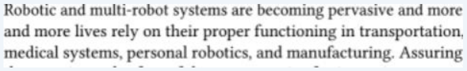
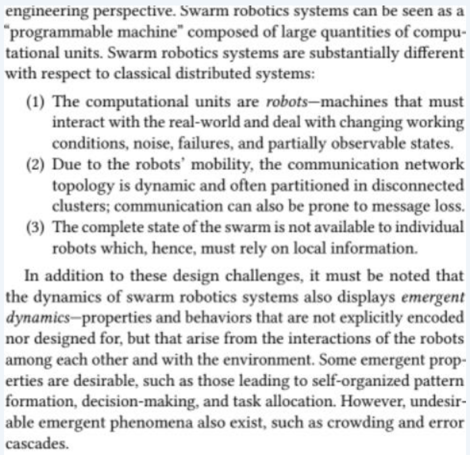
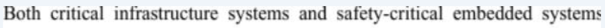
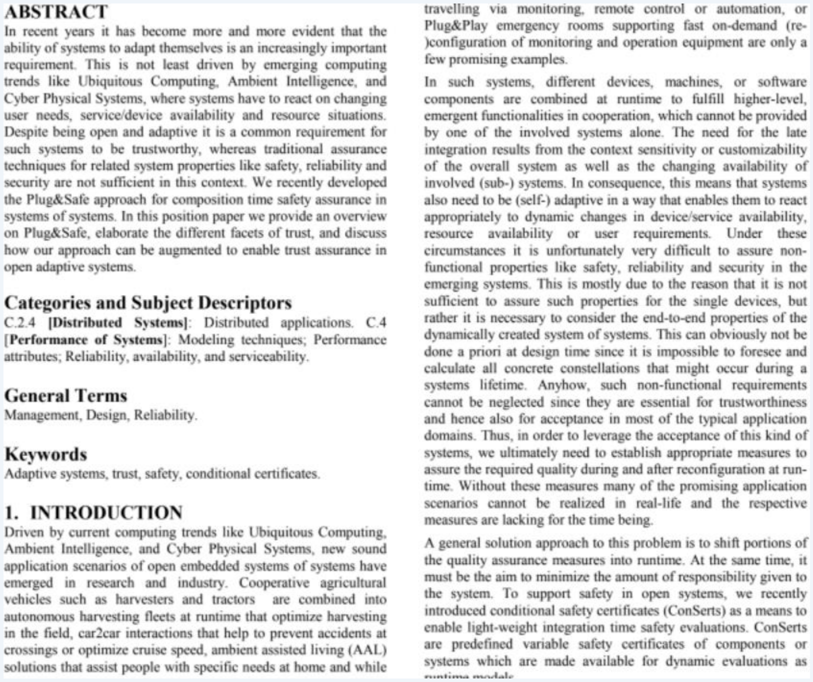
Summaries with Coded Segments - SafeSecMergedFinalCoding_latest.mx22

Code	Coded segments	Summary
System and Its Properties	 Robotic and multi-robot systems are becoming pervasive and more and more lives rely on their proper functioning in transportation, medical systems, personal robotics, and manufacturing. Assuring Engineering safety in swarm robotics: 1 - 1 (0)  engineering perspective. Swarm robotics systems can be seen as a "programmable machine" composed of large quantities of computational units. Swarm robotics systems are substantially different with respect to classical distributed systems: (1) The computational units are *robots*—machines that must interact with the real-world and deal with changing working conditions, noise, failures, and partially observable states. (2) Due to the robots' mobility, the communication network topology is dynamic and often partitioned in disconnected clusters; communication can also be prone to message loss. (3) The complete state of the swarm is not available to individual robots which, hence, must rely on local information. In addition to these design challenges, it must be noted that the dynamics of swarm robotics systems also displays *emergent dynamics*—properties and behaviors that are not explicitly encoded nor designed for, but that arise from the interactions of the robots among each other and with the environment. Some emergent properties are desirable, such as those leading to self-organized pattern formation, decision-making, and task allocation. However, undesirable emergent phenomena also exist, such as crowding and error cascades. Engineering safety in swarm robotics: 1 - 1 (0)	
System and Its Properties\Domain	 Both critical infrastructure systems and safety-critical embedded systems Achieving critical system survivability through software archit: 2 - 2 (0)	



ABSTRACT

In recent years it has become more and more evident that the ability of systems to adapt themselves is an increasingly important requirement. This is not least driven by emerging computing trends like Ubiquitous Computing, Ambient Intelligence, and Cyber Physical Systems, where systems have to react on changing user needs, service/device availability and resource situations. Despite being open and adaptive it is a common requirement for such systems to be trustworthy, whereas traditional assurance techniques for related system properties like safety, reliability and security are not sufficient in this context. We recently developed the Plug&Safe approach for composition time safety assurance in systems of systems. In this position paper we provide an overview on Plug&Safe, elaborate the different facets of trust, and discuss how our approach can be augmented to enable trust assurance in open adaptive systems.

Categories and Subject Descriptors

C.2.4 **[Distributed Systems]**: Distributed applications. C.4 [**Performance of Systems**]: Modeling techniques; Performance attributes; Reliability, availability, and serviceability.

General Terms

Management, Design, Reliability.

Keywords

Adaptive systems, trust, safety, conditional certificates.

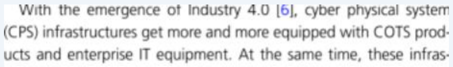
1. INTRODUCTION

Driven by current computing trends like Ubiquitous Computing, Ambient Intelligence, and Cyber Physical Systems, new sound application scenarios of open embedded systems of systems have emerged in research and industry. Cooperative agricultural vehicles such as harvesters and tractors are combined into autonomous harvesting fleets at runtime that optimize harvesting in the field, car2car interactions that help to prevent accidents at crossings or optimize cruise speed, ambient assisted living (AAL) solutions that assist people with specific needs at home and while travelling via monitoring, remote control or automation, or Plug&Play emergency rooms supporting fast on-demand (re-)configuration of monitoring and operation equipment are only a few promising examples.

In such systems, different devices, machines, or software components are combined at runtime to fulfill higher-level, emergent functionalities in cooperation, which cannot be provided by one of the involved systems alone. The need for the late integration results from the context sensitivity or customizability of the overall system as well as the changing availability of involved (sub-) systems. In consequence, this means that systems also need to be (self-) adaptive in a way that enables them to react appropriately to dynamic changes in device/service availability, resource availability or user requirements. Under these circumstances it is unfortunately very difficult to assure non-functional properties like safety, reliability and security in the emerging systems. This is mostly due to the reason that it is not sufficient to assure such properties for the single devices, but rather it is necessary to consider the end-to-end properties of the dynamically created system of systems. This can obviously not be done a priori at design time since it is impossible to foresee and calculate all concrete constellations that might occur during a systems lifetime. Anyhow, such non-functional requirements cannot be neglected since they are essential for trustworthiness and hence also for acceptance in most of the typical application domains. Thus, in order to leverage the acceptance of this kind of systems, we ultimately need to establish appropriate measures to assure the required quality during and after reconfiguration at run-time. Without these measures many of the promising application scenarios cannot be realized in real-life and the respective measures are lacking for the time being.

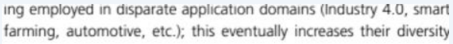
A general solution approach to this problem is to shift portions of the quality assurance measures into runtime. At the same time, it must be the aim to minimize the amount of responsibility given to the system. To support safety in open systems, we recently introduced conditional safety certificates (ConSerts) as a means to enable light-weight integration time safety evaluations. ConSerts are predefined variable safety certificates of components or systems which are made available for dynamic evaluations as runtime models.

Approaching runtime trust assurance in open adaptive systems: 1 - 1 (0)



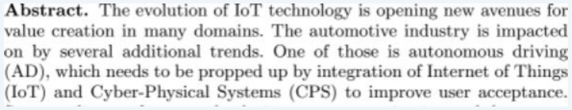
With the emergence of Industry 4.0 [6], cyber physical system (CPS) infrastructures get more and more equipped with COTS products and enterprise IT equipment. At the same time, these infras-

Countering targeted cyber-physical attacks using anomaly detect: 1 - 1 (0)



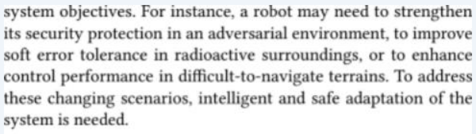
ing employed in disparate application domains (Industry 4.0, smart farming, automotive, etc.); this eventually increases their diversity

Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)

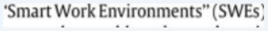


Abstract. The evolution of IoT technology is opening new avenues for value creation in many domains. The automotive industry is impacted on by several additional trends. One of those is autonomous driving (AD), which needs to be propped up by integration of Internet of Things (IoT) and Cyber-Physical Systems (CPS) to improve user acceptance.

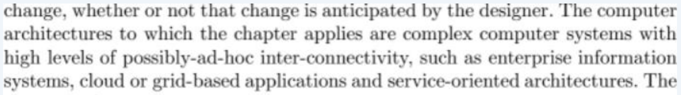
Digital Twins for Dependability Improvement of Autonomous Drivi: 2 - 2 (0)


system objectives. For instance, a robot may need to strengthen its security protection in an adversarial environment, to improve soft error tolerance in radioactive surroundings, or to enhance control performance in difficult-to-navigate terrains. To address these changing scenarios, intelligent and safe adaptation of the system is needed.

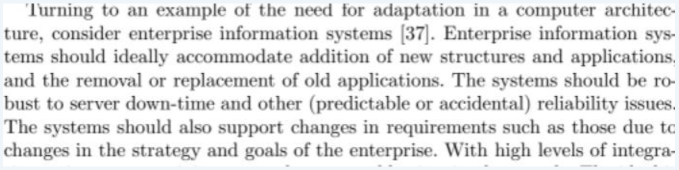
Know the unknowns addressing disturbances and uncertainties in: 7 - 7 (0)


‘Smart Work Environments” (SWEs)

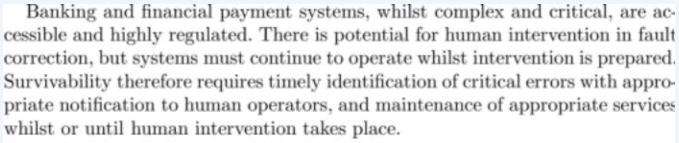
Ontology development for run-time safety management methodology: 1 - 1 (0)


change, whether or not that change is anticipated by the designer. The computer architectures to which the chapter applies are complex computer systems with high levels of possibly-ad-hoc inter-connectivity, such as enterprise information systems, cloud or grid-based applications and service-oriented architectures. The

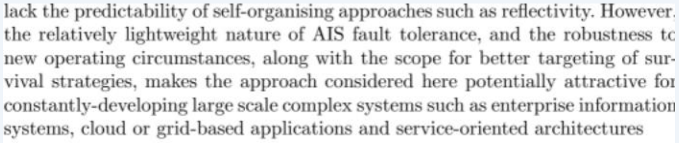
Self-organisation for Survival in Complex Computer Architecture: 1 - 1 (0)


Turning to an example of the need for adaptation in a computer architecture, consider enterprise information systems [37]. Enterprise information systems should ideally accommodate addition of new structures and applications, and the removal or replacement of old applications. The systems should be robust to server down-time and other (predictable or accidental) reliability issues. The systems should also support changes in requirements such as those due to changes in the strategy and goals of the enterprise. With high levels of integra-

Self-organisation for Survival in Complex Computer Architecture: 4 - 4 (0)

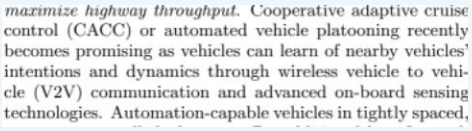

Banking and financial payment systems, whilst complex and critical, are accessible and highly regulated. There is potential for human intervention in fault correction, but systems must continue to operate whilst intervention is prepared. Survivability therefore requires timely identification of critical errors with appropriate notification to human operators, and maintenance of appropriate services whilst or until human intervention takes place.

Self-organisation for Survival in Complex Computer Architecture: 10 - 10 (0)


lack the predictability of self-organising approaches such as reflectivity. However, the relatively lightweight nature of AIS fault tolerance, and the robustness to new operating circumstances, along with the scope for better targeting of survival strategies, makes the approach considered here potentially attractive for constantly-developing large scale complex systems such as enterprise information systems, cloud or grid-based applications and service-oriented architectures

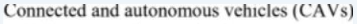
Self-organisation for Survival in Complex Computer Architecture: 16 - 16 (0)

System and Its Properties\Domain\Automotive



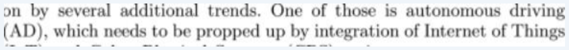
maximize highway throughput. Cooperative adaptive cruise control (CACC) or automated vehicle platooning recently becomes promising as vehicles can learn of nearby vehicles' intentions and dynamics through wireless vehicle to vehicle (V2V) communication and advanced on-board sensing technologies. Automation-capable vehicles in tightly spaced,

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 1 - 1 (0)



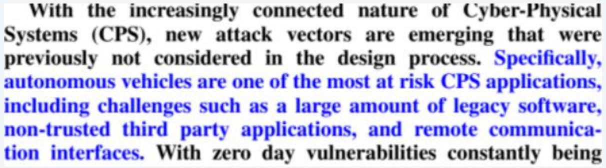
Connected and autonomous vehicles (CAVs)

A simplified approach for dynamic security risk management in c: 2 - 2 (0)



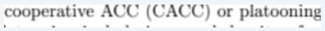
on by several additional trends. One of those is autonomous driving (AD), which needs to be propped up by integration of Internet of Things

Digital Twins for Dependability Improvement of Autonomous Drivi: 2 - 2 (0)



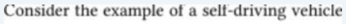
With the increasingly connected nature of Cyber-Physical Systems (CPS), new attack vectors are emerging that were previously not considered in the design process. Specifically, autonomous vehicles are one of the most at risk CPS applications, including challenges such as a large amount of legacy software, non-trusted third party applications, and remote communication interfaces. With zero day vulnerabilities constantly being

Integrated moving target defense and control reconfiguration fo: 1 - 1 (0)



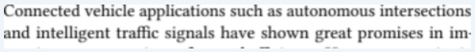
cooperative ACC (CACC) or platooning

Is your commute driving you crazy a study of misbehavior in veh: 1 - 1 (0)



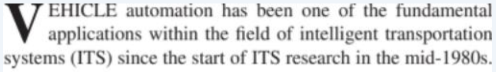
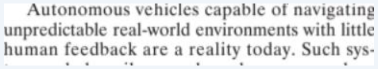
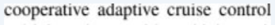
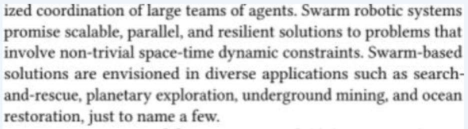
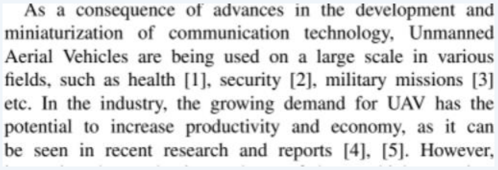
Consider the example of a self-driving vehicle

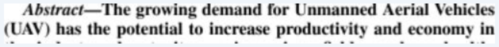
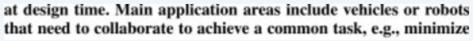
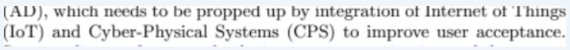
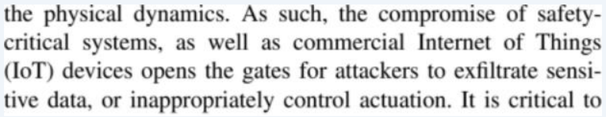
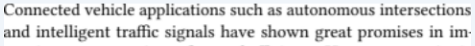
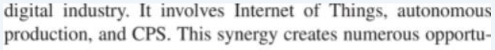
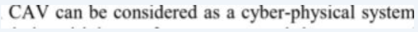
Know the unknowns addressing disturbances and uncertainties in: 1 - 1 (0)

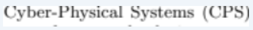
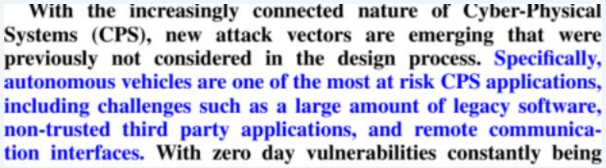
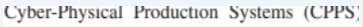
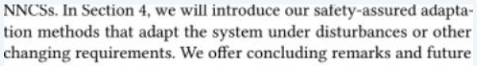
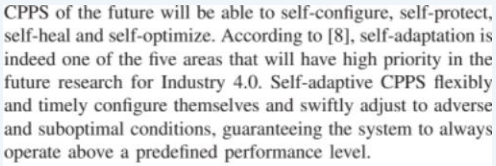


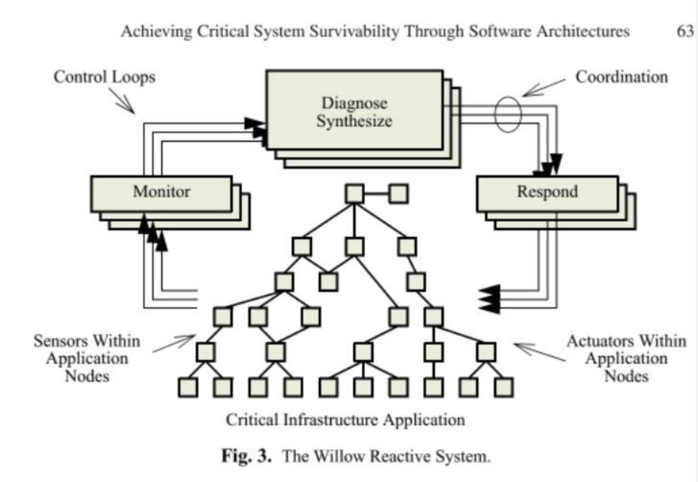
Connected vehicle applications such as autonomous intersections and intelligent traffic signals have shown great promises in im-

Network and system level security in connected vehicle applicat: 1 - 1 (0)

	 VEHICLE automation has been one of the fundamental applications within the field of intelligent transportation systems (ITS) since the start of ITS research in the mid-1980s. Potential cyberattacks on automated vehicles: 1 - 1 (0)
	 Connected and automated vehicles SARA Security Automotive Risk Analysis Method: 1 - 1 (0)
	 Autonomous vehicles capable of navigating unpredictable real-world environments with little human feedback are a reality today. Such sys- Security vulnerabilities of connected vehicle streams and their: 1 - 1 (0)
	 Level 3 (L3) Automated Driving (AD) systems TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)
	 cooperative adaptive cruise control Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)
	 vehicles Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)
System and Its Properties\Domain\UAV & other Robotics	 ized coordination of large teams of agents. Swarm robotic systems promise scalable, parallel, and resilient solutions to problems that involve non-trivial space-time dynamic constraints. Swarm-based solutions are envisioned in diverse applications such as search-and-rescue, planetary exploration, underground mining, and ocean restoration, just to name a few. Engineering safety in swarm robotics: 1 - 1 (0)
	 As a consequence of advances in the development and miniaturization of communication technology, Unmanned Aerial Vehicles are being used on a large scale in various fields, such as health [1], security [2], military missions [3] etc. In the industry, the growing demand for UAV has the potential to increase productivity and economy, as it can be seen in recent research and reports [4], [5]. However, STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)

	 *Abstract*—The growing demand for Unmanned Aerial Vehicles (UAV) has the potential to increase productivity and economy in STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)
	 at design time. Main application areas include vehicles or robots that need to collaborate to achieve a common task, e.g., minimize Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)
	unmanned aerial systems (UAS) VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)
System and Its Properties\Domain\Internet of Things	 (AD), which needs to be propped up by integration of Internet of Things (IoT) and Cyber-Physical Systems (CPS) to improve user acceptance. Digital Twins for Dependability Improvement of Autonomous Drivi: 2 - 2 (0)
	 the physical dynamics. As such, the compromise of safety-critical systems, as well as commercial Internet of Things (IoT) devices opens the gates for attackers to exfiltrate sensitive data, or inappropriately control actuation. It is critical to Integrated moving target defense and control reconfiguration fo: 1 - 1 (0)
	 Connected vehicle applications such as autonomous intersections and intelligent traffic signals have shown great promises in im- Network and system level security in connected vehicle applicat: 1 - 1 (0)
	Internet of Things (IoT) Ontology development for run-time safety management methodology: 1 - 1 (0)
	 digital industry. It involves Internet of Things, autonomous production, and CPS. This synergy creates numerous opportu- Protecting cyber physical production systems using anomaly dete: 1 - 1 (0)
System and Its Properties\Domain\critical	 CAV can be considered as a cyber-physical system

infrastructure & production systems	A simplified approach for dynamic security risk management in c: 2 - 2 (0)  Cyber-Physical Systems (CPS) Digital Twins for Dependability Improvement of Autonomous Drivi: 2 - 2 (0)  With the increasingly connected nature of Cyber-Physical Systems (CPS), new attack vectors are emerging that were previously not considered in the design process. Specifically, autonomous vehicles are one of the most at risk CPS applications, including challenges such as a large amount of legacy software, non-trusted third party applications, and remote communication interfaces. With zero day vulnerabilities constantly being Integrated moving target defense and control reconfiguration fo: 1 - 1 (0)  Cyber-Physical Production Systems (CPPS) Protecting cyber physical production systems using anomaly dete: 1 - 1 (0)
System and Its Properties\Self-Adaptation	 NNCSs. In Section 4, we will introduce our safety-assured adaptation methods that adapt the system under disturbances or other changing requirements. We offer concluding remarks and future Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)  CPPS of the future will be able to self-configure, self-protect, self-heal and self-optimize. According to [8], self-adaptation is indeed one of the five areas that will have high priority in the future research for Industry 4.0. Self-adaptive CPPS flexibly and timely configure themselves and swiftly adjust to adverse and suboptimal conditions, guaranteeing the system to always operate above a predefined performance level. Protecting cyber physical production systems using anomaly dete: 1 - 1 (0)
System and Its Properties\Self-Adaptation\Architecture	

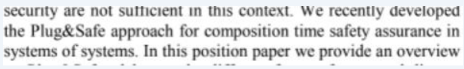


Achieving Critical System Survivability Through Software Architectures 63

Control Loops
Diagnose Synthesize
Coordination
Monitor
Respond
Sensors Within Application Nodes
Actuators Within Application Nodes
Critical Infrastructure Application

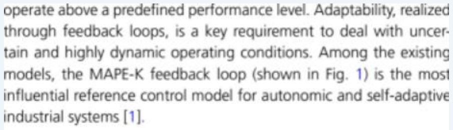
Fig. 3. The Willow Reactive System.

Achieving critical system survivability through software archit: 13: 29|395 - 13: 386|639 (0)



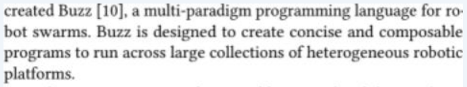
security are not sufficient in this context. We recently developed the Plug&Safe approach for composition time safety assurance in systems of systems. In this position paper we provide an overview

Approaching runtime trust assurance in open adaptive systems: 1 - 1 (0)



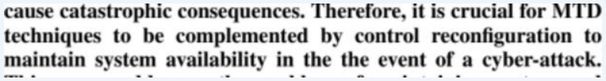
operate above a predefined performance level. Adaptability, realized through feedback loops, is a key requirement to deal with uncertain and highly dynamic operating conditions. Among the existing models, the MAPE-K feedback loop (shown in Fig. 1) is the most influential reference control model for autonomic and self-adaptive industrial systems [1].

Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)



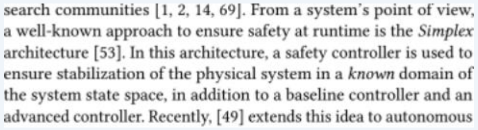
created Buzz [10], a multi-paradigm programming language for robot swarms. Buzz is designed to create concise and composable programs to run across large collections of heterogeneous robotic platforms.

Engineering safety in swarm robotics: 2 - 2 (0)



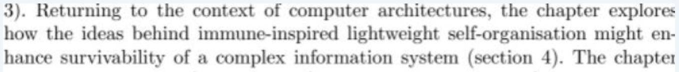
cause catastrophic consequences. Therefore, it is crucial for MTD techniques to be complemented by control reconfiguration to maintain system availability in the the event of a cyber-attack.

Integrated moving target defense and control reconfiguration fo: 1 - 1 (0)



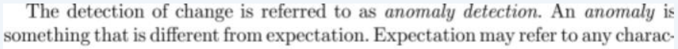
search communities [1, 2, 14, 69]. From a system's point of view, a well-known approach to ensure safety at runtime is the *Simplex* architecture [53]. In this architecture, a safety controller is used to ensure stabilization of the physical system in a *known* domain of the system state space, in addition to a baseline controller and an advanced controller. Recently, [49] extends this idea to autonomous

Know the unknowns addressing disturbances and uncertainties in: 7 - 7 (0)



3). Returning to the context of computer architectures, the chapter explores how the ideas behind immune-inspired lightweight self-organisation might enhance survivability of a complex information system (section 4). The chapter

Self-organisation for Survival in Complex Computer Architecture: 2 - 2 (0)



The detection of change is referred to as *anomaly detection*. An *anomaly* is something that is different from expectation. Expectation may refer to any charac-

Self-organisation for Survival in Complex Computer Architecture: 4 - 4 (0)

System and Its Properties\Self-Adaptation\Architecture\Mape-K or Variant

Achieving Critical System Survivability Through Software Architectures 63

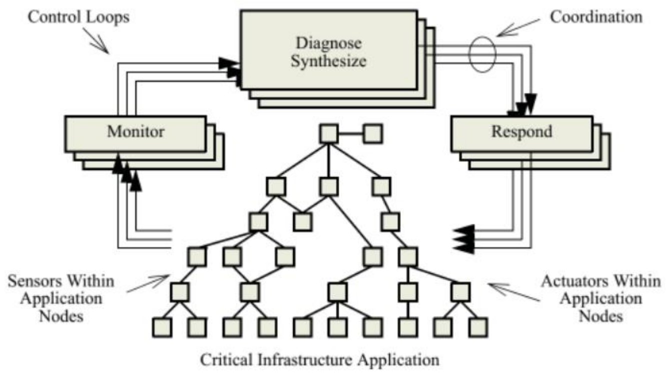
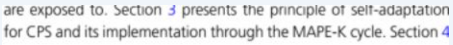


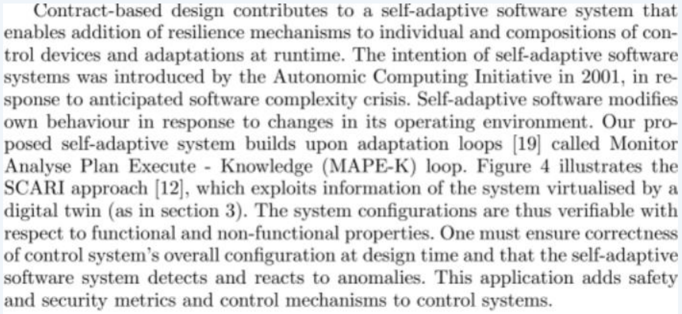
Fig. 3. The Willow Reactive System.

Achieving critical system survivability through software archit: 13: 29|395 - 13: 386|639 (0)



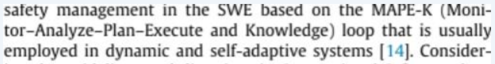
are exposed to. Section 3 presents the principle of self-adaptation for CPS and its implementation through the MAPE-K cycle. Section 4

Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)



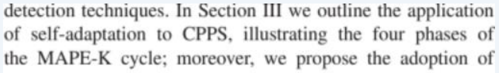
Contract-based design contributes to a self-adaptive software system that enables addition of resilience mechanisms to individual and compositions of control devices and adaptations at runtime. The intention of self-adaptive software systems was introduced by the Autonomic Computing Initiative in 2001, in response to anticipated software complexity crisis. Self-adaptive software modifies own behaviour in response to changes in its operating environment. Our proposed self-adaptive system builds upon adaptation loops [19] called Monitor Analyse Plan Execute - Knowledge (MAPE-K) loop. Figure 4 illustrates the SCARI approach [12], which exploits information of the system virtualised by a digital twin (as in section 3). The system configurations are thus verifiable with respect to functional and non-functional properties. One must ensure correctness of control system's overall configuration at design time and that the self-adaptive software system detects and reacts to anomalies. This application adds safety and security metrics and control mechanisms to control systems.

Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



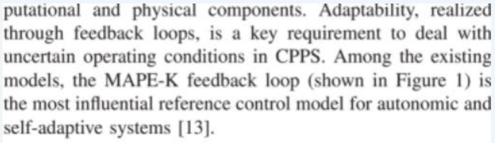
safety management in the SWE based on the MAPE-K (Monitor–Analyze–Plan–Execute and Knowledge) loop that is usually employed in dynamic and self-adaptive systems [14]. Consider-

Ontology development for run-time safety management methodology: 2 - 2 (0)



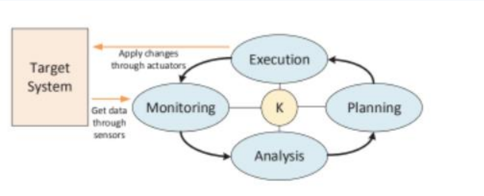
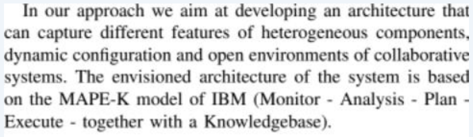
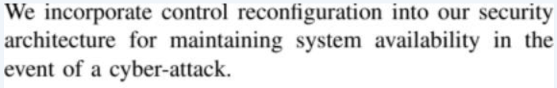
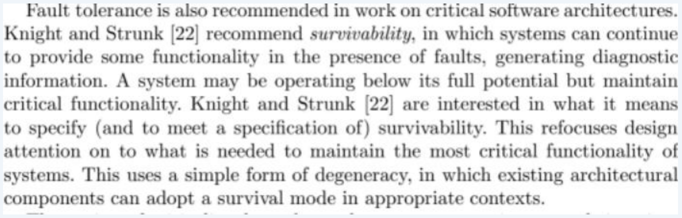
detection techniques. In Section III we outline the application of self-adaptation to CPPS, illustrating the four phases of the MAPE-K cycle; moreover, we propose the adoption of

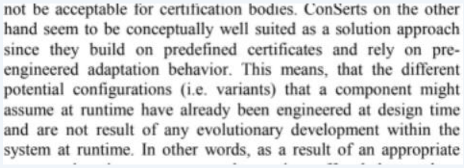
Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)



putational and physical components. Adaptability, realized through feedback loops, is a key requirement to deal with uncertain operating conditions in CPPS. Among the existing models, the MAPE-K feedback loop (shown in Figure 1) is the most influential reference control model for autonomic and self-adaptive systems [13].

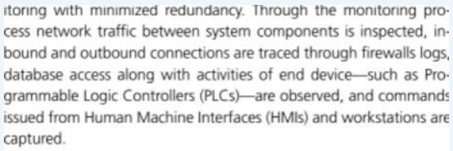
Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)

	 Target System Apply changes through actuators Get data through sensors Execution Monitoring K Planning Analysis Fig. 1. MAPE-K cycle. Protecting cyber physical production systems using anomaly dete: 2: 318|190 - 2: 566|306 (0)
	 In our approach we aim at developing an architecture that can capture different features of heterogeneous components, dynamic configuration and open environments of collaborative systems. The envisioned architecture of the system is based on the MAPE-K model of IBM (Monitor - Analysis - Plan - Execute - together with a Knowledgebase). Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)
System and Its Properties\Self-Adaptation\Architecture\No Specific Architecture	 We incorporate control reconfiguration into our security architecture for maintaining system availability in the event of a cyber-attack. Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)
	 Fault tolerance is also recommended in work on critical software architectures. Knight and Strunk [22] recommend *survivability*, in which systems can continue to provide some functionality in the presence of faults, generating diagnostic information. A system may be operating below its full potential but maintain critical functionality. Knight and Strunk [22] are interested in what it means to specify (and to meet a specification of) survivability. This refocuses design attention on to what is needed to maintain the most critical functionality of systems. This uses a simple form of degeneracy, in which existing architectural components can adopt a survival mode in appropriate contexts. Self-organisation for Survival in Complex Computer Architecture: 9 - 9 (0)
System and Its Properties\Self-Adaptation\Way of Implementation	



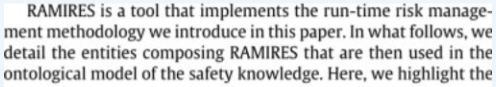
not be acceptable for certification bodies. ConSerts on the other hand seem to be conceptually well suited as a solution approach since they build on predefined certificates and rely on pre-engineered adaptation behavior. This means, that the different potential configurations (i.e. variants) that a component might assume at runtime have already been engineered at design time and are not result of any evolutionary development within the system at runtime. In other words, as a result of an appropriate

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



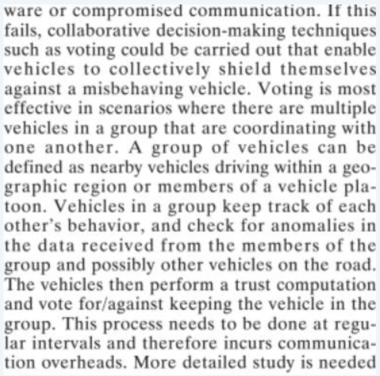
itoring with minimized redundancy. Through the monitoring process network traffic between system components is inspected, inbound and outbound connections are traced through firewalls logs, database access along with activities of end device—such as Programmable Logic Controllers (PLCs)—are observed, and commands issued from Human Machine Interfaces (HMIs) and workstations are captured.

Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)



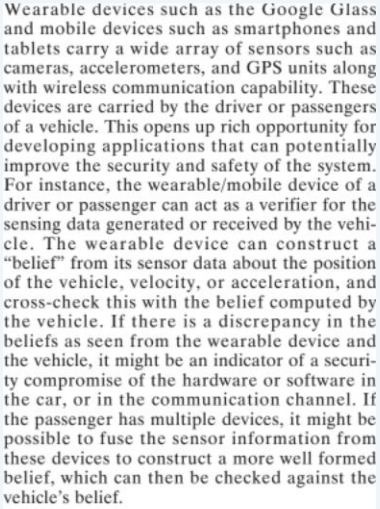
RAMIRES is a tool that implements the run-time risk management methodology we introduce in this paper. In what follows, we detail the entities composing RAMIRES that are then used in the ontological model of the safety knowledge. Here, we highlight the

Ontology development for run-time safety management methodology: 3 - 3 (0)



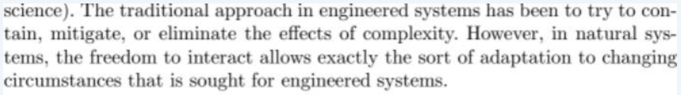
ware or compromised communication. If this fails, collaborative decision-making techniques such as voting could be carried out that enable vehicles to collectively shield themselves against a misbehaving vehicle. Voting is most effective in scenarios where there are multiple vehicles in a group that are coordinating with one another. A group of vehicles can be defined as nearby vehicles driving within a geographic region or members of a vehicle platoon. Vehicles in a group keep track of each other's behavior, and check for anomalies in the data received from the members of the group and possibly other vehicles on the road. The vehicles then perform a trust computation and vote for/against keeping the vehicle in the group. This process needs to be done at regular intervals and therefore incurs communication overheads. More detailed study is needed

Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)



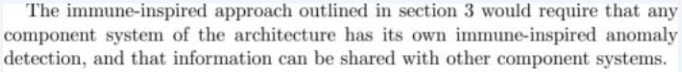
Wearable devices such as the Google Glass and mobile devices such as smartphones and tablets carry a wide array of sensors such as cameras, accelerometers, and GPS units along with wireless communication capability. These devices are carried by the driver or passengers of a vehicle. This opens up rich opportunity for developing applications that can potentially improve the security and safety of the system. For instance, the wearable/mobile device of a driver or passenger can act as a verifier for the sensing data generated or received by the vehicle. The wearable device can construct a "belief" from its sensor data about the position of the vehicle, velocity, or acceleration, and cross-check this with the belief computed by the vehicle. If there is a discrepancy in the beliefs as seen from the wearable device and the vehicle, it might be an indicator of a security compromise of the hardware or software in the car, or in the communication channel. If the passenger has multiple devices, it might be possible to fuse the sensor information from these devices to construct a more well formed belief, which can then be checked against the vehicle's belief.

Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)



science). The traditional approach in engineered systems has been to try to contain, mitigate, or eliminate the effects of complexity. However, in natural systems, the freedom to interact allows exactly the sort of adaptation to changing circumstances that is sought for engineered systems.

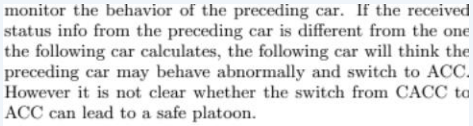
Self-organisation for Survival in Complex Computer Architecture: 2 - 2 (0)



The immune-inspired approach outlined in section 3 would require that any component system of the architecture has its own immune-inspired anomaly detection, and that information can be shared with other component systems.

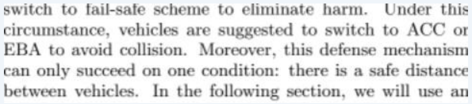
Self-organisation for Survival in Complex Computer Architecture: 12 - 12 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Limitation

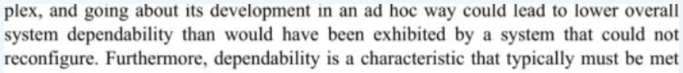
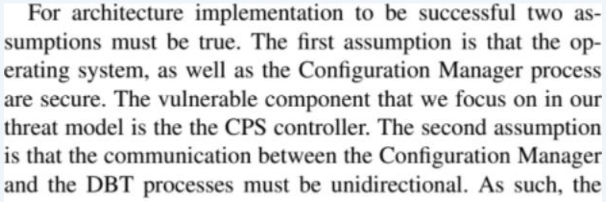
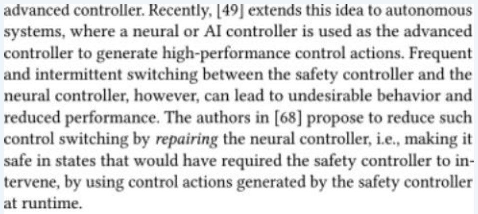
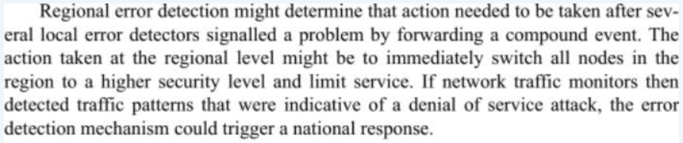
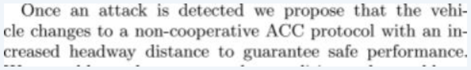


monitor the behavior of the preceding car. If the received status info from the preceding car is different from the one the following car calculates, the following car will think the preceding car may behave abnormally and switch to ACC. However it is not clear whether the switch from CACC to ACC can lead to a safe platoon.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)

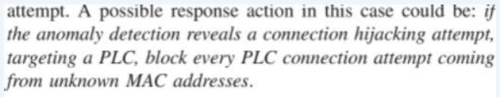


switch to fail-safe scheme to eliminate harm. Under this circumstance, vehicles are suggested to switch to ACC or EBA to avoid collision. Moreover, this defense mechanism can only succeed on one condition: there is a safe distance between vehicles. In the following section, we will use an

	A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)
	 plex, and going about its development in an ad hoc way could lead to lower overall system dependability than would have been exhibited by a system that could not reconfigure. Furthermore, dependability is a characteristic that typically must be met Achieving critical system survivability through software archit: 19 - 19 (0)
	 For architecture implementation to be successful two assumptions must be true. The first assumption is that the operating system, as well as the Configuration Manager process are secure. The vulnerable component that we focus on in our threat model is the the CPS controller. The second assumption is that the communication between the Configuration Manager and the DBT processes must be unidirectional. As such, the Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)
	 advanced controller. Recently, [49] extends this idea to autonomous systems, where a neural or AI controller is used as the advanced controller to generate high-performance control actions. Frequent and intermittent switching between the safety controller and the neural controller, however, can lead to undesirable behavior and reduced performance. The authors in [68] propose to reduce such control switching by *repairing* the neural controller, i.e., making it safe in states that would have required the safety controller to intervene, by using control actions generated by the safety controller at runtime. Know the unknowns addressing disturbances and uncertainties in: 7 - 7 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Increased Vigilance\Increase Security Level	 Regional error detection might determine that action needed to be taken after several local error detectors signalled a problem by forwarding a compound event. The action taken at the regional level might be to immediately switch all nodes in the region to a higher security level and limit service. If network traffic monitors then detected traffic patterns that were indicative of a denial of service attack, the error detection mechanism could trigger a national response. Achieving critical system survivability through software archit: 15 - 15 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Increased	 Once an attack is detected we propose that the vehicle changes to a non-cooperative ACC protocol with an increased headway distance to guarantee safe performance. Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)

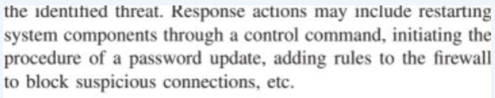
Vigilance\Increase Safety Margin

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Increased Vigilance\Block Communication



attempt. A possible response action in this case could be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses.*

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)



the identified threat. Response actions may include restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)

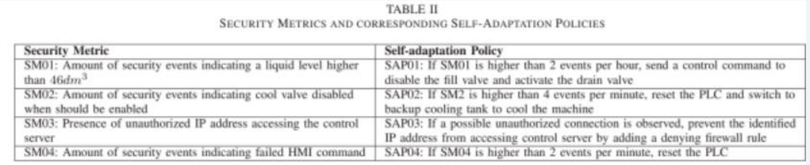
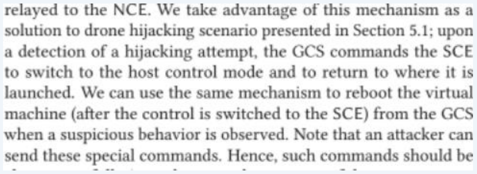


TABLE II
SECURITY METRICS AND CORRESPONDING SELF-ADAPTATION POLICIES

Security Metric	Self-adaptation Policy
SM01: Amount of security events indicating a liquid level higher than $46dm^3$	SAP01: If SM01 is higher than 2 events per hour, send a control command to disable the fill valve and activate the drain valve
SM02: Amount of security events indicating cool valve disabled when should be enabled	SAP02: If SM2 is higher than 4 events per minute, reset the PLC and switch to backup cooling tank to cool the machine
SM03: Presence of unauthorized IP address accessing the control server	SAP03: If a possible unauthorized connection is observed, prevent the identified IP address from accessing control server by adding a denying firewall rule
SM04: Amount of security events indicating failed HMI command	SAP04: If SM04 is higher than 2 events per minute, reset the PLC

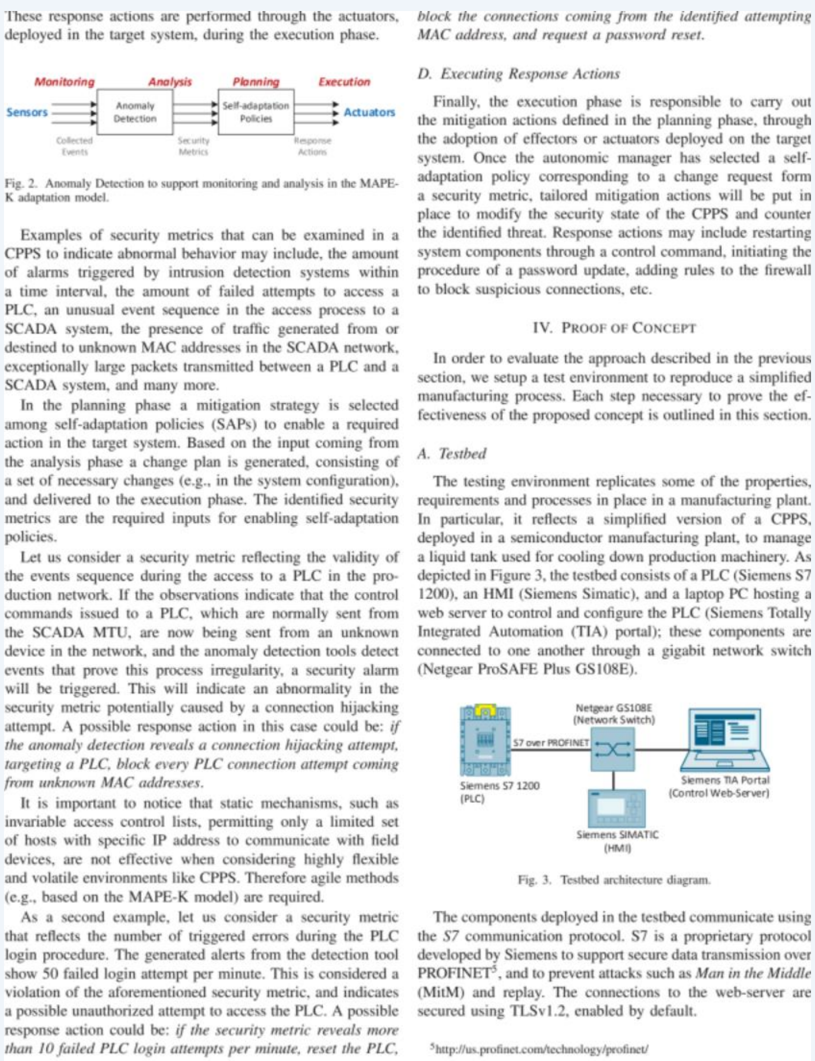
Protecting cyber physical production systems using anomaly dete: 8: 33|687 - 8: 550|791 (0)



relayed to the NCE. We take advantage of this mechanism as a solution to drone hijacking scenario presented in Section 5.1; upon a detection of a hijacking attempt, the GCS commands the SCE to switch to the host control mode and to return to where it is launched. We can use the same mechanism to reboot the virtual machine (after the control is switched to the SCE) from the GCS when a suspicious behavior is observed. Note that an attacker can send these special commands. Hence, such commands should be

VirtualDrone virtual sensing actuation and communication for at: 4 - 4 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Stop Operation\Restart System



These response actions are performed through the actuators, deployed in the target system, during the execution phase.

Monitoring
Analysis
Planning
Execution
Sensors
Anomaly Detection
Self-adaptation Policies
Actuators
Collected Events
Security Metrics
Response Actions

Fig. 2. Anomaly Detection to support monitoring and analysis in the MAPE-K adaptation model.

Examples of security metrics that can be examined in a CPPS to indicate abnormal behavior may include, the amount of alarms triggered by intrusion detection systems within a time interval, the amount of failed attempts to access a PLC, an unusual event sequence in the access process to a SCADA system, the presence of traffic generated from or destined to unknown MAC addresses in the SCADA network, exceptionally large packets transmitted between a PLC and a SCADA system, and many more.

In the planning phase a mitigation strategy is selected among self-adaptation policies (SAPs) to enable a required action in the target system. Based on the input coming from the analysis phase a change plan is generated, consisting of a set of necessary changes (e.g., in the system configuration), and delivered to the execution phase. The identified security metrics are the required inputs for enabling self-adaptation policies.

Let us consider a security metric reflecting the validity of the events sequence during the access to a PLC in the production network. If the observations indicate that the control commands issued to a PLC, which are normally sent from the SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tools detect events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric potentially caused by a connection hijacking attempt. A possible response action in this case could be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses.*

It is important to notice that static mechanisms, such as invariable access control lists, permitting only a limited set of hosts with specific IP address to communicate with field devices, are not effective when considering highly flexible and volatile environments like CPPS. Therefore agile methods (e.g., based on the MAPE-K model) are required.

As a second example, let us consider a security metric that reflects the number of triggered errors during the PLC login procedure. The generated alerts from the detection tool show 50 failed login attempt per minute. This is considered a violation of the aforementioned security metric, and indicates a possible unauthorized attempt to access the PLC. A possible response action could be: *if the security metric reveals more than 10 failed PLC login attempts per minute, reset the PLC, block the connections coming from the identified attempting MAC address, and request a password reset.*

D. Executing Response Actions

Finally, the execution phase is responsible to carry out the mitigation actions defined in the planning phase, through the adoption of effectors or actuators deployed on the target system. Once the autonomic manager has selected a self-adaptation policy corresponding to a change request form a security metric, tailored mitigation actions will be put in place to modify the security state of the CPPS and counter the identified threat. Response actions may include restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

IV. Proof of Concept

In order to evaluate the approach described in the previous section, we setup a test environment to reproduce a simplified manufacturing process. Each step necessary to prove the effectiveness of the proposed concept is outlined in this section.

A. Testbed

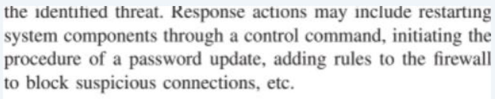
The testing environment replicates some of the properties, requirements and processes in place in a manufacturing plant. In particular, it reflects a simplified version of a CPPS, deployed in a semiconductor manufacturing plant, to manage a liquid tank used for cooling down production machinery. As depicted in Figure 3, the testbed consists of a PLC (Siemens S7 1200), an HMI (Siemens Simatic), and a laptop PC hosting a web server to control and configure the PLC (Siemens Totally Integrated Automation (TIA) portal); these components are connected to one another through a gigabit network switch (Netgear ProSAFE Plus GS108E).

Fig. 3. Testbed architecture diagram.

The components deployed in the testbed communicate using the *S7* communication protocol. S7 is a proprietary protocol developed by Siemens to support secure data transmission over PROFINET[5], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

[5] http://us.profinet.com/technology/profinet/

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)


the identified threat. Response actions may include restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)

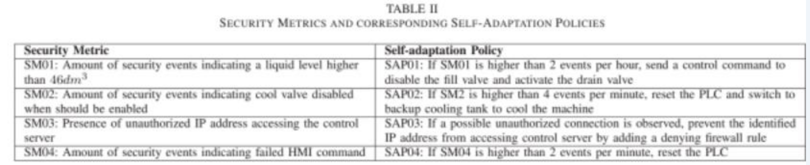
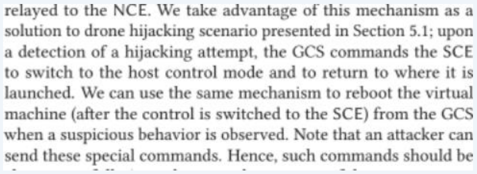


TABLE II

SECURITY METRICS AND CORRESPONDING SELF-ADAPTATION POLICIES

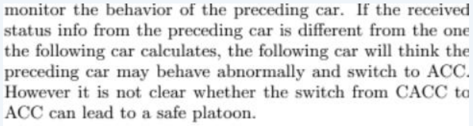
Security Metric	Self-adaptation Policy
SM01: Amount of security events indicating a liquid level higher than $46dm^3$	SAP01: If SM01 is higher than 2 events per hour, send a control command to disable the fill valve and activate the drain valve
SM02: Amount of security events indicating cool valve disabled when should be enabled	SAP02: If SM2 is higher than 4 events per minute, reset the PLC and switch to backup cooling tank to cool the machine
SM03: Presence of unauthorized IP address accessing the control server	SAP03: If a possible unauthorized connection is observed, prevent the identified IP address from accessing control server by adding a denying firewall rule
SM04: Amount of security events indicating failed HMI command	SAP04: If SM04 is higher than 2 events per minute, reset the PLC

Protecting cyber physical production systems using anomaly dete: 8: 33|687 - 8: 550|791 (0)

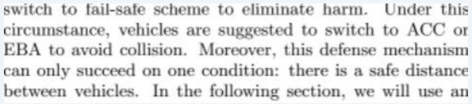

relayed to the NCE. We take advantage of this mechanism as a solution to drone hijacking scenario presented in Section 5.1; upon a detection of a hijacking attempt, the GCS commands the SCE to switch to the host control mode and to return to where it is launched. We can use the same mechanism to reboot the virtual machine (after the control is switched to the SCE) from the GCS when a suspicious behavior is observed. Note that an attacker can send these special commands. Hence, such commands should be

VirtualDrone virtual sensing actuation and communication for at: 4 - 4 (0)

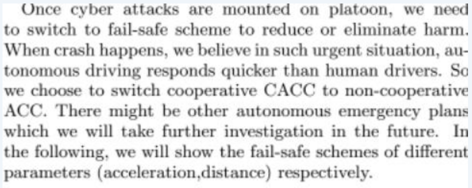
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Stop Operation\End Cooperation


monitor the behavior of the preceding car. If the received status info from the preceding car is different from the one the following car calculates, the following car will think the preceding car may behave abnormally and switch to ACC. However it is not clear whether the switch from CACC to ACC can lead to a safe platoon.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)

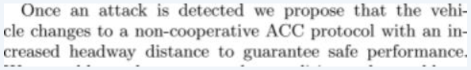

switch to fail-safe scheme to eliminate harm. Under this circumstance, vehicles are suggested to switch to ACC or EBA to avoid collision. Moreover, this defense mechanism can only succeed on one condition: there is a safe distance between vehicles. In the following section, we will use an

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)



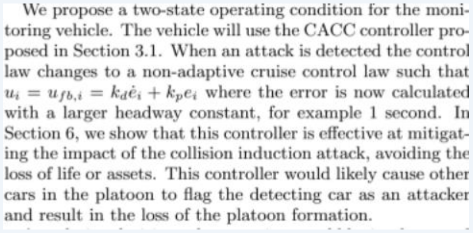
Once cyber attacks are mounted on platoon, we need to switch to fail-safe scheme to reduce or eliminate harm. When crash happens, we believe in such urgent situation, autonomous driving responds quicker than human drivers. So we choose to switch cooperative CACC to non-cooperative ACC. There might be other autonomous emergency plans which we will take further investigation in the future. In the following, we will show the fail-safe schemes of different parameters (acceleration,distance) respectively.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 8 - 8 (0)



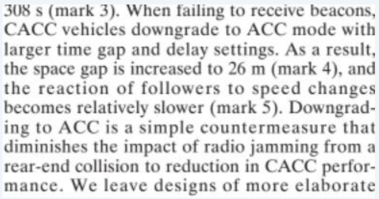
Once an attack is detected we propose that the vehicle changes to a non-cooperative ACC protocol with an increased headway distance to guarantee safe performance.

Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)



We propose a two-state operating condition for the monitoring vehicle. The vehicle will use the CACC controller proposed in Section 3.1. When an attack is detected the control law changes to a non-adaptive cruise control law such that $u_i = u_{fb,i} = k_d\dot{e}_i + k_p e_i$ where the error is now calculated with a larger headway constant, for example 1 second. In Section 6, we show that this controller is effective at mitigating the impact of the collision induction attack, avoiding the loss of life or assets. This controller would likely cause other cars in the platoon to flag the detecting car as an attacker and result in the loss of the platoon formation.

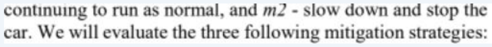
Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



308 s (mark 3). When failing to receive beacons, CACC vehicles downgrade to ACC mode with larger time gap and delay settings. As a result, the space gap is increased to 26 m (mark 4), and the reaction of followers to speed changes becomes relatively slower (mark 5). Downgrading to ACC is a simple countermeasure that diminishes the impact of radio jamming from a rear-end collision to reduction in CACC performance. We leave designs of more elaborate

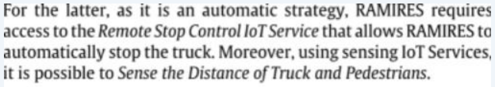
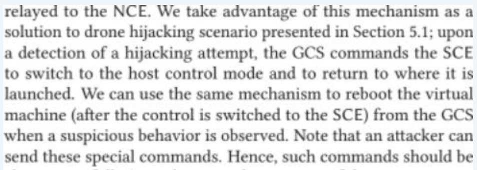
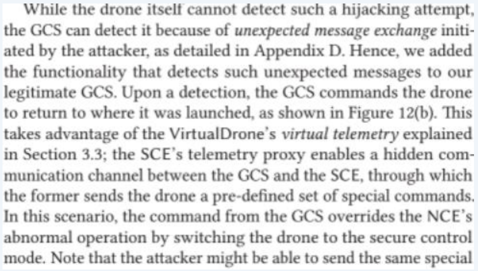
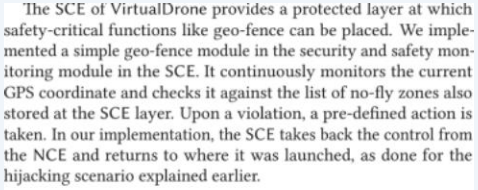
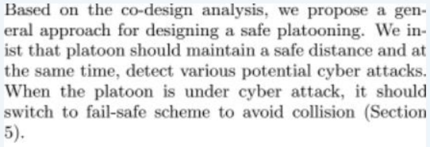
Security vulnerabilities of connected vehicle streams and their: 5 - 5 (0)

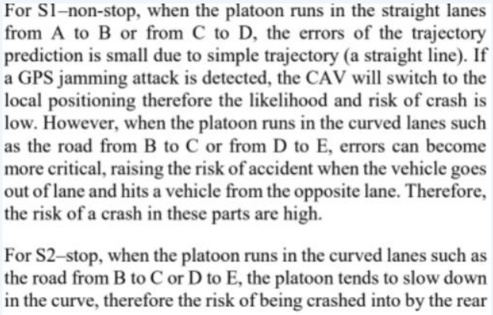
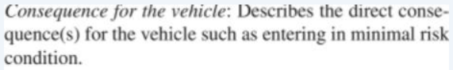
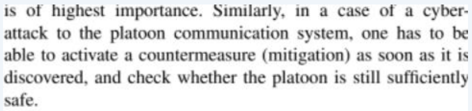
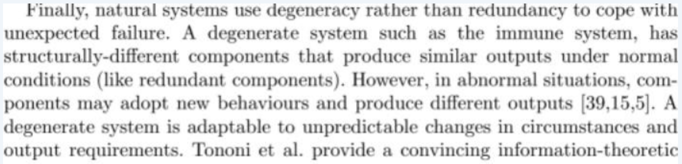
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Stop Operation\Stop Operation

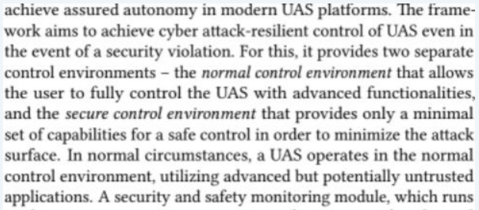


continuing to run as normal, and *m2* - slow down and stop the car. We will evaluate the three following mitigation strategies:

A simplified approach for dynamic security risk management in c: 7 - 7 (0)

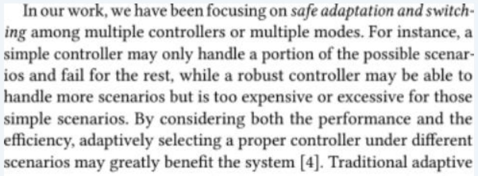
	 For the latter, as it is an automatic strategy, RAMIRES requires access to the *Remote Stop Control IoT Service* that allows RAMIRES to automatically stop the truck. Moreover, using sensing IoT Services, it is possible to *Sense the Distance of Truck and Pedestrians.* Ontology development for run-time safety management methodology: 7 - 7 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Stop Operation\Return to Base	 relayed to the NCE. We take advantage of this mechanism as a solution to drone hijacking scenario presented in Section 5.1; upon a detection of a hijacking attempt, the GCS commands the SCE to switch to the host control mode and to return to where it is launched. We can use the same mechanism to reboot the virtual machine (after the control is switched to the SCE) from the GCS when a suspicious behavior is observed. Note that an attacker can send these special commands. Hence, such commands should be VirtualDrone virtual sensing actuation and communication for at: 4 - 4 (0)  While the drone itself cannot detect such a hijacking attempt, the GCS can detect it because of *unexpected message exchange* initiated by the attacker, as detailed in Appendix D. Hence, we added the functionality that detects such unexpected messages to our legitimate GCS. Upon a detection, the GCS commands the drone to return to where it was launched, as shown in Figure 12(b). This takes advantage of the VirtualDrone's *virtual telemetry* explained in Section 3.3; the SCE's telemetry proxy enables a hidden communication channel between the GCS and the SCE, through which the former sends the drone a pre-defined set of special commands. In this scenario, the command from the GCS overrides the NCE's abnormal operation by switching the drone to the secure control mode. Note that the attacker might be able to send the same special VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)  The SCE of VirtualDrone provides a protected layer at which safety-critical functions like geo-fence can be placed. We implemented a simple geo-fence module in the security and safety monitoring module in the SCE. It continuously monitors the current GPS coordinate and checks it against the list of no-fly zones also stored at the SCE layer. Upon a violation, a pre-defined action is taken. In our implementation, the SCE takes back the control from the NCE and returns to where it was launched, as done for the hijacking scenario explained earlier. VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Not Specified	 Based on the co-design analysis, we propose a general approach for designing a safe platooning. We insist that platoon should maintain a safe distance and at the same time, detect various potential cyber attacks. When the platoon is under cyber attack, it should switch to fail-safe scheme to avoid collision (Section 5). A Functional Co-Design towards Safe and Secure Vehicle Platooni: 2 - 2 (0)

	 For S1–non-stop, when the platoon runs in the straight lanes from A to B or from C to D, the errors of the trajectory prediction is small due to simple trajectory (a straight line). If a GPS jamming attack is detected, the CAV will switch to the local positioning therefore the likelihood and risk of crash is low. However, when the platoon runs in the curved lanes such as the road from B to C or from D to E, errors can become more critical, raising the risk of accident when the vehicle goes out of lane and hits a vehicle from the opposite lane. Therefore, the risk of a crash in these parts are high. For S2–stop, when the platoon runs in the curved lanes such as the road from B to C or D to E, the platoon tends to slow down in the curve, therefore the risk of being crashed into by the rear A simplified approach for dynamic security risk management in c: 7 - 7 (0)
	 *Consequence for the vehicle*: Describes the direct consequence(s) for the vehicle such as entering in minimal risk condition. Potential cyberattacks on automated vehicles: 4 - 4 (0)
	 is of highest importance. Similarly, in a case of a cyber-attack to the platoon communication system, one has to be able to activate a countermeasure (mitigation) as soon as it is discovered, and check whether the platoon is still sufficiently safe. Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Service Reduction\Degeneracy	 Finally, natural systems use degeneracy rather than redundancy to cope with unexpected failure. A degenerate system such as the immune system, has structurally-different components that produce similar outputs under normal conditions (like redundant components). However, in abnormal situations, components may adopt new behaviours and produce different outputs [39,15,5]. A degenerate system is adaptable to unpredictable changes in circumstances and output requirements. Tononi et al. provide a convincing information-theoretic Self-organisation for Survival in Complex Computer Architecture: 5 - 5 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Service Reduction\Minimal Operation	


achieve assured autonomy in modern UAS platforms. The framework aims to achieve cyber attack-resilient control of UAS even in the event of a security violation. For this, it provides two separate control environments – the *normal control environment* that allows the user to fully control the UAS with advanced functionalities, and the *secure control environment* that provides only a minimal set of capabilities for a safe control in order to minimize the attack surface. In normal circumstances, a UAS operates in the normal control environment, utilizing advanced but potentially untrusted applications. A security and safety monitoring module, which runs

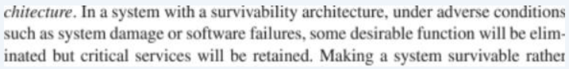
VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Service Reduction\Save Energy, Computation Time

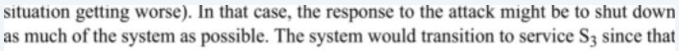

In our work, we have been focusing on *safe adaptation and switching* among multiple controllers or multiple modes. For instance, a simple controller may only handle a portion of the possible scenarios and fail for the rest, while a robust controller may be able to handle more scenarios but is too expensive or excessive for those simple scenarios. By considering both the performance and the efficiency, adaptively selecting a proper controller under different scenarios may greatly benefit the system [4]. Traditional adaptive

Know the unknowns addressing disturbances and uncertainties in: 7 - 7 (0)

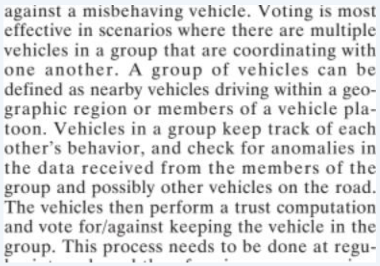
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Service Reduction\Remove Component


chitecture. In a system with a survivability architecture, under adverse conditions such as system damage or software failures, some desirable function will be eliminated but critical services will be retained. Making a system survivable rather

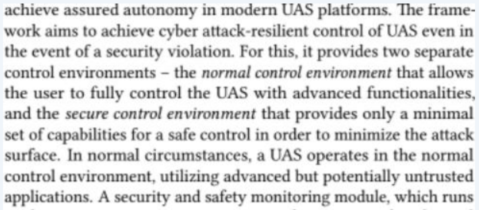
Achieving critical system survivability through software archit: 1 - 1 (0)


situation getting worse). In that case, the response to the attack might be to shut down as much of the system as possible. The system would transition to service S_3 since that

Achieving critical system survivability through software archit: 10 - 10 (0)


against a misbehaving vehicle. Voting is most effective in scenarios where there are multiple vehicles in a group that are coordinating with one another. A group of vehicles can be defined as nearby vehicles driving within a geographic region or members of a vehicle platoon. Vehicles in a group keep track of each other's behavior, and check for anomalies in the data received from the members of the group and possibly other vehicles on the road. The vehicles then perform a trust computation and vote for/against keeping the vehicle in the group. This process needs to be done at regu-

Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)



achieve assured autonomy in modern UAS platforms. The framework aims to achieve cyber attack-resilient control of UAS even in the event of a security violation. For this, it provides two separate control environments – the *normal control environment* that allows the user to fully control the UAS with advanced functionalities, and the *secure control environment* that provides only a minimal set of capabilities for a safe control in order to minimize the attack surface. In normal circumstances, a UAS operates in the normal control environment, utilizing advanced but potentially untrusted applications. A security and safety monitoring module, which runs

VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Redundancy

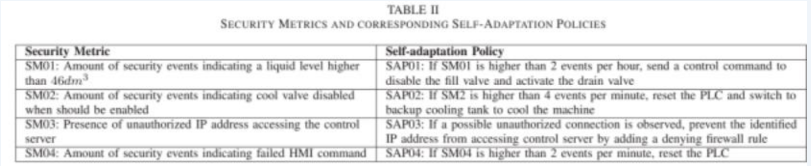
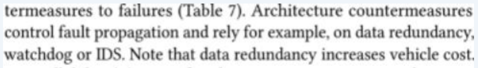


TABLE II
SECURITY METRICS AND CORRESPONDING SELF-ADAPTATION POLICIES

Security Metric	Self-adaptation Policy
SM01: Amount of security events indicating a liquid level higher than $46dm^3$	SAP01: If SM01 is higher than 2 events per hour, send a control command to disable the fill valve and activate the drain valve
SM02: Amount of security events indicating cool valve disabled when should be enabled	SAP02: If SM2 is higher than 4 events per minute, reset the PLC and switch to backup cooling tank to cool the machine
SM03: Presence of unauthorized IP address accessing the control server	SAP03: If a possible unauthorized connection is observed, prevent the identified IP address from accessing control server by adding a denying firewall rule
SM04: Amount of security events indicating failed HMI command	SAP04: If SM04 is higher than 2 events per minute, reset the PLC

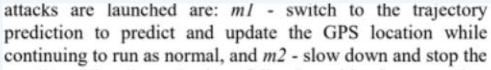
Protecting cyber physical production systems using anomaly dete: 8: 33|687 - 8: 550|791 (0)



termeasures to failures (Table 7). Architecture countermeasures control fault propagation and rely for example, on data redundancy, watchdog or IDS. Note that data redundancy increases vehicle cost.

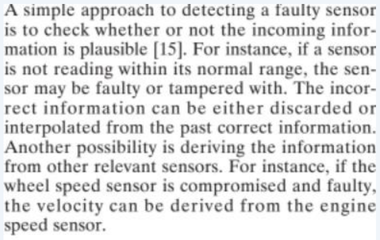
SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Redundancy\State Estimation



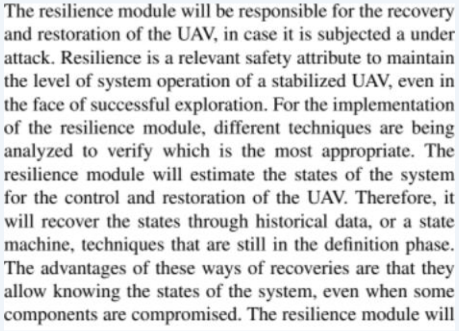
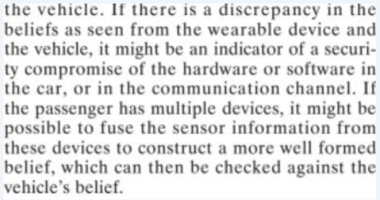
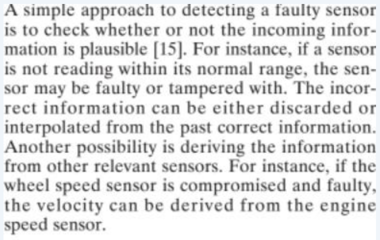
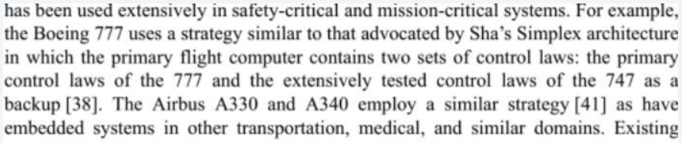
attacks are launched are: *m1* - switch to the trajectory prediction to predict and update the GPS location while continuing to run as normal, and *m2* - slow down and stop the

A simplified approach for dynamic security risk management in c: 7 - 7 (0)

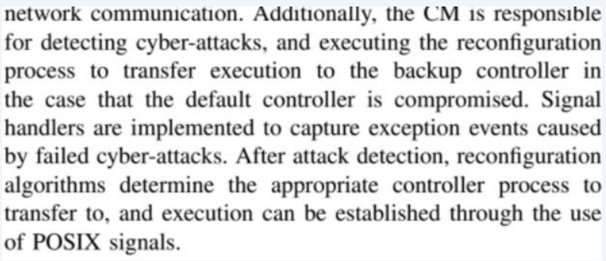


A simple approach to detecting a faulty sensor is to check whether or not the incoming information is plausible [15]. For instance, if a sensor is not reading within its normal range, the sensor may be faulty or tampered with. The incorrect information can be either discarded or interpolated from the past correct information. Another possibility is deriving the information from other relevant sensors. For instance, if the wheel speed sensor is compromised and faulty, the velocity can be derived from the engine speed sensor.

Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)

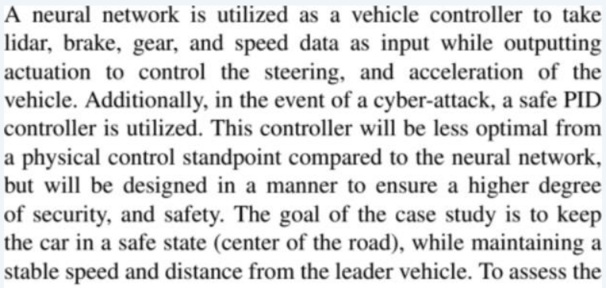
	 The resilience module will be responsible for the recovery and restoration of the UAV, in case it is subjected a under attack. Resilience is a relevant safety attribute to maintain the level of system operation of a stabilized UAV, even in the face of successful exploration. For the implementation of the resilience module, different techniques are being analyzed to verify which is the most appropriate. The resilience module will estimate the states of the system for the control and restoration of the UAV. Therefore, it will recover the states through historical data, or a state machine, techniques that are still in the definition phase. The advantages of these ways of recoveries are that they allow knowing the states of the system, even when some components are compromised. The resilience module will STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Redundancy\Redundant Sensor	 the vehicle. If there is a discrepancy in the beliefs as seen from the wearable device and the vehicle, it might be an indicator of a security compromise of the hardware or software in the car, or in the communication channel. If the passenger has multiple devices, it might be possible to fuse the sensor information from these devices to construct a more well formed belief, which can then be checked against the vehicle's belief. Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)  A simple approach to detecting a faulty sensor is to check whether or not the incoming information is plausible [15]. For instance, if a sensor is not reading within its normal range, the sensor may be faulty or tampered with. The incorrect information can be either discarded or interpolated from the past correct information. Another possibility is deriving the information from other relevant sensors. For instance, if the wheel speed sensor is compromised and faulty, the velocity can be derived from the engine speed sensor. Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Redundancy\Redundant Controller	 has been used extensively in safety-critical and mission-critical systems. For example, the Boeing 777 uses a strategy similar to that advocated by Sha's Simplex architecture in which the primary flight computer contains two sets of control laws: the primary control laws of the 777 and the extensively tested control laws of the 747 as a backup [38]. The Airbus A330 and A340 employ a similar strategy [41] as have embedded systems in other transportation, medical, and similar domains. Existing

Achieving critical system survivability through software archit: 17 - 17 (0)



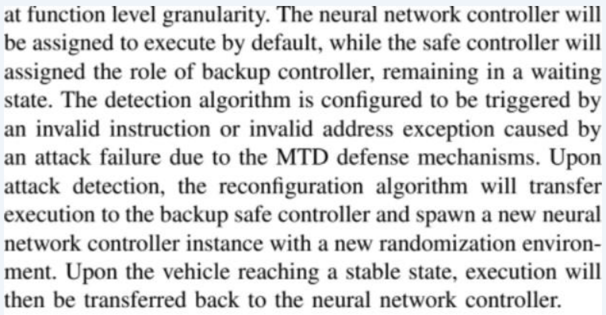
network communication. Additionally, the CM is responsible for detecting cyber-attacks, and executing the reconfiguration process to transfer execution to the backup controller in the case that the default controller is compromised. Signal handlers are implemented to capture exception events caused by failed cyber-attacks. After attack detection, reconfiguration algorithms determine the appropriate controller process to transfer to, and execution can be established through the use of POSIX signals.

Integrated moving target defense and control reconfiguration fo: 4 - 4 (0)



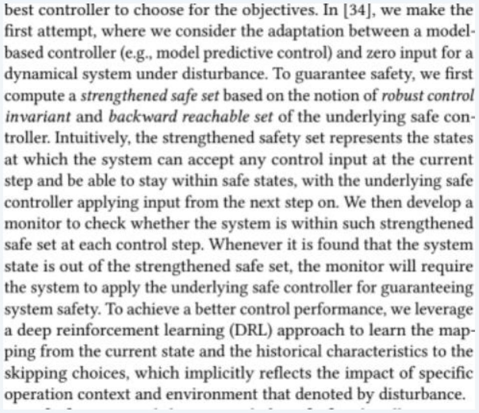
A neural network is utilized as a vehicle controller to take lidar, brake, gear, and speed data as input while outputting actuation to control the steering, and acceleration of the vehicle. Additionally, in the event of a cyber-attack, a safe PID controller is utilized. This controller will be less optimal from a physical control standpoint compared to the neural network, but will be designed in a manner to ensure a higher degree of security, and safety. The goal of the case study is to keep the car in a safe state (center of the road), while maintaining a stable speed and distance from the leader vehicle. To assess the

Integrated moving target defense and control reconfiguration fo: 7 - 7 (0)



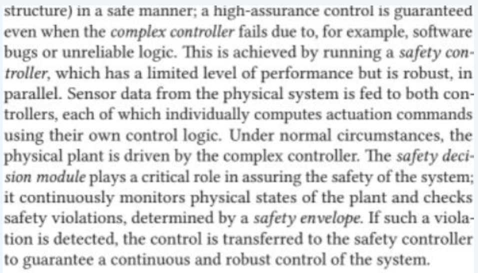
at function level granularity. The neural network controller will be assigned to execute by default, while the safe controller will assigned the role of backup controller, remaining in a waiting state. The detection algorithm is configured to be triggered by an invalid instruction or invalid address exception caused by an attack failure due to the MTD defense mechanisms. Upon attack detection, the reconfiguration algorithm will transfer execution to the backup safe controller and spawn a new neural network controller instance with a new randomization environment. Upon the vehicle reaching a stable state, execution will then be transferred back to the neural network controller.

Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)



best controller to choose for the objectives. In [34], we make the first attempt, where we consider the adaptation between a model-based controller (e.g., model predictive control) and zero input for a dynamical system under disturbance. To guarantee safety, we first compute a *strengthened safe set* based on the notion of *robust control invariant* and *backward reachable set* of the underlying safe controller. Intuitively, the strengthened safety set represents the states at which the system can accept any control input at the current step and be able to stay within safe states, with the underlying safe controller applying input from the next step on. We then develop a monitor to check whether the system is within such strengthened safe set at each control step. Whenever it is found that the system state is out of the strengthened safe set, the monitor will require the system to apply the underlying safe controller for guaranteeing system safety. To achieve a better control performance, we leverage a deep reinforcement learning (DRL) approach to learn the mapping from the current state and the historical characteristics to the skipping choices, which implicitly reflects the impact of specific operation context and environment that denoted by disturbance.

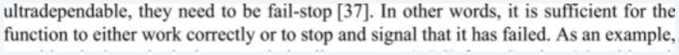
Know the unknowns addressing disturbances and uncertainties in: 8 - 8 (0)



structure) in a safe manner; a high-assurance control is guaranteed even when the *complex controller* fails due to, for example, software bugs or unreliable logic. This is achieved by running a *safety controller*, which has a limited level of performance but is robust, in parallel. Sensor data from the physical system is fed to both controllers, each of which individually computes actuation commands using their own control logic. Under normal circumstances, the physical plant is driven by the complex controller. The *safety decision module* plays a critical role in assuring the safety of the system; it continuously monitors physical states of the plant and checks safety violations, determined by a *safety envelope*. If such a violation is detected, the control is transferred to the safety controller to guarantee a continuous and robust control of the system.

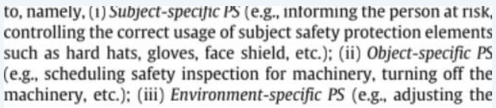
VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Other



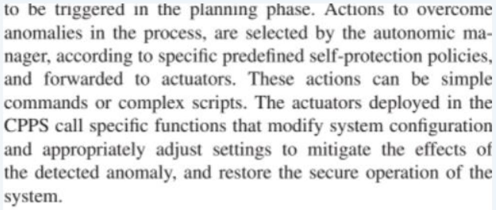
ultradependable, they need to be fail-stop [37]. In other words, it is sufficient for the function to either work correctly or to stop and signal that it has failed. As an example,

Achieving critical system survivability through software archit: 18 - 18 (0)



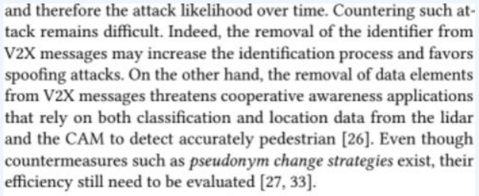
to, namely, (i) *Subject-specific PS* (e.g., informing the person at risk, controlling the correct usage of subject safety protection elements such as hard hats, gloves, face shield, etc.); (ii) *Object-specific PS* (e.g., scheduling safety inspection for machinery, turning off the machinery, etc.); (iii) *Environment-specific PS* (e.g., adjusting the

Ontology development for run-time safety management methodology: 5 - 5 (0)



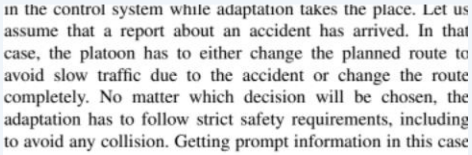
to be triggered in the planning phase. Actions to overcome anomalies in the process, are selected by the autonomic manager, according to specific predefined self-protection policies, and forwarded to actuators. These actions can be simple commands or complex scripts. The actuators deployed in the CPPS call specific functions that modify system configuration and appropriately adjust settings to mitigate the effects of the detected anomaly, and restore the secure operation of the system.

Protecting cyber physical production systems using anomaly dete: 7 - 7 (0)



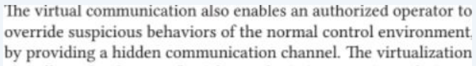
and therefore the attack likelihood over time. Countering such attack remains difficult. Indeed, the removal of the identifier from V2X messages may increase the identification process and favors spoofing attacks. On the other hand, the removal of data elements from V2X messages threatens cooperative awareness applications that rely on both classification and location data from the lidar and the CAM to detect accurately pedestrian [26]. Even though countermeasures such as *pseudonym change strategies* exist, their efficiency still need to be evaluated [27, 33].

SARA Security Automotive Risk Analysis Method: 9 - 9 (0)



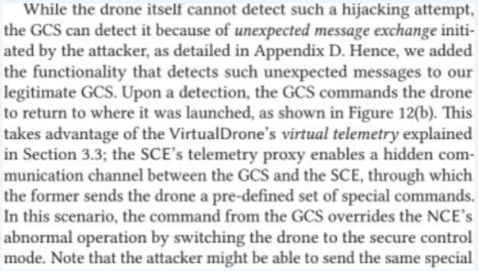
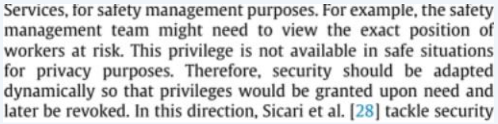
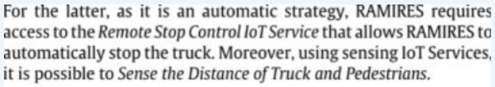
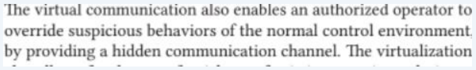
in the control system while adaptation takes the place. Let us assume that a report about an accident has arrived. In that case, the platoon has to either change the planned route to avoid slow traffic due to the accident or change the route completely. No matter which decision will be chosen, the adaptation has to follow strict safety requirements, including to avoid any collision. Getting prompt information in this case

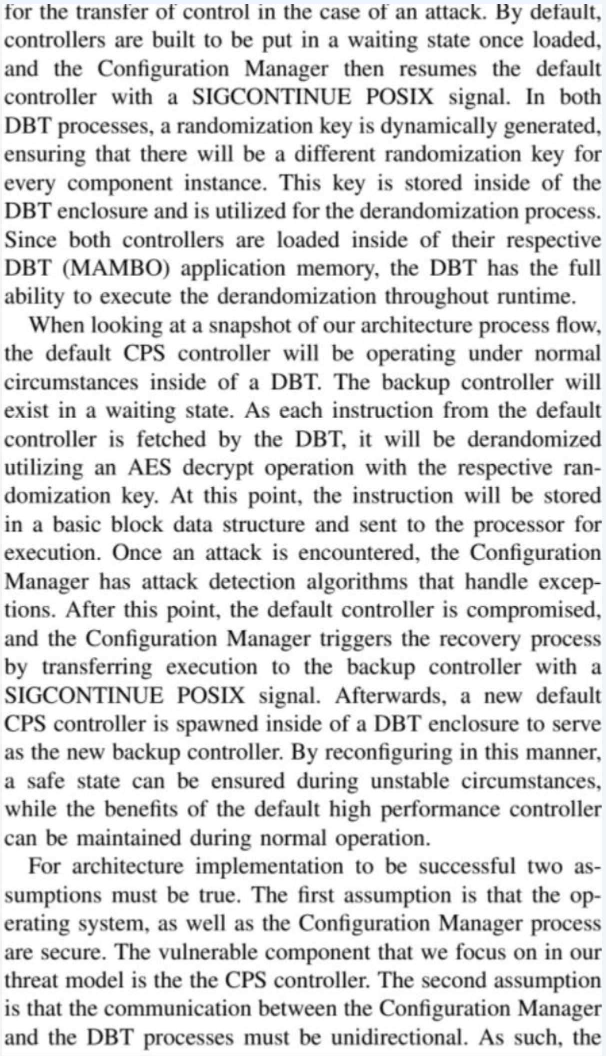
Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)



The virtual communication also enables an authorized operator to override suspicious behaviors of the normal control environment by providing a hidden communication channel. The virtualization

VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)

	 While the drone itself cannot detect such a hijacking attempt, the GCS can detect it because of *unexpected message exchange* initiated by the attacker, as detailed in Appendix D. Hence, we added the functionality that detects such unexpected messages to our legitimate GCS. Upon a detection, the GCS commands the drone to return to where it was launched, as shown in Figure 12(b). This takes advantage of the VirtualDrone's *virtual telemetry* explained in Section 3.3; the SCE's telemetry proxy enables a hidden communication channel between the GCS and the SCE, through which the former sends the drone a pre-defined set of special commands. In this scenario, the command from the GCS overrides the NCE's abnormal operation by switching the drone to the secure control mode. Note that the attacker might be able to send the same special VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Other\Grant Priviledges	 Services, for safety management purposes. For example, the safety management team might need to view the exact position of workers at risk. This privilege is not available in safe situations for privacy purposes. Therefore, security should be adapted dynamically so that privileges would be granted upon need and later be revoked. In this direction, Sicari et al. [28] tackle security Ontology development for run-time safety management methodology: 3 - 3 (0)  For the latter, as it is an automatic strategy, RAMIRES requires access to the *Remote Stop Control IoT Service* that allows RAMIRES to automatically stop the truck. Moreover, using sensing IoT Services, it is possible to *Sense the Distance of Truck and Pedestrians*. Ontology development for run-time safety management methodology: 7 - 7 (0)  The virtual communication also enables an authorized operator to override suspicious behaviors of the normal control environment, by providing a hidden communication channel. The virtualization VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Strategy\Other\Code Decryption	



for the transfer of control in the case of an attack. By default, controllers are built to be put in a waiting state once loaded, and the Configuration Manager then resumes the default controller with a SIGCONTINUE POSIX signal. In both DBT processes, a randomization key is dynamically generated, ensuring that there will be a different randomization key for every component instance. This key is stored inside of the DBT enclosure and is utilized for the derandomization process. Since both controllers are loaded inside of their respective DBT (MAMBO) application memory, the DBT has the full ability to execute the derandomization throughout runtime.

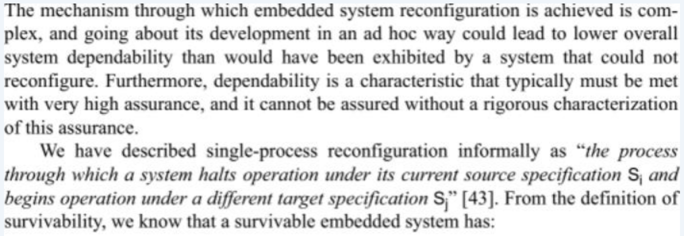
When looking at a snapshot of our architecture process flow, the default CPS controller will be operating under normal circumstances inside of a DBT. The backup controller will exist in a waiting state. As each instruction from the default controller is fetched by the DBT, it will be derandomized utilizing an AES decrypt operation with the respective randomization key. At this point, the instruction will be stored in a basic block data structure and sent to the processor for execution. Once an attack is encountered, the Configuration Manager has attack detection algorithms that handle exceptions. After this point, the default controller is compromised, and the Configuration Manager triggers the recovery process by transferring execution to the backup controller with a SIGCONTINUE POSIX signal. Afterwards, a new default CPS controller is spawned inside of a DBT enclosure to serve as the new backup controller. By reconfiguring in this manner, a safe state can be ensured during unstable circumstances, while the benefits of the default high performance controller can be maintained during normal operation.

For architecture implementation to be successful two assumptions must be true. The first assumption is that the operating system, as well as the Configuration Manager process are secure. The vulnerable component that we focus on in our threat model is the the CPS controller. The second assumption is that the communication between the Configuration Manager and the DBT processes must be unidirectional. As such, the

Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation

Realization\Adaptation Definition

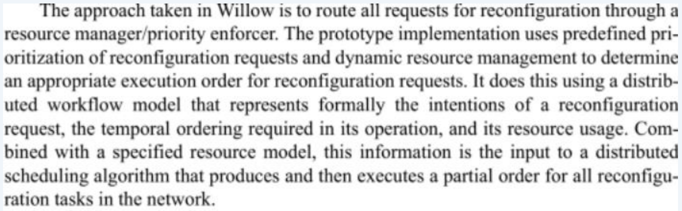


The mechanism through which embedded system reconfiguration is achieved is complex, and going about its development in an ad hoc way could lead to lower overall system dependability than would have been exhibited by a system that could not reconfigure. Furthermore, dependability is a characteristic that typically must be met with very high assurance, and it cannot be assured without a rigorous characterization of this assurance.

We have described single-process reconfiguration informally as "*the process through which a system halts operation under its current source specification* S_i *and begins operation under a different target specification* S_j" [43]. From the definition of survivability, we know that a survivable embedded system has:

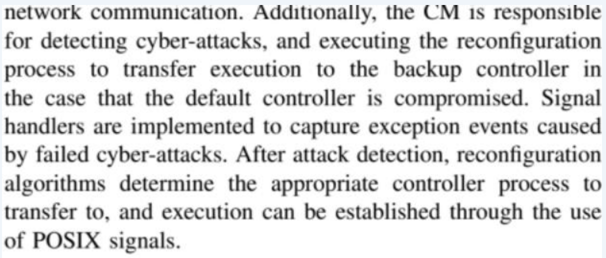
Achieving critical system survivability through software archit: 19 - 19 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Realization\Execute



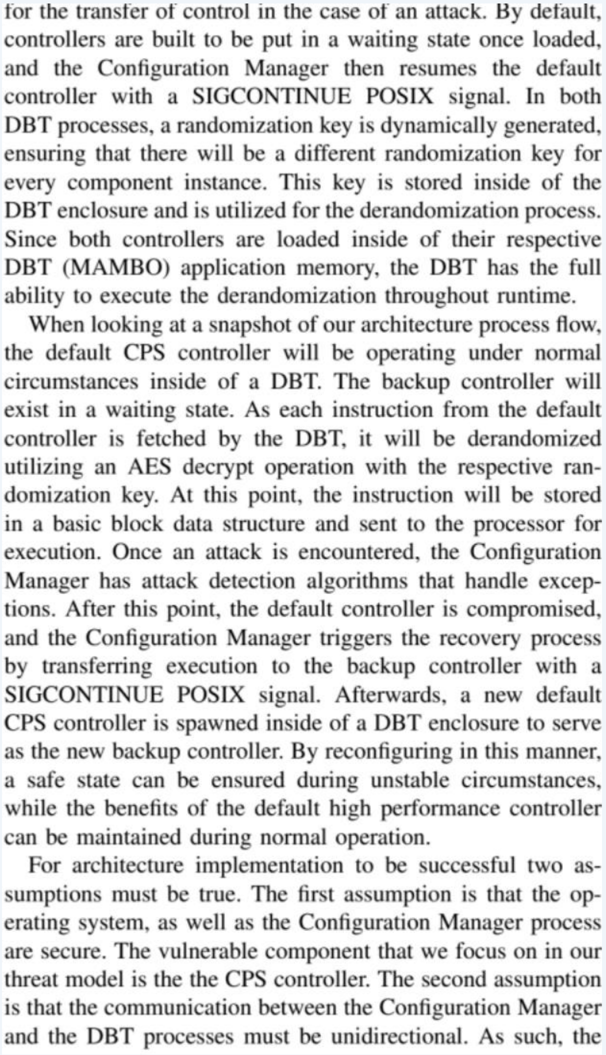
The approach taken in Willow is to route all requests for reconfiguration through a resource manager/priority enforcer. The prototype implementation uses predefined prioritization of reconfiguration requests and dynamic resource management to determine an appropriate execution order for reconfiguration requests. It does this using a distributed workflow model that represents formally the intentions of a reconfiguration request, the temporal ordering required in its operation, and its resource usage. Combined with a specified resource model, this information is the input to a distributed scheduling algorithm that produces and then executes a partial order for all reconfiguration tasks in the network.

Achieving critical system survivability through software archit: 17 - 17 (0)



network communication. Additionally, the CM is responsible for detecting cyber-attacks, and executing the reconfiguration process to transfer execution to the backup controller in the case that the default controller is compromised. Signal handlers are implemented to capture exception events caused by failed cyber-attacks. After attack detection, reconfiguration algorithms determine the appropriate controller process to transfer to, and execution can be established through the use of POSIX signals.

Integrated moving target defense and control reconfiguration fo: 4 - 4 (0)

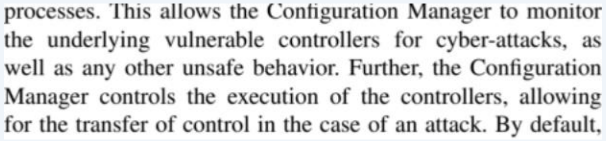


for the transfer of control in the case of an attack. By default, controllers are built to be put in a waiting state once loaded, and the Configuration Manager then resumes the default controller with a SIGCONTINUE POSIX signal. In both DBT processes, a randomization key is dynamically generated, ensuring that there will be a different randomization key for every component instance. This key is stored inside of the DBT enclosure and is utilized for the derandomization process. Since both controllers are loaded inside of their respective DBT (MAMBO) application memory, the DBT has the full ability to execute the derandomization throughout runtime.

When looking at a snapshot of our architecture process flow, the default CPS controller will be operating under normal circumstances inside of a DBT. The backup controller will exist in a waiting state. As each instruction from the default controller is fetched by the DBT, it will be derandomized utilizing an AES decrypt operation with the respective randomization key. At this point, the instruction will be stored in a basic block data structure and sent to the processor for execution. Once an attack is encountered, the Configuration Manager has attack detection algorithms that handle exceptions. After this point, the default controller is compromised, and the Configuration Manager triggers the recovery process by transferring execution to the backup controller with a SIGCONTINUE POSIX signal. Afterwards, a new default CPS controller is spawned inside of a DBT enclosure to serve as the new backup controller. By reconfiguring in this manner, a safe state can be ensured during unstable circumstances, while the benefits of the default high performance controller can be maintained during normal operation.

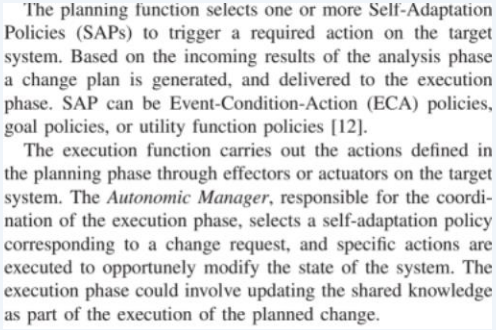
For architecture implementation to be successful two assumptions must be true. The first assumption is that the operating system, as well as the Configuration Manager process are secure. The vulnerable component that we focus on in our threat model is the the CPS controller. The second assumption is that the communication between the Configuration Manager and the DBT processes must be unidirectional. As such, the

Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)



processes. This allows the Configuration Manager to monitor the underlying vulnerable controllers for cyber-attacks, as well as any other unsafe behavior. Further, the Configuration Manager controls the execution of the controllers, allowing for the transfer of control in the case of an attack. By default,

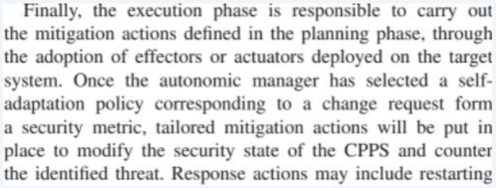
Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)



The planning function selects one or more Self-Adaptation Policies (SAPs) to trigger a required action on the target system. Based on the incoming results of the analysis phase a change plan is generated, and delivered to the execution phase. SAP can be Event-Condition-Action (ECA) policies, goal policies, or utility function policies [12].

The execution function carries out the actions defined in the planning phase through effectors or actuators on the target system. The *Autonomic Manager*, responsible for the coordination of the execution phase, selects a self-adaptation policy corresponding to a change request, and specific actions are executed to opportunely modify the state of the system. The execution phase could involve updating the shared knowledge as part of the execution of the planned change.

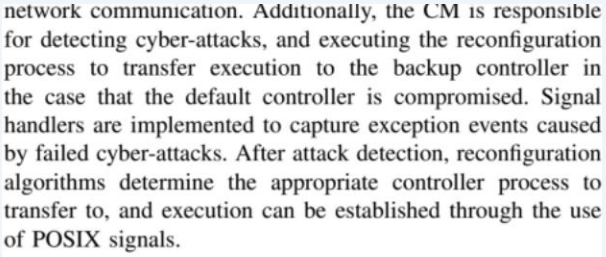
Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)



Finally, the execution phase is responsible to carry out the mitigation actions defined in the planning phase, through the adoption of effectors or actuators deployed on the target system. Once the autonomic manager has selected a self-adaptation policy corresponding to a change request form a security metric, tailored mitigation actions will be put in place to modify the security state of the CPPS and counter the identified threat. Response actions may include restarting

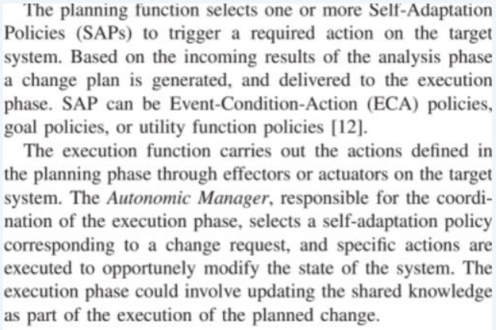
Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Realization\Plan



network communication. Additionally, the CM is responsible for detecting cyber-attacks, and executing the reconfiguration process to transfer execution to the backup controller in the case that the default controller is compromised. Signal handlers are implemented to capture exception events caused by failed cyber-attacks. After attack detection, reconfiguration algorithms determine the appropriate controller process to transfer to, and execution can be established through the use of POSIX signals.

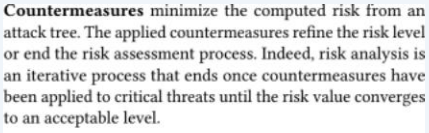
Integrated moving target defense and control reconfiguration fo: 4 - 4 (0)



The planning function selects one or more Self-Adaptation Policies (SAPs) to trigger a required action on the target system. Based on the incoming results of the analysis phase a change plan is generated, and delivered to the execution phase. SAP can be Event-Condition-Action (ECA) policies, goal policies, or utility function policies [12].

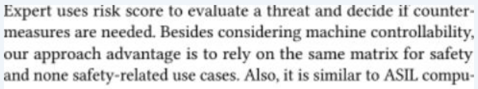
The execution function carries out the actions defined in the planning phase through effectors or actuators on the target system. The *Autonomic Manager*, responsible for the coordination of the execution phase, selects a self-adaptation policy corresponding to a change request, and specific actions are executed to opportunely modify the state of the system. The execution phase could involve updating the shared knowledge as part of the execution of the planned change.

Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)



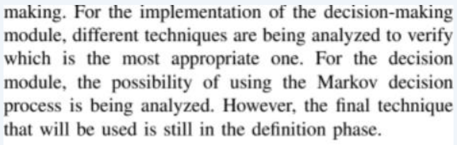
Countermeasures minimize the computed risk from an attack tree. The applied countermeasures refine the risk level or end the risk assessment process. Indeed, risk analysis is an iterative process that ends once countermeasures have been applied to critical threats until the risk value converges to an acceptable level.

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



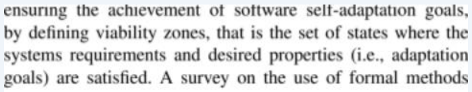
Expert uses risk score to evaluate a threat and decide if countermeasures are needed. Besides considering machine controllability, our approach advantage is to rely on the same matrix for safety and none safety-related use cases. Also, it is similar to ASIL compu-

SARA Security Automotive Risk Analysis Method: 7 - 7 (0)



making. For the implementation of the decision-making module, different techniques are being analyzed to verify which is the most appropriate one. For the decision module, the possibility of using the Markov decision process is being analyzed. However, the final technique that will be used is still in the definition phase.

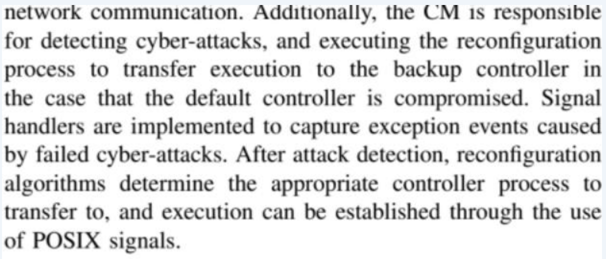
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)



ensuring the achievement of software self-adaptation goals, by defining viability zones, that is the set of states where the systems requirements and desired properties (i.e., adaptation goals) are satisfied. A survey on the use of formal methods

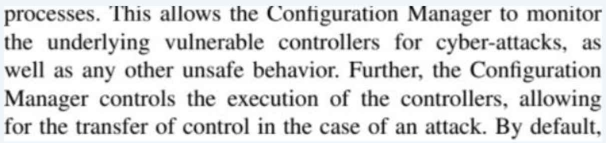
Towards a Framework for Safe and Secure Adaptive Collaborative: 2 - 2 (0)

System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Realization\Monitor



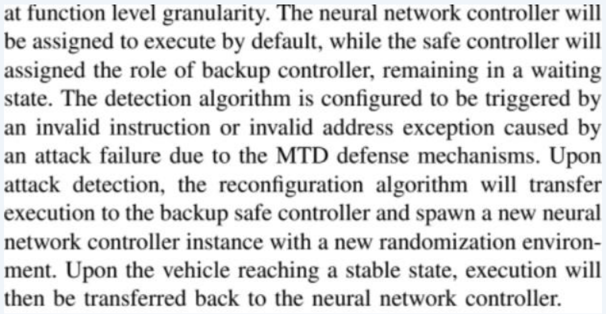
network communication. Additionally, the CM is responsible for detecting cyber-attacks, and executing the reconfiguration process to transfer execution to the backup controller in the case that the default controller is compromised. Signal handlers are implemented to capture exception events caused by failed cyber-attacks. After attack detection, reconfiguration algorithms determine the appropriate controller process to transfer to, and execution can be established through the use of POSIX signals.

Integrated moving target defense and control reconfiguration fo: 4 - 4 (0)



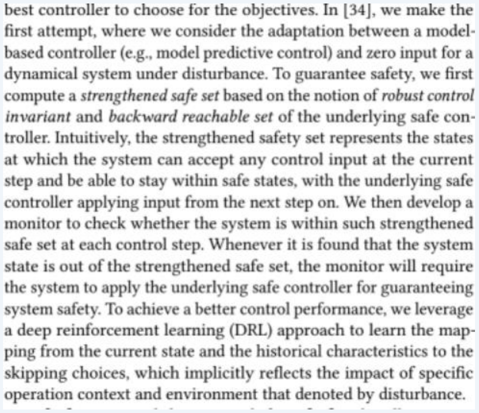
processes. This allows the Configuration Manager to monitor the underlying vulnerable controllers for cyber-attacks, as well as any other unsafe behavior. Further, the Configuration Manager controls the execution of the controllers, allowing for the transfer of control in the case of an attack. By default,

Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)



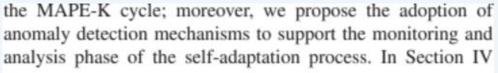
at function level granularity. The neural network controller will be assigned to execute by default, while the safe controller will assigned the role of backup controller, remaining in a waiting state. The detection algorithm is configured to be triggered by an invalid instruction or invalid address exception caused by an attack failure due to the MTD defense mechanisms. Upon attack detection, the reconfiguration algorithm will transfer execution to the backup safe controller and spawn a new neural network controller instance with a new randomization environment. Upon the vehicle reaching a stable state, execution will then be transferred back to the neural network controller.

Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)



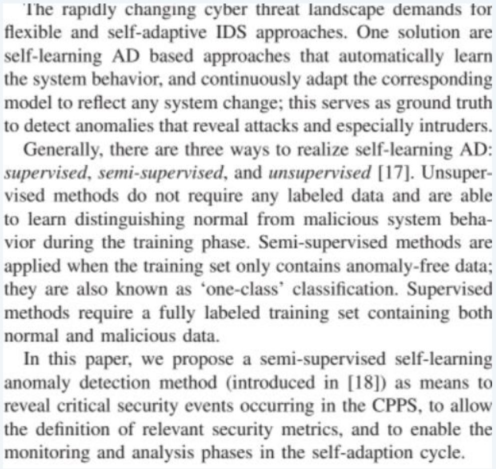
best controller to choose for the objectives. In [34], we make the first attempt, where we consider the adaptation between a model-based controller (e.g., model predictive control) and zero input for a dynamical system under disturbance. To guarantee safety, we first compute a *strengthened safe set* based on the notion of *robust control invariant* and *backward reachable set* of the underlying safe controller. Intuitively, the strengthened safety set represents the states at which the system can accept any control input at the current step and be able to stay within safe states, with the underlying safe controller applying input from the next step on. We then develop a monitor to check whether the system is within such strengthened safe set at each control step. Whenever it is found that the system state is out of the strengthened safe set, the monitor will require the system to apply the underlying safe controller for guaranteeing system safety. To achieve a better control performance, we leverage a deep reinforcement learning (DRL) approach to learn the mapping from the current state and the historical characteristics to the skipping choices, which implicitly reflects the impact of specific operation context and environment that denoted by disturbance.

Know the unknowns addressing disturbances and uncertainties in: 8 - 8 (0)



the MAPE-K cycle; moreover, we propose the adoption of anomaly detection mechanisms to support the monitoring and analysis phase of the self-adaptation process. In Section IV

Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)

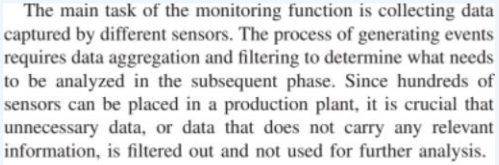
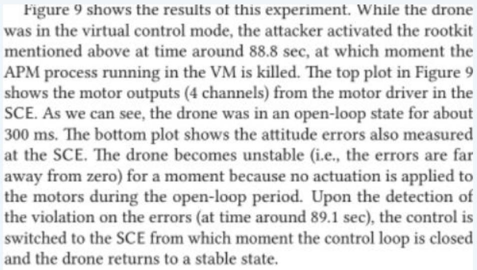
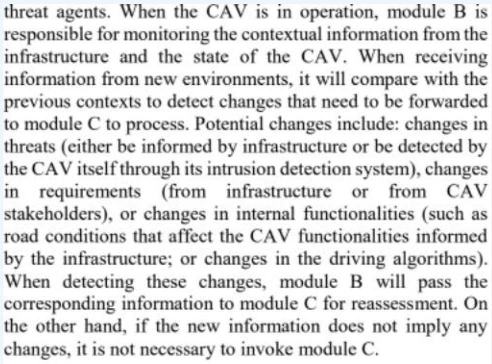


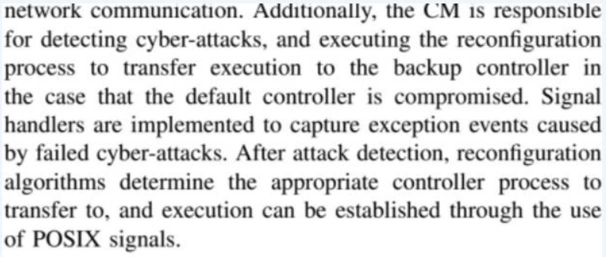
The rapidly changing cyber threat landscape demands for flexible and self-adaptive IDS approaches. One solution are self-learning AD based approaches that automatically learn the system behavior, and continuously adapt the corresponding model to reflect any system change; this serves as ground truth to detect anomalies that reveal attacks and especially intruders.

Generally, there are three ways to realize self-learning AD: *supervised*, *semi-supervised*, and *unsupervised* [17]. Unsupervised methods do not require any labeled data and are able to learn distinguishing normal from malicious system behavior during the training phase. Semi-supervised methods are applied when the training set only contains anomaly-free data; they are also known as 'one-class' classification. Supervised methods require a fully labeled training set containing both normal and malicious data.

In this paper, we propose a semi-supervised self-learning anomaly detection method (introduced in [18]) as means to reveal critical security events occurring in the CPPS, to allow the definition of relevant security metrics, and to enable the monitoring and analysis phases in the self-adaption cycle.

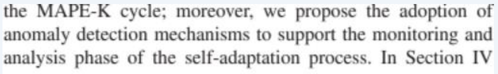
Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)

	 The main task of the monitoring function is collecting data captured by different sensors. The process of generating events requires data aggregation and filtering to determine what needs to be analyzed in the subsequent phase. Since hundreds of sensors can be placed in a production plant, it is crucial that unnecessary data, or data that does not carry any relevant information, is filtered out and not used for further analysis. Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)  Figure 9 shows the results of this experiment. While the drone was in the virtual control mode, the attacker activated the rootkit mentioned above at time around 88.8 sec, at which moment the APM process running in the VM is killed. The top plot in Figure 9 shows the motor outputs (4 channels) from the motor driver in the SCE. As we can see, the drone was in an open-loop state for about 300 ms. The bottom plot shows the attitude errors also measured at the SCE. The drone becomes unstable (i.e., the errors are far away from zero) for a moment because no actuation is applied to the motors during the open-loop period. Upon the detection of the violation on the errors (at time around 89.1 sec), the control is switched to the SCE from which moment the control loop is closed and the drone returns to a stable state. VirtualDrone virtual sensing actuation and communication for at: 7 - 7 (0)
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Realization\Analyze	 threat agents. When the CAV is in operation, module B is responsible for monitoring the contextual information from the infrastructure and the state of the CAV. When receiving information from new environments, it will compare with the previous contexts to detect changes that need to be forwarded to module C to process. Potential changes include: changes in threats (either be informed by infrastructure or be detected by the CAV itself through its intrusion detection system), changes in requirements (from infrastructure or from CAV stakeholders), or changes in internal functionalities (such as road conditions that affect the CAV functionalities informed by the infrastructure; or changes in the driving algorithms). When detecting these changes, module B will pass the corresponding information to module C for reassessment. On the other hand, if the new information does not imply any changes, it is not necessary to invoke module C. A simplified approach for dynamic security risk management in c: 5 - 5 (0)



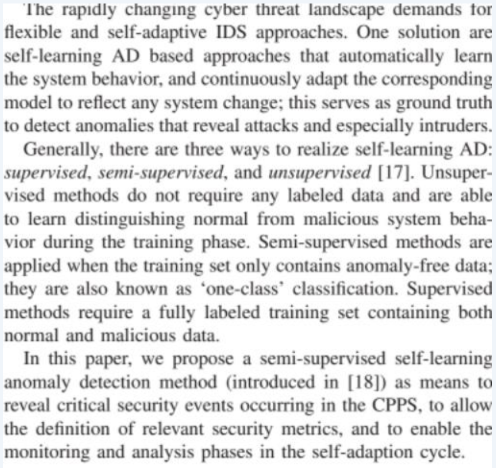
network communication. Additionally, the CM is responsible for detecting cyber-attacks, and executing the reconfiguration process to transfer execution to the backup controller in the case that the default controller is compromised. Signal handlers are implemented to capture exception events caused by failed cyber-attacks. After attack detection, reconfiguration algorithms determine the appropriate controller process to transfer to, and execution can be established through the use of POSIX signals.

Integrated moving target defense and control reconfiguration fo: 4 - 4 (0)



the MAPE-K cycle; moreover, we propose the adoption of anomaly detection mechanisms to support the monitoring and analysis phase of the self-adaptation process. In Section IV

Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)

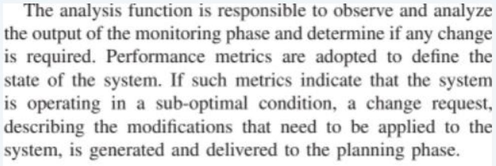


The rapidly changing cyber threat landscape demands for flexible and self-adaptive IDS approaches. One solution are self-learning AD based approaches that automatically learn the system behavior, and continuously adapt the corresponding model to reflect any system change; this serves as ground truth to detect anomalies that reveal attacks and especially intruders.

Generally, there are three ways to realize self-learning AD: *supervised*, *semi-supervised*, and *unsupervised* [17]. Unsupervised methods do not require any labeled data and are able to learn distinguishing normal from malicious system behavior during the training phase. Semi-supervised methods are applied when the training set only contains anomaly-free data; they are also known as 'one-class' classification. Supervised methods require a fully labeled training set containing both normal and malicious data.

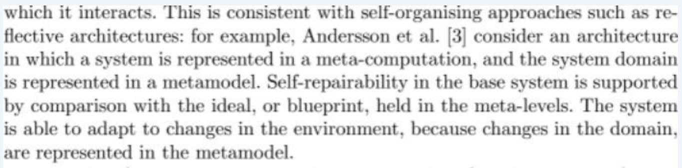
In this paper, we propose a semi-supervised self-learning anomaly detection method (introduced in [18]) as means to reveal critical security events occurring in the CPPS, to allow the definition of relevant security metrics, and to enable the monitoring and analysis phases in the self-adaption cycle.

Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)



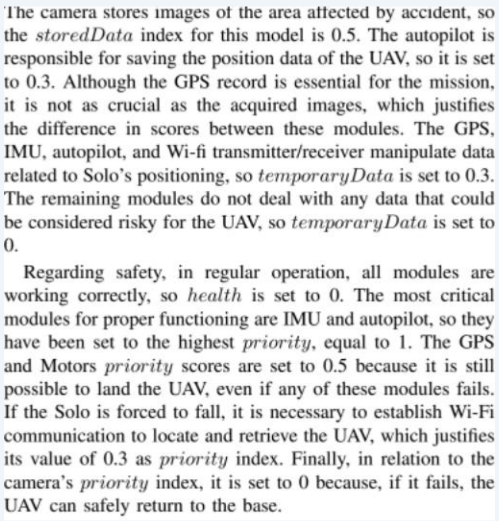
The analysis function is responsible to observe and analyze the output of the monitoring phase and determine if any change is required. Performance metrics are adopted to define the state of the system. If such metrics indicate that the system is operating in a sub-optimal condition, a change request, describing the modifications that need to be applied to the system, is generated and delivered to the planning phase.

Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)



which it interacts. This is consistent with self-organising approaches such as reflective architectures: for example, Andersson et al. [3] consider an architecture in which a system is represented in a meta-computation, and the system domain is represented in a metamodel. Self-repairability in the base system is supported by comparison with the ideal, or blueprint, held in the meta-levels. The system is able to adapt to changes in the environment, because changes in the domain, are represented in the metamodel.

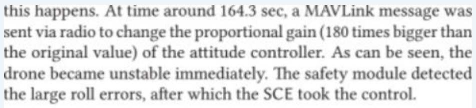
Self-organisation for Survival in Complex Computer Architecture: 3 - 3 (0)



The camera stores images of the area affected by accident, so the *storedData* index for this model is 0.5. The autopilot is responsible for saving the position data of the UAV, so it is set to 0.3. Although the GPS record is essential for the mission, it is not as crucial as the acquired images, which justifies the difference in scores between these modules. The GPS, IMU, autopilot, and Wi-fi transmitter/receiver manipulate data related to Solo's positioning, so *temporaryData* is set to 0.3. The remaining modules do not deal with any data that could be considered risky for the UAV, so *temporaryData* is set to 0.

Regarding safety, in regular operation, all modules are working correctly, so *health* is set to 0. The most critical modules for proper functioning are IMU and autopilot, so they have been set to the highest *priority*, equal to 1. The GPS and Motors *priority* scores are set to 0.5 because it is still possible to land the UAV, even if any of these modules fails. If the Solo is forced to fall, it is necessary to establish Wi-Fi communication to locate and retrieve the UAV, which justifies its value of 0.3 as *priority* index. Finally, in relation to the camera's *priority* index, it is set to 0 because, if it fails, the UAV can safely return to the base.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)

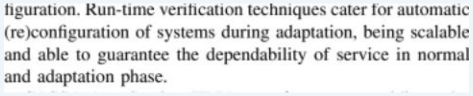


this happens. At time around 164.3 sec, a MAVLink message was sent via radio to change the proportional gain (180 times bigger than the original value) of the attitude controller. As can be seen, the drone became unstable immediately. The safety module detected the large roll errors, after which the SCE took the control.

VirtualDrone virtual sensing actuation and communication for at: 7 - 7 (0)

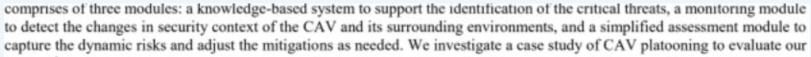
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation

Realization\Adaptation Verification

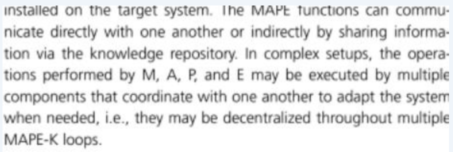

figuration. Run-time verification techniques cater for automatic (re)configuration of systems during adaptation, being scalable and able to guarantee the dependability of service in normal and adaptation phase.

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)

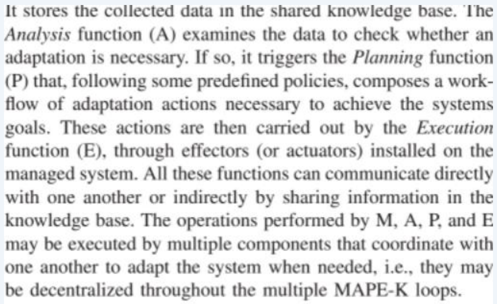
System and Its Properties\Self-Adaptation\Way of Implementation\Adaptation Realization\Adaptation Components


comprises of three modules: a knowledge-based system to support the identification of the critical threats, a monitoring module to detect the changes in security context of the CAV and its surrounding environments, and a simplified assessment module to capture the dynamic risks and adjust the mitigations as needed. We investigate a case study of CAV platooning to evaluate our

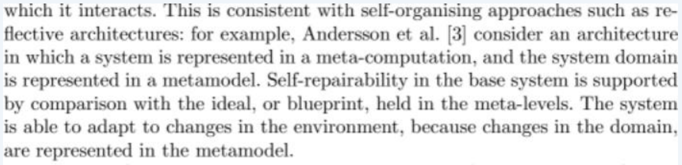
A simplified approach for dynamic security risk management in c: 2 - 2 (0)


installed on the target system. The MAPE functions can communicate directly with one another or indirectly by sharing information via the knowledge repository. In complex setups, the operations performed by M, A, P, and E may be executed by multiple components that coordinate with one another to adapt the system when needed, i.e., they may be decentralized throughout multiple MAPE-K loops.

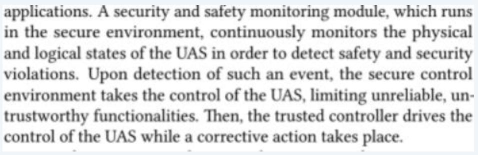
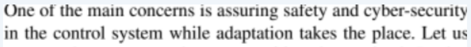
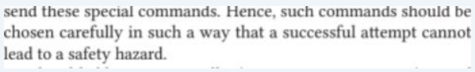
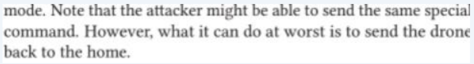
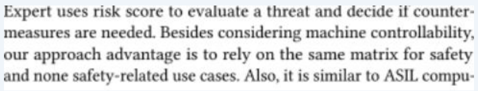
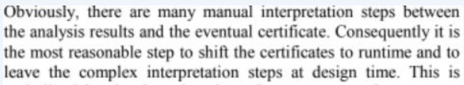
Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)

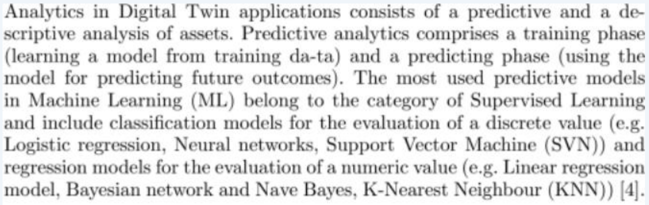

It stores the collected data in the shared knowledge base. The *Analysis* function (A) examines the data to check whether an adaptation is necessary. If so, it triggers the *Planning* function (P) that, following some predefined policies, composes a workflow of adaptation actions necessary to achieve the systems goals. These actions are then carried out by the *Execution* function (E), through effectors (or actuators) installed on the managed system. All these functions can communicate directly with one another or indirectly by sharing information in the knowledge base. The operations performed by M, A, P, and E may be executed by multiple components that coordinate with one another to adapt the system when needed, i.e., they may be decentralized throughout the multiple MAPE-K loops.

Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)


which it interacts. This is consistent with self-organising approaches such as reflective architectures: for example, Andersson et al. [3] consider an architecture in which a system is represented in a meta-computation, and the system domain is represented in a metamodel. Self-repairability in the base system is supported by comparison with the ideal, or blueprint, held in the meta-levels. The system is able to adapt to changes in the environment, because changes in the domain, are represented in the metamodel.

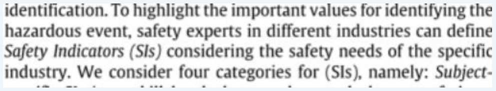
Self-organisation for Survival in Complex Computer Architecture: 3 - 3 (0)

	 applications. A security and safety monitoring module, which runs in the secure environment, continuously monitors the physical and logical states of the UAS in order to detect safety and security violations. Upon detection of such an event, the secure control environment takes the control of the UAS, limiting unreliable, untrustworthy functionalities. Then, the trusted controller drives the control of the UAS while a corrective action takes place. VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)
System and Its Properties\Self-Adaptation\Attack on Self-Adaptation	 One of the main concerns is assuring safety and cyber-security in the control system while adaptation takes the place. Let us Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)  send these special commands. Hence, such commands should be chosen carefully in such a way that a successful attempt cannot lead to a safety hazard. VirtualDrone virtual sensing actuation and communication for at: 4 - 4 (0)  mode. Note that the attacker might be able to send the same special command. However, what it can do at worst is to send the drone back to the home. VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)
System and Its Properties\Degree of Automation\Manual	 Expert uses risk score to evaluate a threat and decide if countermeasures are needed. Besides considering machine controllability, our approach advantage is to rely on the same matrix for safety and none safety-related use cases. Also, it is similar to ASIL compu- SARA Security Automotive Risk Analysis Method: 7 - 7 (0)
System and Its Properties\Degree of Automation\Semi-automated	 Obviously, there are many manual interpretation steps between the analysis results and the eventual certificate. Consequently it is the most reasonable step to shift the certificates to runtime and to leave the complex interpretation steps at design time. This is Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)



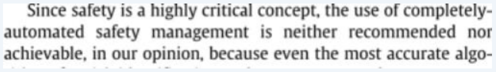
Analytics in Digital Twin applications consists of a predictive and a descriptive analysis of assets. Predictive analytics comprises a training phase (learning a model from training da-ta) and a predicting phase (using the model for predicting future outcomes). The most used predictive models in Machine Learning (ML) belong to the category of Supervised Learning and include classification models for the evaluation of a discrete value (e.g. Logistic regression, Neural networks, Support Vector Machine (SVN)) and regression models for the evaluation of a numeric value (e.g. Linear regression model, Bayesian network and Nave Bayes, K-Nearest Neighbour (KNN)) [4].

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)



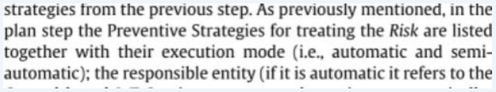
identification. To highlight the important values for identifying the hazardous event, safety experts in different industries can define *Safety Indicators (SIs)* considering the safety needs of the specific industry. We consider four categories for (SIs), namely: *Subject-*

Ontology development for run-time safety management methodology: 5 - 5 (0)



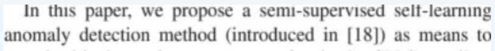
Since safety is a highly critical concept, the use of completely-automated safety management is neither recommended nor achievable, in our opinion, because even the most accurate algo-

Ontology development for run-time safety management methodology: 6 - 6 (0)



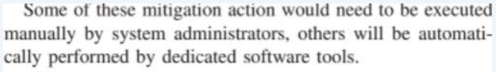
strategies from the previous step. As previously mentioned, in the plan step the Preventive Strategies for treating the *Risk* are listed together with their execution mode (i.e., automatic and semi-automatic); the responsible entity (if it is automatic it refers to the

Ontology development for run-time safety management methodology: 6 - 6 (0)



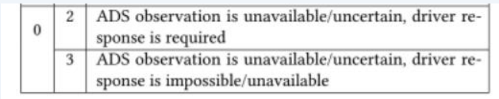
In this paper, we propose a semi-supervised self-learning anomaly detection method (introduced in [18]) as means to

Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)



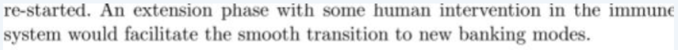
Some of these mitigation action would need to be executed manually by system administrators, others will be automatically performed by dedicated software tools.

Protecting cyber physical production systems using anomaly dete: 8 - 8 (0)



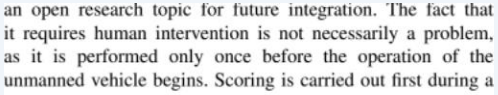
0	2	ADS observation is unavailable/uncertain, driver response is required
	3	ADS observation is unavailable/uncertain, driver response is impossible/unavailable

SARA Security Automotive Risk Analysis Method: 7: 308|585 - 7: 563|631 (0)



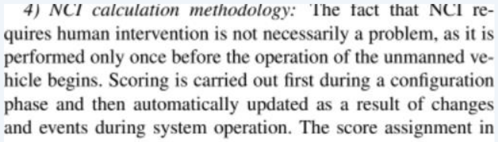
re-started. An extension phase with some human intervention in the immune system would facilitate the smooth transition to new banking modes.

Self-organisation for Survival in Complex Computer Architecture: 14 - 14 (0)



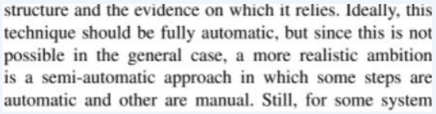
an open research topic for future integration. The fact that it requires human intervention is not necessarily a problem, as it is performed only once before the operation of the unmanned vehicle begins. Scoring is carried out first during a

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)



4) NCI calculation methodology: The fact that NCI requires human intervention is not necessarily a problem, as it is performed only once before the operation of the unmanned vehicle begins. Scoring is carried out first during a configuration phase and then automatically updated as a result of changes and events during system operation. The score assignment in

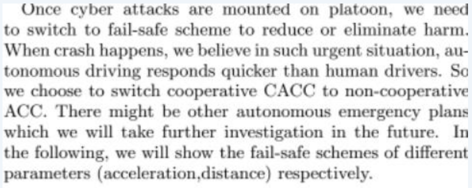
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)



structure and the evidence on which it relies. Ideally, this technique should be fully automatic, but since this is not possible in the general case, a more realistic ambition is a semi-automatic approach in which some steps are automatic and other are manual. Still, for some system

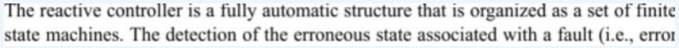
Towards a Framework for Safe and Secure Adaptive Collaborative: 5 - 5 (0)

System and Its Properties\Degree of Automation\Fully-automated



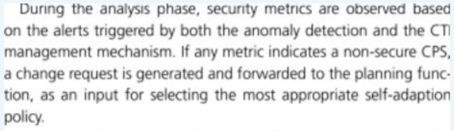
Once cyber attacks are mounted on platoon, we need to switch to fail-safe scheme to reduce or eliminate harm. When crash happens, we believe in such urgent situation, autonomous driving responds quicker than human drivers. So we choose to switch cooperative CACC to non-cooperative ACC. There might be other autonomous emergency plans which we will take further investigation in the future. In the following, we will show the fail-safe schemes of different parameters (acceleration,distance) respectively.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 8 - 8 (0)



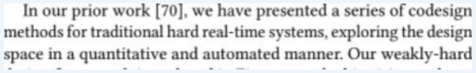
The reactive controller is a fully automatic structure that is organized as a set of finite state machines. The detection of the erroneous state associated with a fault (i.e., error

Achieving critical system survivability through software archit: 14 - 14 (0)



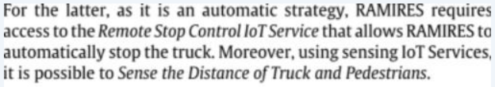
During the analysis phase, security metrics are observed based on the alerts triggered by both the anomaly detection and the CTI management mechanism. If any metric indicates a non-secure CPS, a change request is generated and forwarded to the planning function, as an input for selecting the most appropriate self-adaption policy.

Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)



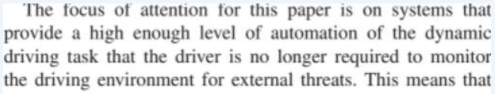
In our prior work [70], we have presented a series of codesign methods for traditional hard real-time systems, exploring the design space in a quantitative and automated manner. Our weakly-hard

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



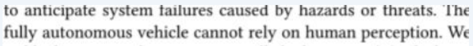
For the latter, as it is an automatic strategy, RAMIRES requires access to the *Remote Stop Control IoT Service* that allows RAMIRES to automatically stop the truck. Moreover, using sensing IoT Services, it is possible to *Sense the Distance of Truck and Pedestrians.*

Ontology development for run-time safety management methodology: 7 - 7 (0)



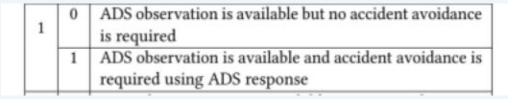
The focus of attention for this paper is on systems that provide a high enough level of automation of the dynamic driving task that the driver is no longer required to monitor the driving environment for external threats. This means that

Potential cyberattacks on automated vehicles: 3 - 3 (0)



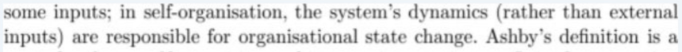
to anticipate system failures caused by hazards or threats. The fully autonomous vehicle cannot rely on human perception. We

SARA Security Automotive Risk Analysis Method: 7 - 7 (0)



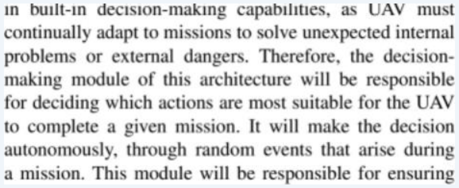
1	0	ADS observation is available but no accident avoidance is required
	1	ADS observation is available and accident avoidance is required using ADS response

SARA Security Automotive Risk Analysis Method: 7: 306|631 - 7: 566|674 (0)

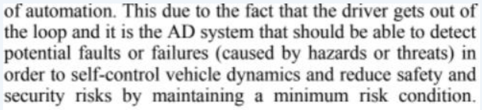


some inputs; in self-organisation, the system's dynamics (rather than external inputs) are responsible for organisational state change. Ashby's definition is a

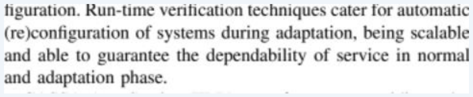
Self-organisation for Survival in Complex Computer Architecture: 3 - 3 (0)


in built-in decision-making capabilities, as UAV must continually adapt to missions to solve unexpected internal problems or external dangers. Therefore, the decision-making module of this architecture will be responsible for deciding which actions are most suitable for the UAV to complete a given mission. It will make the decision autonomously, through random events that arise during a mission. This module will be responsible for ensuring

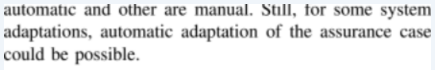
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)


of automation. This due to the fact that the driver gets out of the loop and it is the AD system that should be able to detect potential faults or failures (caused by hazards or threats) in order to self-control vehicle dynamics and reduce safety and security risks by maintaining a minimum risk condition.

TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)

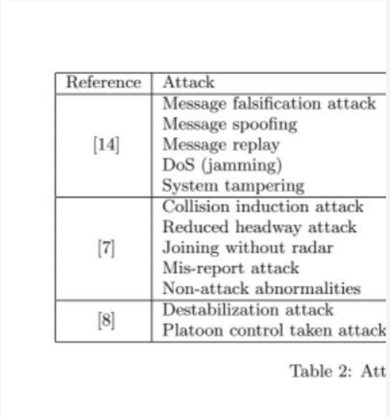

figuration. Run-time verification techniques cater for automatic (re)configuration of systems during adaptation, being scalable and able to guarantee the dependability of service in normal and adaptation phase.

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)


automatic and other are manual. Still, for some system adaptations, automatic adaptation of the assurance case could be possible.

Towards a Framework for Safe and Secure Adaptive Collaborative: 5 - 5 (0)

Integration\Security Attacks\CIA\Confidentiality



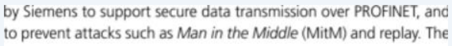
Reference	Attack
[14]	Message falsification attack Message spoofing Message replay DoS (jamming) System tampering
[7]	Collision induction attack Reduced headway attack Joining without radar Mis-report attack Non-attack abnormalities
[8]	Destabilization attack Platoon control taken attack

Table 2: Att

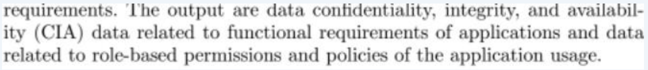
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 97|565 - 5: 295|775 (0)

spoofing

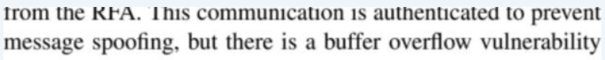
A simplified approach for dynamic security risk management in c: 6 - 6 (0)


by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The

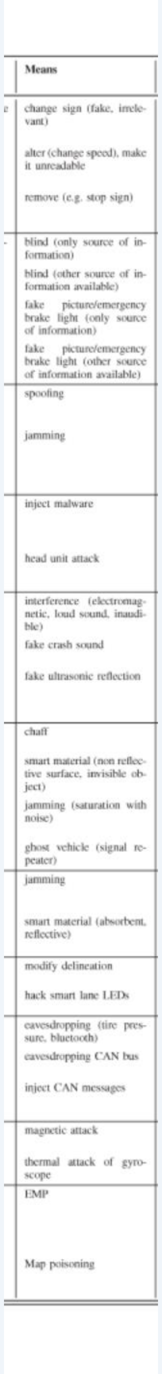
Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)


requirements. The output are data confidentiality, integrity, and availability (CIA) data related to functional requirements of applications and data related to role-based permissions and policies of the application usage.

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)

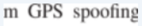

from the RFA. This communication is authenticated to prevent message spoofing, but there is a buffer overflow vulnerability

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)

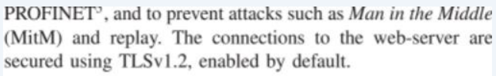


	Means
e	change sign (fake, irrelevant) alter (change speed), make it unreadable remove (e.g. stop sign)
-	blind (only source of information) blind (other source of information available) fake picture/emergency brake light (only source of information) fake picture/emergency brake light (other source of information available)
	spoofing jamming
	inject malware head unit attack
	interference (electromagnetic, loud sound, inaudible) fake crash sound fake ultrasonic reflection
	chaff smart material (non reflective surface, invisible object) jamming (saturation with noise) ghost vehicle (signal repeater)
	jamming smart material (absorbent, reflective)
	modify delineation hack smart lane LEDs
	eavesdropping (tire pressure, bluetooth) eavesdropping CAN bus inject CAN messages
	magnetic attack thermal attack of gyroscope
	EMP Map poisoning

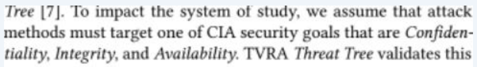
Potential cyberattacks on automated vehicles: 5: 77|25 - 5: 156|727 (0)


m GPS spoofing

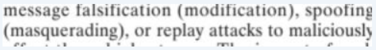
Potential cyberattacks on automated vehicles: 6 - 6 (0)


PROFINET[3], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

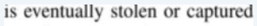
Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)


Tree [7]. To impact the system of study, we assume that attack methods must target one of CIA security goals that are *Confidentiality, Integrity*, and *Availability.* TVRA *Threat Tree* validates this

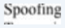
SARA Security Automotive Risk Analysis Method: 5 - 5 (0)


message falsification (modification), spoofing (masquerading), or replay attacks to maliciously

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

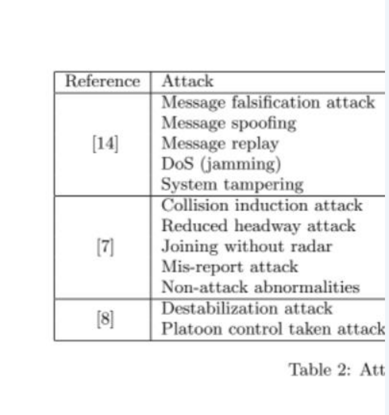

is eventually stolen or captured

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)


Spoofing

TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)

Integration\Security Attacks\CIA\Integrity



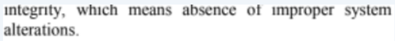
Reference	Attack
[14]	Message falsification attack Message spoofing Message replay DoS (jamming) System tampering
[7]	Collision induction attack Reduced headway attack Joining without radar Mis-report attack Non-attack abnormalities
[8]	Destabilization attack Platoon control taken attack

Table 2: Att

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 97|565 - 5: 295|775 (0)

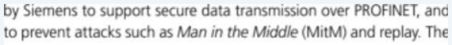
tampering

A simplified approach for dynamic security risk management in c: 6 - 6 (0)



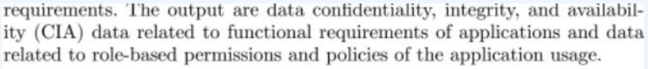
integrity, which means absence of improper system alterations.

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



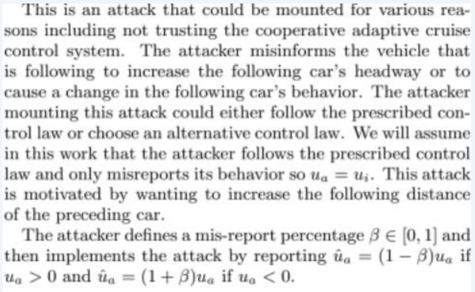
by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



requirements. The output are data confidentiality, integrity, and availability (CIA) data related to functional requirements of applications and data related to role-based permissions and policies of the application usage.

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)



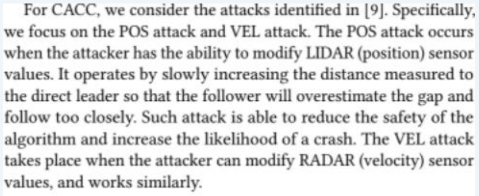
This is an attack that could be mounted for various reasons including not trusting the cooperative adaptive cruise control system. The attacker misinforms the vehicle that is following to increase the following car's headway or to cause a change in the following car's behavior. The attacker mounting this attack could either follow the prescribed control law or choose an alternative control law. We will assume in this work that the attacker follows the prescribed control law and only misreports its behavior so $u_a = u_i$. This attack is motivated by wanting to increase the following distance of the preceding car.

The attacker defines a mis-report percentage $\beta \in [0, 1]$ and then implements the attack by reporting $\hat{u}_a = (1 - \beta)u_a$ if $u_a > 0$ and $\hat{u}_a = (1 + \beta)u_a$ if $u_a < 0$.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)

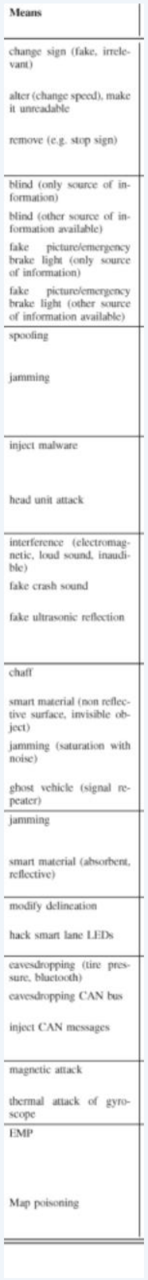
al *fault data injection* attacks

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



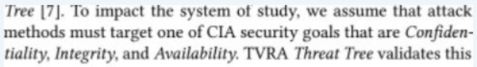
For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)



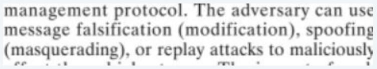
Means
change sign (fake, irrelevant) alter (change speed), make it unreadable remove (e.g. stop sign)
blind (only source of information) blind (other source of information available) fake picture/emergency brake light (only source of information) fake picture/emergency brake light (other source of information available)
spoofing jamming
inject malware head unit attack
interference (electromagnetic, loud sound, inaudible) fake crash sound fake ultrasonic reflection
chaff smart material (non reflective surface, invisible object) jamming (saturation with noise) ghost vehicle (signal repeater)
jamming smart material (absorbent, reflective)
modify delineation hack smart lane LEDs
eavesdropping (tire pressure, bluetooth) eavesdropping CAN bus inject CAN messages
magnetic attack thermal attack of gyroscope
EMP Map poisoning

Potential cyberattacks on automated vehicles: 5: 85|43 - 5: 157|696 (0)



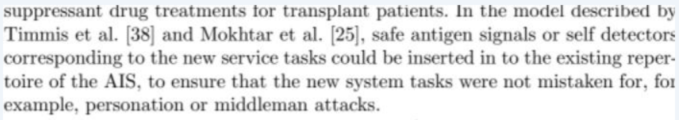
Tree [7]. To impact the system of study, we assume that attack methods must target one of CIA security goals that are *Confidentiality, Integrity,* and *Availability.* TVRA *Threat Tree* validates this

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



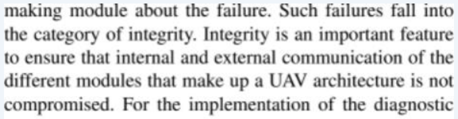
management protocol. The adversary can use message falsification (modification), spoofing (masquerading), or replay attacks to maliciously

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)



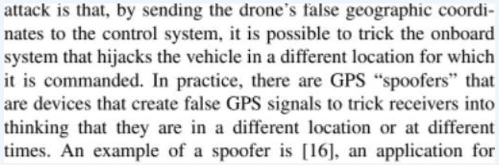
suppressant drug treatments for transplant patients. In the model described by Timmis et al. [38] and Mokhtar et al. [25], safe antigen signals or self detectors corresponding to the new service tasks could be inserted in to the existing repertoire of the AIS, to ensure that the new system tasks were not mistaken for, for example, personation or middleman attacks.

Self-organisation for Survival in Complex Computer Architecture: 13 - 13 (0)



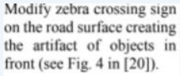
making module about the failure. Such failures fall into the category of integrity. Integrity is an important feature to ensure that internal and external communication of the different modules that make up a UAV architecture is not compromised. For the implementation of the diagnostic

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)



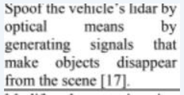
attack is that, by sending the drone's false geographic coordinates to the control system, it is possible to trick the onboard system that hijacks the vehicle in a different location for which it is commanded. In practice, there are GPS "spoofers" that are devices that create false GPS signals to trick receivers into thinking that they are in a different location or at different times. An example of a spoofer is [16], an application for

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 6 - 6 (0)



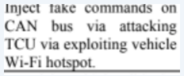
Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).

TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



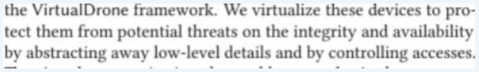
Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].

TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.

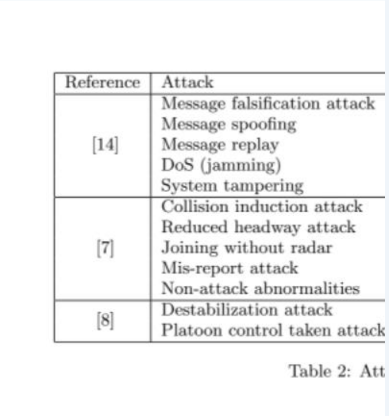
TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



the VirtualDrone framework. We virtualize these devices to protect them from potential threats on the integrity and availability by abstracting away low-level details and by controlling accesses.

VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)

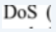
Integration\Security Attacks\CIA\Availability



Reference	Attack
[14]	Message falsification attack Message spoofing Message replay DoS (jamming) System tampering
[7]	Collision induction attack Reduced headway attack Joining without radar Mis-report attack Non-attack abnormalities
[8]	Destabilization attack Platoon control taken attack

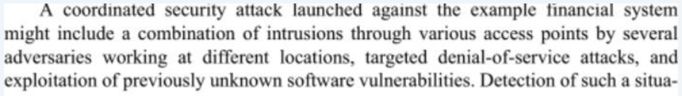
Table 2: Att

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 97|565 - 5: 295|775 (0)



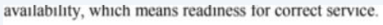
DoS (

A simplified approach for dynamic security risk management in c: 6 - 6 (0)



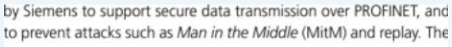
A coordinated security attack launched against the example financial system might include a combination of intrusions through various access points by several adversaries working at different locations, targeted denial-of-service attacks, and exploitation of previously unknown software vulnerabilities. Detection of such a situa-

Achieving critical system survivability through software archit: 14 - 14 (0)



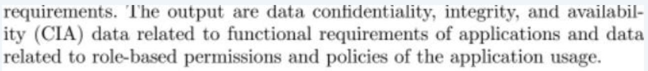
availability, which means readiness for correct service.

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



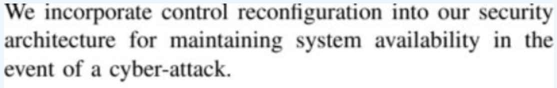
by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



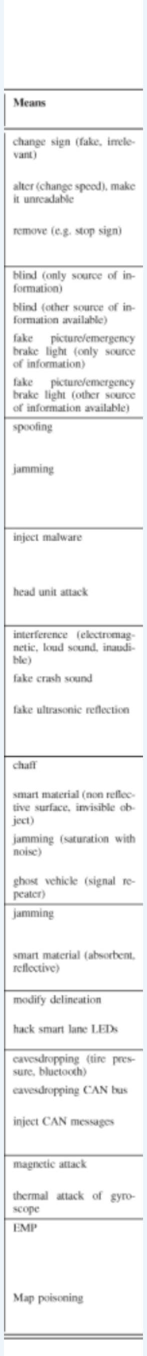
requirements. The output are data confidentiality, integrity, and availability (CIA) data related to functional requirements of applications and data related to role-based permissions and policies of the application usage.

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)



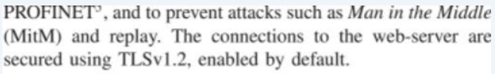
We incorporate control reconfiguration into our security architecture for maintaining system availability in the event of a cyber-attack.

Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)



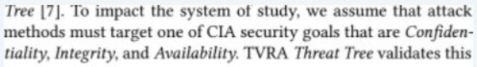
Means
change sign (fake, irrelevant) alter (change speed), make it unreadable remove (e.g. stop sign)
blind (only source of information) blind (other source of information available) fake picture/emergency brake light (only source of information) fake picture/emergency brake light (other source of information available)
spoofing jamming
inject malware head unit attack
interference (electromagnetic, loud sound, inaudible) fake crash sound fake ultrasonic reflection
chaff smart material (non reflective surface, invisible object) jamming (saturation with noise) ghost vehicle (signal repeater)
jamming smart material (absorbent, reflective)
modify delineation hack smart lane LEDs
eavesdropping (tire pressure, bluetooth) eavesdropping CAN bus inject CAN messages
magnetic attack thermal attack of gyroscope
EMP Map poisoning

Potential cyberattacks on automated vehicles: 5: 83|52 - 5: 154|744 (0)



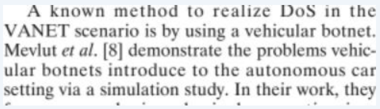
PROFINET[3], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)



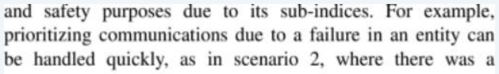
Tree [7]. To impact the system of study, we assume that attack methods must target one of CIA security goals that are *Confidentiality, Integrity*, and *Availability*. TVRA *Threat Tree* validates this

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



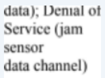
A known method to realize DoS in the VANET scenario is by using a vehicular botnet. Mevlut *et al.* [8] demonstrate the problems vehicular botnets introduce to the autonomous car setting via a simulation study. In their work, they

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



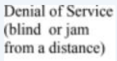
and safety purposes due to its sub-indices. For example, prioritizing communications due to a failure in an entity can be handled quickly, as in scenario 2, where there was a

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)



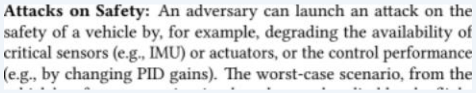
data); Denial of
Service (jam
sensor
data channel)

TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



Denial of Service
(blind or jam
from a distance)

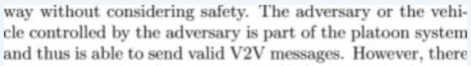
TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the

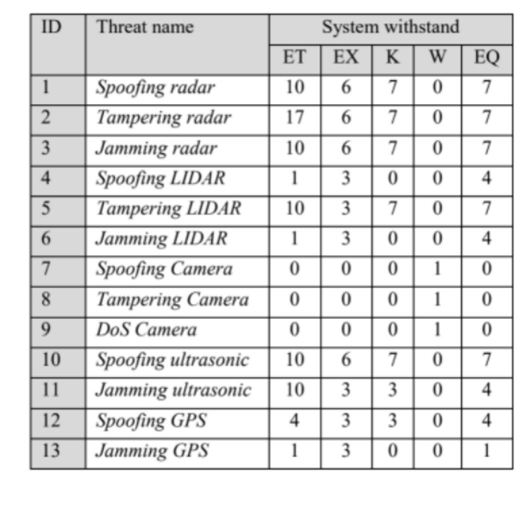
VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

Integration\Security Attacks\Attack Surface\Remote Access



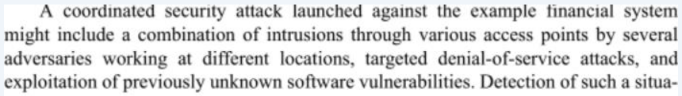
way without considering safety. The adversary or the vehicle controlled by the adversary is part of the platoon system and thus is able to send valid V2V messages. However, there

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)



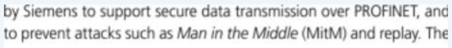
ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

A simplified approach for dynamic security risk management in c: 7: 30|65 - 7: 302|322 (0)



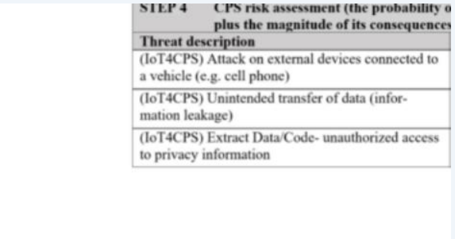
A coordinated security attack launched against the example financial system might include a combination of intrusions through various access points by several adversaries working at different locations, targeted denial-of-service attacks, and exploitation of previously unknown software vulnerabilities. Detection of such a situa-

Achieving critical system survivability through software archit: 14 - 14 (0)



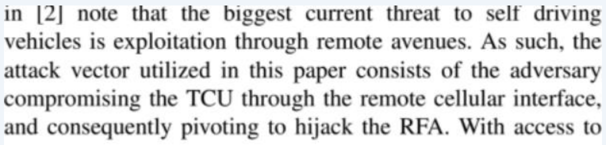
by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



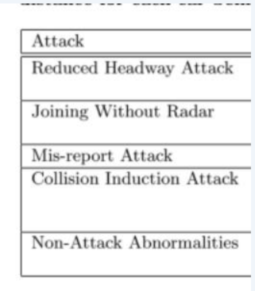
STEP 4 CPS risk assessment (the probability o plus the magnitude of its consequences
Threat description
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)
(IoT4CPS) Unintended transfer of data (information leakage)
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information

Digital Twins for Dependability Improvement of Autonomous Drivi: 10: 86|189 - 10: 317|306 (0)



in [2] note that the biggest current threat to self driving vehicles is exploitation through remote avenues. As such, the attack vector utilized in this paper consists of the adversary compromising the TCU through the remote cellular interface, and consequently pivoting to hijack the RFA. With access to

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)

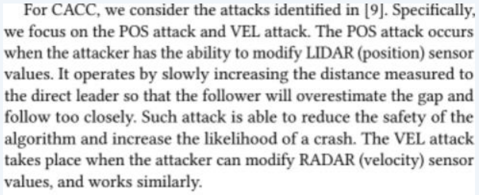


Attack
Reduced Headway Attack
Joining Without Radar
Mis-report Attack
Collision Induction Attack
Non-Attack Abnormalities

Is your commute driving you crazy a study of misbehavior in veh: 5: 44|399 - 5: 173|543 (0)

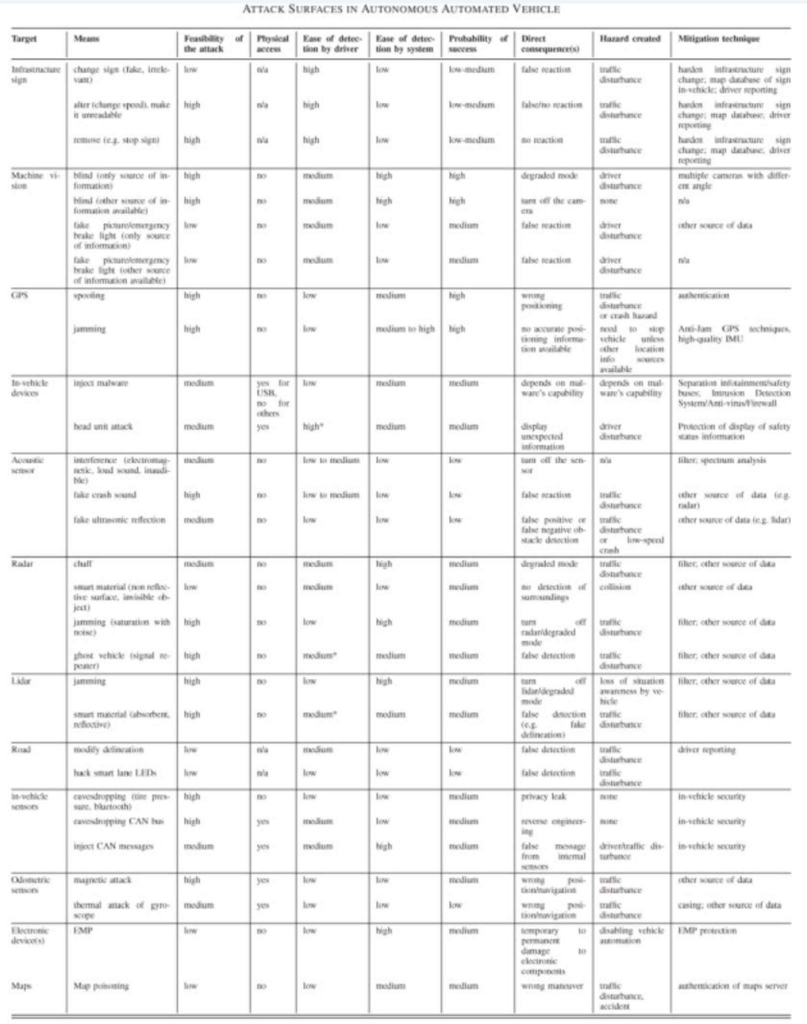
al *fault data injection* attacks

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

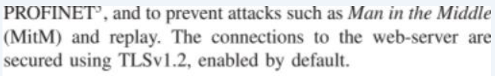
Network and system level security in connected vehicle applicat: 3 - 3 (0)



ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 40|55 - 5: 562|714 (0)

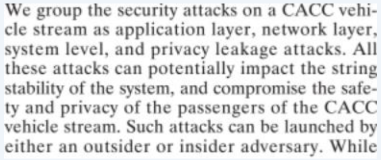


PROFINET[3], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)

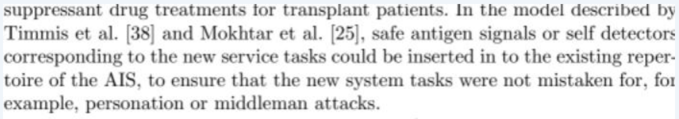
Vehicle Tracking

SARA Security Automotive Risk Analysis Method: 8 - 8 (0)



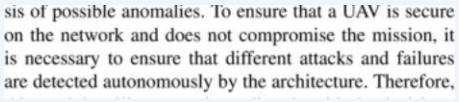
We group the security attacks on a CACC vehicle stream as application layer, network layer, system level, and privacy leakage attacks. All these attacks can potentially impact the string stability of the system, and compromise the safety and privacy of the passengers of the CACC vehicle stream. Such attacks can be launched by either an outsider or insider adversary. While

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)



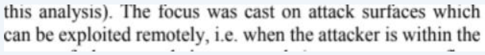
suppressant drug treatments for transplant patients. In the model described by Timmis et al. [38] and Mokhtar et al. [25], safe antigen signals or self detectors corresponding to the new service tasks could be inserted in to the existing repertoire of the AIS, to ensure that the new system tasks were not mistaken for, for example, personation or middleman attacks.

Self-organisation for Survival in Complex Computer Architecture: 13 - 13 (0)



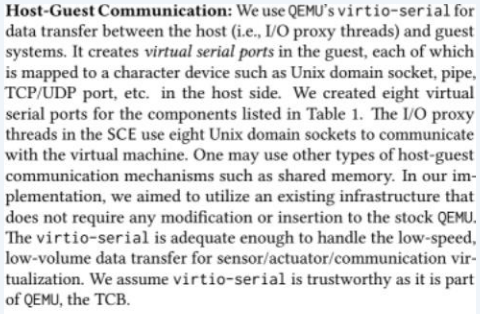
sis of possible anomalies. To ensure that a UAV is secure on the network and does not compromise the mission, it is necessary to ensure that different attacks and failures are detected autonomously by the architecture. Therefore,

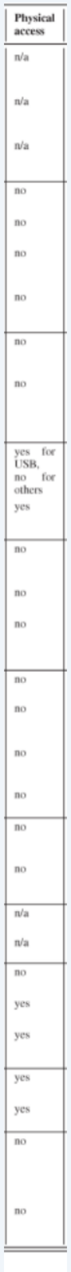
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)



this analysis). The focus was cast on attack surfaces which can be exploited remotely, i.e. when the attacker is within the

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

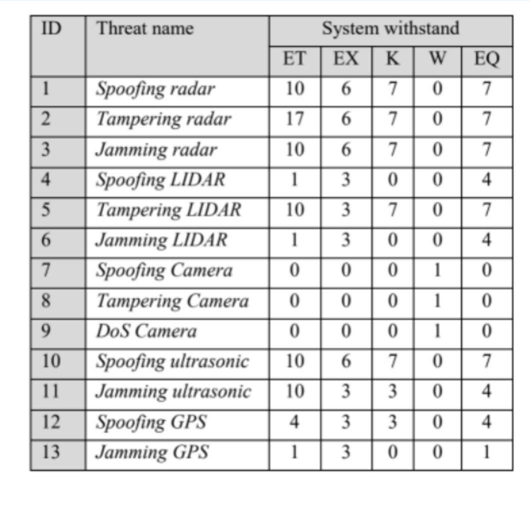
	 **Host-Guest Communication:** We use QEMU's `virtio-serial` for data transfer between the host (i.e., I/O proxy threads) and guest systems. It creates *virtual serial ports* in the guest, each of which is mapped to a character device such as Unix domain socket, pipe, TCP/UDP port, etc. in the host side. We created eight virtual serial ports for the components listed in Table 1. The I/O proxy threads in the SCE use eight Unix domain sockets to communicate with the virtual machine. One may use other types of host-guest communication mechanisms such as shared memory. In our implementation, we aimed to utilize an existing infrastructure that does not require any modification or insertion to the stock QEMU. The `virtio-serial` is adequate enough to handle the low-speed, low-volume data transfer for sensor/actuator/communication virtualization. We assume `virtio-serial` is trustworthy as it is part of QEMU, the TCB. VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)
Integration\Security Attacks\Attack Surface\Physical Access	



Physical access
n/a
n/a
n/a
no
no
no
no
no
no
yes for USB, no for others
yes
no
no
no
no
no
no
no
no
no
n/a
n/a
no
yes
yes
yes
yes
no
no

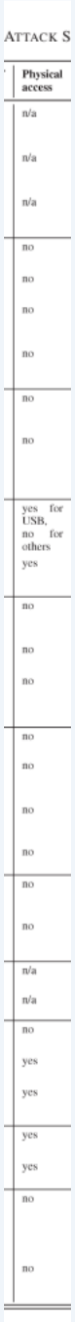
Potential cyberattacks on automated vehicles: 5: 201|50 - 5: 233|699 (0)

Integration\Security Attacks\Attack Surface\Close Proximity



ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

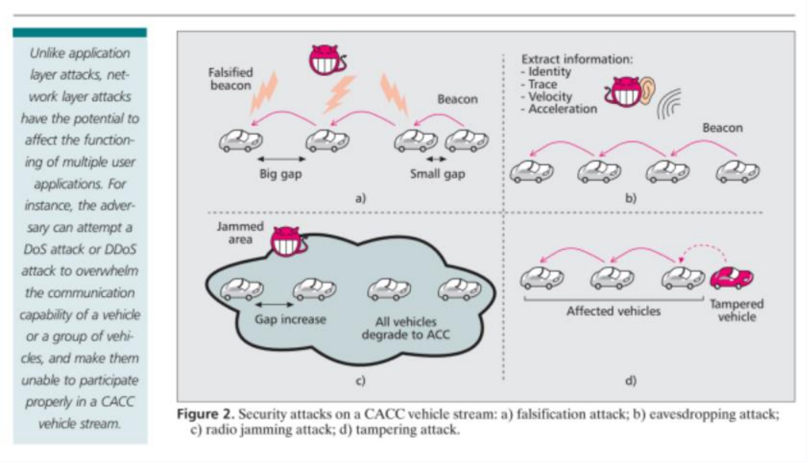
A simplified approach for dynamic security risk management in c: 7: 30|65 - 7: 302|322 (0)



ATTACK S

Physical access
n/a
n/a
n/a
no
no
no
no
no
no
yes for USB, no for others
yes
no
no
no
no
no
no
no
no
no
n/a
n/a
no
yes
yes
yes
yes
no
no

Potential cyberattacks on automated vehicles: 5: 197|54 - 5: 231|729 (0)

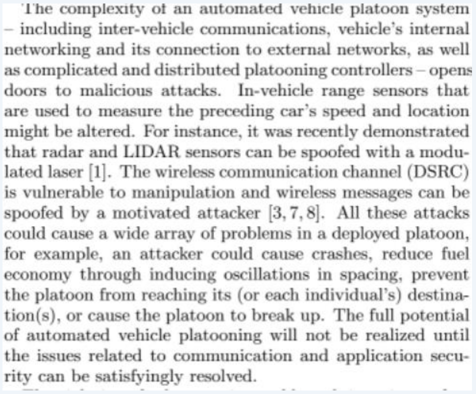


Unlike application layer attacks, network layer attacks have the potential to affect the functioning of multiple user applications. For instance, the adversary can attempt a DoS attack or DDoS attack to overwhelm the communication capability of a vehicle or a group of vehicles, and make them unable to participate properly in a CACC vehicle stream.

Figure 2. Security attacks on a CACC vehicle stream: a) falsification attack; b) eavesdropping attack; c) radio jamming attack; d) tampering attack.

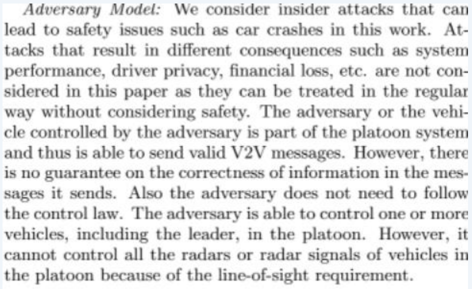
Security vulnerabilities of connected vehicle streams and their: 3: 68|512 - 3: 588|805 (0)

Integration\Security Attacks\Attack Mechanisms



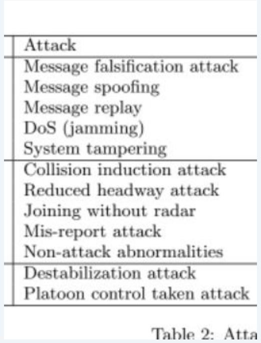
The complexity of an automated vehicle platoon system – including inter-vehicle communications, vehicle's internal networking and its connection to external networks, as well as complicated and distributed platooning controllers – opens doors to malicious attacks. In-vehicle range sensors that are used to measure the preceding car's speed and location might be altered. For instance, it was recently demonstrated that radar and LIDAR sensors can be spoofed with a modulated laser [1]. The wireless communication channel (DSRC) is vulnerable to manipulation and wireless messages can be spoofed by a motivated attacker [3, 7, 8]. All these attacks could cause a wide array of problems in a deployed platoon, for example, an attacker could cause crashes, reduce fuel economy through inducing oscillations in spacing, prevent the platoon from reaching its (or each individual's) destination(s), or cause the platoon to break up. The full potential of automated vehicle platooning will not be realized until the issues related to communication and application security can be satisfyingly resolved.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 1 - 1 (0)



Adversary Model: We consider insider attacks that can lead to safety issues such as car crashes in this work. Attacks that result in different consequences such as system performance, driver privacy, financial loss, etc. are not considered in this paper as they can be treated in the regular way without considering safety. The adversary or the vehicle controlled by the adversary is part of the platoon system and thus is able to send valid V2V messages. However, there is no guarantee on the correctness of information in the messages it sends. Also the adversary does not need to follow the control law. The adversary is able to control one or more vehicles, including the leader, in the platoon. However, it cannot control all the radars or radar signals of vehicles in the platoon because of the line-of-sight requirement.

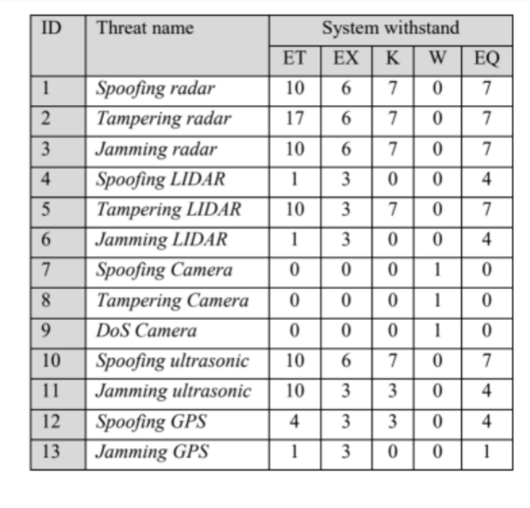
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)



Attack
Message falsification attack
Message spoofing
Message replay
DoS (jamming)
System tampering
Collision induction attack
Reduced headway attack
Joining without radar
Mis-report attack
Non-attack abnormalities
Destabilization attack
Platoon control taken attack

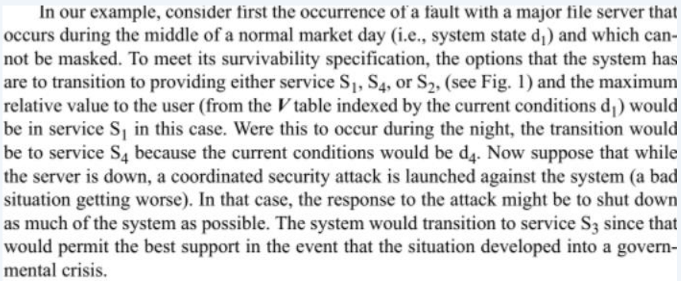
Table 2: Atta

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 170|586 - 5: 299|756 (0)



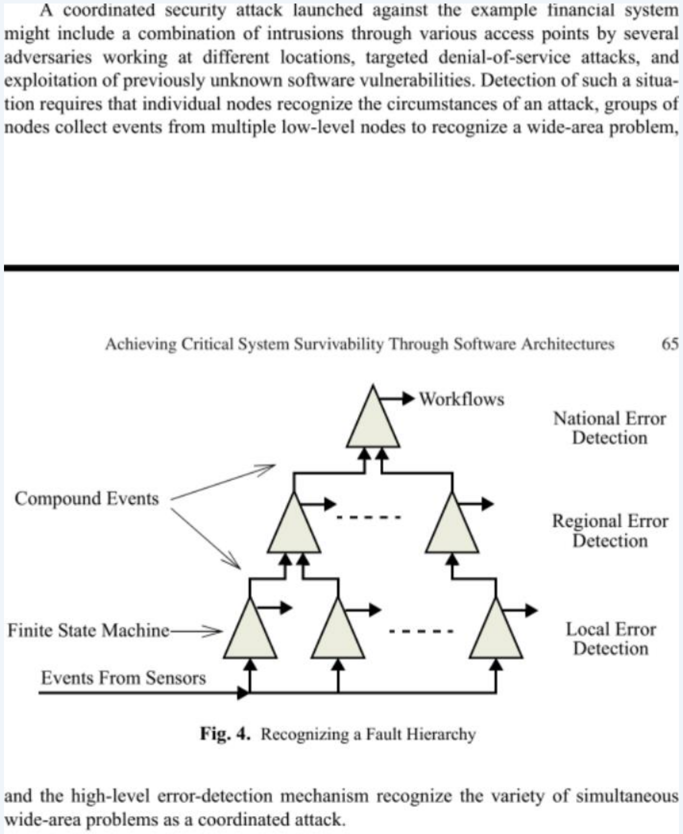
ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

A simplified approach for dynamic security risk management in c: 7: 30|65 - 7: 302|322 (0)



In our example, consider first the occurrence of a fault with a major file server that occurs during the middle of a normal market day (i.e., system state d_1) and which cannot be masked. To meet its survivability specification, the options that the system has are to transition to providing either service S_1, S_4, or S_2, (see Fig. 1) and the maximum relative value to the user (from the ***V*** table indexed by the current conditions d_1) would be in service S_1 in this case. Were this to occur during the night, the transition would be to service S_4 because the current conditions would be d_4. Now suppose that while the server is down, a coordinated security attack is launched against the system (a bad situation getting worse). In that case, the response to the attack might be to shut down as much of the system as possible. The system would transition to service S_3 since that would permit the best support in the event that the situation developed into a governmental crisis.

Achieving critical system survivability through software archit: 10 - 10 (0)



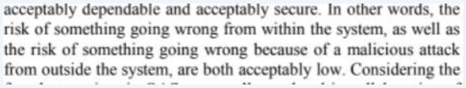
A coordinated security attack launched against the example financial system might include a combination of intrusions through various access points by several adversaries working at different locations, targeted denial-of-service attacks, and exploitation of previously unknown software vulnerabilities. Detection of such a situation requires that individual nodes recognize the circumstances of an attack, groups of nodes collect events from multiple low-level nodes to recognize a wide-area problem,

Achieving Critical System Survivability Through Software Architectures 65

Fig. 4. Recognizing a Fault Hierarchy

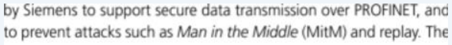
and the high-level error-detection mechanism recognize the variety of simultaneous wide-area problems as a coordinated attack.

Achieving critical system survivability through software archit: 14 - 15 (0)



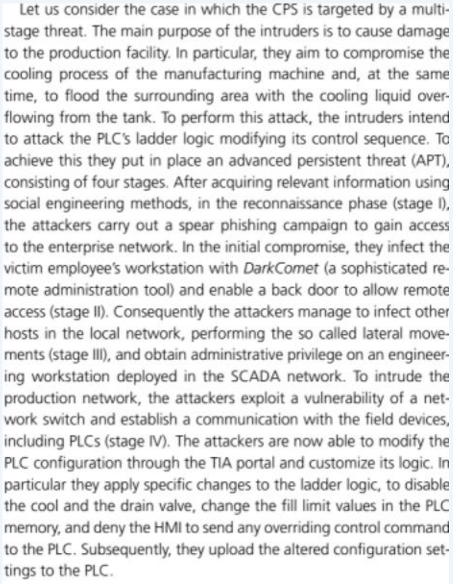
acceptably dependable and acceptably secure. In other words, the risk of something going wrong from within the system, as well as the risk of something going wrong because of a malicious attack from outside the system, are both acceptably low. Considering the

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



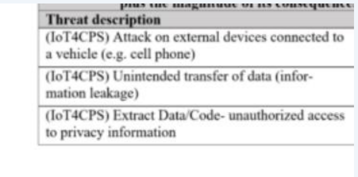
by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using social engineering methods, in the reconnaissance phase (stage I), the attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with *DarkComet* (a sophisticated remote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral movements (stage III), and obtain administrative privilege on an engineering workstation deployed in the SCADA network. To intrude the production network, the attackers exploit a vulnerability of a network switch and establish a communication with the field devices, including PLCs (stage IV). The attackers are now able to modify the PLC configuration through the TIA portal and customize its logic. In particular they apply specific changes to the ladder logic, to disable the cool and the drain valve, change the fill limit values in the PLC memory, and deny the HMI to send any overriding control command to the PLC. Subsequently, they upload the altered configuration settings to the PLC.

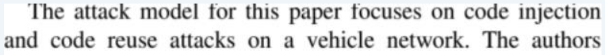
Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)



plus the magnitude of its consequence

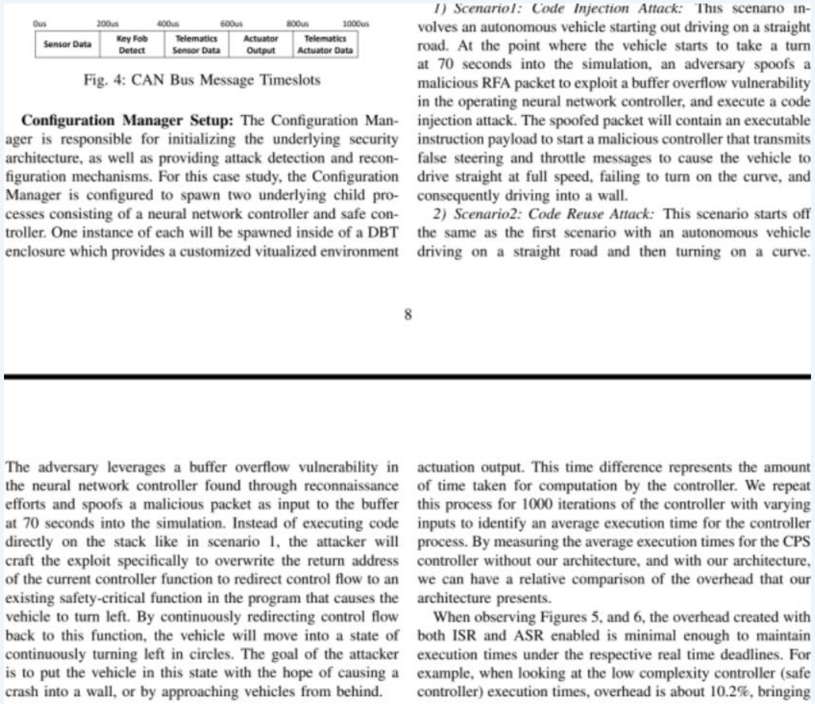
Threat description
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)
(IoT4CPS) Unintended transfer of data (information leakage)
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information

Digital Twins for Dependability Improvement of Autonomous Drivi: 10: 134|209 - 10: 316|295 (0)



The attack model for this paper focuses on code injection and code reuse attacks on a vehicle network. The authors

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)



0us
200us
400us
600us
800us
1000us
Sensor Data
Key Fob Detect
Telematics Sensor Data
Actuator Output
Telematics Actuator Data

Fig. 4: CAN Bus Message Timeslots

Configuration Manager Setup: The Configuration Manager is responsible for initializing the underlying security architecture, as well as providing attack detection and reconfiguration mechanisms. For this case study, the Configuration Manager is configured to spawn two underlying child processes consisting of a neural network controller and safe controller. One instance of each will be spawned inside of a DBT enclosure which provides a customized vitualized environment

1) Scenario1: Code Injection Attack: This scenario involves an autonomous vehicle starting out driving on a straight road. At the point where the vehicle starts to take a turn at 70 seconds into the simulation, an adversary spoofs a malicious RFA packet to exploit a buffer overflow vulnerability in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall.

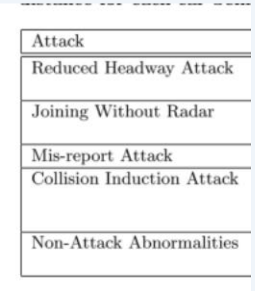
2) Scenario2: Code Reuse Attack: This scenario starts off the same as the first scenario with an autonomous vehicle driving on a straight road and then turning on a curve.

The adversary leverages a buffer overflow vulnerability in the neural network controller found through reconnaissance efforts and spoofs a malicious packet as input to the buffer at 70 seconds into the simulation. Instead of executing code directly on the stack like in scenario 1, the attacker will craft the exploit specifically to overwrite the return address of the current controller function to redirect control flow to an existing safety-critical function in the program that causes the vehicle to turn left. By continuously redirecting control flow back to this function, the vehicle will move into a state of continuously turning left in circles. The goal of the attacker is to put the vehicle in this state with the hope of causing a crash into a wall, or by approaching vehicles from behind.

actuation output. This time difference represents the amount of time taken for computation by the controller. We repeat this process for 1000 iterations of the controller with varying inputs to identify an average execution time for the controller process. By measuring the average execution times for the CPS controller without our architecture, and with our architecture, we can have a relative comparison of the overhead that our architecture presents.

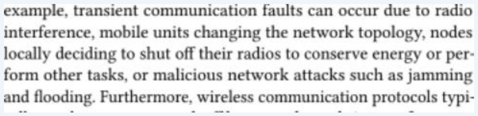
When observing Figures 5, and 6, the overhead created with both ISR and ASR enabled is minimal enough to maintain execution times under the respective real time deadlines. For example, when looking at the low complexity controller (safe controller) execution times, overhead is about 10.2%, bringing

Integrated moving target defense and control reconfiguration fo: 8 - 9 (0)



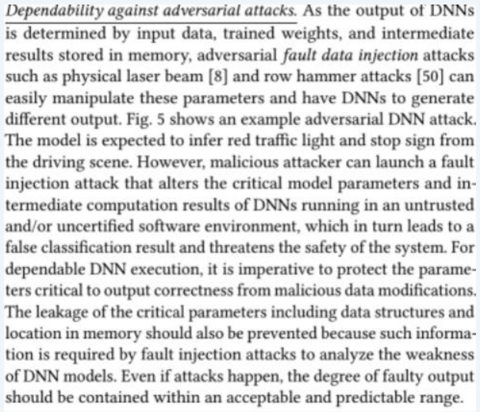
Attack
Reduced Headway Attack
Joining Without Radar
Mis-report Attack
Collision Induction Attack
Non-Attack Abnormalities

Is your commute driving you crazy a study of misbehavior in veh: 5: 44|399 - 5: 173|543 (0)



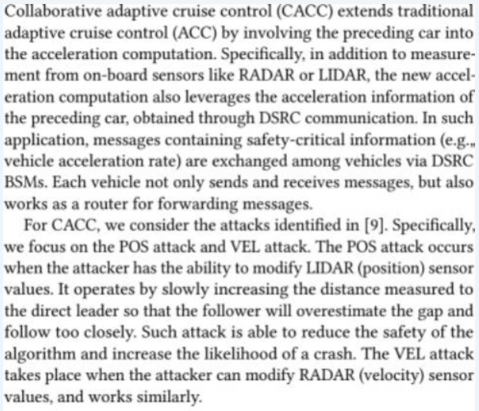
example, transient communication faults can occur due to radio interference, mobile units changing the network topology, nodes locally deciding to shut off their radios to conserve energy or perform other tasks, or malicious network attacks such as jamming and flooding. Furthermore, wireless communication protocols typi-

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



Dependability against adversarial attacks. As the output of DNNs is determined by input data, trained weights, and intermediate results stored in memory, adversarial *fault data injection* attacks such as physical laser beam [8] and row hammer attacks [50] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack. The model is expected to infer red traffic light and stop sign from the driving scene. However, malicious attacker can launch a fault injection attack that alters the critical model parameters and intermediate computation results of DNNs running in an untrusted and/or uncertified software environment, which in turn leads to a false classification result and threatens the safety of the system. For dependable DNN execution, it is imperative to protect the parameters critical to output correctness from malicious data modifications. The leakage of the critical parameters including data structures and location in memory should also be prevented because such information is required by fault injection attacks to analyze the weakness of DNN models. Even if attacks happen, the degree of faulty output should be contained within an acceptable and predictable range.

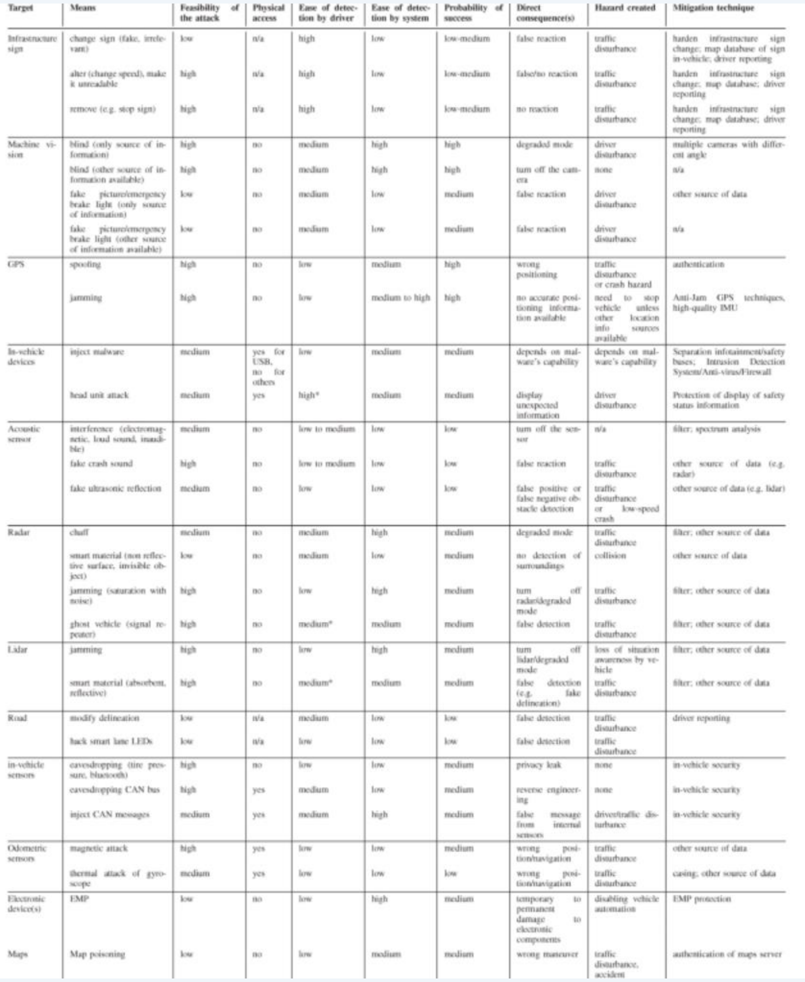
Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



Collaborative adaptive cruise control (CACC) extends traditional adaptive cruise control (ACC) by involving the preceding car into the acceleration computation. Specifically, in addition to measurement from on-board sensors like RADAR or LIDAR, the new acceleration computation also leverages the acceleration information of the preceding car, obtained through DSRC communication. In such application, messages containing safety-critical information (e.g., vehicle acceleration rate) are exchanged among vehicles via DSRC BSMs. Each vehicle not only sends and receives messages, but also works as a router for forwarding messages.

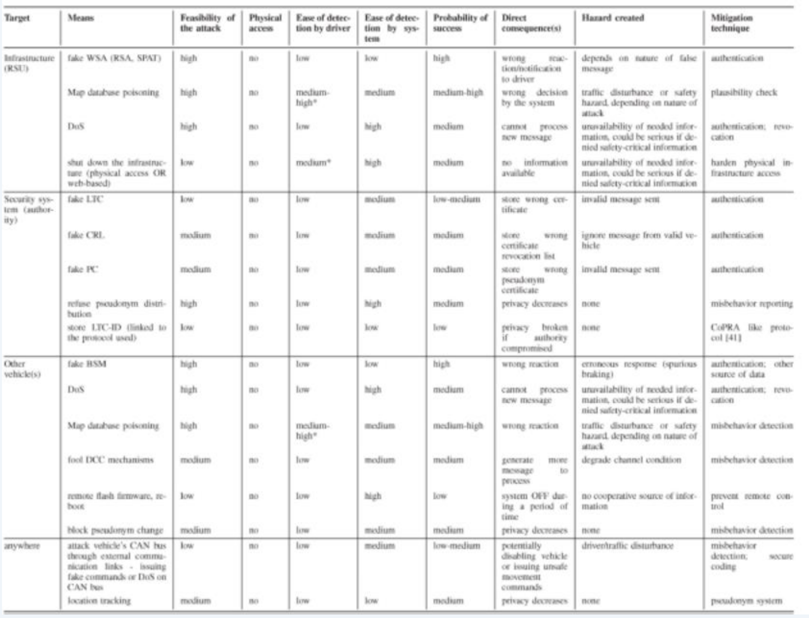
For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)



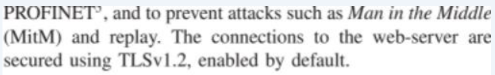
Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 41|66 - 5: 564|694 (0)



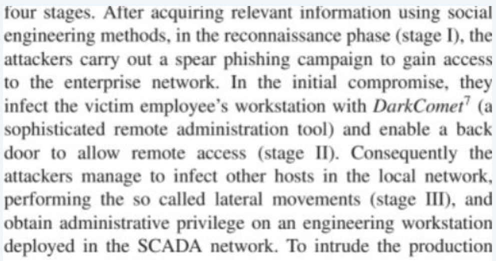
Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure (RSU)	fake WSA (RSA, SPAT)	high	no	low	low	high	wrong reaction/notification to driver	depends on nature of false message	authentication
	Map database poisoning	high	no	medium-high*	medium	medium-high	wrong decision by the system	traffic disturbance or safety hazard, depending on nature of attack	plausibility check
	DoS	high	no	low	high	medium	cannot process new message	unavailability of needed information, could be serious if denied safety-critical information	authentication; revocation
	shut down the infrastructure (physical access OR web-based)	low	no	medium*	high	medium	no information available	unavailability of needed information, could be serious if denied safety-critical information	harden physical infrastructure access
Security system (authority)	fake LTC	low	no	low	medium	low-medium	store wrong certificate	invalid message sent	authentication
	fake CRL	medium	no	low	medium	medium	store wrong certificate revocation list	ignore message from valid vehicle	authentication
	fake PC	medium	no	low	medium	medium	store wrong pseudonym certificate	invalid message sent	authentication
	refuse pseudonym distribution	high	no	low	high	medium	privacy decreases	none	misbehavior reporting
	store LTC-ID (linked to the protocol used)	low	no	low	low	low	privacy broken if authority compromised	none	CoPRA like protocol [41]
Other vehicle(s)	fake BSM	high	no	low	low	high	wrong reaction	erroneous response (spurious braking)	authentication; other source of data
	DoS	high	no	low	high	medium	cannot process new message	unavailability of needed information, could be serious if denied safety-critical information	authentication; revocation
	Map database poisoning	high	no	medium-high*	medium	medium-high	wrong reaction	traffic disturbance or safety hazard, depending on nature of attack	misbehavior detection
	fool DCC mechanisms	medium	no	low	medium	medium	generate more message to process	degrade channel condition	misbehavior detection
	remote flash firmware, reboot	low	no	low	high	low	system OFF during a period of time	no cooperative source of information	prevent remote control
	block pseudonym change	medium	no	low	medium	medium	privacy decreases	none	misbehavior detection
anywhere	attack vehicle's CAN bus through external communication links - issuing fake commands or DoS on CAN bus	low	no	low	medium	low-medium	potentially disabling vehicle or issuing unsafe movement commands	driver/traffic disturbance	misbehavior detection; secure coding
	location tracking	medium	no	low	low	medium	privacy decreases	none	pseudonym system

Potential cyberattacks on automated vehicles: 8: 41|315 - 8: 556|703 (0)



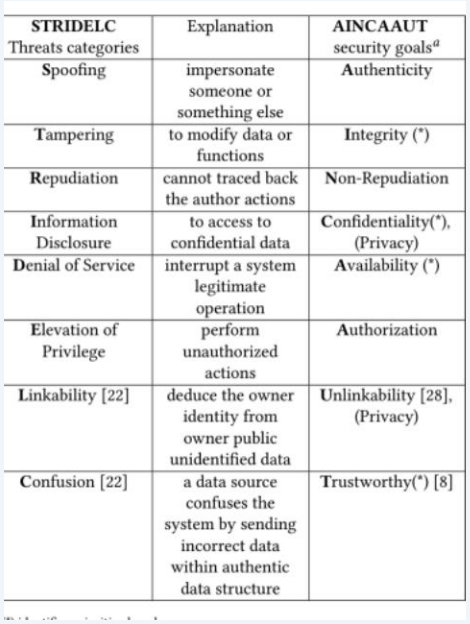
PROFINET[5], and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)



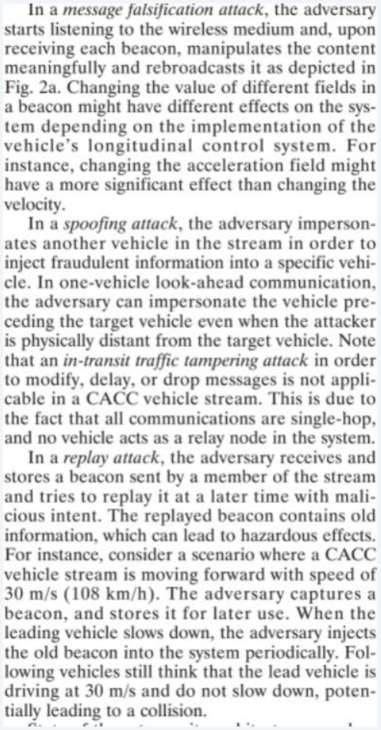
four stages. After acquiring relevant information using social engineering methods, in the reconnaissance phase (stage I), the attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with *DarkComet*[7] (a sophisticated remote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral movements (stage III), and obtain administrative privilege on an engineering workstation deployed in the SCADA network. To intrude the production

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



STRIDELC Threats categories	Explanation	**AINCAAUT** security goals[a]
Spoofing	impersonate someone or something else	**A**uthenticity
Tampering	to modify data or functions	**I**ntegrity (*)
Repudiation	cannot traced back the author actions	**N**on-Repudiation
Information Disclosure	to access to confidential data	**C**onfidentiality(*), (Privacy)
Denial of Service	interrupt a system legitimate operation	**A**vailability (*)
Elevation of Privilege	perform unauthorized actions	**A**uthorization
Linkability [22]	deduce the owner identity from owner public unidentified data	**U**nlinkability [28], (Privacy)
Confusion [22]	a data source confuses the system by sending incorrect data within authentic data structure	**T**rustworthy(*) [8]

SARA Security Automotive Risk Analysis Method: 5: 58|379 - 5: 295|693 (0)

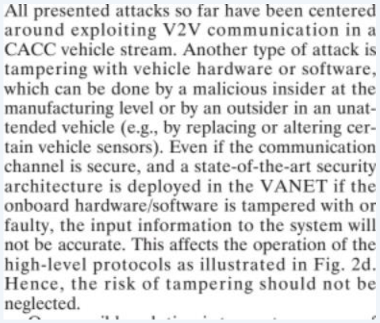


In a *message falsification attack*, the adversary starts listening to the wireless medium and, upon receiving each beacon, manipulates the content meaningfully and rebroadcasts it as depicted in Fig. 2a. Changing the value of different fields in a beacon might have different effects on the system depending on the implementation of the vehicle's longitudinal control system. For instance, changing the acceleration field might have a more significant effect than changing the velocity.

In a *spoofing attack*, the adversary impersonates another vehicle in the stream in order to inject fraudulent information into a specific vehicle. In one-vehicle look-ahead communication, the adversary can impersonate the vehicle preceding the target vehicle even when the attacker is physically distant from the target vehicle. Note that an *in-transit traffic tampering attack* in order to modify, delay, or drop messages is not applicable in a CACC vehicle stream. This is due to the fact that all communications are single-hop, and no vehicle acts as a relay node in the system.

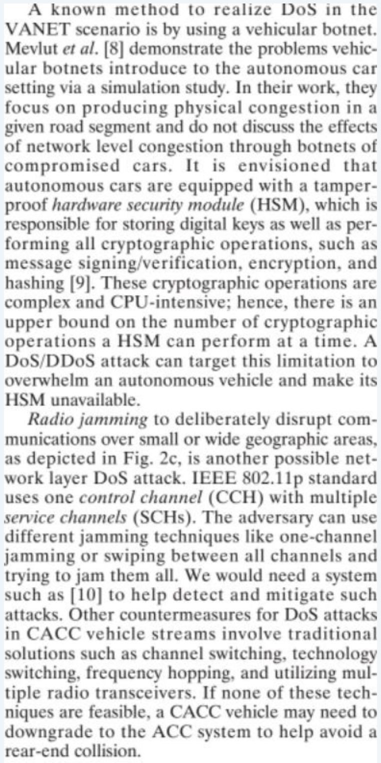
In a *replay attack*, the adversary receives and stores a beacon sent by a member of the stream and tries to replay it at a later time with malicious intent. The replayed beacon contains old information, which can lead to hazardous effects. For instance, consider a scenario where a CACC vehicle stream is moving forward with speed of 30 m/s (108 km/h). The adversary captures a beacon, and stores it for later use. When the leading vehicle slows down, the adversary injects the old beacon into the system periodically. Following vehicles still think that the lead vehicle is driving at 30 m/s and do not slow down, potentially leading to a collision.

Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)



All presented attacks so far have been centered around exploiting V2V communication in a CACC vehicle stream. Another type of attack is tampering with vehicle hardware or software, which can be done by a malicious insider at the manufacturing level or by an outsider in an unattended vehicle (e.g., by replacing or altering certain vehicle sensors). Even if the communication channel is secure, and a state-of-the-art security architecture is deployed in the VANET if the onboard hardware/software is tampered with or faulty, the input information to the system will not be accurate. This affects the operation of the high-level protocols as illustrated in Fig. 2d. Hence, the risk of tampering should not be neglected.

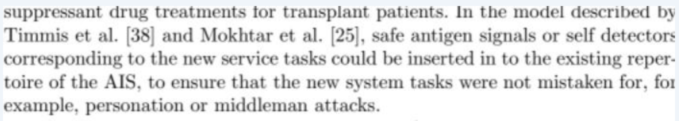
Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



A known method to realize DoS in the VANET scenario is by using a vehicular botnet. Mevlut *et al*. [8] demonstrate the problems vehicular botnets introduce to the autonomous car setting via a simulation study. In their work, they focus on producing physical congestion in a given road segment and do not discuss the effects of network level congestion through botnets of compromised cars. It is envisioned that autonomous cars are equipped with a tamper-proof *hardware security module* (HSM), which is responsible for storing digital keys as well as performing all cryptographic operations, such as message signing/verification, encryption, and hashing [9]. These cryptographic operations are complex and CPU-intensive; hence, there is an upper bound on the number of cryptographic operations a HSM can perform at a time. A DoS/DDoS attack can target this limitation to overwhelm an autonomous vehicle and make its HSM unavailable.

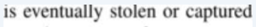
Radio jamming to deliberately disrupt communications over small or wide geographic areas, as depicted in Fig. 2c, is another possible network layer DoS attack. IEEE 802.11p standard uses one *control channel* (CCH) with multiple *service channels* (SCHs). The adversary can use different jamming techniques like one-channel jamming or swiping between all channels and trying to jam them all. We would need a system such as [10] to help detect and mitigate such attacks. Other countermeasures for DoS attacks in CACC vehicle streams involve traditional solutions such as channel switching, technology switching, frequency hopping, and utilizing multiple radio transceivers. If none of these techniques are feasible, a CACC vehicle may need to downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



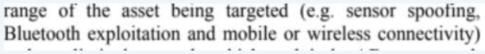
suppressant drug treatments for transplant patients. In the model described by Timmis et al. [38] and Mokhtar et al. [25], safe antigen signals or self detectors corresponding to the new service tasks could be inserted in to the existing repertoire of the AIS, to ensure that the new system tasks were not mistaken for, for example, personation or middleman attacks.

Self-organisation for Survival in Complex Computer Architecture: 13 - 13 (0)



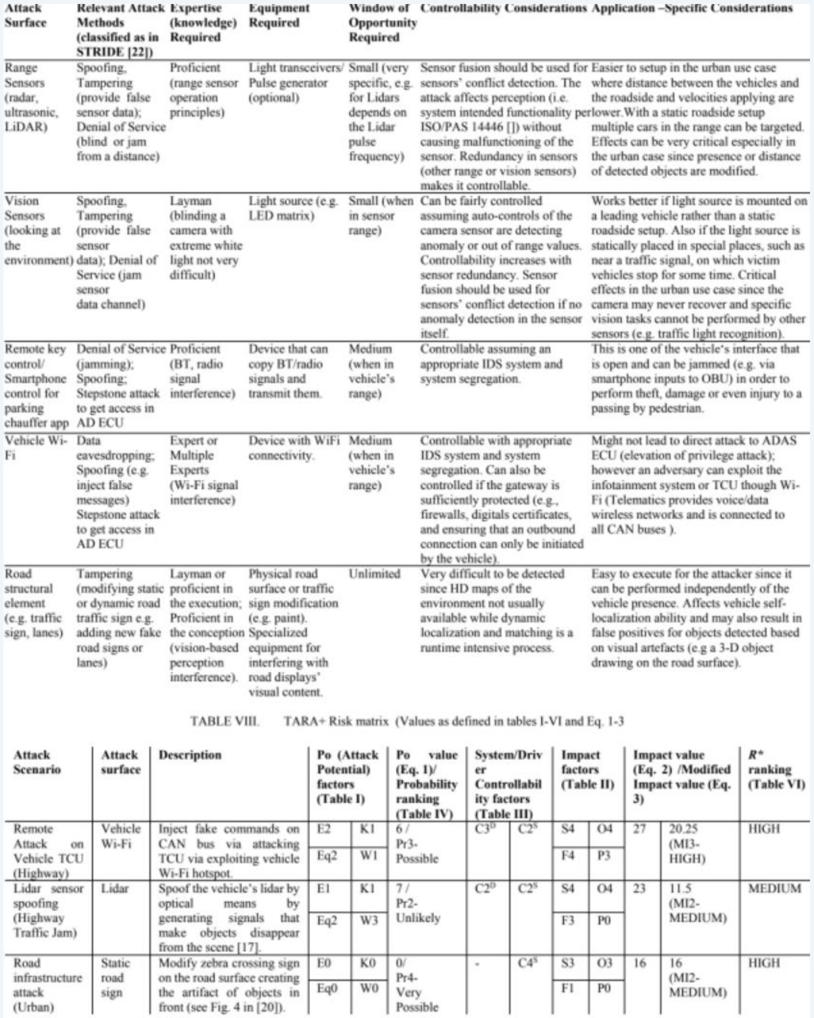
is eventually stolen or captured

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)



range of the asset being targeted (e.g. sensor spoofing, Bluetooth exploitation and mobile or wireless connectivity)

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

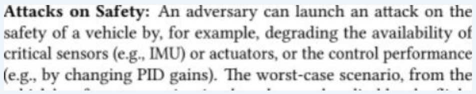


Attack Surface	Relevant Attack Methods (classified as in STRIDE [22])	Expertise (knowledge) Required	Equipment Required	Window of Opportunity Required	Controllability Considerations	Application –Specific Considerations
Range Sensors (radar, ultrasonic, LiDAR)	Spoofing, Tampering (provide false sensor data); Denial of Service (blind or jam from a distance)	Proficient (range sensor operation principles)	Light transceivers/ Pulse generator (optional)	Small (very specific, e.g. for Lidars depends on the Lidar pulse frequency)	Sensor fusion should be used for sensors' conflict detection. The attack affects perception (i.e. system intended functionality per ISO/PAS 14446 []) without causing malfunctioning of the sensor. Redundancy in sensors (other range or vision sensors) makes it controllable.	Easier to setup in the urban use case where distance between the vehicles and the roadside and velocities applying are lower.With a static roadside setup multiple cars in the range can be targeted. Effects can be very critical especially in the urban case since presence or distance of detected objects are modified.
Vision Sensors (looking at the environment)	Spoofing, Tampering (provide false sensor data); Denial of Service (jam sensor data channel)	Layman (blinding a camera with extreme white light not very difficult)	Light source (e.g. LED matrix)	Small (when in sensor range)	Can be fairly controlled assuming auto-controls of the camera sensor are detecting anomaly or out of range values. Controllability increases with sensor redundancy. Sensor fusion should be used for sensors' conflict detection if no anomaly detection in the sensor itself.	Works better if light source is mounted on a leading vehicle rather than a static roadside setup. Also if the light source is statically placed in special places, such as near a traffic signal, on which victim vehicles stop for some time. Critical effects in the urban use case since the camera may never recover and specific vision tasks cannot be performed by other sensors (e.g. traffic light recognition).
Remote key control/ Smartphone control for parking chauffer app	Denial of Service (jamming); Spoofing; Stepstone attack to get access in AD ECU	Proficient (BT, radio signal interference)	Device that can copy BT/radio signals and transmit them.	Medium (when in vehicle's range)	Controllable assuming an appropriate IDS system and system segregation.	This is one of the vehicle's interface that is open and can be jammed (e.g. via smartphone inputs to OBU) in order to perform theft, damage or even injury to a passing by pedestrian.
Vehicle Wi-Fi	Data eavesdropping; Spoofing (e.g. inject false messages) Stepstone attack to get access in AD ECU	Expert or Multiple Experts (Wi-Fi signal interference)	Device with WiFi connectivity.	Medium (when in vehicle's range)	Controllable with appropriate IDS system and system segregation. Can also be controlled if the gateway is sufficiently protected (e.g., firewalls, digitals certificates, and ensuring that an outbound connection can only be initiated by the vehicle).	Might not lead to direct attack to ADAS ECU (elevation of privilege attack); however an adversary can exploit the infotainment system or TCU though Wi-Fi (Telematics provides voice/data wireless networks and is connected to all CAN buses).
Road structural element (e.g. traffic sign, lanes)	Tampering (modifying static or dynamic road traffic sign e.g. adding new fake road signs or lanes)	Layman or proficient in the execution; Proficient in the conception (vision-based perception interference).	Physical road surface or traffic sign modification (e.g. paint). Specialized equipment for interfering with road displays' visual content.	Unlimited	Very difficult to be detected since HD maps of the environment not usually available while dynamic localization and matching is a runtime intensive process.	Easy to execute for the attacker since it can be performed independently of the vehicle presence. Affects vehicle self-localization ability and may also result in false positives for objects detected based on visual artefacts (e.g a 3-D object drawing on the road surface).

TABLE VIII. TARA+ Risk matrix (Values as defined in tables I-VI and Eq. 1-3

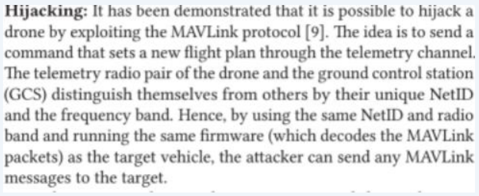
Attack Scenario	Attack surface	Description	Po (Attack Potential) factors (Table I)		Po value (Eq. 1)/ Probability ranking (Table IV)	System/Driver Controllability factors (Table III)		Impact factors (Table II)		Impact value (Eq. 2) /Modified Impact value (Eq. 3)		R^* ranking (Table VI)
Remote Attack on Vehicle TCU (Highway)	Vehicle Wi-Fi	Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.	E2	K1	6 / Pr3- Possible	$C3^D$	$C2^S$	S4	O4	27	20.25 (MI3- HIGH)	HIGH
			Eq2	W1				F4	P3			
Lidar sensor spoofing (Highway Traffic Jam)	Lidar	Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].	E1	K1	7 / Pr2- Unlikely	$C2^D$	$C2^S$	S4	O4	23	11.5 (MI2- MEDIUM)	MEDIUM
			Eq2	W3				F3	P0			
Road infrastructure attack (Urban)	Static road sign	Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).	E0	K0	0/ Pr4- Very Possible	-	$C4^S$	S3	O3	16	16 (MI2- MEDIUM)	HIGH
			Eq0	W0				F1	P0			

TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)



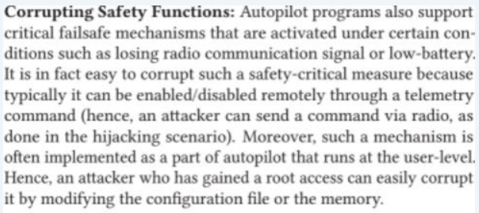
Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the

VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)



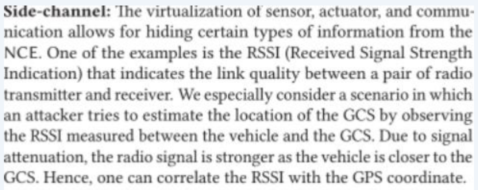
Hijacking: It has been demonstrated that it is possible to hijack a drone by exploiting the MAVLink protocol [9]. The idea is to send a command that sets a new flight plan through the telemetry channel. The telemetry radio pair of the drone and the ground control station (GCS) distinguish themselves from others by their unique NetID and the frequency band. Hence, by using the same NetID and radio band and running the same firmware (which decodes the MAVLink packets) as the target vehicle, the attacker can send any MAVLink messages to the target.

VirtualDrone virtual sensing actuation and communication for at: 7 - 7 (0)



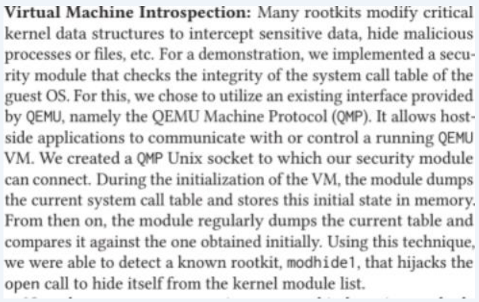
Corrupting Safety Functions: Autopilot programs also support critical failsafe mechanisms that are activated under certain conditions such as losing radio communication signal or low-battery. It is in fact easy to corrupt such a safety-critical measure because typically it can be enabled/disabled remotely through a telemetry command (hence, an attacker can send a command via radio, as done in the hijacking scenario). Moreover, such a mechanism is often implemented as a part of autopilot that runs at the user-level. Hence, an attacker who has gained a root access can easily corrupt it by modifying the configuration file or the memory.

VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)



Side-channel: The virtualization of sensor, actuator, and communication allows for hiding certain types of information from the NCE. One of the examples is the RSSI (Received Signal Strength Indication) that indicates the link quality between a pair of radio transmitter and receiver. We especially consider a scenario in which an attacker tries to estimate the location of the GCS by observing the RSSI measured between the vehicle and the GCS. Due to signal attenuation, the radio signal is stronger as the vehicle is closer to the GCS. Hence, one can correlate the RSSI with the GPS coordinate.

VirtualDrone virtual sensing actuation and communication for at: 8 - 8 (0)



Virtual Machine Introspection: Many rootkits modify critical kernel data structures to intercept sensitive data, hide malicious processes or files, etc. For a demonstration, we implemented a security module that checks the integrity of the system call table of the guest OS. For this, we chose to utilize an existing interface provided by `QEMU`, namely the QEMU Machine Protocol (`QMP`). It allows host-side applications to communicate with or control a running `QEMU` VM. We created a `QMP` Unix socket to which our security module can connect. During the initialization of the VM, the module dumps the current system call table and stores this initial state in memory. From then on, the module regularly dumps the current table and compares it against the one obtained initially. Using this technique, we were able to detect a known rootkit, `modhide1`, that hijacks the open call to hide itself from the kernel module list.

VirtualDrone virtual sensing actuation and communication for at: 9 - 9 (0)

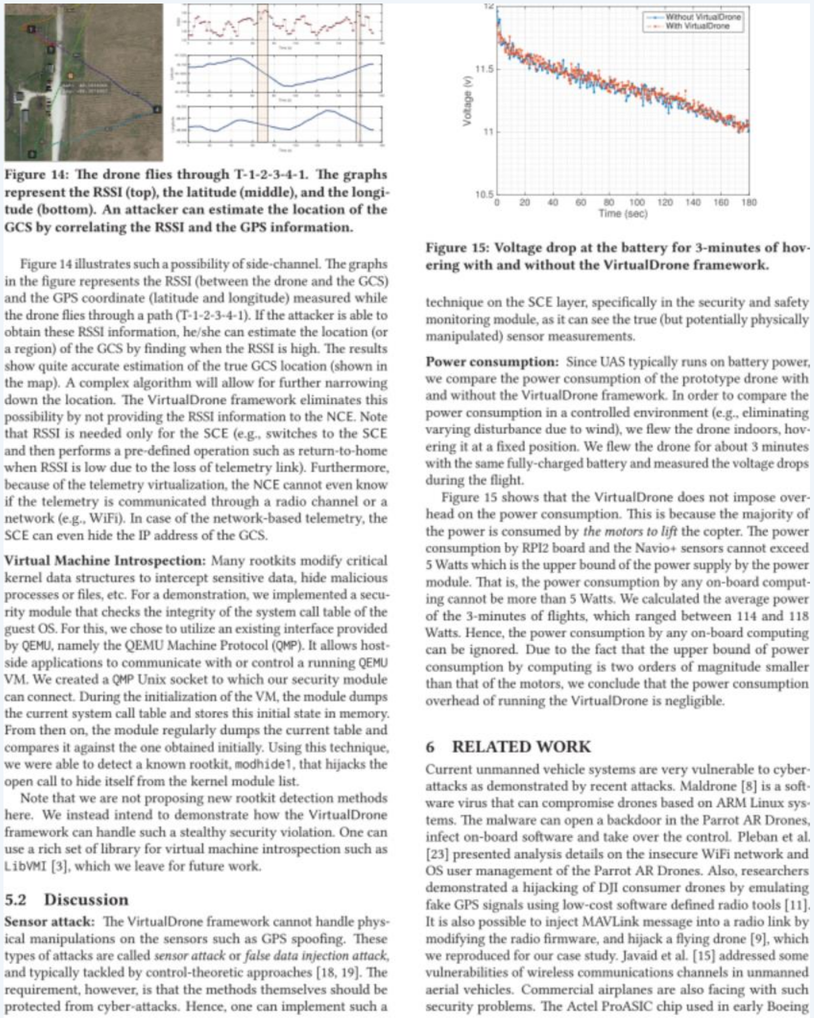


Figure 14: The drone flies through T-1-2-3-4-1. The graphs represent the RSSI (top), the latitude (middle), and the longitude (bottom). An attacker can estimate the location of the GCS by correlating the RSSI and the GPS information.

Figure 14 illustrates such a possibility of side-channel. The graphs in the figure represents the RSSI (between the drone and the GCS) and the GPS coordinate (latitude and longitude) measured while the drone flies through a path (T-1-2-3-4-1). If the attacker is able to obtain these RSSI information, he/she can estimate the location (or a region) of the GCS by finding when the RSSI is high. The results show quite accurate estimation of the true GCS location (shown in the map). A complex algorithm will allow for further narrowing down the location. The VirtualDrone framework eliminates this possibility by not providing the RSSI information to the NCE. Note that RSSI is needed only for the SCE (e.g., switches to the SCE and then performs a pre-defined operation such as return-to-home when RSSI is low due to the loss of telemetry link). Furthermore, because of the telemetry virtualization, the NCE cannot even know if the telemetry is communicated through a radio channel or a network (e.g., WiFi). In case of the network-based telemetry, the SCE can even hide the IP address of the GCS.

Virtual Machine Introspection: Many rootkits modify critical kernel data structures to intercept sensitive data, hide malicious processes or files, etc. For a demonstration, we implemented a security module that checks the integrity of the system call table of the guest OS. For this, we chose to utilize an existing interface provided by QEMU, namely the QEMU Machine Protocol (QMP). It allows host-side applications to communicate with or control a running QEMU VM. We created a QMP Unix socket to which our security module can connect. During the initialization of the VM, the module dumps the current system call table and stores this initial state in memory. From then on, the module regularly dumps the current table and compares it against the one obtained initially. Using this technique, we were able to detect a known rootkit, modhide1, that hijacks the open call to hide itself from the kernel module list.

Note that we are not proposing new rootkit detection methods here. We instead intend to demonstrate how the VirtualDrone framework can handle such a stealthy security violation. One can use a rich set of library for virtual machine introspection such as LibVMI [3], which we leave for future work.

5.2 Discussion

Sensor attack: The VirtualDrone framework cannot handle physical manipulations on the sensors such as GPS spoofing. These types of attacks are called *sensor attack* or *false data injection attack*, and typically tackled by control-theoretic approaches [18, 19]. The requirement, however, is that the methods themselves should be protected from cyber-attacks. Hence, one can implement such a technique on the SCE layer, specifically in the security and safety monitoring module, as it can see the true (but potentially physically manipulated) sensor measurements.

Figure 15: Voltage drop at the battery for 3-minutes of hovering with and without the VirtualDrone framework.

Power consumption: Since UAS typically runs on battery power, we compare the power consumption of the prototype drone with and without the VirtualDrone framework. In order to compare the power consumption in a controlled environment (e.g., eliminating varying disturbance due to wind), we flew the drone indoors, hovering it at a fixed position. We flew the drone for about 3 minutes with the same fully-charged battery and measured the voltage drops during the flight.

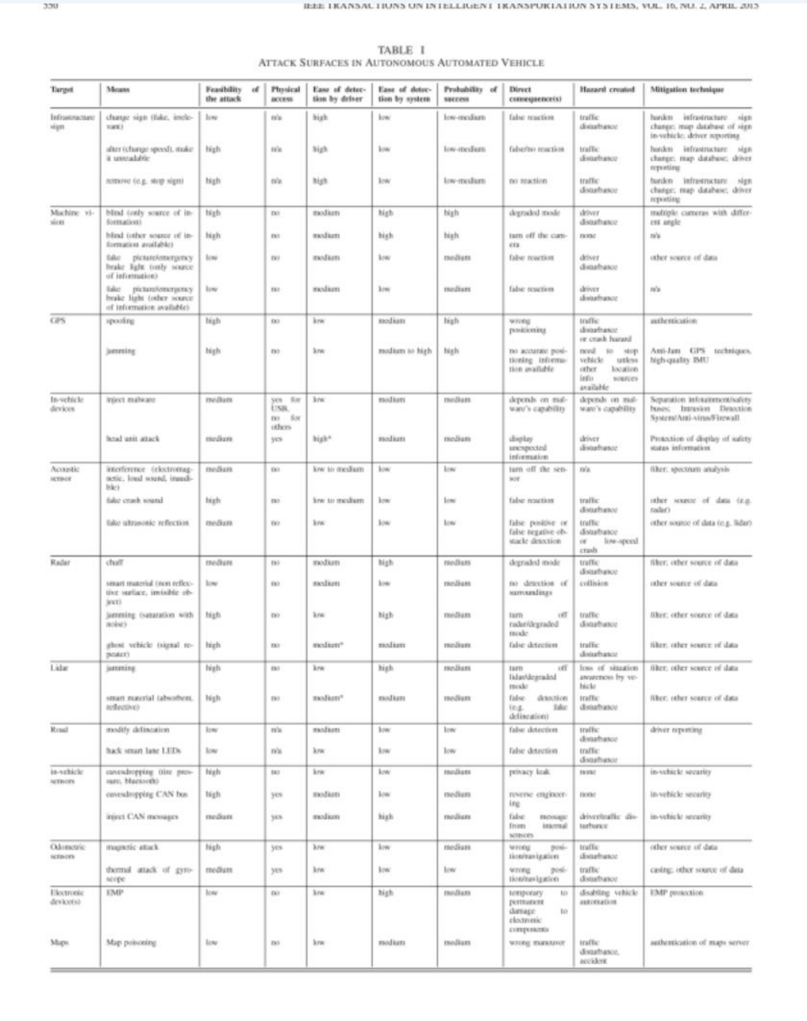
Figure 15 shows that the VirtualDrone does not impose overhead on the power consumption. This is because the majority of the power is consumed by *the motors to lift* the copter. The power consumption by RPI2 board and the Navio+ sensors cannot exceed 5 Watts which is the upper bound of the power supply by the power module. That is, the power consumption by any on-board computing cannot be more than 5 Watts. We calculated the average power of the 3-minutes of flights, which ranged between 114 and 118 Watts. Hence, the power consumption by any on-board computing can be ignored. Due to the fact that the upper bound of power consumption by computing is two orders of magnitude smaller than that of the motors, we conclude that the power consumption overhead of running the VirtualDrone is negligible.

6 RELATED WORK

Current unmanned vehicle systems are very vulnerable to cyber-attacks as demonstrated by recent attacks. Maldrone [8] is a software virus that can compromise drones based on ARM Linux systems. The malware can open a backdoor in the Parrot AR Drones, infect on-board software and take over the control. Pleban et al. [23] presented analysis details on the insecure WiFi network and OS user management of the Parrot AR Drones. Also, researchers demonstrated a hijacking of DJI consumer drones by emulating fake GPS signals using low-cost software defined radio tools [11]. It is also possible to inject MAVLink message into a radio link by modifying the radio firmware, and hijack a flying drone [9], which we reproduced for our case study. Javaid et al. [15] addressed some vulnerabilities of wireless communications channels in unmanned aerial vehicles. Commercial airplanes are also facing with such security problems. The Actel ProASIC chip used in early Boeing

VirtualDrone virtual sensing actuation and communication for at: 9 - 9 (0)

Integration\Security Attacks\Attack Mechanisms\Abuse Existing Functionality

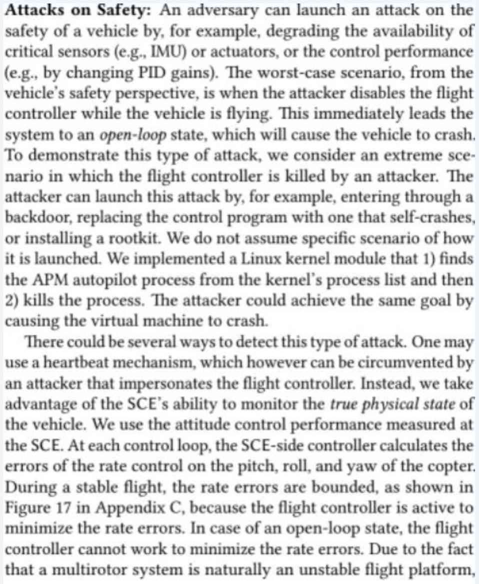


550 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)

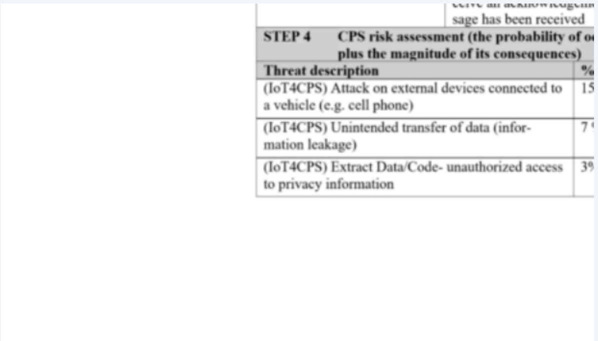


Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the vehicle's safety perspective, is when the attacker disables the flight controller while the vehicle is flying. This immediately leads the system to an *open-loop* state, which will cause the vehicle to crash. To demonstrate this type of attack, we consider an extreme scenario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how it is launched. We implemented a Linux kernel module that 1) finds the APM autopilot process from the kernel's process list and then 2) kills the process. The attacker could achieve the same goal by causing the virtual machine to crash.

There could be several ways to detect this type of attack. One may use a heartbeat mechanism, which however can be circumvented by an attacker that impersonates the flight controller. Instead, we take advantage of the SCE's ability to monitor the *true physical state* of the vehicle. We use the attitude control performance measured at the SCE. At each control loop, the SCE-side controller calculates the errors of the rate control on the pitch, roll, and yaw of the copter. During a stable flight, the rate errors are bounded, as shown in Figure 17 in Appendix C, because the flight controller is active to minimize the rate errors. In case of an open-loop state, the flight controller cannot work to minimize the rate errors. Due to the fact that a multirotor system is naturally an unstable flight platform,

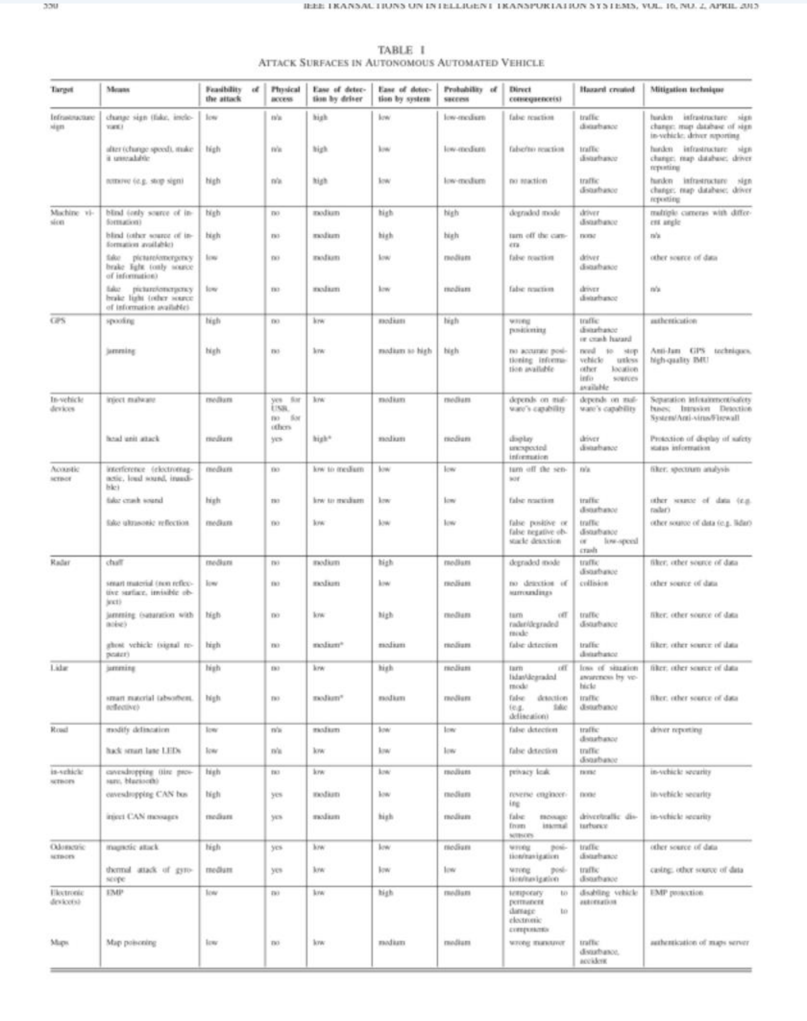
VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

Integration\Security Attacks\Attack Mechanisms\Collect and Analyze Information



	ceive an acknowledgem sage has been received
STEP 4	**CPS risk assessment (the probability of o plus the magnitude of its consequences)**
Threat description	**%**
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)	15
(IoT4CPS) Unintended transfer of data (information leakage)	7
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information	3%

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 23|152 - 9: 327|321 (0)



550 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)

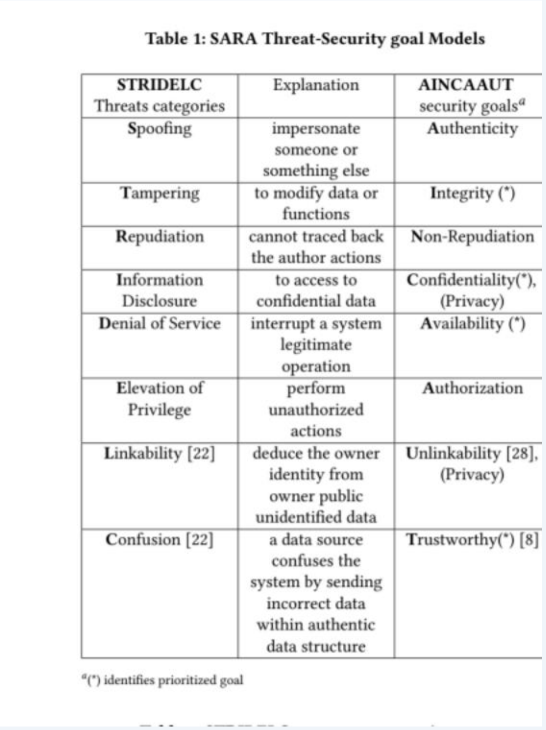


Table 1: SARA Threat-Security goal Models

STRIDELC Threats categories	Explanation	**AINCAAUT** security goals[a]
Spoofing	impersonate someone or something else	**A**uthenticity
Tampering	to modify data or functions	**I**ntegrity (*)
Repudiation	cannot traced back the author actions	**N**on-Repudiation
Information Disclosure	to access to confidential data	**C**onfidentiality(*), (Privacy)
Denial of Service	interrupt a system legitimate operation	**A**vailability (*)
Elevation of Privilege	perform unauthorized actions	**A**uthorization
Linkability [22]	deduce the owner identity from owner public unidentified data	**U**nlinkability [28], (Privacy)
Confusion [22]	a data source confuses the system by sending incorrect data within authentic data structure	**T**rustworthy(*) [8]

[a](*) identifies prioritized goal

SARA Security Automotive Risk Analysis Method: 5: 12|354 - 5: 290|723 (0)

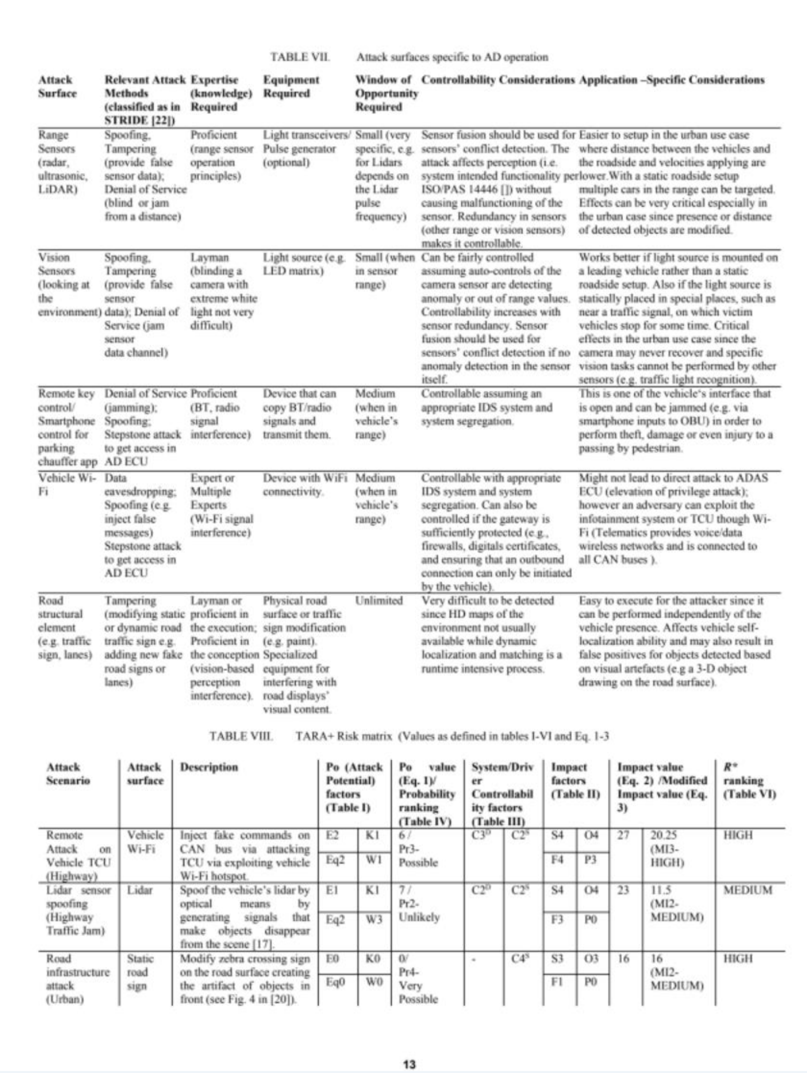


TABLE VII. Attack surfaces specific to AD operation

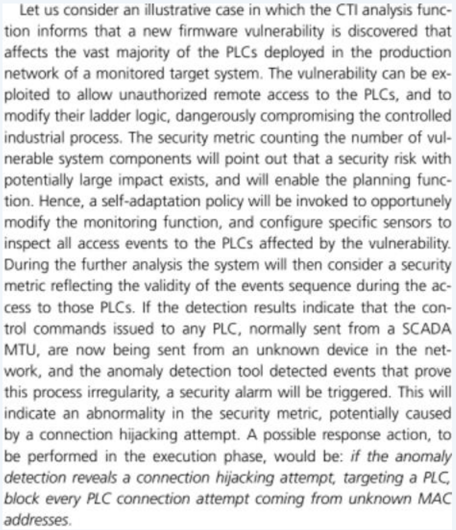
Attack Surface	Relevant Attack Methods (classified as in STRIDE [22])	Expertise (knowledge) Required	Equipment Required	Window of Opportunity Required	Controllability Considerations	Application –Specific Considerations
Range Sensors (radar, ultrasonic, LiDAR)	Spoofing, Tampering (provide false sensor data); Denial of Service (blind or jam from a distance)	Proficient (range sensor operation principles)	Light transceivers/ Pulse generator (optional)	Small (very specific, e.g. for Lidars depends on the Lidar pulse frequency)	Sensor fusion should be used for sensors' conflict detection. The attack affects perception (i.e. system intended functionality per ISO/PAS 14446 []) without causing malfunctioning of the sensor. Redundancy in sensors (other range or vision sensors) makes it controllable.	Easier to setup in the urban use case where distance between the vehicles and the roadside and velocities applying are lower.With a static roadside setup multiple cars in the range can be targeted. Effects can be very critical especially in the urban case since presence or distance of detected objects are modified.
Vision Sensors (looking at the environment)	Spoofing, Tampering (provide false sensor data); Denial of Service (jam sensor data channel)	Layman (blinding a camera with extreme white light not very difficult)	Light source (e.g. LED matrix)	Small (when in sensor range)	Can be fairly controlled assuming auto-controls of the camera sensor are detecting anomaly or out of range values. Controllability increases with sensor redundancy. Sensor fusion should be used for sensors' conflict detection if no anomaly detection in the sensor itself.	Works better if light source is mounted on a leading vehicle rather than a static roadside setup. Also if the light source is statically placed in special places, such as near a traffic signal, on which victim vehicles stop for some time. Critical effects in the urban use case since the camera may never recover and specific vision tasks cannot be performed by other sensors (e.g. traffic light recognition).
Remote key control/ Smartphone control for parking chauffer app	Denial of Service (jamming); Spoofing; Stepstone attack to get access in AD ECU	Proficient (BT, radio signal interference)	Device that can copy BT/radio signals and transmit them.	Medium (when in vehicle's range)	Controllable assuming an appropriate IDS system and system segregation.	This is one of the vehicle's interface that is open and can be jammed (e.g. via smartphone inputs to OBU) in order to perform theft, damage or even injury to a passing by pedestrian.
Vehicle Wi-Fi	Data eavesdropping; Spoofing (e.g. inject false messages) Stepstone attack to get access in AD ECU	Expert or Multiple Experts (Wi-Fi signal interference)	Device with WiFi connectivity.	Medium (when in vehicle's range)	Controllable with appropriate IDS system and system segregation. Can also be controlled if the gateway is sufficiently protected (e.g., firewalls, digitals certificates, and ensuring that an outbound connection can only be initiated by the vehicle).	Might not lead to direct attack to ADAS ECU (elevation of privilege attack); however an adversary can exploit the infotainment system or TCU though Wi-Fi (Telematics provides voice/data wireless networks and is connected to all CAN buses).
Road structural element (e.g. traffic sign, lanes)	Tampering (modifying static or dynamic road traffic sign e.g. adding new fake road signs or lanes)	Layman or proficient in the execution; Proficient in the conception (vision-based perception interference).	Physical road surface or traffic sign modification (e.g. paint). Specialized equipment for interfering with road displays' visual content.	Unlimited	Very difficult to be detected since HD maps of the environment not usually available while dynamic localization and matching is a runtime intensive process.	Easy to execute for the attacker since it can be performed independently of the vehicle presence. Affects vehicle self-localization ability and may also result in false positives for objects detected based on visual artefacts (e.g a 3-D object drawing on the road surface).

TABLE VIII. TARA+ Risk matrix (Values as defined in tables I-VI and Eq. 1-3

Attack Scenario	Attack surface	Description	Po (Attack Potential) factors (Table I)		Po value (Eq. 1)/ Probability ranking (Table IV)	System/Driver Controllability factors (Table III)		Impact factors (Table II)		Impact value (Eq. 2) /Modified Impact value (Eq. 3)		*R** ranking (Table VI)
Remote Attack on Vehicle TCU (Highway)	Vehicle Wi-Fi	Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.	E2 Eq2	K1 W1	6 / Pr3- Possible	$C3^{D}$	$C2^{S}$	S4 F4	O4 P3	27	20.25 (MI3- HIGH)	HIGH
Lidar sensor spoofing (Highway Traffic Jam)	Lidar	Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].	E1 Eq2	K1 W3	7 / Pr2- Unlikely	$C2^{D}$	$C2^{S}$	S4 F3	O4 P0	23	11.5 (MI2- MEDIUM)	MEDIUM
Road infrastructure attack (Urban)	Static road sign	Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).	E0 Eq0	K0 W0	0/ Pr4- Very Possible	-	$C4^{S}$	S3 F1	O3 P0	16	16 (MI2- MEDIUM)	HIGH

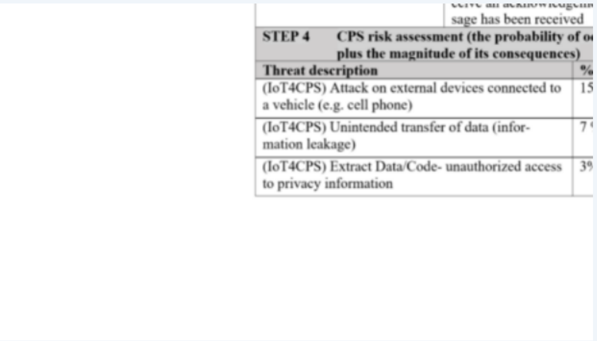
TARA Controllability-aware Threat Analysis and Risk Assessment: 6: 28|38 - 6: 576|763 (0)

Integration\Security Attacks\Attack Mechanisms\Subvert Access Control



Let us consider an illustrative case in which the CTI analysis function informs that a new firmware vulnerability is discovered that affects the vast majority of the PLCs deployed in the production network of a monitored target system. The vulnerability can be exploited to allow unauthorized remote access to the PLCs, and to modify their ladder logic, dangerously compromising the controlled industrial process. The security metric counting the number of vulnerable system components will point out that a security risk with potentially large impact exists, and will enable the planning function. Hence, a self-adaptation policy will be invoked to opportunely modify the monitoring function, and configure specific sensors to inspect all access events to the PLCs affected by the vulnerability. During the further analysis the system will then consider a security metric reflecting the validity of the events sequence during the access to those PLCs. If the detection results indicate that the control commands issued to any PLC, normally sent from a SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tool detected events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric, potentially caused by a connection hijacking attempt. A possible response action, to be performed in the execution phase, would be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses.*

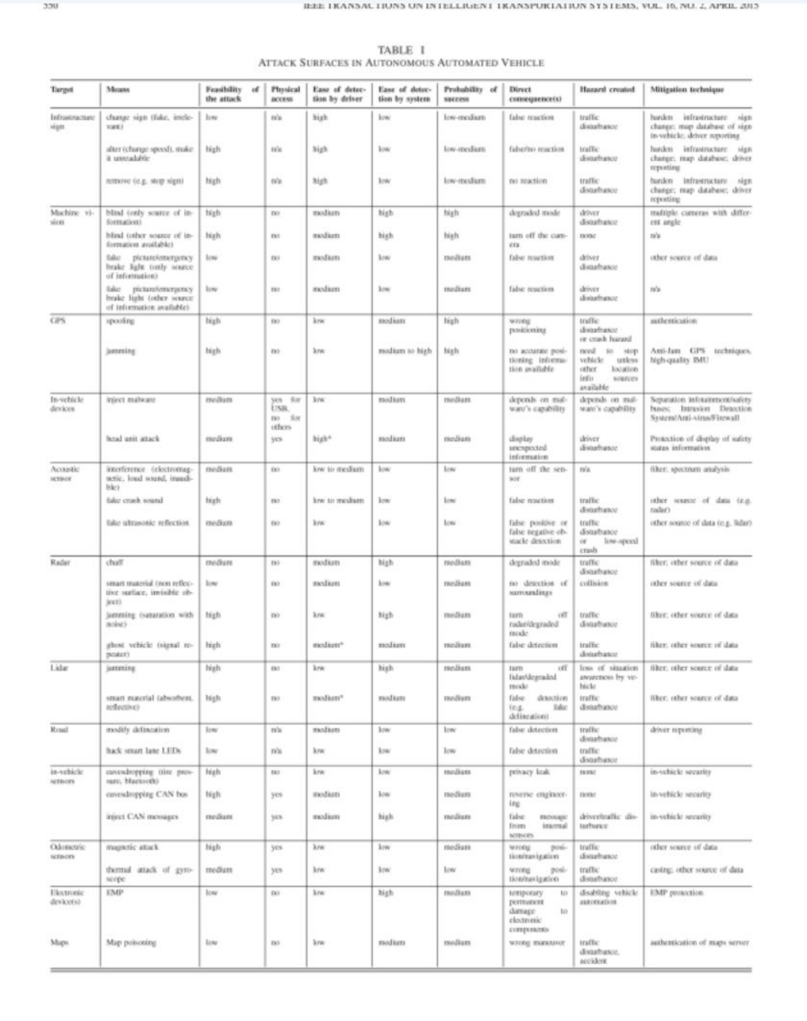
Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



ceive an acknowledgem
sage has been received

STEP 4 CPS risk assessment (the probability of o plus the magnitude of its consequences)	
Threat description	**%**
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)	15
(IoT4CPS) Unintended transfer of data (information leakage)	7
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information	3

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 23|152 - 9: 327|321 (0)

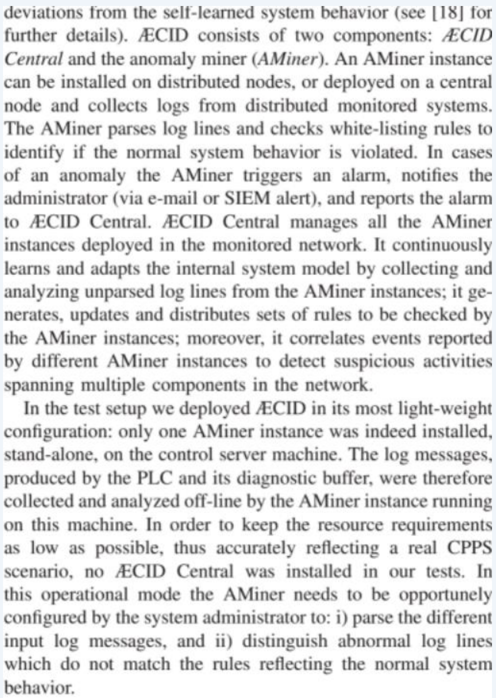


550 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong manoeuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)



deviations from the self-learned system behavior (see [18] for further details). ÆCID consists of two components: *ÆCID Central* and the anomaly miner (*AMiner*). An AMiner instance can be installed on distributed nodes, or deployed on a central node and collects logs from distributed monitored systems. The AMiner parses log lines and checks white-listing rules to identify if the normal system behavior is violated. In cases of an anomaly the AMiner triggers an alarm, notifies the administrator (via e-mail or SIEM alert), and reports the alarm to ÆCID Central. ÆCID Central manages all the AMiner instances deployed in the monitored network. It continuously learns and adapts the internal system model by collecting and analyzing unparsed log lines from the AMiner instances; it generates, updates and distributes sets of rules to be checked by the AMiner instances; moreover, it correlates events reported by different AMiner instances to detect suspicious activities spanning multiple components in the network.

In the test setup we deployed ÆCID in its most light-weight configuration: only one AMiner instance was indeed installed, stand-alone, on the control server machine. The log messages, produced by the PLC and its diagnostic buffer, were therefore collected and analyzed off-line by the AMiner instance running on this machine. In order to keep the resource requirements as low as possible, thus accurately reflecting a real CPPS scenario, no ÆCID Central was installed in our tests. In this operational mode the AMiner needs to be opportunely configured by the system administrator to: i) parse the different input log messages, and ii) distinguish abnormal log lines which do not match the rules reflecting the normal system behavior.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)

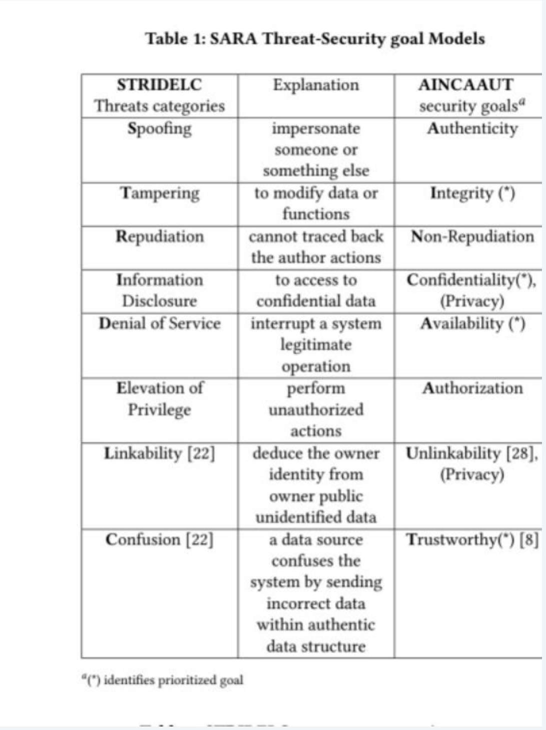
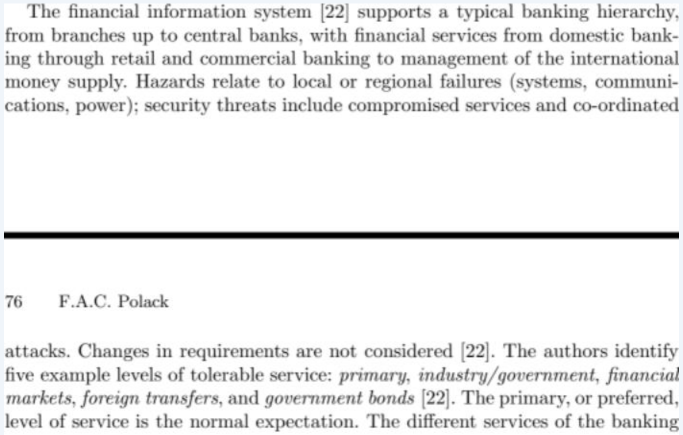


Table 1: SARA Threat-Security goal Models

STRIDELC Threats categories	Explanation	AINCAAUT security goals[a]
Spoofing	impersonate someone or something else	Authenticity
Tampering	to modify data or functions	Integrity (*)
Repudiation	cannot traced back the author actions	Non-Repudiation
Information Disclosure	to access to confidential data	Confidentiality(*), (Privacy)
Denial of Service	interrupt a system legitimate operation	Availability (*)
Elevation of Privilege	perform unauthorized actions	Authorization
Linkability [22]	deduce the owner identity from owner public unidentified data	Unlinkability [28], (Privacy)
Confusion [22]	a data source confuses the system by sending incorrect data within authentic data structure	Trustworthy(*) [8]

[a] (*) identifies prioritized goal

SARA Security Automotive Risk Analysis Method: 5: 12|354 - 5: 290|723 (0)



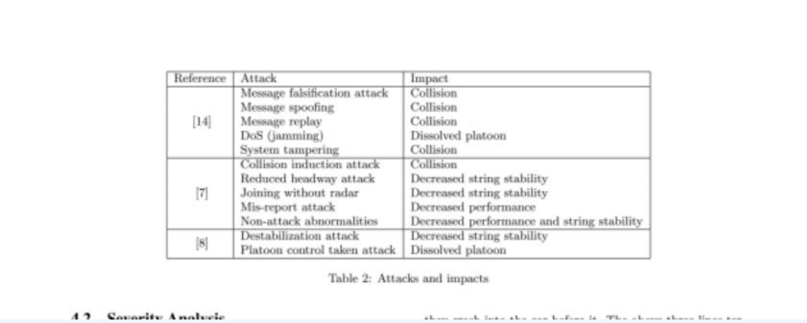
The financial information system [22] supports a typical banking hierarchy, from branches up to central banks, with financial services from domestic banking through retail and commercial banking to management of the international money supply. Hazards relate to local or regional failures (systems, communications, power); security threats include compromised services and co-ordinated

76 F.A.C. Polack

attacks. Changes in requirements are not considered [22]. The authors identify five example levels of tolerable service: *primary*, *industry/government*, *financial markets*, *foreign transfers*, and *government bonds* [22]. The primary, or preferred, level of service is the normal expectation. The different services of the banking

Self-organisation for Survival in Complex Computer Architecture: 10 - 11 (0)

Integration\Security Attacks\Attack Mechanisms\Inject Unexpected Items

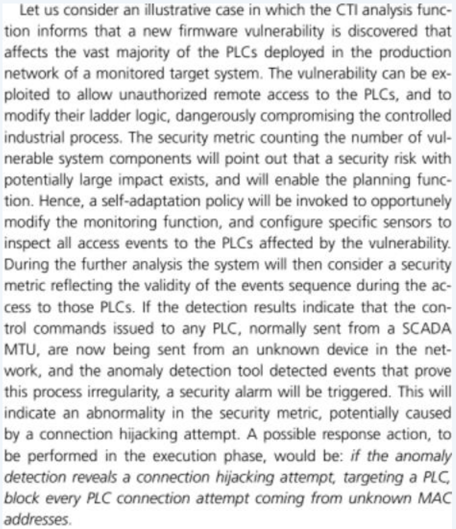


Reference	Attack	Impact
[14]	Message falsification attack	Collision
	Message spoofing	Collision
	Message replay	Collision
	DoS (jamming)	Dissolved platoon
	System tampering	Collision
[7]	Collision induction attack	Collision
	Reduced headway attack	Decreased string stability
	Joining without radar	Decreased string stability
	Mis-report attack	Decreased performance
	Non-attack abnormalities	Decreased performance and string stability
[8]	Destabilization attack	Decreased string stability
	Platoon control taken attack	Dissolved platoon

Table 2: Attacks and impacts

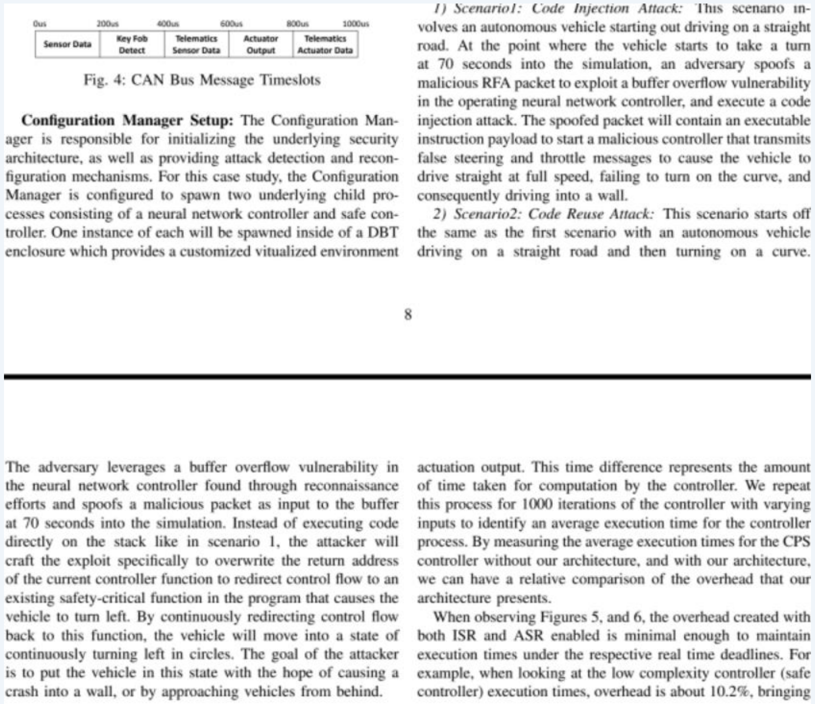
4.2 Severity Analysis

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 1|558 - 5: 601|791 (0)



Let us consider an illustrative case in which the CTI analysis function informs that a new firmware vulnerability is discovered that affects the vast majority of the PLCs deployed in the production network of a monitored target system. The vulnerability can be exploited to allow unauthorized remote access to the PLCs, and to modify their ladder logic, dangerously compromising the controlled industrial process. The security metric counting the number of vulnerable system components will point out that a security risk with potentially large impact exists, and will enable the planning function. Hence, a self-adaptation policy will be invoked to opportunely modify the monitoring function, and configure specific sensors to inspect all access events to the PLCs affected by the vulnerability. During the further analysis the system will then consider a security metric reflecting the validity of the events sequence during the access to those PLCs. If the detection results indicate that the control commands issued to any PLC, normally sent from a SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tool detected events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric, potentially caused by a connection hijacking attempt. A possible response action, to be performed in the execution phase, would be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses*.

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



0us 200us 400us 600us 800us 1000us

Sensor Data	Key Fob Detect	Telematics Sensor Data	Actuator Output	Telematics Actuator Data

Fig. 4: CAN Bus Message Timeslots

Configuration Manager Setup: The Configuration Manager is responsible for initializing the underlying security architecture, as well as providing attack detection and reconfiguration mechanisms. For this case study, the Configuration Manager is configured to spawn two underlying child processes consisting of a neural network controller and safe controller. One instance of each will be spawned inside of a DBT enclosure which provides a customized vitualized environment

1) Scenario1: Code Injection Attack: This scenario involves an autonomous vehicle starting out driving on a straight road. At the point where the vehicle starts to take a turn at 70 seconds into the simulation, an adversary spoofs a malicious RFA packet to exploit a buffer overflow vulnerability in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall.

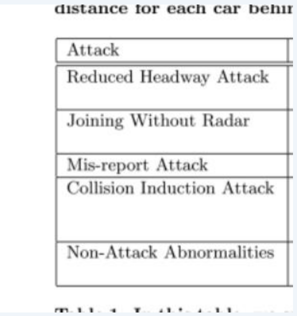
2) Scenario2: Code Reuse Attack: This scenario starts off the same as the first scenario with an autonomous vehicle driving on a straight road and then turning on a curve.

The adversary leverages a buffer overflow vulnerability in the neural network controller found through reconnaissance efforts and spoofs a malicious packet as input to the buffer at 70 seconds into the simulation. Instead of executing code directly on the stack like in scenario 1, the attacker will craft the exploit specifically to overwrite the return address of the current controller function to redirect control flow to an existing safety-critical function in the program that causes the vehicle to turn left. By continuously redirecting control flow back to this function, the vehicle will move into a state of continuously turning left in circles. The goal of the attacker is to put the vehicle in this state with the hope of causing a crash into a wall, or by approaching vehicles from behind.

actuation output. This time difference represents the amount of time taken for computation by the controller. We repeat this process for 1000 iterations of the controller with varying inputs to identify an average execution time for the controller process. By measuring the average execution times for the CPS controller without our architecture, and with our architecture, we can have a relative comparison of the overhead that our architecture presents.

When observing Figures 5, and 6, the overhead created with both ISR and ASR enabled is minimal enough to maintain execution times under the respective real time deadlines. For example, when looking at the low complexity controller (safe controller) execution times, overhead is about 10.2%, bringing

Integrated moving target defense and control reconfiguration fo: 8 - 9 (0)

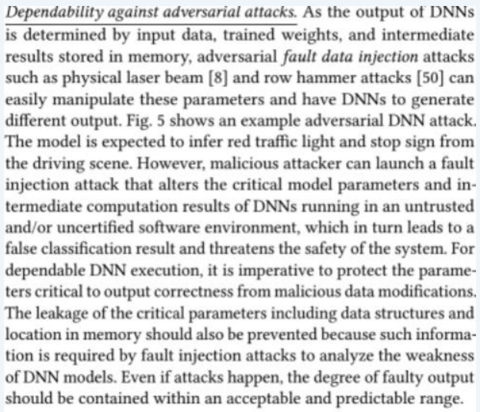


distance for each car behir

Attack
Reduced Headway Attack
Joining Without Radar
Mis-report Attack
Collision Induction Attack
Non-Attack Abnormalities

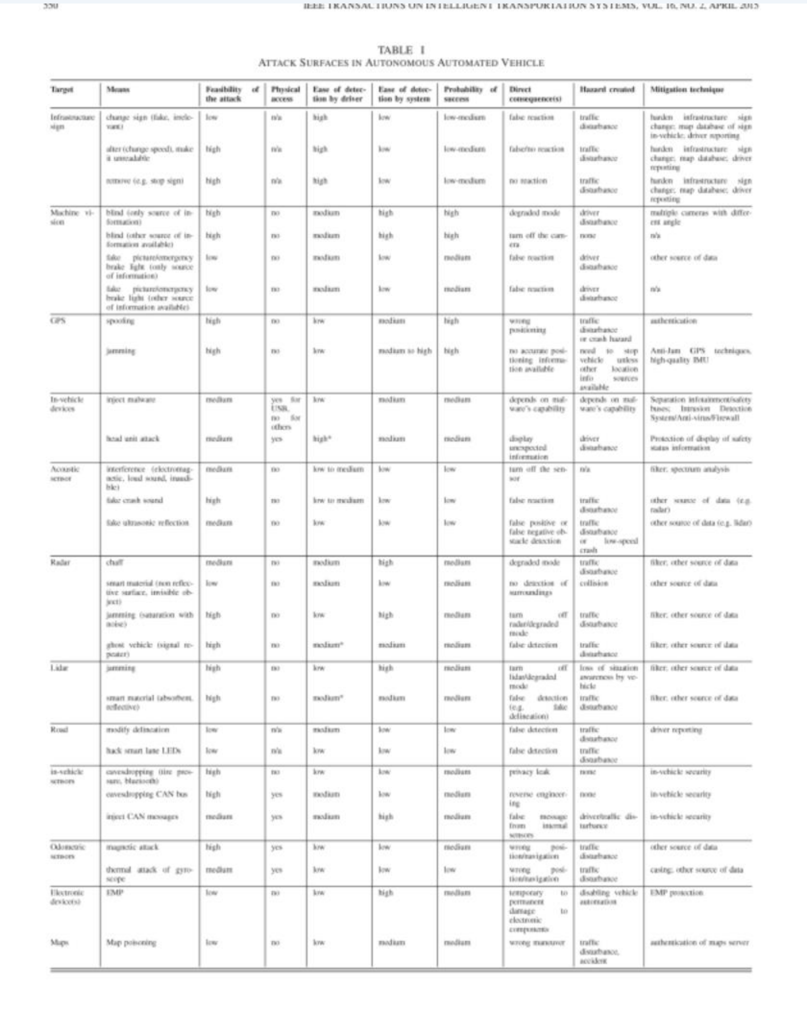
Table 1. In this table

Is your commute driving you crazy a study of misbehavior in veh: 5: 26|393 - 5: 176|547 (0)



Dependability against adversarial attacks. As the output of DNNs is determined by input data, trained weights, and intermediate results stored in memory, adversarial *fault data injection* attacks such as physical laser beam [8] and row hammer attacks [50] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack. The model is expected to infer red traffic light and stop sign from the driving scene. However, malicious attacker can launch a fault injection attack that alters the critical model parameters and intermediate computation results of DNNs running in an untrusted and/or uncertified software environment, which in turn leads to a false classification result and threatens the safety of the system. For dependable DNN execution, it is imperative to protect the parameters critical to output correctness from malicious data modifications. The leakage of the critical parameters including data structures and location in memory should also be prevented because such information is required by fault injection attacks to analyze the weakness of DNN models. Even if attacks happen, the degree of faulty output should be contained within an acceptable and predictable range.

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)

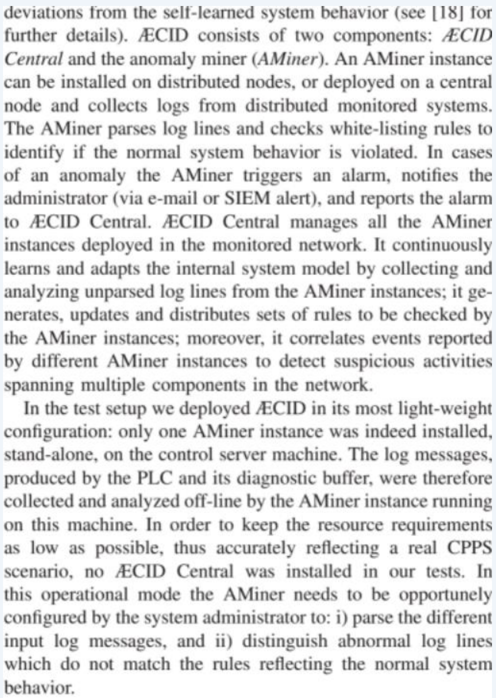


550 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong manouver	traffic disturbance, accident	authentication of maps server

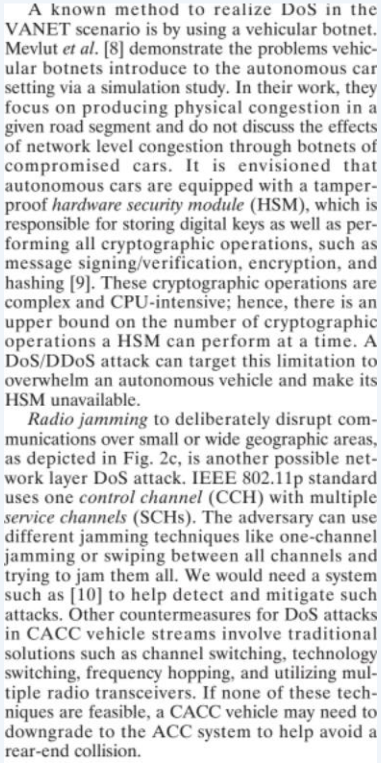
Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)



deviations from the self-learned system behavior (see [18] for further details). ÆCID consists of two components: *ÆCID Central* and the anomaly miner (*AMiner*). An AMiner instance can be installed on distributed nodes, or deployed on a central node and collects logs from distributed monitored systems. The AMiner parses log lines and checks white-listing rules to identify if the normal system behavior is violated. In cases of an anomaly the AMiner triggers an alarm, notifies the administrator (via e-mail or SIEM alert), and reports the alarm to ÆCID Central. ÆCID Central manages all the AMiner instances deployed in the monitored network. It continuously learns and adapts the internal system model by collecting and analyzing unparsed log lines from the AMiner instances; it generates, updates and distributes sets of rules to be checked by the AMiner instances; moreover, it correlates events reported by different AMiner instances to detect suspicious activities spanning multiple components in the network.

In the test setup we deployed ÆCID in its most light-weight configuration: only one AMiner instance was indeed installed, stand-alone, on the control server machine. The log messages, produced by the PLC and its diagnostic buffer, were therefore collected and analyzed off-line by the AMiner instance running on this machine. In order to keep the resource requirements as low as possible, thus accurately reflecting a real CPPS scenario, no ÆCID Central was installed in our tests. In this operational mode the AMiner needs to be opportunely configured by the system administrator to: i) parse the different input log messages, and ii) distinguish abnormal log lines which do not match the rules reflecting the normal system behavior.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



A known method to realize DoS in the VANET scenario is by using a vehicular botnet. Mevlut *et al*. [8] demonstrate the problems vehicular botnets introduce to the autonomous car setting via a simulation study. In their work, they focus on producing physical congestion in a given road segment and do not discuss the effects of network level congestion through botnets of compromised cars. It is envisioned that autonomous cars are equipped with a tamper-proof *hardware security module* (HSM), which is responsible for storing digital keys as well as performing all cryptographic operations, such as message signing/verification, encryption, and hashing [9]. These cryptographic operations are complex and CPU-intensive; hence, there is an upper bound on the number of cryptographic operations a HSM can perform at a time. A DoS/DDoS attack can target this limitation to overwhelm an autonomous vehicle and make its HSM unavailable.

Radio jamming to deliberately disrupt communications over small or wide geographic areas, as depicted in Fig. 2c, is another possible network layer DoS attack. IEEE 802.11p standard uses one *control channel* (CCH) with multiple *service channels* (SCHs). The adversary can use different jamming techniques like one-channel jamming or swiping between all channels and trying to jam them all. We would need a system such as [10] to help detect and mitigate such attacks. Other countermeasures for DoS attacks in CACC vehicle streams involve traditional solutions such as channel switching, technology switching, frequency hopping, and utilizing multiple radio transceivers. If none of these techniques are feasible, a CACC vehicle may need to downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)

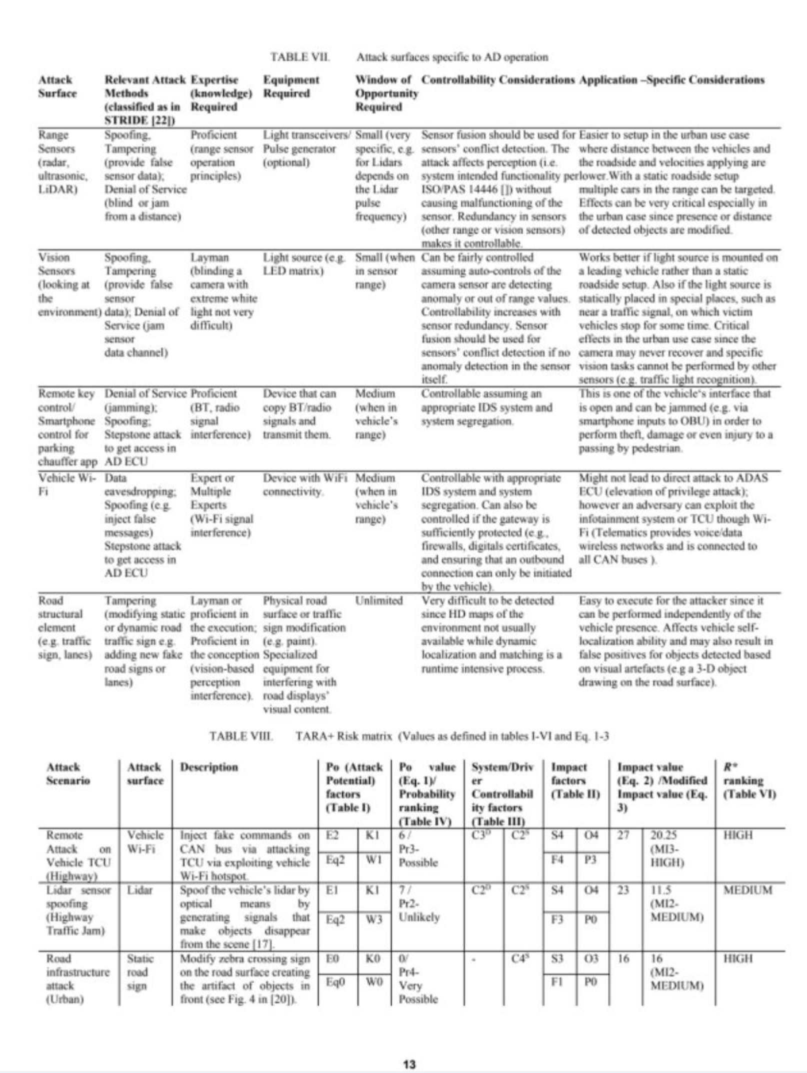


TABLE VII. Attack surfaces specific to AD operation

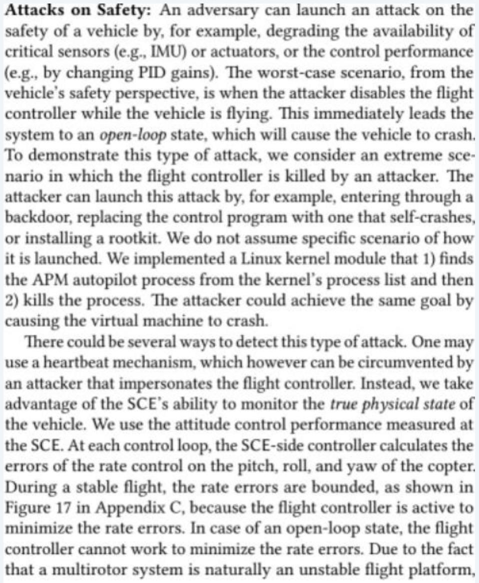
Attack Surface	Relevant Attack Methods (classified as in STRIDE [22])	Expertise (knowledge) Required	Equipment Required	Window of Opportunity Required	Controllability Considerations	Application –Specific Considerations
Range Sensors (radar, ultrasonic, LiDAR)	Spoofing, Tampering (provide false sensor data); Denial of Service (blind or jam from a distance)	Proficient (range sensor operation principles)	Light transceivers/ Pulse generator (optional)	Small (very specific, e.g. for Lidars depends on the Lidar pulse frequency)	Sensor fusion should be used for sensors' conflict detection. The attack affects perception (i.e. system intended functionality per ISO/PAS 14446 []) without causing malfunctioning of the sensor. Redundancy in sensors (other range or vision sensors) makes it controllable.	Easier to setup in the urban use case where distance between the vehicles and the roadside and velocities applying are lower.With a static roadside setup multiple cars in the range can be targeted. Effects can be very critical especially in the urban case since presence or distance of detected objects are modified.
Vision Sensors (looking at the environment)	Spoofing, Tampering (provide false sensor data); Denial of Service (jam sensor data channel)	Layman (blinding a camera with extreme white light not very difficult)	Light source (e.g. LED matrix)	Small (when in sensor range)	Can be fairly controlled assuming auto-controls of the camera sensor are detecting anomaly or out of range values. Controllability increases with sensor redundancy. Sensor fusion should be used for sensors' conflict detection if no anomaly detection in the sensor itself.	Works better if light source is mounted on a leading vehicle rather than a static roadside setup. Also if the light source is statically placed in special places, such as near a traffic signal, on which victim vehicles stop for some time. Critical effects in the urban use case since the camera may never recover and specific vision tasks cannot be performed by other sensors (e.g. traffic light recognition).
Remote key control/ Smartphone control for parking chauffer app	Denial of Service (jamming); Spoofing; Stepstone attack to get access in AD ECU	Proficient (BT, radio signal interference)	Device that can copy BT/radio signals and transmit them.	Medium (when in vehicle's range)	Controllable assuming an appropriate IDS system and system segregation.	This is one of the vehicle's interface that is open and can be jammed (e.g. via smartphone inputs to OBU) in order to perform theft, damage or even injury to a passing by pedestrian.
Vehicle Wi-Fi	Data eavesdropping; Spoofing (e.g. inject false messages) Stepstone attack to get access in AD ECU	Expert or Multiple Experts (Wi-Fi signal interference)	Device with WiFi connectivity.	Medium (when in vehicle's range)	Controllable with appropriate IDS system and system segregation. Can also be controlled if the gateway is sufficiently protected (e.g., firewalls, digitals certificates, and ensuring that an outbound connection can only be initiated by the vehicle).	Might not lead to direct attack to ADAS ECU (elevation of privilege attack); however an adversary can exploit the infotainment system or TCU though Wi-Fi (Telematics provides voice/data wireless networks and is connected to all CAN buses).
Road structural element (e.g. traffic sign, lanes)	Tampering (modifying static or dynamic road traffic sign e.g. adding new fake road signs or lanes)	Layman or proficient in the execution; Proficient in the conception (vision-based perception interference).	Physical road surface or traffic sign modification (e.g. paint). Specialized equipment for interfering with road displays' visual content.	Unlimited	Very difficult to be detected since HD maps of the environment not usually available while dynamic localization and matching is a runtime intensive process.	Easy to execute for the attacker since it can be performed independently of the vehicle presence. Affects vehicle self-localization ability and may also result in false positives for objects detected based on visual artefacts (e.g a 3-D object drawing on the road surface).

TABLE VIII. TARA+ Risk matrix (Values as defined in tables I-VI and Eq. 1-3

Attack Scenario	Attack surface	Description	Po (Attack Potential) factors (Table I)		Po value (Eq. 1)/ Probability ranking (Table IV)	System/Driver Controllability factors (Table III)		Impact factors (Table II)		Impact value (Eq. 2) /Modified Impact value (Eq. 3)		*R** ranking (Table VI)
Remote Attack on Vehicle TCU (Highway)	Vehicle Wi-Fi	Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.	E2 Eq2	K1 W1	6 / Pr3-Possible	$C3^{D}$	$C2^{S}$	S4 F4	O4 P3	27	20.25 (MI3-HIGH)	HIGH
Lidar sensor spoofing (Highway Traffic Jam)	Lidar	Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].	E1 Eq2	K1 W3	7 / Pr2-Unlikely	$C2^{D}$	$C2^{S}$	S4 F3	O4 P0	23	11.5 (MI2-MEDIUM)	MEDIUM
Road infrastructure attack (Urban)	Static road sign	Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).	E0 Eq0	K0 W0	0/ Pr4-Very Possible	-	$C4^{S}$	S3 F1	O3 P0	16	16 (MI2-MEDIUM)	HIGH

13

TARA Controllability-aware Threat Analysis and Risk Assessment: 6: 28|38 - 6: 576|763 (0)

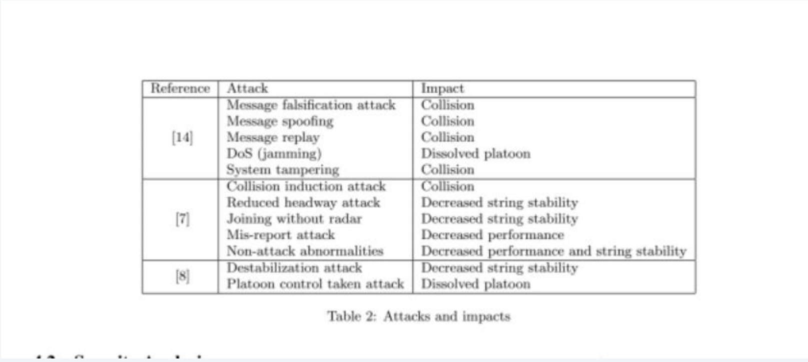


Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the vehicle's safety perspective, is when the attacker disables the flight controller while the vehicle is flying. This immediately leads the system to an *open-loop* state, which will cause the vehicle to crash. To demonstrate this type of attack, we consider an extreme scenario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how it is launched. We implemented a Linux kernel module that 1) finds the APM autopilot process from the kernel's process list and then 2) kills the process. The attacker could achieve the same goal by causing the virtual machine to crash.

There could be several ways to detect this type of attack. One may use a heartbeat mechanism, which however can be circumvented by an attacker that impersonates the flight controller. Instead, we take advantage of the SCE's ability to monitor the *true physical state* of the vehicle. We use the attitude control performance measured at the SCE. At each control loop, the SCE-side controller calculates the errors of the rate control on the pitch, roll, and yaw of the copter. During a stable flight, the rate errors are bounded, as shown in Figure 17 in Appendix C, because the flight controller is active to minimize the rate errors. In case of an open-loop state, the flight controller cannot work to minimize the rate errors. Due to the fact that a multirotor system is naturally an unstable flight platform,

VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

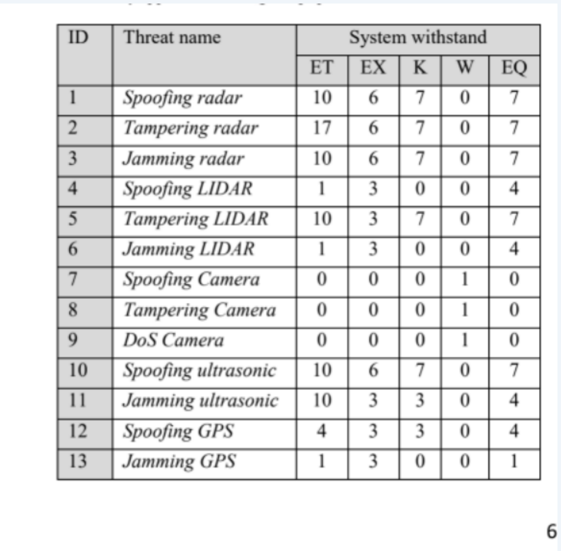
Integration\Security Attacks\Attack Mechanisms\Engage in Deceptive Interactions



Reference	Attack	Impact
[14]	Message falsification attack	Collision
	Message spoofing	Collision
	Message replay	Collision
	DoS (jamming)	Dissolved platoon
	System tampering	Collision
[7]	Collision induction attack	Collision
	Reduced headway attack	Decreased string stability
	Joining without radar	Decreased string stability
	Mis-report attack	Decreased performance
	Non-attack abnormalities	Decreased performance and string stability
[8]	Destabilization attack	Decreased string stability
	Platoon control taken attack	Dissolved platoon

Table 2: Attacks and impacts

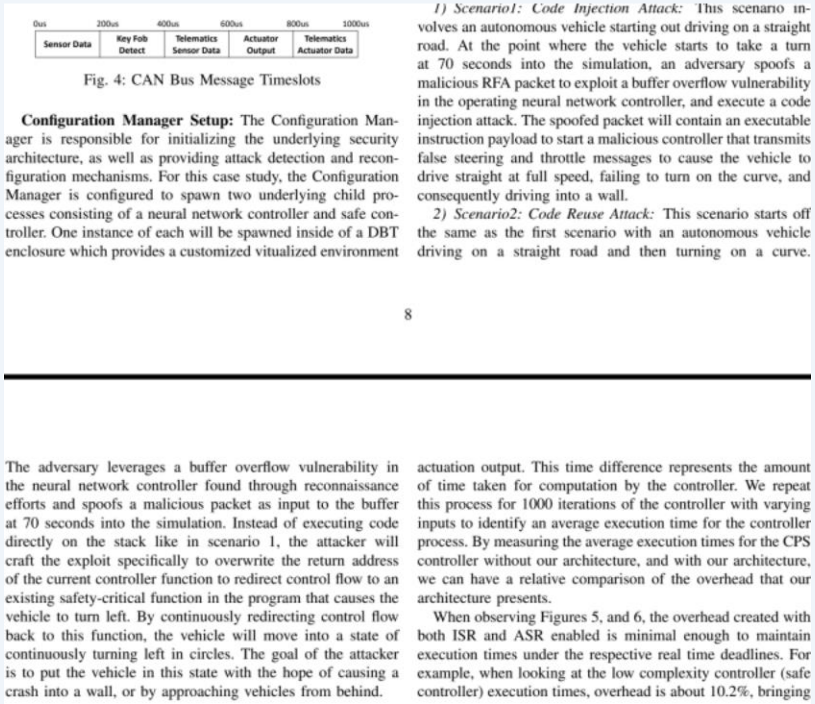
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 33|561 - 5: 558|790 (0)



ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

6

A simplified approach for dynamic security risk management in c: 7: 16|48 - 7: 302|325 (0)



0us 200us 400us 600us 800us 1000us

Sensor Data	Key Fob Detect	Telematics Sensor Data	Actuator Output	Telematics Actuator Data

Fig. 4: CAN Bus Message Timeslots

Configuration Manager Setup: The Configuration Manager is responsible for initializing the underlying security architecture, as well as providing attack detection and reconfiguration mechanisms. For this case study, the Configuration Manager is configured to spawn two underlying child processes consisting of a neural network controller and safe controller. One instance of each will be spawned inside of a DBT enclosure which provides a customized vitualized environment

1) Scenario1: Code Injection Attack: This scenario involves an autonomous vehicle starting out driving on a straight road. At the point where the vehicle starts to take a turn at 70 seconds into the simulation, an adversary spoofs a malicious RFA packet to exploit a buffer overflow vulnerability in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall.

2) Scenario2: Code Reuse Attack: This scenario starts off the same as the first scenario with an autonomous vehicle driving on a straight road and then turning on a curve.

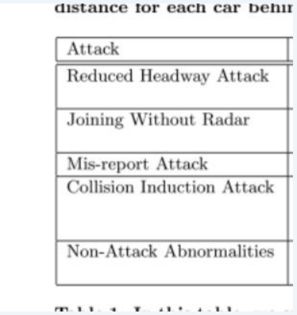
8

The adversary leverages a buffer overflow vulnerability in the neural network controller found through reconnaissance efforts and spoofs a malicious packet as input to the buffer at 70 seconds into the simulation. Instead of executing code directly on the stack like in scenario 1, the attacker will craft the exploit specifically to overwrite the return address of the current controller function to redirect control flow to an existing safety-critical function in the program that causes the vehicle to turn left. By continuously redirecting control flow back to this function, the vehicle will move into a state of continuously turning left in circles. The goal of the attacker is to put the vehicle in this state with the hope of causing a crash into a wall, or by approaching vehicles from behind.

actuation output. This time difference represents the amount of time taken for computation by the controller. We repeat this process for 1000 iterations of the controller with varying inputs to identify an average execution time for the controller process. By measuring the average execution times for the CPS controller without our architecture, and with our architecture, we can have a relative comparison of the overhead that our architecture presents.

When observing Figures 5, and 6, the overhead created with both ISR and ASR enabled is minimal enough to maintain execution times under the respective real time deadlines. For example, when looking at the low complexity controller (safe controller) execution times, overhead is about 10.2%, bringing

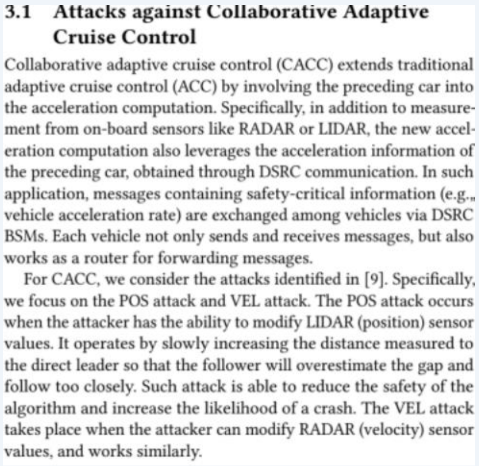
Integrated moving target defense and control reconfiguration fo: 8 - 9 (0)



distance for each car behi

Attack
Reduced Headway Attack
Joining Without Radar
Mis-report Attack
Collision Induction Attack
Non-Attack Abnormalities

Is your commute driving you crazy a study of misbehavior in veh: 5: 26|393 - 5: 176|547 (0)

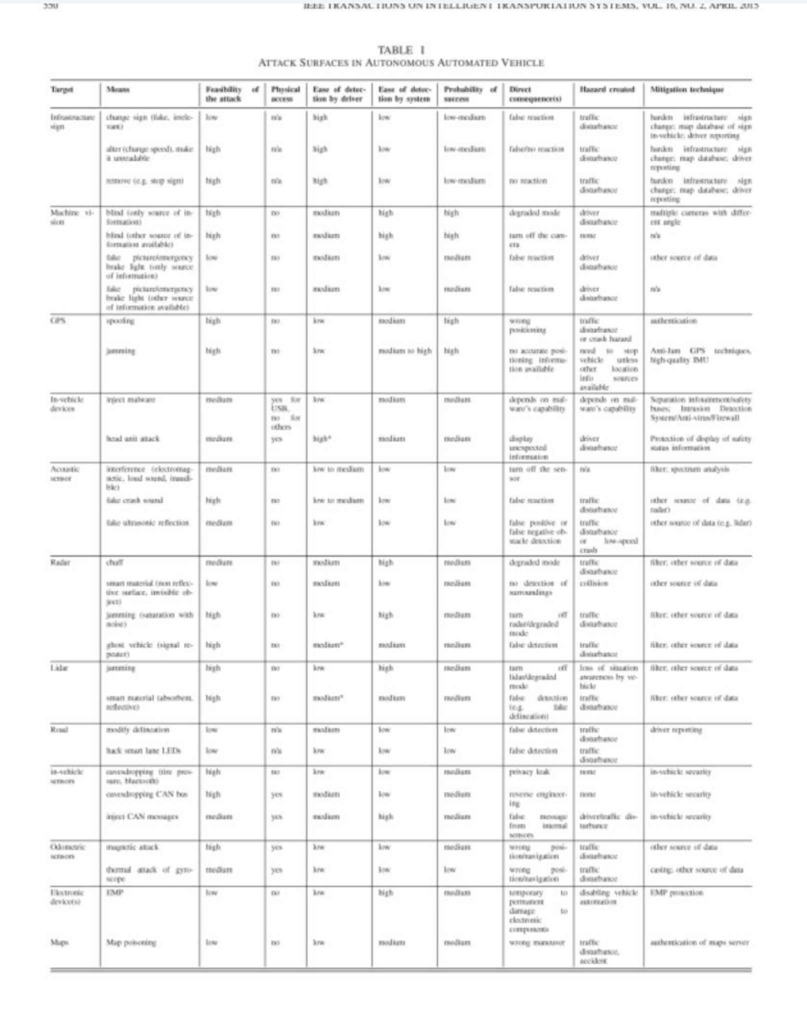


3.1 Attacks against Collaborative Adaptive Cruise Control

Collaborative adaptive cruise control (CACC) extends traditional adaptive cruise control (ACC) by involving the preceding car into the acceleration computation. Specifically, in addition to measurement from on-board sensors like RADAR or LIDAR, the new acceleration computation also leverages the acceleration information of the preceding car, obtained through DSRC communication. In such application, messages containing safety-critical information (e.g.,, vehicle acceleration rate) are exchanged among vehicles via DSRC BSMs. Each vehicle not only sends and receives messages, but also works as a router for forwarding messages.

For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)



550 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong manouver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)

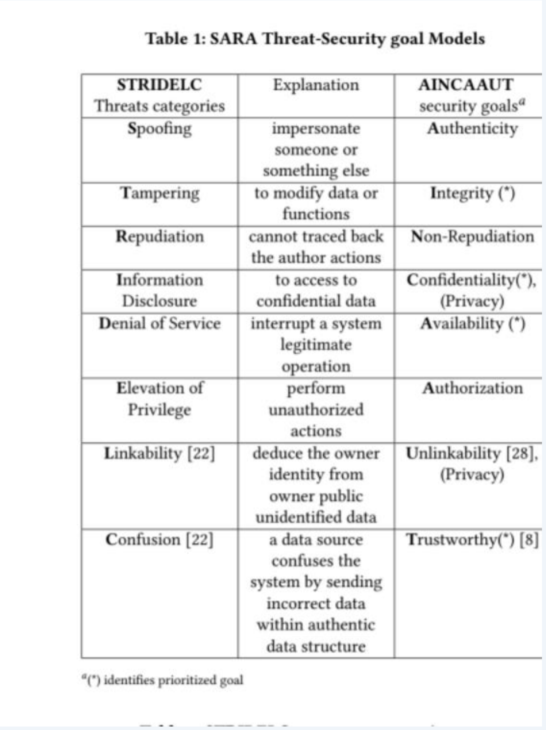
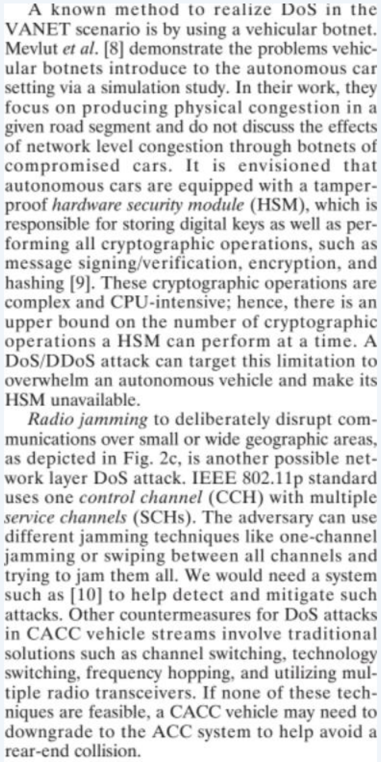


Table 1: SARA Threat-Security goal Models

STRIDELC Threats categories	Explanation	**AINCAAUT** security goals[a]
Spoofing	impersonate someone or something else	**A**uthenticity
Tampering	to modify data or functions	**I**ntegrity (*)
Repudiation	cannot traced back the author actions	**N**on-Repudiation
Information Disclosure	to access to confidential data	**C**onfidentiality(*), (Privacy)
Denial of Service	interrupt a system legitimate operation	**A**vailability (*)
Elevation of Privilege	perform unauthorized actions	**A**uthorization
Linkability [22]	deduce the owner identity from owner public unidentified data	**U**nlinkability [28], (Privacy)
Confusion [22]	a data source confuses the system by sending incorrect data within authentic data structure	**T**rustworthy(*) [8]

[a] (*) identifies prioritized goal

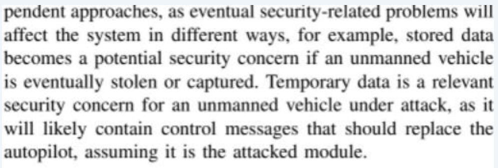
SARA Security Automotive Risk Analysis Method: 5: 12|354 - 5: 290|723 (0)



A known method to realize DoS in the VANET scenario is by using a vehicular botnet. Mevlut *et al*. [8] demonstrate the problems vehicular botnets introduce to the autonomous car setting via a simulation study. In their work, they focus on producing physical congestion in a given road segment and do not discuss the effects of network level congestion through botnets of compromised cars. It is envisioned that autonomous cars are equipped with a tamper-proof *hardware security module* (HSM), which is responsible for storing digital keys as well as performing all cryptographic operations, such as message signing/verification, encryption, and hashing [9]. These cryptographic operations are complex and CPU-intensive; hence, there is an upper bound on the number of cryptographic operations a HSM can perform at a time. A DoS/DDoS attack can target this limitation to overwhelm an autonomous vehicle and make its HSM unavailable.

Radio jamming to deliberately disrupt communications over small or wide geographic areas, as depicted in Fig. 2c, is another possible network layer DoS attack. IEEE 802.11p standard uses one *control channel* (CCH) with multiple *service channels* (SCHs). The adversary can use different jamming techniques like one-channel jamming or swiping between all channels and trying to jam them all. We would need a system such as [10] to help detect and mitigate such attacks. Other countermeasures for DoS attacks in CACC vehicle streams involve traditional solutions such as channel switching, technology switching, frequency hopping, and utilizing multiple radio transceivers. If none of these techniques are feasible, a CACC vehicle may need to downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



pendent approaches, as eventual security-related problems will affect the system in different ways, for example, stored data becomes a potential security concern if an unmanned vehicle is eventually stolen or captured. Temporary data is a relevant security concern for an unmanned vehicle under attack, as it will likely contain control messages that should replace the autopilot, assuming it is the attacked module.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)

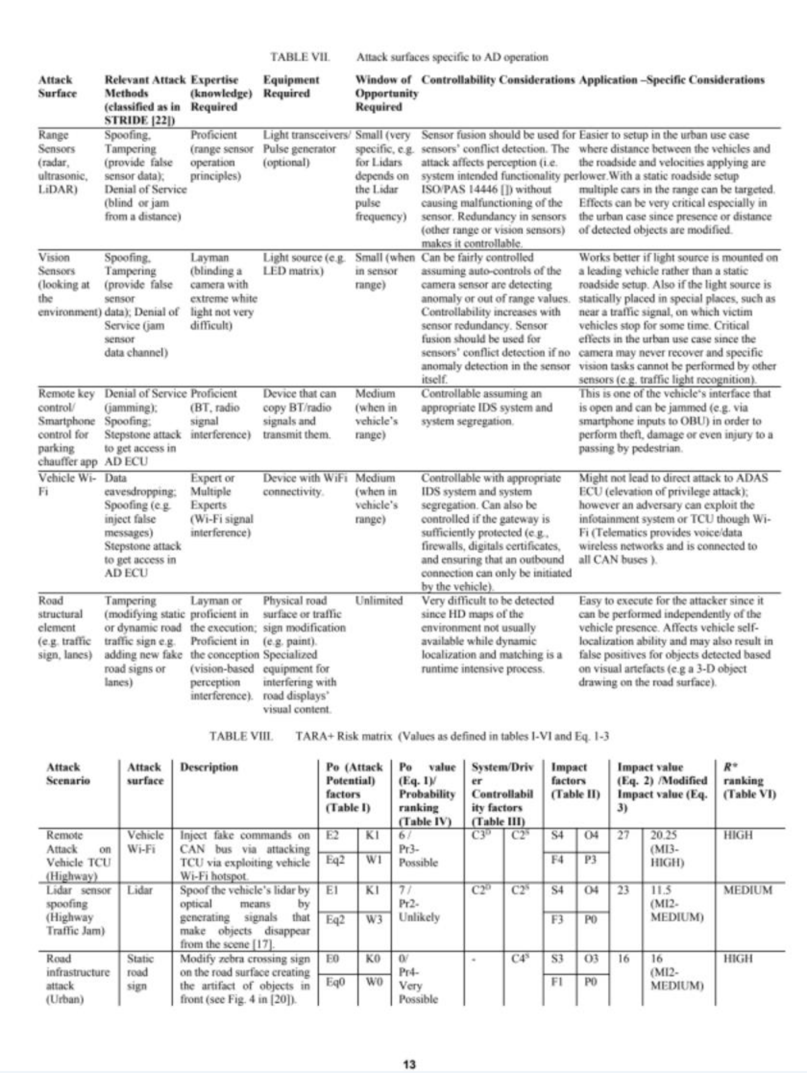


TABLE VII. Attack surfaces specific to AD operation

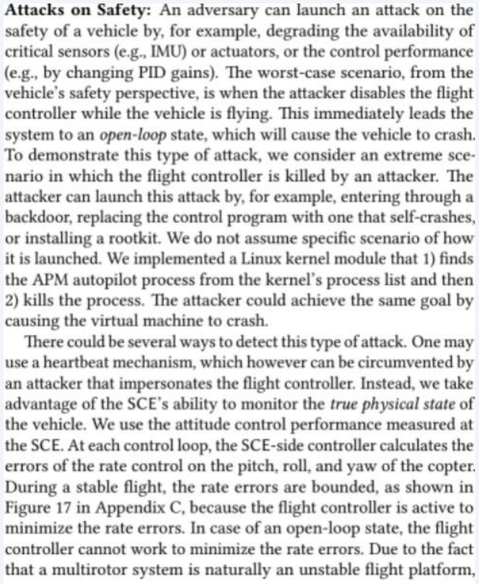
Attack Surface	Relevant Attack Methods (classified as in STRIDE [22])	Expertise (knowledge) Required	Equipment Required	Window of Opportunity Required	Controllability Considerations	Application –Specific Considerations
Range Sensors (radar, ultrasonic, LiDAR)	Spoofing, Tampering (provide false sensor data); Denial of Service (blind or jam from a distance)	Proficient (range sensor operation principles)	Light transceivers/ Pulse generator (optional)	Small (very specific, e.g. for Lidars depends on the Lidar pulse frequency)	Sensor fusion should be used for sensors' conflict detection. The attack affects perception (i.e. system intended functionality per ISO/PAS 14446 []) without causing malfunctioning of the sensor. Redundancy in sensors (other range or vision sensors) makes it controllable.	Easier to setup in the urban use case where distance between the vehicles and the roadside and velocities applying are lower.With a static roadside setup multiple cars in the range can be targeted. Effects can be very critical especially in the urban case since presence or distance of detected objects are modified.
Vision Sensors (looking at the environment)	Spoofing, Tampering (provide false sensor data); Denial of Service (jam sensor data channel)	Layman (blinding a camera with extreme white light not very difficult)	Light source (e.g. LED matrix)	Small (when in sensor range)	Can be fairly controlled assuming auto-controls of the camera sensor are detecting anomaly or out of range values. Controllability increases with sensor redundancy. Sensor fusion should be used for sensors' conflict detection if no anomaly detection in the sensor itself.	Works better if light source is mounted on a leading vehicle rather than a static roadside setup. Also if the light source is statically placed in special places, such as near a traffic signal, on which victim vehicles stop for some time. Critical effects in the urban use case since the camera may never recover and specific vision tasks cannot be performed by other sensors (e.g. traffic light recognition).
Remote key control/ Smartphone control for parking chauffer app	Denial of Service (jamming); Spoofing; Stepstone attack to get access in AD ECU	Proficient (BT, radio signal interference)	Device that can copy BT/radio signals and transmit them.	Medium (when in vehicle's range)	Controllable assuming an appropriate IDS system and system segregation.	This is one of the vehicle's interface that is open and can be jammed (e.g. via smartphone inputs to OBU) in order to perform theft, damage or even injury to a passing by pedestrian.
Vehicle Wi-Fi	Data eavesdropping; Spoofing (e.g. inject false messages) Stepstone attack to get access in AD ECU	Expert or Multiple Experts (Wi-Fi signal interference)	Device with WiFi connectivity.	Medium (when in vehicle's range)	Controllable with appropriate IDS system and system segregation. Can also be controlled if the gateway is sufficiently protected (e.g., firewalls, digitals certificates, and ensuring that an outbound connection can only be initiated by the vehicle).	Might not lead to direct attack to ADAS ECU (elevation of privilege attack); however an adversary can exploit the infotainment system or TCU though Wi-Fi (Telematics provides voice/data wireless networks and is connected to all CAN buses).
Road structural element (e.g. traffic sign, lanes)	Tampering (modifying static or dynamic road traffic sign e.g. adding new fake road signs or lanes)	Layman or proficient in the execution; Proficient in the conception (vision-based perception interference).	Physical road surface or traffic sign modification (e.g. paint). Specialized equipment for interfering with road displays' visual content.	Unlimited	Very difficult to be detected since HD maps of the environment not usually available while dynamic localization and matching is a runtime intensive process.	Easy to execute for the attacker since it can be performed independently of the vehicle presence. Affects vehicle self-localization ability and may also result in false positives for objects detected based on visual artefacts (e.g a 3-D object drawing on the road surface).

TABLE VIII. TARA+ Risk matrix (Values as defined in tables I-VI and Eq. 1-3

Attack Scenario	Attack surface	Description	Po (Attack Potential) factors (Table I)		Po value (Eq. 1)/ Probability ranking (Table IV)	System/Driver Controllability factors (Table III)		Impact factors (Table II)		Impact value (Eq. 2) /Modified Impact value (Eq. 3)		*R** ranking (Table VI)
Remote Attack on Vehicle TCU (Highway)	Vehicle Wi-Fi	Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.	E2 Eq2	K1 W1	6 / Pr3-Possible	$C3^D$	$C2^S$	S4 F4	O4 P3	27	20.25 (MI3-HIGH)	HIGH
Lidar sensor spoofing (Highway Traffic Jam)	Lidar	Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].	E1 Eq2	K1 W3	7 / Pr2-Unlikely	$C2^D$	$C2^S$	S4 F3	O4 P0	23	11.5 (MI2-MEDIUM)	MEDIUM
Road infrastructure attack (Urban)	Static road sign	Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).	E0 Eq0	K0 W0	0/ Pr4-Very Possible	-	$C4^S$	S3 F1	O3 P0	16	16 (MI2-MEDIUM)	HIGH

13

TARA Controllability-aware Threat Analysis and Risk Assessment: 6: 28|38 - 6: 576|763 (0)

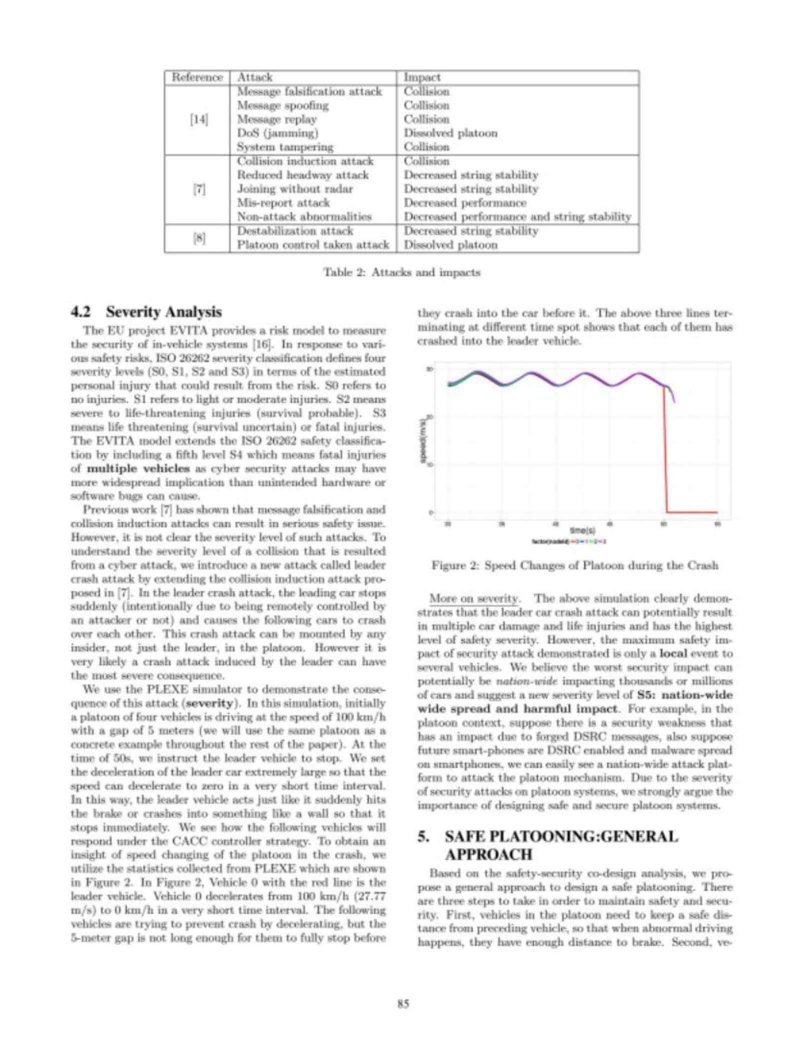


Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the vehicle's safety perspective, is when the attacker disables the flight controller while the vehicle is flying. This immediately leads the system to an *open-loop* state, which will cause the vehicle to crash. To demonstrate this type of attack, we consider an extreme scenario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how it is launched. We implemented a Linux kernel module that 1) finds the APM autopilot process from the kernel's process list and then 2) kills the process. The attacker could achieve the same goal by causing the virtual machine to crash.

There could be several ways to detect this type of attack. One may use a heartbeat mechanism, which however can be circumvented by an attacker that impersonates the flight controller. Instead, we take advantage of the SCE's ability to monitor the *true physical state* of the vehicle. We use the attitude control performance measured at the SCE. At each control loop, the SCE-side controller calculates the errors of the rate control on the pitch, roll, and yaw of the copter. During a stable flight, the rate errors are bounded, as shown in Figure 17 in Appendix C, because the flight controller is active to minimize the rate errors. In case of an open-loop state, the flight controller cannot work to minimize the rate errors. Due to the fact that a multirotor system is naturally an unstable flight platform,

VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

Integration\Security Attacks\Attack Mechanisms\Manipulate System Resources



Reference	Attack	Impact
[14]	Message falsification attack	Collision
	Message spoofing	Collision
	Message replay	Collision
	DoS (jamming)	Dissolved platoon
	System tampering	Collision
[7]	Collision induction attack	Collision
	Reduced headway attack	Decreased string stability
	Joining without radar	Decreased string stability
	Mis-report attack	Decreased performance
	Non-attack abnormalities	Decreased performance and string stability
[8]	Destabilization attack	Decreased string stability
	Platoon control taken attack	Dissolved platoon

Table 2: Attacks and impacts

4.2 Severity Analysis

The EU project EVITA provides a risk model to measure the security of in-vehicle systems [16]. In response to various safety risks, ISO 26262 severity classification defines four severity levels (S0, S1, S2 and S3) in terms of the estimated personal injury that could result from the risk. S0 refers to no injuries. S1 refers to light or moderate injuries. S2 means severe to life-threatening injuries (survival probable). S3 means life threatening (survival uncertain) or fatal injuries. The EVITA model extends the ISO 26262 safety classification by including a fifth level S4 which means fatal injuries of **multiple vehicles** as cyber security attacks may have more widespread implication than unintended hardware or software bugs can cause.

Previous work [7] has shown that message falsification and collision induction attacks can result in serious safety issue. However, it is not clear the severity level of such attacks. To understand the severity level of a collision that is resulted from a cyber attack, we introduce a new attack called leader crash attack by extending the collision induction attack proposed in [7]. In the leader crash attack, the leading car stops suddenly (intentionally due to being remotely controlled by an attacker or not) and causes the following cars to crash over each other. This crash attack can be mounted by any insider, not just the leader, in the platoon. However it is very likely a crash attack induced by the leader can have the most severe consequence.

We use the PLEXE simulator to demonstrate the consequence of this attack (**severity**). In this simulation, initially a platoon of four vehicles is driving at the speed of 100 km/h with a gap of 5 meters (we will use the same platoon as a concrete example throughout the rest of the paper). At the time of 50s, we instruct the leader vehicle to stop. We set the deceleration of the leader car extremely large so that the speed can decelerate to zero in a very short time interval. In this way, the leader vehicle acts just like it suddenly hits the brake or crashes into something like a wall so that it stops immediately. We see how the following vehicles will respond under the CACC controller strategy. To obtain an insight of speed changing of the platoon in the crash, we utilize the statistics collected from PLEXE which are shown in Figure 2. In Figure 2, Vehicle 0 with the red line is the leader vehicle. Vehicle 0 decelerates from 100 km/h (27.77 m/s) to 0 km/h in a very short time interval. The following vehicles are trying to prevent crash by decelerating, but the 5-meter gap is not long enough for them to fully stop before they crash into the car before it. The above three lines terminating at different time spot shows that each of them has crashed into the leader vehicle.

Figure 2: Speed Changes of Platoon during the Crash

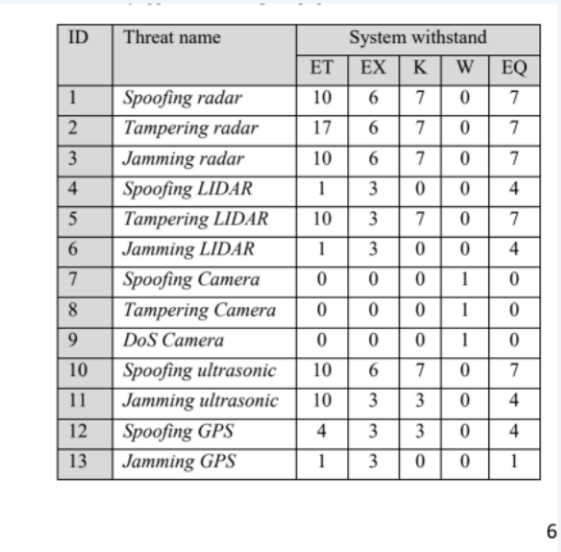
More on severity. The above simulation clearly demonstrates that the leader car crash attack can potentially result in multiple car damage and life injuries and has the highest level of safety severity. However, the maximum safety impact of security attack demonstrated is only a **local** event to several vehicles. We believe the worst security impact can potentially be *nation-wide* impacting thousands or millions of cars and suggest a new severity level of **S5: nation-wide wide spread and harmful impact**. For example, in the platoon context, suppose there is a security weakness that has an impact due to forged DSRC messages, also suppose future smart-phones are DSRC enabled and malware spread on smartphones, we can easily see a nation-wide attack platform to attack the platoon mechanism. Due to the severity of security attacks on platoon systems, we strongly argue the importance of designing safe and secure platoon systems.

5. SAFE PLATOONING:GENERAL APPROACH

Based on the safety-security co-design analysis, we propose a general approach to design a safe platooning. There are three steps to take in order to maintain safety and security. First, vehicles in the platoon need to keep a safe distance from preceding vehicle, so that when abnormal driving happens, they have enough distance to brake. Second, ve-

85

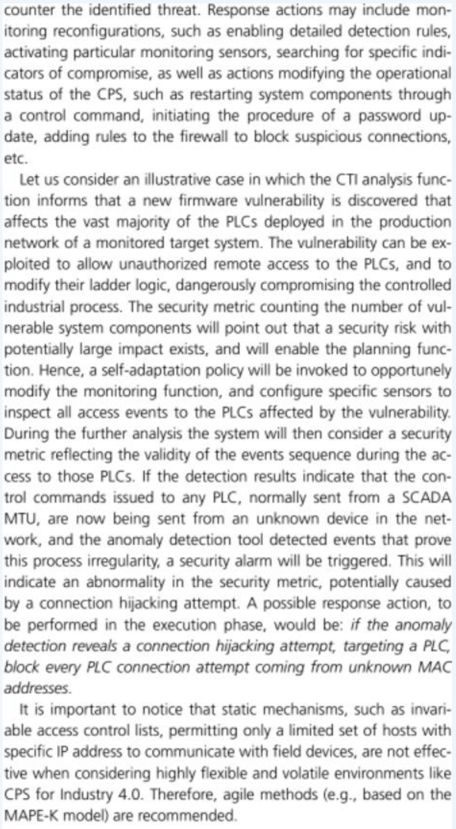
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 301|2562 - 5: 497|2745 (0)



ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

6

A simplified approach for dynamic security risk management in c: 7: 16|48 - 7: 302|325 (0)

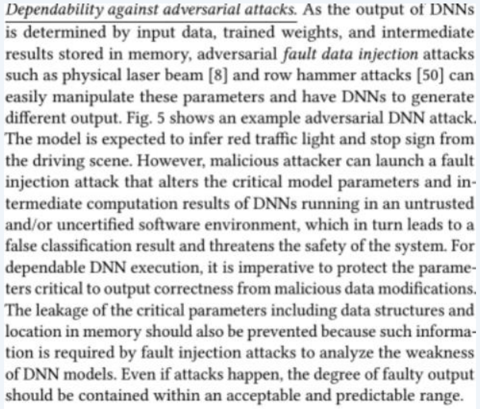


counter the identified threat. Response actions may include monitoring reconfigurations, such as enabling detailed detection rules, activating particular monitoring sensors, searching for specific indicators of compromise, as well as actions modifying the operational status of the CPS, such as restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

Let us consider an illustrative case in which the CTI analysis function informs that a new firmware vulnerability is discovered that affects the vast majority of the PLCs deployed in the production network of a monitored target system. The vulnerability can be exploited to allow unauthorized remote access to the PLCs, and to modify their ladder logic, dangerously compromising the controlled industrial process. The security metric counting the number of vulnerable system components will point out that a security risk with potentially large impact exists, and will enable the planning function. Hence, a self-adaptation policy will be invoked to opportunely modify the monitoring function, and configure specific sensors to inspect all access events to the PLCs affected by the vulnerability. During the further analysis the system will then consider a security metric reflecting the validity of the events sequence during the access to those PLCs. If the detection results indicate that the control commands issued to any PLC, normally sent from a SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tool detected events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric, potentially caused by a connection hijacking attempt. A possible response action, to be performed in the execution phase, would be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses.*

It is important to notice that static mechanisms, such as invariable access control lists, permitting only a limited set of hosts with specific IP address to communicate with field devices, are not effective when considering highly flexible and volatile environments like CPS for Industry 4.0. Therefore, agile methods (e.g., based on the MAPE-K model) are recommended.

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



Dependability against adversarial attacks. As the output of DNNs is determined by input data, trained weights, and intermediate results stored in memory, adversarial *fault data injection* attacks such as physical laser beam [8] and row hammer attacks [50] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack. The model is expected to infer red traffic light and stop sign from the driving scene. However, malicious attacker can launch a fault injection attack that alters the critical model parameters and intermediate computation results of DNNs running in an untrusted and/or uncertified software environment, which in turn leads to a false classification result and threatens the safety of the system. For dependable DNN execution, it is imperative to protect the parameters critical to output correctness from malicious data modifications. The leakage of the critical parameters including data structures and location in memory should also be prevented because such information is required by fault injection attacks to analyze the weakness of DNN models. Even if attacks happen, the degree of faulty output should be contained within an acceptable and predictable range.

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)

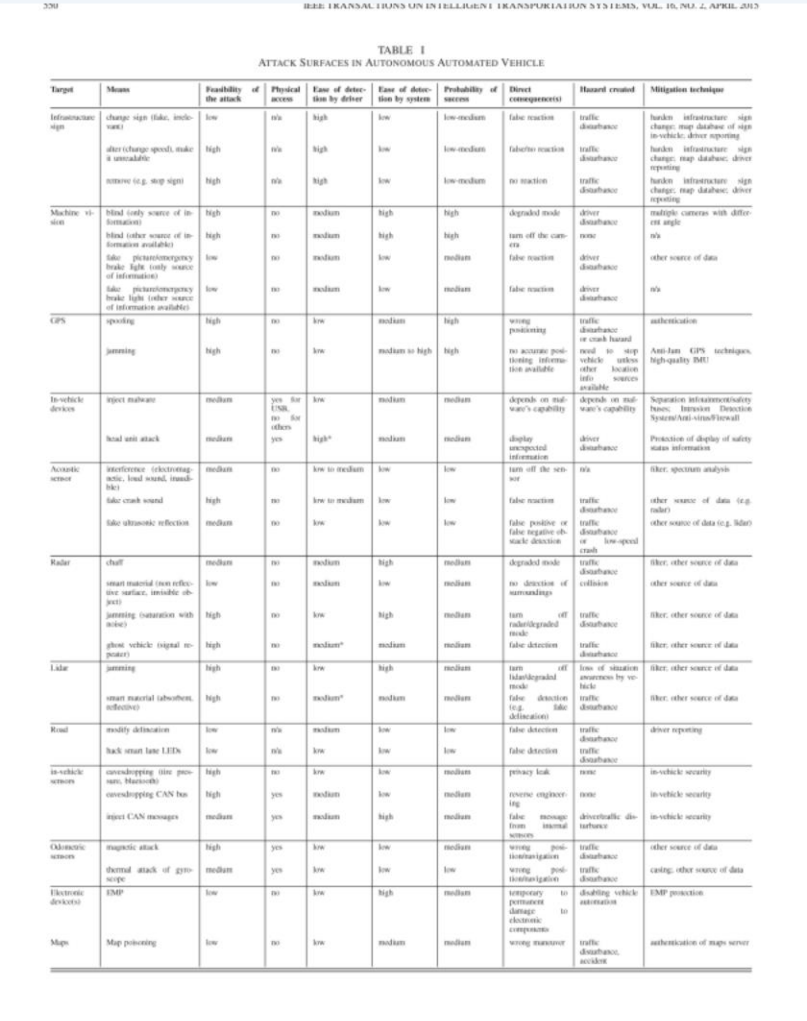
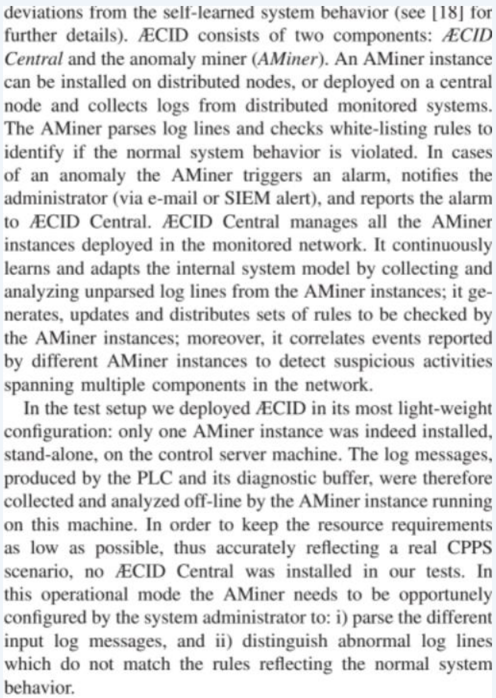


TABLE I
ATTACK SURFACES IN AUTONOMOUS AUTOMATED VEHICLE

Target	Means	Feasibility of the attack	Physical access	Ease of detection by driver	Ease of detection by system	Probability of success	Direct consequence(s)	Hazard created	Mitigation technique
Infrastructure sign	change sign (fake, irrelevant)	low	n/a	high	low	low-medium	false reaction	traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
	alter (change speed), make it unreadable	high	n/a	high	low	low-medium	false/no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
	remove (e.g. stop sign)	high	n/a	high	low	low-medium	no reaction	traffic disturbance	harden infrastructure sign change; map database; driver reporting
Machine vision	blind (only source of information)	high	no	medium	high	high	degraded mode	driver disturbance	multiple cameras with different angle
	blind (other source of information available)	high	no	medium	high	high	turn off the camera	none	n/a
	fake picture/emergency brake light (only source of information)	low	no	medium	low	medium	false reaction	driver disturbance	other source of data
	fake picture/emergency brake light (other source of information available)	low	no	medium	low	medium	false reaction	driver disturbance	n/a
GPS	spoofing	high	no	low	medium	high	wrong positioning	traffic disturbance or crash hazard	authentication
	jamming	high	no	low	medium to high	high	no accurate positioning information available	need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
In-vehicle devices	inject malware	medium	yes for USB, no for others	low	medium	medium	depends on malware's capability	depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
	head unit attack	medium	yes	high*	medium	medium	display unexpected information	driver disturbance	Protection of display of safety status information
Acoustic sensor	interference (electromagnetic, loud sound, inaudible)	medium	no	low to medium	low	low	turn off the sensor	n/a	filter; spectrum analysis
	fake crash sound	high	no	low to medium	low	low	false reaction	traffic disturbance	other source of data (e.g. radar)
	fake ultrasonic reflection	medium	no	low	low	low	false positive or false negative obstacle detection	traffic disturbance or low-speed crash	other source of data (e.g. lidar)
Radar	chaff	medium	no	medium	high	medium	degraded mode	traffic disturbance	filter; other source of data
	smart material (non reflective surface, invisible object)	low	no	medium	low	medium	no detection of surroundings	collision	other source of data
	jamming (saturation with noise)	high	no	low	high	medium	turn off radar/degraded mode	traffic disturbance	filter; other source of data
	ghost vehicle (signal repeater)	high	no	medium*	medium	medium	false detection	traffic disturbance	filter; other source of data
Lidar	jamming	high	no	low	high	medium	turn off lidar/degraded mode	loss of situation awareness by vehicle	filter; other source of data
	smart material (absorbent, reflective)	high	no	medium*	medium	medium	false detection (e.g. fake delineation)	traffic disturbance	filter; other source of data
Road	modify delineation	low	n/a	medium	low	low	false detection	traffic disturbance	driver reporting
	hack smart lane LEDs	low	n/a	low	low	low	false detection	traffic disturbance	
in-vehicle sensors	eavesdropping (tire pressure, bluetooth)	high	no	low	low	medium	privacy leak	none	in-vehicle security
	eavesdropping CAN bus	high	yes	medium	low	medium	reverse engineering	none	in-vehicle security
	inject CAN messages	medium	yes	medium	high	medium	false message from internal sensors	driver/traffic disturbance	in-vehicle security
Odometric sensors	magnetic attack	high	yes	low	low	medium	wrong position/navigation	traffic disturbance	other source of data
	thermal attack of gyroscope	medium	yes	low	low	low	wrong position/navigation	traffic disturbance	casing; other source of data
Electronic device(s)	EMP	low	no	low	high	medium	temporary to permanent damage to electronic components	disabling vehicle automation	EMP protection
Maps	Map poisoning	low	no	low	medium	medium	wrong maneuver	traffic disturbance, accident	authentication of maps server

Potential cyberattacks on automated vehicles: 5: 11|25 - 5: 591|753 (0)



deviations from the self-learned system behavior (see [18] for further details). ÆCID consists of two components: *ÆCID Central* and the anomaly miner (*AMiner*). An AMiner instance can be installed on distributed nodes, or deployed on a central node and collects logs from distributed monitored systems. The AMiner parses log lines and checks white-listing rules to identify if the normal system behavior is violated. In cases of an anomaly the AMiner triggers an alarm, notifies the administrator (via e-mail or SIEM alert), and reports the alarm to ÆCID Central. ÆCID Central manages all the AMiner instances deployed in the monitored network. It continuously learns and adapts the internal system model by collecting and analyzing unparsed log lines from the AMiner instances; it generates, updates and distributes sets of rules to be checked by the AMiner instances; moreover, it correlates events reported by different AMiner instances to detect suspicious activities spanning multiple components in the network.

In the test setup we deployed ÆCID in its most light-weight configuration: only one AMiner instance was indeed installed, stand-alone, on the control server machine. The log messages, produced by the PLC and its diagnostic buffer, were therefore collected and analyzed off-line by the AMiner instance running on this machine. In order to keep the resource requirements as low as possible, thus accurately reflecting a real CPPS scenario, no ÆCID Central was installed in our tests. In this operational mode the AMiner needs to be opportunely configured by the system administrator to: i) parse the different input log messages, and ii) distinguish abnormal log lines which do not match the rules reflecting the normal system behavior.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)

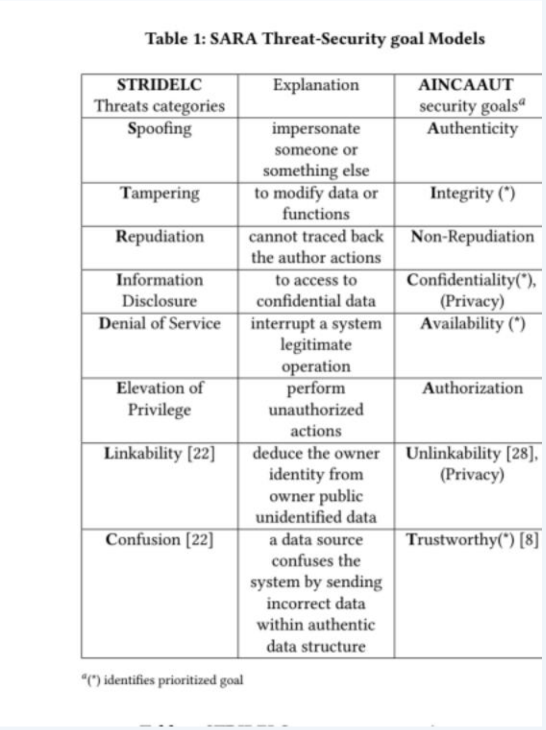
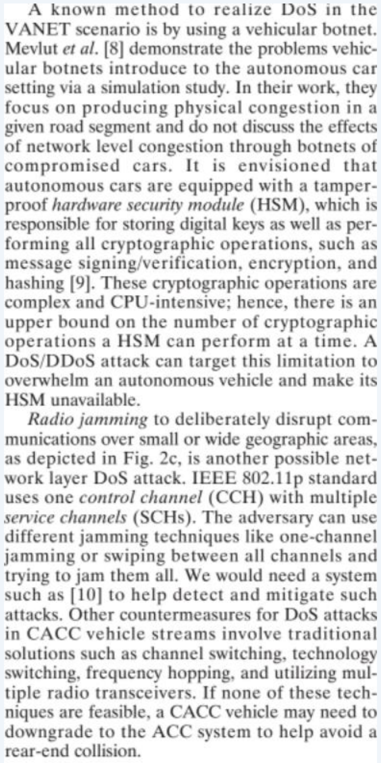


Table 1: SARA Threat-Security goal Models

STRIDELC Threats categories	Explanation	**AINCAAUT** security goals[a]
Spoofing	impersonate someone or something else	**A**uthenticity
Tampering	to modify data or functions	**I**ntegrity (*)
Repudiation	cannot traced back the author actions	**N**on-Repudiation
Information Disclosure	to access to confidential data	**C**onfidentiality(*), (Privacy)
Denial of Service	interrupt a system legitimate operation	**A**vailability (*)
Elevation of Privilege	perform unauthorized actions	**A**uthorization
Linkability [22]	deduce the owner identity from owner public unidentified data	**U**nlinkability [28], (Privacy)
Confusion [22]	a data source confuses the system by sending incorrect data within authentic data structure	**T**rustworthy(*) [8]

[a](*) identifies prioritized goal

SARA Security Automotive Risk Analysis Method: 5: 12|354 - 5: 290|723 (0)



A known method to realize DoS in the VANET scenario is by using a vehicular botnet. Mevlut *et al*. [8] demonstrate the problems vehicular botnets introduce to the autonomous car setting via a simulation study. In their work, they focus on producing physical congestion in a given road segment and do not discuss the effects of network level congestion through botnets of compromised cars. It is envisioned that autonomous cars are equipped with a tamperproof *hardware security module* (HSM), which is responsible for storing digital keys as well as performing all cryptographic operations, such as message signing/verification, encryption, and hashing [9]. These cryptographic operations are complex and CPU-intensive; hence, there is an upper bound on the number of cryptographic operations a HSM can perform at a time. A DoS/DDoS attack can target this limitation to overwhelm an autonomous vehicle and make its HSM unavailable.

Radio jamming to deliberately disrupt communications over small or wide geographic areas, as depicted in Fig. 2c, is another possible network layer DoS attack. IEEE 802.11p standard uses one *control channel* (CCH) with multiple *service channels* (SCHs). The adversary can use different jamming techniques like one-channel jamming or swiping between all channels and trying to jam them all. We would need a system such as [10] to help detect and mitigate such attacks. Other countermeasures for DoS attacks in CACC vehicle streams involve traditional solutions such as channel switching, technology switching, frequency hopping, and utilizing multiple radio transceivers. If none of these techniques are feasible, a CACC vehicle may need to downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)

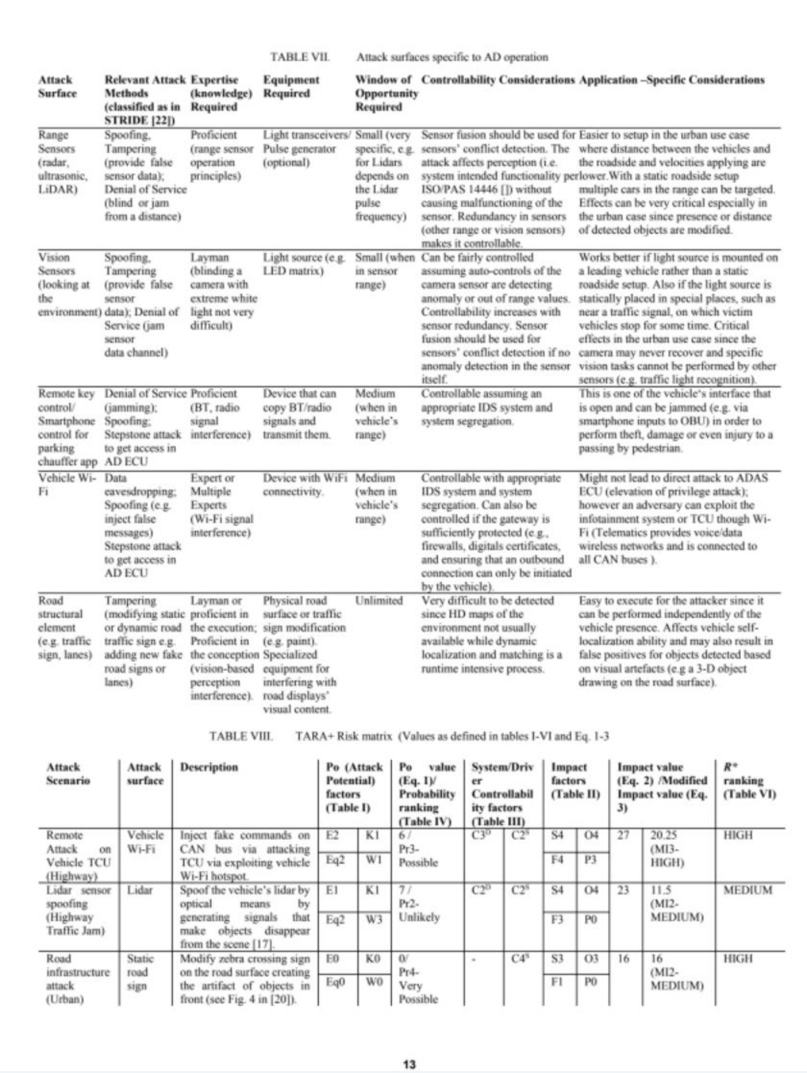


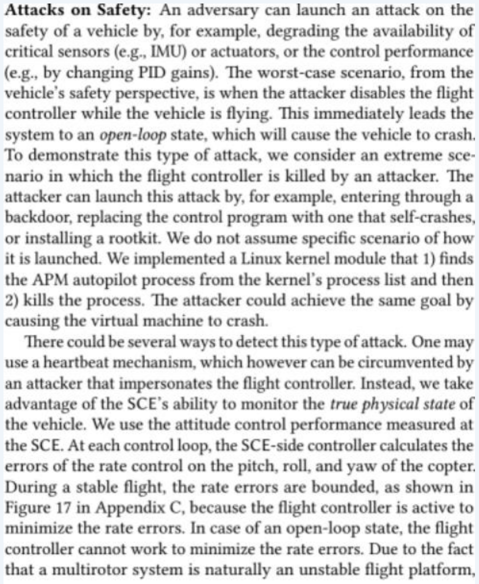
TABLE VII. Attack surfaces specific to AD operation

Attack Surface	Relevant Attack Methods (classified as in STRIDE [22])	Expertise (knowledge) Required	Equipment Required	Window of Opportunity Required	Controllability Considerations	Application –Specific Considerations
Range Sensors (radar, ultrasonic, LiDAR)	Spoofing, Tampering (provide false sensor data); Denial of Service (blind or jam from a distance)	Proficient (range sensor operation principles)	Light transceivers/ Pulse generator (optional)	Small (very specific, e.g. for Lidars depends on the Lidar pulse frequency)	Sensor fusion should be used for sensors' conflict detection. The attack affects perception (i.e. system intended functionality per ISO/PAS 14446 []) without causing malfunctioning of the sensor. Redundancy in sensors (other range or vision sensors) makes it controllable.	Easier to setup in the urban use case where distance between the vehicles and the roadside and velocities applying are lower.With a static roadside setup multiple cars in the range can be targeted. Effects can be very critical especially in the urban case since presence or distance of detected objects are modified.
Vision Sensors (looking at the environment)	Spoofing, Tampering (provide false sensor data); Denial of Service (jam sensor data channel)	Layman (blinding a camera with extreme white light not very difficult)	Light source (e.g. LED matrix)	Small (when in sensor range)	Can be fairly controlled assuming auto-controls of the camera sensor are detecting anomaly or out of range values. Controllability increases with sensor redundancy. Sensor fusion should be used for sensors' conflict detection if no anomaly detection in the sensor itself.	Works better if light source is mounted on a leading vehicle rather than a static roadside setup. Also if the light source is statically placed in special places, such as near a traffic signal, on which victim vehicles stop for some time. Critical effects in the urban use case since the camera may never recover and specific vision tasks cannot be performed by other sensors (e.g. traffic light recognition).
Remote key control/ Smartphone control for parking chauffer app	Denial of Service (jamming); Spoofing; Stepstone attack to get access in AD ECU	Proficient (BT, radio signal interference)	Device that can copy BT/radio signals and transmit them.	Medium (when in vehicle's range)	Controllable assuming an appropriate IDS system and system segregation.	This is one of the vehicle's interface that is open and can be jammed (e.g. via smartphone inputs to OBU) in order to perform theft, damage or even injury to a passing by pedestrian.
Vehicle Wi-Fi	Data eavesdropping; Spoofing (e.g. inject false messages) Stepstone attack to get access in AD ECU	Expert or Multiple Experts (Wi-Fi signal interference)	Device with WiFi connectivity.	Medium (when in vehicle's range)	Controllable with appropriate IDS system and system segregation. Can also be controlled if the gateway is sufficiently protected (e.g., firewalls, digitals certificates, and ensuring that an outbound connection can only be initiated by the vehicle).	Might not lead to direct attack to ADAS ECU (elevation of privilege attack); however an adversary can exploit the infotainment system or TCU though Wi-Fi (Telematics provides voice/data wireless networks and is connected to all CAN buses).
Road structural element (e.g. traffic sign, lanes)	Tampering (modifying static or dynamic road traffic sign e.g. adding new fake road signs or lanes)	Layman or proficient in the execution; Proficient in the conception (vision-based perception interference).	Physical road surface or traffic sign modification (e.g. paint). Specialized equipment for interfering with road displays' visual content.	Unlimited	Very difficult to be detected since HD maps of the environment not usually available while dynamic localization and matching is a runtime intensive process.	Easy to execute for the attacker since it can be performed independently of the vehicle presence. Affects vehicle self-localization ability and may also result in false positives for objects detected based on visual artefacts (e.g a 3-D object drawing on the road surface).

TABLE VIII. TARA+ Risk matrix (Values as defined in tables I-VI and Eq. 1-3

Attack Scenario	Attack surface	Description	Po (Attack Potential) factors (Table I)		Po value (Eq. 1)/ Probability ranking (Table IV)	System/Driver Controllability factors (Table III)		Impact factors (Table II)		Impact value (Eq. 2) /Modified Impact value (Eq. 3)		R^* ranking (Table VI)
Remote Attack on Vehicle TCU (Highway)	Vehicle Wi-Fi	Inject fake commands on CAN bus via attacking TCU via exploiting vehicle Wi-Fi hotspot.	E2 Eq2	K1 W1	6 / Pr3- Possible	$C3^{D}$	$C2^{S}$	S4 F4	O4 P3	27	20.25 (MI3- HIGH)	HIGH
Lidar sensor spoofing (Highway Traffic Jam)	Lidar	Spoof the vehicle's lidar by optical means by generating signals that make objects disappear from the scene [17].	E1 Eq2	K1 W3	7 / Pr2- Unlikely	$C2^{D}$	$C2^{S}$	S4 F3	O4 P0	23	11.5 (MI2- MEDIUM)	MEDIUM
Road infrastructure attack (Urban)	Static road sign	Modify zebra crossing sign on the road surface creating the artifact of objects in front (see Fig. 4 in [20]).	E0 Eq0	K0 W0	0/ Pr4- Very Possible	-	$C4^{S}$	S3 F1	O3 P0	16	16 (MI2- MEDIUM)	HIGH

TARA Controllability-aware Threat Analysis and Risk Assessment: 6: 28|38 - 6: 576|763 (0)

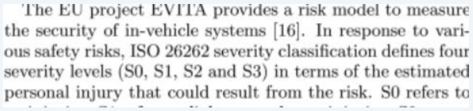


Attacks on Safety: An adversary can launch an attack on the safety of a vehicle by, for example, degrading the availability of critical sensors (e.g., IMU) or actuators, or the control performance (e.g., by changing PID gains). The worst-case scenario, from the vehicle's safety perspective, is when the attacker disables the flight controller while the vehicle is flying. This immediately leads the system to an *open-loop* state, which will cause the vehicle to crash. To demonstrate this type of attack, we consider an extreme scenario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how it is launched. We implemented a Linux kernel module that 1) finds the APM autopilot process from the kernel's process list and then 2) kills the process. The attacker could achieve the same goal by causing the virtual machine to crash.

There could be several ways to detect this type of attack. One may use a heartbeat mechanism, which however can be circumvented by an attacker that impersonates the flight controller. Instead, we take advantage of the SCE's ability to monitor the *true physical state* of the vehicle. We use the attitude control performance measured at the SCE. At each control loop, the SCE-side controller calculates the errors of the rate control on the pitch, roll, and yaw of the copter. During a stable flight, the rate errors are bounded, as shown in Figure 17 in Appendix C, because the flight controller is active to minimize the rate errors. In case of an open-loop state, the flight controller cannot work to minimize the rate errors. Due to the fact that a multirotor system is naturally an unstable flight platform,

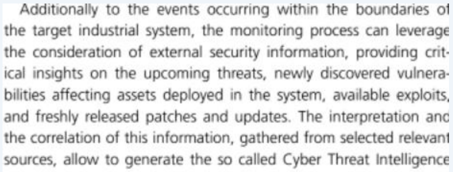
VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

Integration\Security Attacks\Type of Data\Risk Indicators



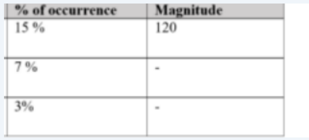
The EU project EVITA provides a risk model to measure the security of in-vehicle systems [16]. In response to various safety risks, ISO 26262 severity classification defines four severity levels (S0, S1, S2 and S3) in terms of the estimated personal injury that could result from the risk. S0 refers to

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)



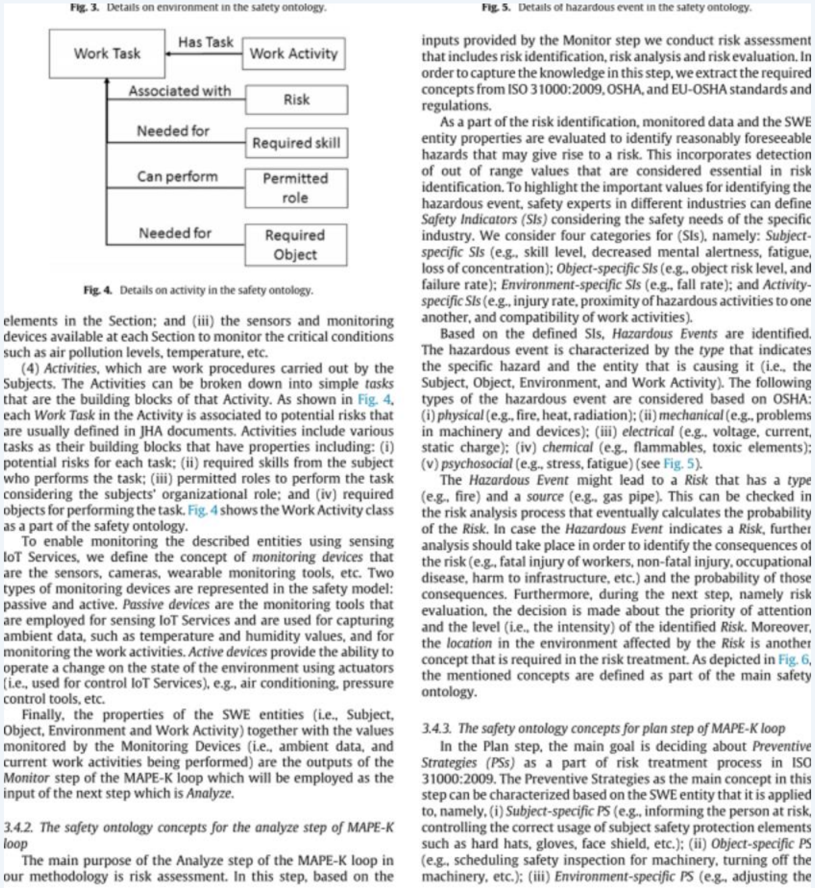
Additionally to the events occurring within the boundaries of the target industrial system, the monitoring process can leverage the consideration of external security information, providing critical insights on the upcoming threats, newly discovered vulnerabilities affecting assets deployed in the system, available exploits, and freshly released patches and updates. The interpretation and the correlation of this information, gathered from selected relevant sources, allow to generate the so called Cyber Threat Intelligence

Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)



% of occurrence	Magnitude
15 %	120
7 %	-
3%	-

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 315|225 - 9: 471|290 (0)



Work Task
Has Task
Work Activity
Associated with
Risk
Needed for
Required skill
Can perform
Permitted role
Needed for
Required Object

Fig. 3. Details on environment in the safety ontology.

Fig. 4. Details on activity in the safety ontology.

Fig. 5. Details of hazardous event in the safety ontology.

elements in the Section; and (iii) the sensors and monitoring devices available at each Section to monitor the critical conditions such as air pollution levels, temperature, etc.

(4) *Activities*, which are work procedures carried out by the Subjects. The Activities can be broken down into simple *tasks* that are the building blocks of that Activity. As shown in Fig. 4, each *Work Task* in the Activity is associated to potential risks that are usually defined in JHA documents. Activities include various tasks as their building blocks that have properties including: (i) potential risks for each task; (ii) required skills from the subject who performs the task; (iii) permitted roles to perform the task considering the subjects' organizational role; and (iv) required objects for performing the task. Fig. 4 shows the Work Activity class as a part of the safety ontology.

To enable monitoring the described entities using sensing IoT Services, we define the concept of *monitoring devices* that are the sensors, cameras, wearable monitoring tools, etc. Two types of monitoring devices are represented in the safety model: passive and active. *Passive devices* are the monitoring tools that are employed for sensing IoT Services and are used for capturing ambient data, such as temperature and humidity values, and for monitoring the work activities. *Active devices* provide the ability to operate a change on the state of the environment using actuators (i.e., used for control IoT Services), e.g., air conditioning, pressure control tools, etc.

Finally, the properties of the SWE entities (i.e., Subject, Object, Environment and Work Activity) together with the values monitored by the Monitoring Devices (i.e., ambient data, and current work activities being performed) are the outputs of the *Monitor* step of the MAPE-K loop which will be employed as the input of the next step which is *Analyze*.

3.4.2. The safety ontology concepts for the analyze step of MAPE-K loop

The main purpose of the Analyze step of the MAPE-K loop in our methodology is risk assessment. In this step, based on the inputs provided by the Monitor step we conduct risk assessment that includes risk identification, risk analysis and risk evaluation. In order to capture the knowledge in this step, we extract the required concepts from ISO 31000:2009, OSHA, and EU-OSHA standards and regulations.

As a part of the risk identification, monitored data and the SWE entity properties are evaluated to identify reasonably foreseeable hazards that may give rise to a risk. This incorporates detection of out of range values that are considered essential in risk identification. To highlight the important values for identifying the hazardous event, safety experts in different industries can define *Safety Indicators (SIs)* considering the safety needs of the specific industry. We consider four categories for (SIs), namely: *Subject-specific SIs* (e.g., skill level, decreased mental alertness, fatigue, loss of concentration); *Object-specific SIs* (e.g., object risk level, and failure rate); *Environment-specific SIs* (e.g., fall rate); and *Activity-specific SIs* (e.g., injury rate, proximity of hazardous activities to one another, and compatibility of work activities).

Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates the specific hazard and the entity that is causing it (i.e., the Subject, Object, Environment, and Work Activity). The following types of the hazardous event are considered based on OSHA: (i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current, static charge); (iv) *chemical* (e.g., flammables, toxic elements); (v) *psychosocial* (e.g., stress, fatigue) (see Fig. 5).

The *Hazardous Event* might lead to a *Risk* that has a *type* (e.g., fire) and a *source* (e.g., gas pipe). This can be checked in the risk analysis process that eventually calculates the probability of the *Risk*. In case the *Hazardous Event* indicates a *Risk*, further analysis should take place in order to identify the consequences of the risk (e.g., fatal injury of workers, non-fatal injury, occupational disease, harm to infrastructure, etc.) and the probability of those consequences. Furthermore, during the next step, namely risk evaluation, the decision is made about the priority of attention and the level (i.e., the intensity) of the identified *Risk*. Moreover, the *location* in the environment affected by the *Risk* is another concept that is required in the risk treatment. As depicted in Fig. 6, the mentioned concepts are defined as part of the main safety ontology.

3.4.3. The safety ontology concepts for plan step of MAPE-K loop

In the Plan step, the main goal is deciding about *Preventive Strategies (PSs)* as a part of risk treatment process in ISO 31000:2009. The Preventive Strategies as the main concept in this step can be characterized based on the SWE entity that it is applied to, namely, (i) *Subject-specific PS* (e.g., informing the person at risk, controlling the correct usage of subject safety protection elements such as hard hats, gloves, face shield, etc.); (ii) *Object-specific PS* (e.g., scheduling safety inspection for machinery, turning off the machinery, etc.); (iii) *Environment-specific PS* (e.g., adjusting the

Ontology development for run-time safety management methodology: 5 - 5 (0)

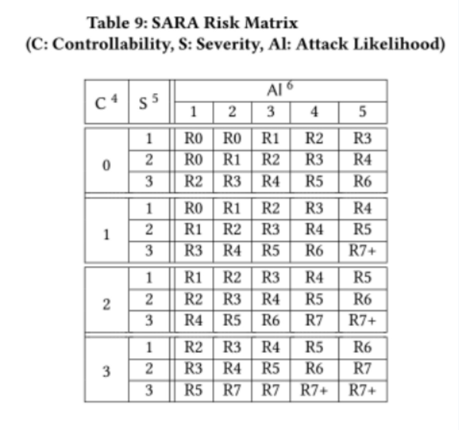
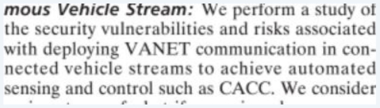


Table 9: SARA Risk Matrix
(C: Controllability, S: Severity, Al: Attack Likelihood)

C 4	S 5	Al 6				
		1	2	3	4	5
0	1	R0	R0	R1	R2	R3
	2	R0	R1	R2	R3	R4
	3	R2	R3	R4	R5	R6
1	1	R0	R1	R2	R3	R4
	2	R1	R2	R3	R4	R5
	3	R3	R4	R5	R6	R7+
2	1	R1	R2	R3	R4	R5
	2	R2	R3	R4	R5	R6
	3	R4	R5	R6	R7	R7+
3	1	R2	R3	R4	R5	R6
	2	R3	R4	R5	R6	R7
	3	R5	R7	R7	R7+	R7+

SARA Security Automotive Risk Analysis Method: 8: 53|381 - 8: 288|598 (0)



mous Vehicle Stream: We perform a study of the security vulnerabilities and risks associated with deploying VANET communication in connected vehicle streams to achieve automated sensing and control such as CACC. We consider

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

Integration\Security Attacks\Type of Data\Threat

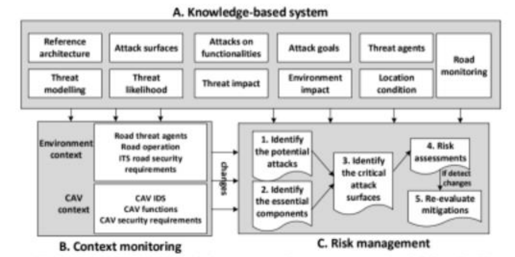
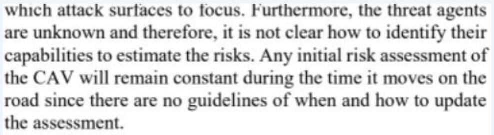


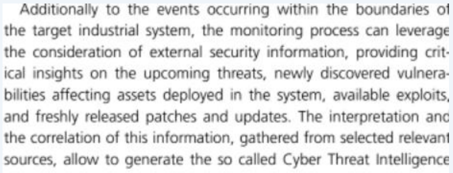
Figure 1. The proposed dynamic risk assessment model for CAV

A simplified approach for dynamic security risk management in c: 4: 36|631 - 4: 300|781 (0)



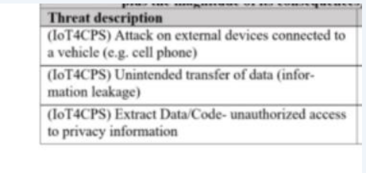
which attack surfaces to focus. Furthermore, the threat agents are unknown and therefore, it is not clear how to identify their capabilities to estimate the risks. Any initial risk assessment of the CAV will remain constant during the time it moves on the road since there are no guidelines of when and how to update the assessment.

A simplified approach for dynamic security risk management in c: 6 - 6 (0)



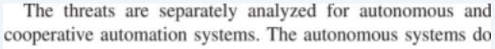
Additionally to the events occurring within the boundaries of the target industrial system, the monitoring process can leverage the consideration of external security information, providing critical insights on the upcoming threats, newly discovered vulnerabilities affecting assets deployed in the system, available exploits, and freshly released patches and updates. The interpretation and the correlation of this information, gathered from selected relevant sources, allow to generate the so called Cyber Threat Intelligence

Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)



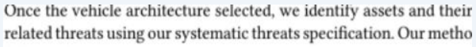
Threat description
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)
(IoT4CPS) Unintended transfer of data (information leakage)
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 133|211 - 9: 319|294 (0)



The threats are separately analyzed for autonomous and cooperative automation systems. The autonomous systems do

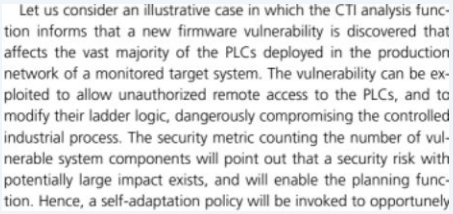
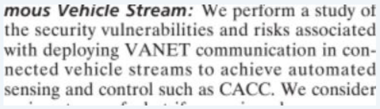
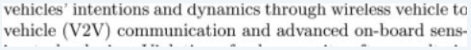
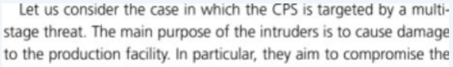
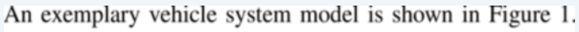
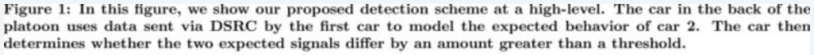
Potential cyberattacks on automated vehicles: 3 - 3 (0)



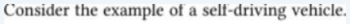
Once the vehicle architecture selected, we identify assets and their related threats using our systematic threats specification. Our metho

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)

Integration\Security Attacks\Type of Data\Vulnerability

	 Let us consider an illustrative case in which the CTI analysis function informs that a new firmware vulnerability is discovered that affects the vast majority of the PLCs deployed in the production network of a monitored target system. The vulnerability can be exploited to allow unauthorized remote access to the PLCs, and to modify their ladder logic, dangerously compromising the controlled industrial process. The security metric counting the number of vulnerable system components will point out that a security risk with potentially large impact exists, and will enable the planning function. Hence, a self-adaptation policy will be invoked to opportunely Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)
	 buffer overflow vulnerability Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)
	 ***mous Vehicle Stream:*** We perform a study of the security vulnerabilities and risks associated with deploying VANET communication in connected vehicle streams to achieve automated sensing and control such as CACC. We consider Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)
Integration\Security Attacks\Type of Data\Asset	 vehicles' intentions and dynamics through wireless vehicle to vehicle (V2V) communication and advanced on-board sens- A Functional Co-Design towards Safe and Secure Vehicle Platooni: 1 - 1 (0)
	 Connected and autonomous vehicles (CAVs) A simplified approach for dynamic security risk management in c: 2 - 2 (0)
	 Let us consider the case in which the CPS is targeted by a multistage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)
	 An exemplary vehicle system model is shown in Figure 1. Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)
	 **Figure 1: In this figure, we show our proposed detection scheme at a high-level. The car in the back of the platoon uses data sent via DSRC by the first car to model the expected behavior of car 2. The car then determines whether the two expected signals differ by an amount greater than a threshold.**

Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



Consider the example of a self-driving vehicle.

Know the unknowns addressing disturbances and uncertainties in: 1 - 1 (0)

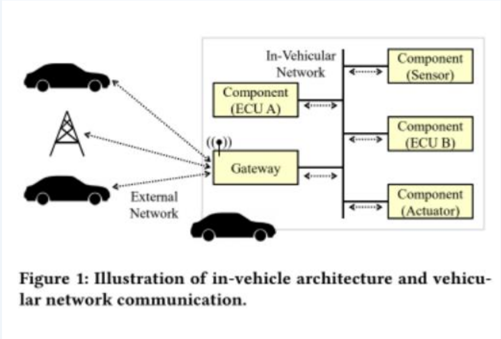
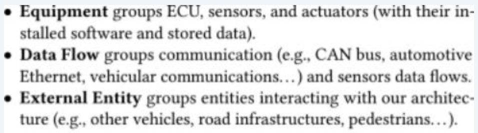


Figure 1: Illustration of in-vehicle architecture and vehicular network communication.

Network and system level security in connected vehicle applicat: 2: 45|555 - 2: 300|725 (0)

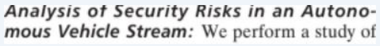
Automated Vehicles

Potential cyberattacks on automated vehicles: 1 - 1 (0)



- **Equipment** groups ECU, sensors, and actuators (with their installed software and stored data).
- **Data Flow** groups communication (e.g., CAN bus, automotive Ethernet, vehicular communications…) and sensors data flows.
- **External Entity** groups entities interacting with our architecture (e.g., other vehicles, road infrastructures, pedestrians…).

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)

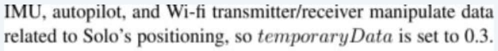


Analysis of Security Risks in an Autonomous Vehicle Stream: We perform a study of

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

two cars.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)



IMU, autopilot, and Wi-fi transmitter/receiver manipulate data related to Solo's positioning, so $temporaryData$ is set to 0.3.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)

vehicle's safety

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

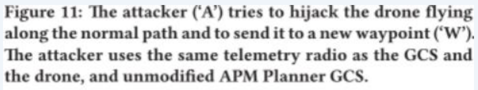
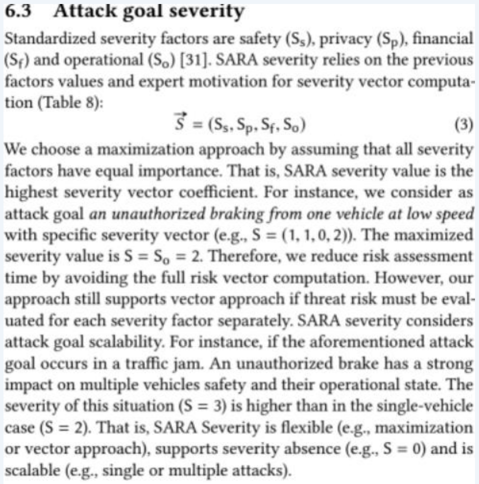


Figure 11: The attacker ('A') tries to hijack the drone flying along the normal path and to send it to a new waypoint ('W'). The attacker uses the same telemetry radio as the GCS and the drone, and unmodified APM Planner GCS.

VirtualDrone virtual sensing actuation and communication for at: 7 - 7 (0)

Integration\Security Attacks\How is Severity Measured\Continous



6.3 Attack goal severity

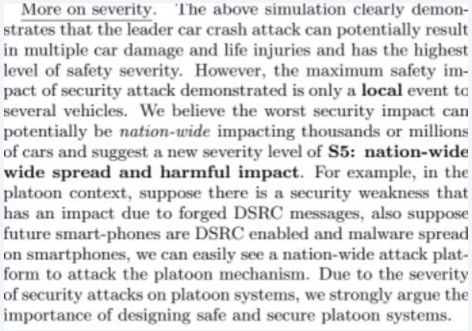
Standardized severity factors are safety (S_s), privacy (S_p), financial (S_f) and operational (S_o) [31]. SARA severity relies on the previous factors values and expert motivation for severity vector computation (Table 8):

$$\vec{S} = (S_s, S_p, S_f, S_o) \quad (3)$$

We choose a maximization approach by assuming that all severity factors have equal importance. That is, SARA severity value is the highest severity vector coefficient. For instance, we consider as attack goal *an unauthorized braking from one vehicle at low speed* with specific severity vector (e.g., $S = (1, 1, 0, 2)$). The maximized severity value is $S = S_o = 2$. Therefore, we reduce risk assessment time by avoiding the full risk vector computation. However, our approach still supports vector approach if threat risk must be evaluated for each severity factor separately. SARA severity considers attack goal scalability. For instance, if the aforementioned attack goal occurs in a traffic jam. An unauthorized brake has a strong impact on multiple vehicles safety and their operational state. The severity of this situation ($S = 3$) is higher than in the single-vehicle case ($S = 2$). That is, SARA Severity is flexible (e.g., maximization or vector approach), supports severity absence (e.g., $S = 0$) and is scalable (e.g., single or multiple attacks).

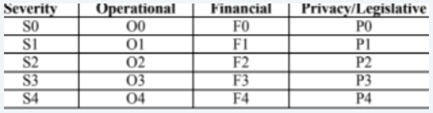
SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

Integration\Security Attacks\How is Severity Measured\Discrete



More on severity. The above simulation clearly demonstrates that the leader car crash attack can potentially result in multiple car damage and life injuries and has the highest level of safety severity. However, the maximum safety impact of security attack demonstrated is only a **local** event to several vehicles. We believe the worst security impact can potentially be *nation-wide* impacting thousands or millions of cars and suggest a new severity level of **S5: nation-wide wide spread and harmful impact**. For example, in the platoon context, suppose there is a security weakness that has an impact due to forged DSRC messages, also suppose future smart-phones are DSRC enabled and malware spread on smartphones, we can easily see a nation-wide attack platform to attack the platoon mechanism. Due to the severity of security attacks on platoon systems, we strongly argue the importance of designing safe and secure platoon systems.

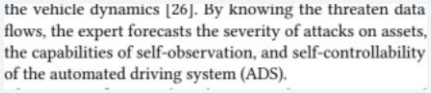
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)



Severity	Operational	Financial	Privacy/Legislative
S0	O0	F0	P0
S1	O1	F1	P1
S2	O2	F2	P2
S3	O3	F3	P3
S4	O4	F4	P4

TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)

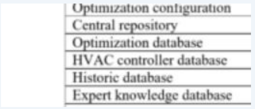
Integration\Security Attacks\Affected Part of Adaptation



the vehicle dynamics [26]. By knowing the threaten data flows, the expert forecasts the severity of attacks on assets, the capabilities of self-observation, and self-controllability of the automated driving system (ADS).

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)

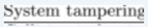
Integration\Security Attacks\Affected Part of Adaptation\Database



Optimization configuration
Central repository
Optimization database
HVAC controller database
Historic database
Expert knowledge database

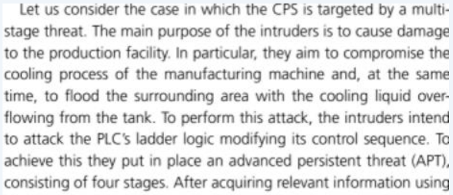
Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 121|467 - 9: 249|516 (0)

Integration\Security Attacks\Affected Part of Adaptation\Controller



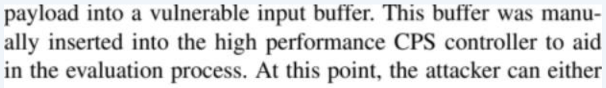
System tampering

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)



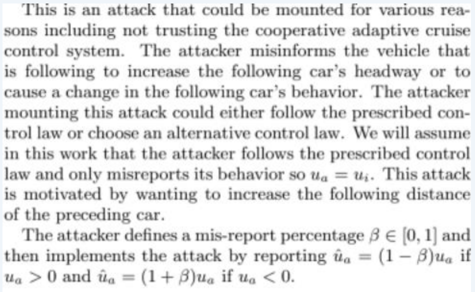
Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using

Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)



payload into a vulnerable input buffer. This buffer was manually inserted into the high performance CPS controller to aid in the evaluation process. At this point, the attacker can either

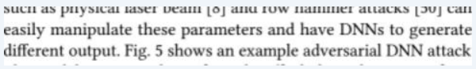
Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)



This is an attack that could be mounted for various reasons including not trusting the cooperative adaptive cruise control system. The attacker misinforms the vehicle that is following to increase the following car's headway or to cause a change in the following car's behavior. The attacker mounting this attack could either follow the prescribed control law or choose an alternative control law. We will assume in this work that the attacker follows the prescribed control law and only misreports its behavior so $u_a = u_i$. This attack is motivated by wanting to increase the following distance of the preceding car.

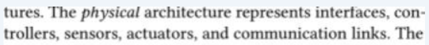
The attacker defines a mis-report percentage $\beta \in [0, 1]$ and then implements the attack by reporting $\hat{u}_a = (1 - \beta)u_a$ if $u_a > 0$ and $\hat{u}_a = (1 + \beta)u_a$ if $u_a < 0$.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



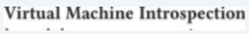
such as physical laser beam [8] and row hammer attacks [30] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



tures. The *physical* architecture represents interfaces, controllers, sensors, actuators, and communication links. The

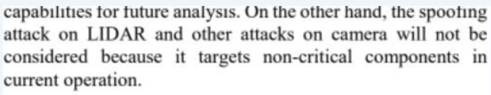
SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



Virtual Machine Introspection

VirtualDrone virtual sensing actuation and communication for at: 9 - 9 (0)

Integration\Security Attacks\Affected Part of Adaptation\Sensors



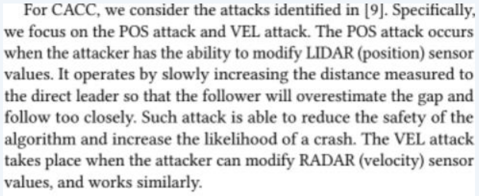
capabilities for future analysis. On the other hand, the spoofing attack on LIDAR and other attacks on camera will not be considered because it targets non-critical components in current operation.

A simplified approach for dynamic security risk management in c: 7 - 7 (0)

Sensors

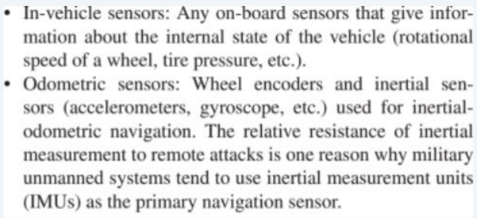
Actuators

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 138|450 - 9: 250|466 (0)



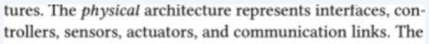
For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)



- In-vehicle sensors: Any on-board sensors that give information about the internal state of the vehicle (rotational speed of a wheel, tire pressure, etc.).
- Odometric sensors: Wheel encoders and inertial sensors (accelerometers, gyroscope, etc.) used for inertial-odometric navigation. The relative resistance of inertial measurement to remote attacks is one reason why military unmanned systems tend to use inertial measurement units (IMUs) as the primary navigation sensor.

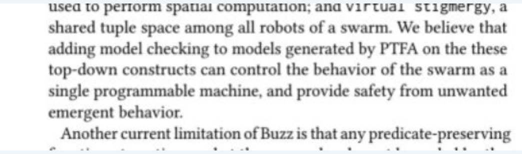
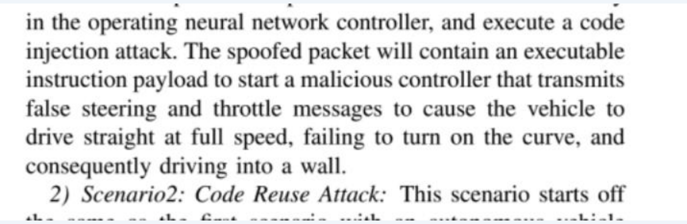
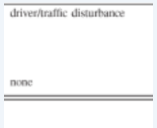
Potential cyberattacks on automated vehicles: 6 - 6 (0)



tures. The *physical* architecture represents interfaces, controllers, sensors, actuators, and communication links. The

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)

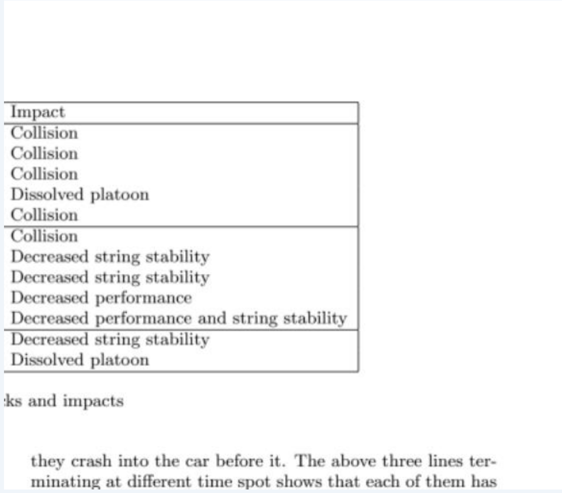
Range
Sensors
(radar,
ultrasonic,
LiDAR)

	TARA Controllability-aware Threat Analysis and Risk Assessment: 6 - 6 (0)
	 irtual Sensing and Actuation: C VirtualDrone virtual sensing actuation and communication for at: 3 - 3 (0)
Integration\Safety Hazards\Hazard Cause\Other	 used to perform spatial computation; and `virtual stigmergy`, a shared tuple space among all robots of a swarm. We believe that adding model checking to models generated by PTFA on the these top-down constructs can control the behavior of the swarm as a single programmable machine, and provide safety from unwanted emergent behavior. Another current limitation of Buzz is that any predicate-preserving Engineering safety in swarm robotics: 4: 34|213 - 4: 302|287 (0)
	 in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall. 2) *Scenario2: Code Reuse Attack:* This scenario starts off Integrated moving target defense and control reconfiguration fo: 8: 301|92 - 8: 589|182 (0)
	(i) physical (e.g., fire, heat, radiation); (ii) mechanical (e.g., problems in machinery and devices); (iii) electrical (e.g., voltage, current, static charge); (iv) chemical (e.g., flammables, toxic elements); (v) psychosocial (e.g., stress, fatigue) (see Fig. 5). Ontology development for run-time safety management methodology: 5 - 5 (0)
	 driver/traffic disturbance none Potential cyberattacks on automated vehicles: 8: 408|297 - 8: 484|361 (0)
	n the location. The VirtualDrone framework eliminates this

possibility by not providing the RSSI information to the NCE. Note that RSSI is needed only for the SCE (e.g., switches to the SCE and then performs a pre-defined operation such as return-to-home when RSSI is low due to the loss of telemetry link). Furthermore, because of the telemetry virtualization, the NCE cannot even know if the telemetry is communicated through a radio channel or a network (e.g., WiFi). In case of the network-based telemetry, the SCE can even hide the IP address of the GCS.

VirtualDrone virtual sensing actuation and communication for at: 9 - 9 (0)

Integration\Safety Hazards\Hazard Cause\String Stability Decrease

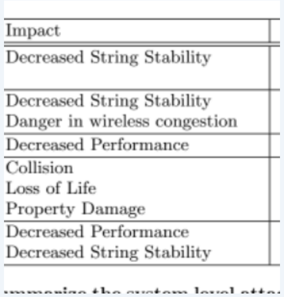


Impact
Collision
Collision
Collision
Dissolved platoon
Collision
Collision
Decreased string stability
Decreased string stability
Decreased performance
Decreased performance and string stability
Decreased string stability
Dissolved platoon

ks and impacts

they crash into the car before it. The above three lines terminating at different time spot shows that each of them has

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 303|540 - 5: 589|790 (0)



Impact
Decreased String Stability
Decreased String Stability Danger in wireless congestion
Decreased Performance
Collision Loss of Life Property Damage
Decreased Performance Decreased String Stability

ummarize the system level atta

Is your commute driving you crazy a study of misbehavior in veh: 5: 178|389 - 5: 319|539 (0)

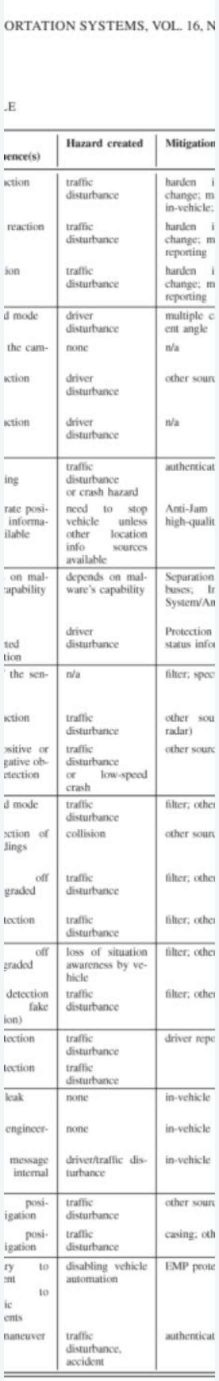
Integration\Safety Hazards\Hazard Cause\Crash

) Scenario1: Code Injection Attack: This scenario in-volves an autonomous vehicle
starting out driving on a straight
road. At the point where the vehicle starts to take a turn
at 70 seconds into the simulation, an adversary spoofs a
malicious RFA packet to exploit a buffer overflow vulnerability
in the operating neural network controller, and execute a code
injection attack. The spoofed packet will contain an executable
instruction payload to start a malicious controller that transmits
false steering and throttle messages to cause the vehicle to
drive straight at full speed, failing to turn on the curve, and
consequently driving into a wall.

Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)

For CACC, we consider the attacks identified in [9]. Specifically,
we focus on the POS attack and VEL attack. The POS attack occurs
when the attacker has the ability to modify LIDAR (position) sensor
values. It operates by slowly increasing the distance measured to
the direct leader so that the follower will overestimate the gap and
follow too closely. Such attack is able to reduce the safety of the
algorithm and increase the likelihood of a crash. The VEL attack
takes place when the attacker can modify RADAR (velocity) sensor
values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)

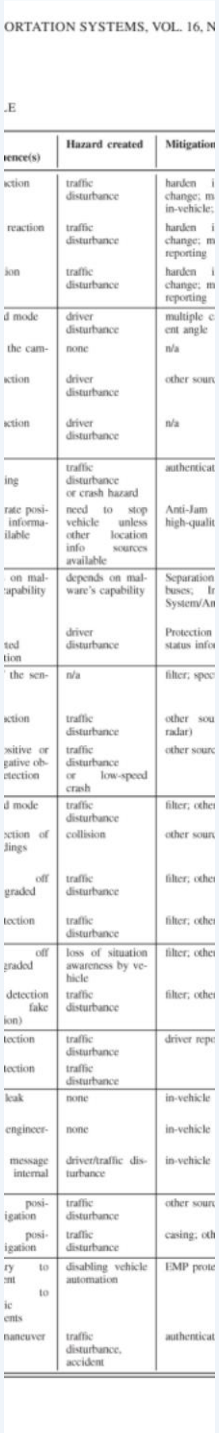


ORTATION SYSTEMS, VOL. 16, N

.E

ence(s)	Hazard created	Mitigation
ction	traffic disturbance	harden i change; m in-vehicle;
reaction	traffic disturbance	harden i change; m reporting
ion	traffic disturbance	harden i change; m reporting
d mode	driver disturbance	multiple c ent angle
the cam-	none	n/a
ction	driver disturbance	other sourc
ction	driver disturbance	n/a
ing	traffic disturbance or crash hazard	authenticat
rate posi- informa- ilable	need to stop vehicle unless other location info sources available	Anti-Jam high-qualit
on mal- apability	depends on mal-ware's capability	Separation buses; Ir System/An
ted tion	driver disturbance	Protection status info
the sen-	n/a	filter; spec
ction	traffic disturbance	other sou radar)
sitive or gative ob- etection	traffic disturbance or low-speed crash	other sourc
d mode	traffic disturbance	filter; other
ction of lings	collision	other sourc
off graded	traffic disturbance	filter; other
tection	traffic disturbance	filter; other
off graded	loss of situation awareness by ve-hicle	filter; other
detection fake ion)	traffic disturbance	filter; other
tection	traffic disturbance	driver repo
tection	traffic disturbance	
leak	none	in-vehicle
engineer-	none	in-vehicle
message internal	driver/traffic dis-turbance	in-vehicle
posi- igation	traffic disturbance	other sourc
posi- igation	traffic disturbance	casing; oth
ry to ent to ic ents	disabling vehicle automation	EMP prote
naneuver	traffic disturbance, accident	authenticat

	Potential cyberattacks on automated vehicles: 5: 396|18 - 5: 504|765 (0)
	seriously impact the vehicle’s safety as the on-board system may make erroneous control decisions threatening other vehicles on road causing multiple injuries or deaths inflicting reputation and financial TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)
Integration\Safety Hazards\Hazard Cause\Component Failure	[13]) define domain specific processes and methods for development of safety-critical embedded systems. They minimise systematic failures at development (e.g. unimplemented requirement) and to control random failures during opera-tion (e.g. component breakdown). These standards rely on quality management Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



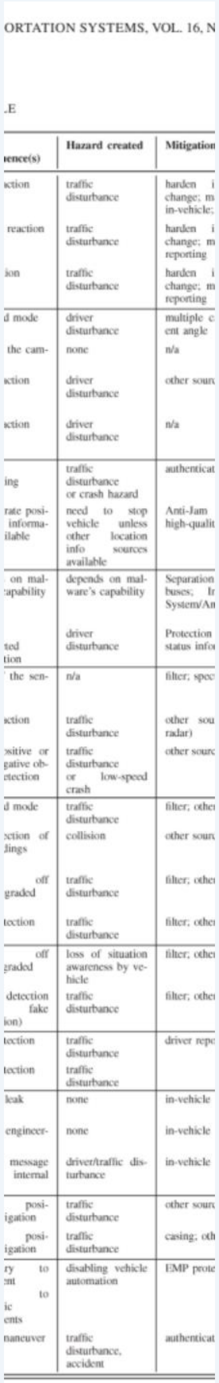
ORTATION SYSTEMS, VOL. 16, N

.E

ence(s)	Hazard created	Mitigation
ction	traffic disturbance	harden i change; m in-vehicle;
reaction	traffic disturbance	harden i change; m reporting
ion	traffic disturbance	harden i change; m reporting
d mode	driver disturbance	multiple c ent angle
the cam-	none	n/a
ction	driver disturbance	other sourc
ction	driver disturbance	n/a
ing	traffic disturbance or crash hazard	authenticat
rate posi- informa- ilable	need to stop vehicle unless other location info sources available	Anti-Jam high-qualit
on mal- apability	depends on mal-ware's capability	Separation buses; Ir System/An
ted tion	driver disturbance	Protection status infor
the sen-	n/a	filter; spec
ction	traffic disturbance	other sou radar)
sitive or gative ob- etection	traffic disturbance or low-speed crash	other sourc
d mode	traffic disturbance	filter; other
ction of lings	collision	other sourc
off graded	traffic disturbance	filter; other
tection	traffic disturbance	filter; other
off graded	loss of situation awareness by vehicle	filter; other
detection fake ion)	traffic disturbance	filter; other
tection	traffic disturbance	driver repo
tection	traffic disturbance	
leak	none	in-vehicle
engineer-	none	in-vehicle
message internal	driver/traffic disturbance	in-vehicle
posi- igation	traffic disturbance	other sourc
posi- igation	traffic disturbance	casing; oth
ry to ent to ic ents	disabling vehicle automation	EMP prote
naneuver	traffic disturbance, accident	authenticat

	Potential cyberattacks on automated vehicles: 5: 396|18 - 5: 504|765 (0)
	To demonstrate this type of attack, we consider an extreme sce-nario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how it is launched. We implemented a Linux kernel module that 1) finds the APM autopilot process from the kernel's process list and then 2) kills the process. The attacker could achieve the VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)
Integration\Safety Hazards\Hazard Cause\Overheating	Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid over-flowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using social engineering methods, in the reconnaissance phase (stage I), the attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with DarkComet (a sophisticated re-mote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral move-ments (stage III), and obtain administrative privilege on an engineer-ing workstation deployed in the SCADA network. To intrude the production network, the attackers exploit a vulnerability of a net-work switch and establish a communication with the field devices, including PLCs (stage IV). The attackers are now able to modify the PLC configuration through the TIA portal and customize its logic. In particular they apply specific changes to the ladder logic, to disable the cool and the drain valve, change the fill limit values in the PLC memory, and deny the HMI to send any overriding control command

	to the PLC. Subsequently, they upload the altered configuration set-tings to the PLC. Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)
	Figure 6 illustrates how the cooling process changes after the PLC starts operating following the new logic (compare this diagram with Figure 5). Supposing that the new logic is uploaded before a new cooling cycle starts (as illustrated with the red dashed line), although the temperature sensor will keep sending high temperatures measurements to the PLC, the cool valve will not be open, and the manufacturing machine will not be cooled down. Assuming that no further safety precautions are enabled, this will continue until the manufacturing machine will overheat and stop operating. Meanwhile, the liquid level in the tank will increase because the fill valve will remain open, making the liquid overflow the tank, and subsequently flood the area around the tank. Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)
Integration\Safety Hazards\Hazard Cause\System Failure	the server is down, a coordinated security attack is launched against the system (a bad situation getting worse). In that case, the response to the attack might be to shut down as much of the system as possible. The system would transition to service S3 since that would permit the best support in the event that the situation developed into a govern-mental crisis. Achieving critical system survivability through software archit: 10 - 10 (0)



ORTATION SYSTEMS, VOL. 16, N

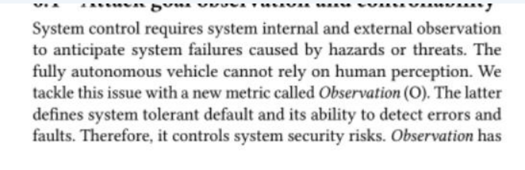
.E

ence(s)	Hazard created	Mitigation
ction	traffic disturbance	harden i change; m in-vehicle;
reaction	traffic disturbance	harden i change; m reporting
ion	traffic disturbance	harden i change; m reporting
d mode	driver disturbance	multiple c ent angle
the cam-	none	n/a
ction	driver disturbance	other sour
ction	driver disturbance	n/a
ing	traffic disturbance or crash hazard	authenticat
rate posi- informa- ilable	need to stop vehicle unless other location info sources available	Anti-Jam high-qualit
on mal- apability	depends on malware's capability	Separation buses; Ir System/An
ted tion	driver disturbance	Protection status info
the sen-	n/a	filter; spec
ction	traffic disturbance	other sou radar)
sitive or gative ob- etection	traffic disturbance or low-speed crash	other sourc
d mode	traffic disturbance	filter; othe
ction of dings	collision	other sour
off graded	traffic disturbance	filter; othe
tection	traffic disturbance	filter; othe
off graded	loss of situation awareness by vehicle	filter; othe
detection fake ion)	traffic disturbance	filter; othe
tection	traffic disturbance	driver repo
tection	traffic disturbance	
leak	none	in-vehicle
engineer-	none	in-vehicle
message internal	driver/traffic disturbance	in-vehicle
posi- igation	traffic disturbance	other sour
posi- igation	traffic disturbance	casing; oth
ry to ent to ic ents	disabling vehicle automation	EMP prote
naneuver	traffic disturbance, accident	authenticat

Potential cyberattacks on automated vehicles: 5: 396|18 - 5: 504|765 (0)

Figure 6 illustrates how the cooling process changes after the PLC starts operating following the new logic (compare this diagram with Figure 5). Supposing that the new logic is uploaded before a new cooling cycle starts (as illustrated with the red dashed line), although the temperature sensor will keep sending high temperatures measurements to the PLC, the cool valve will not be open, and the manufacturing machine will not be cooled down. Assuming that no further safety precautions are enabled, this will continue until the manufacturing machine will overheat and stop operating. Meanwhile, the liquid level in the tank will increase because the fill valve will remain open, making the liquid overflow the tank, and subsequently flood the area around the tank.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



6.1 Attack goal observation and controllability

System control requires system internal and external observation to anticipate system failures caused by hazards or threats. The fully autonomous vehicle cannot rely on human perception. We tackle this issue with a new metric called *Observation* (O). The latter defines system tolerant default and its ability to detect errors and faults. Therefore, it controls system security risks. *Observation* has

SARA Security Automotive Risk Analysis Method: 7: 37|70 - 7: 306|155 (0)

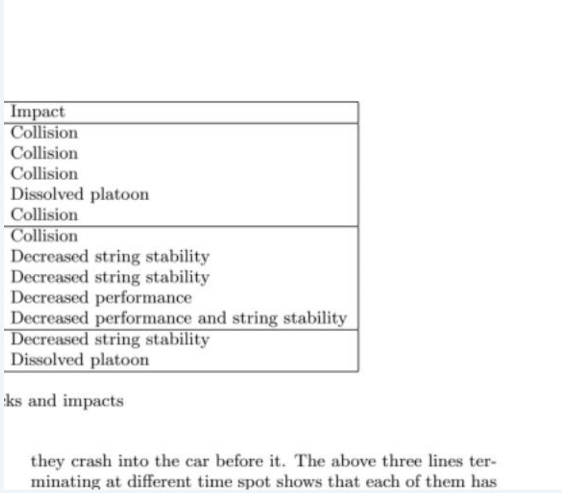
Hazards relate to local or regional failures (systems, communi-cations, power)

Self-organisation for Survival in Complex Computer Architecture: 10 - 10 (0)

mitigated by the system as system goes into fail-operational mode (sufficient vehicle level redundancy to continue full operation; no reliance on the driver; applicable to L4 and higher).

TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)

Integration\Safety Hazards\Hazard Cause\Performance Alteration



Impact
Collision
Collision
Collision
Dissolved platoon
Collision
Collision
Decreased string stability
Decreased string stability
Decreased performance
Decreased performance and string stability
Decreased string stability
Dissolved platoon

:ks and impacts

they crash into the car before it. The above three lines terminating at different time spot shows that each of them has

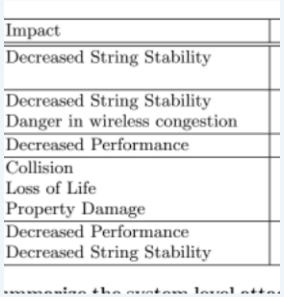
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 303|540 - 5: 589|790 (0)

In (ABCD), given the spoofing attacks on LIDAR, the knowledge-based system suggests that attackers aim to spoof nearby objects, which is part of a larger aim of slowing down the CAV (e.g. see the attack tree in Figure 5). On the other hand, LIDAR spoofing is the only attack that is recorded, therefore we assume that attacker capabilities equal to system

A simplified approach for dynamic security risk management in c: 6 - 6 (0)

) Scenario1: Code Injection Attack: This scenario in-volves an autonomous vehicle starting out driving on a straight road. At the point where the vehicle starts to take a turn at 70 seconds into the simulation, an adversary spoofs a malicious RFA packet to exploit a buffer overflow vulnerability in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall.

Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)



Impact
Decreased String Stability
Decreased String Stability Danger in wireless congestion
Decreased Performance
Collision Loss of Life Property Damage
Decreased Performance Decreased String Stability

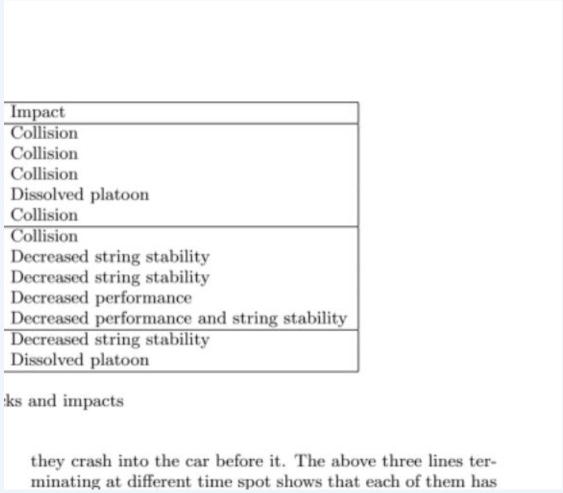
ummarize the system level atta

Is your commute driving you crazy a study of misbehavior in veh: 5: 178|389 - 5: 319|539 (0)

instance, changing the acceleration field might have a more significant effect than changing the velocity

Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)

Integration\Safety Hazards\Hazard Cause\Collision

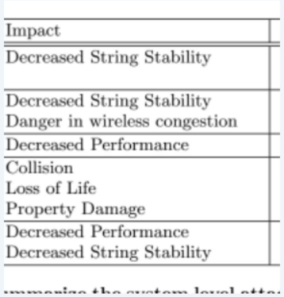


Impact
Collision Collision Collision Dissolved platoon Collision
Collision Decreased string stability Decreased string stability Decreased performance Decreased performance and string stability
Decreased string stability Dissolved platoon

ks and impacts

they crash into the car before it. The above three lines terminating at different time spot shows that each of them has

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 303|540 - 5: 589|790 (0)



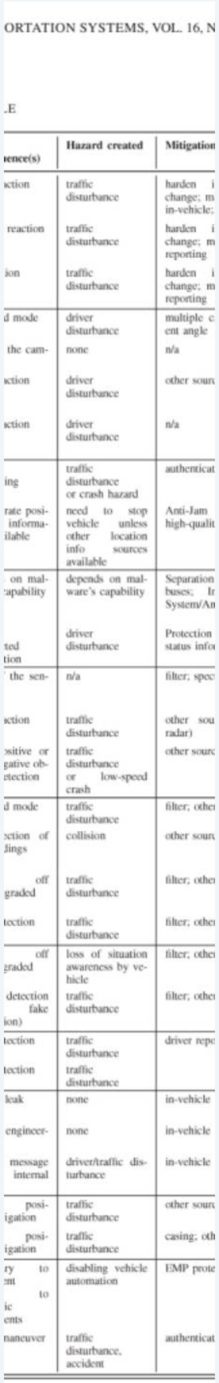
Impact
Decreased String Stability
Decreased String Stability Danger in wireless congestion
Decreased Performance
Collision Loss of Life Property Damage
Decreased Performance Decreased String Stability

ummarize the system level atta

Is your commute driving you crazy a study of misbehavior in veh: 5: 178|389 - 5: 319|539 (0)

) Only partial agreement reached. Unsafe to
merge as vehicles 1 and 2 may collide in the middle lane. (b)
Global consensus/agreement reached. Safe to merge. (c) Par-tial agreement reached
among vehicles 1, 2 and 4. Vehicle 2
may merge into the middle lane based on its own sensors
and the communication with 4, but the case is not as safe as
in case (b). (d) Partial agreement reached between vehicles
1 and 2, and between 3 and 4. Vehicle 2 may still be able to
merge into the middle lane based on its own sensors, but the
case is not as safe as the ones in (b) and (c)

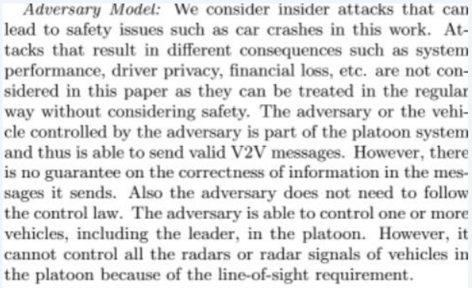
Network and system level security in connected vehicle applicat: 5 - 5 (0)

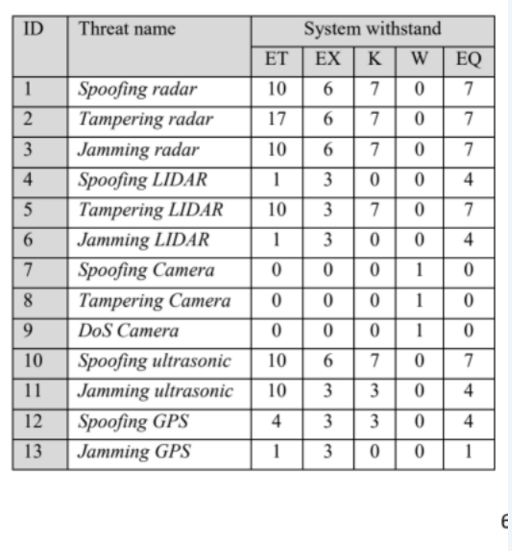


ORTATION SYSTEMS, VOL. 16, N

.E

ience(s)	Hazard created	Mitigation
ction	traffic disturbance	harden i change; m in-vehicle;
reaction	traffic disturbance	harden i change; m reporting
ion	traffic disturbance	harden i change; m reporting
d mode	driver disturbance	multiple c ent angle
the cam-	none	n/a
ction	driver disturbance	other sour
ction	driver disturbance	n/a
ing	traffic disturbance or crash hazard	authenticat
rate posi- informa- ilable	need to stop vehicle unless other location info sources available	Anti-Jam high-qualit
on mal- apability	depends on malware's capability	Separation buses; Ir System/An
ted tion	driver disturbance	Protection status info
the sen-	n/a	filter; spec
ction	traffic disturbance	other sou radar)
sitive or gative ob- etection	traffic disturbance or low-speed crash	other sourc
d mode	traffic disturbance	filter; other
ction of lings	collision	other sour
off graded	traffic disturbance	filter; other
tection	traffic disturbance	filter; other
off graded	loss of situation awareness by vehicle	filter; other
detection fake ion)	traffic disturbance	filter; other
tection	traffic disturbance	driver repo
tection	traffic disturbance	
leak	none	in-vehicle
engineer-	none	in-vehicle
message internal	driver/traffic disturbance	in-vehicle
posi- igation	traffic disturbance	other sour
posi- igation	traffic disturbance	casing; oth
ry to ent to ic ents	disabling vehicle automation	EMP prote
maneuver	traffic disturbance, accident	authenticat

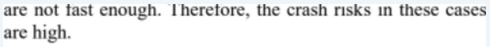
<table>
<tr><td rowspan="3"></td><td>Potential cyberattacks on automated vehicles: 5: 396|18 - 5: 504|765 (0)</td></tr>
<tr><td>leading vehicle slows down, the adversary injects
the old beacon into the system periodically. Fol-lowing vehicles still think that the
lead vehicle is
driving at 30 m/s and do not slow down, poten-tially leading to a collision.
Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)</td></tr>
<tr><td>f an accident involving two cars. I
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)</td></tr>
<tr><td rowspan="2">Integration\Safety Hazards\Hazard Source\External</td><td>
<em>Adversary Model:</em> We consider insider attacks that can lead to safety issues such as car crashes in this work. Attacks that result in different consequences such as system performance, driver privacy, financial loss, etc. are not considered in this paper as they can be treated in the regular way without considering safety. The adversary or the vehicle controlled by the adversary is part of the platoon system and thus is able to send valid V2V messages. However, there is no guarantee on the correctness of information in the messages it sends. Also the adversary does not need to follow the control law. The adversary is able to control one or more vehicles, including the leader, in the platoon. However, it cannot control all the radars or radar signals of vehicles in the platoon because of the line-of-sight requirement.
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)</td></tr>
<tr><td></td></tr>
</table>



ID	Threat name	System withstand				
		ET	EX	K	W	EQ
1	*Spoofing radar*	10	6	7	0	7
2	*Tampering radar*	17	6	7	0	7
3	*Jamming radar*	10	6	7	0	7
4	*Spoofing LIDAR*	1	3	0	0	4
5	*Tampering LIDAR*	10	3	7	0	7
6	*Jamming LIDAR*	1	3	0	0	4
7	*Spoofing Camera*	0	0	0	1	0
8	*Tampering Camera*	0	0	0	1	0
9	*DoS Camera*	0	0	0	1	0
10	*Spoofing ultrasonic*	10	6	7	0	7
11	*Jamming ultrasonic*	10	3	3	0	4
12	*Spoofing GPS*	4	3	3	0	4
13	*Jamming GPS*	1	3	0	0	1

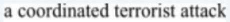
6

A simplified approach for dynamic security risk management in c: 7: 39|43 - 7: 300|323 (0)



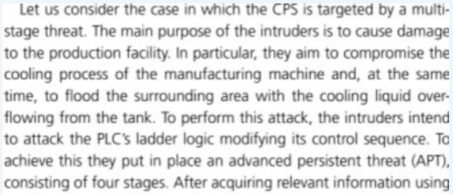
are not fast enough. Therefore, the crash risks in these cases are high.

A simplified approach for dynamic security risk management in c: 8 - 8 (0)



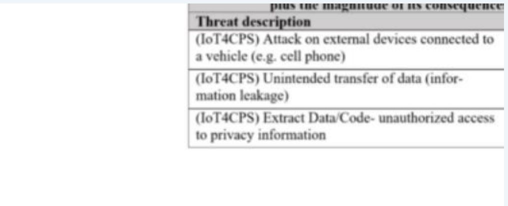
a coordinated terrorist attack

Achieving critical system survivability through software archit: 14 - 14 (0)



Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using

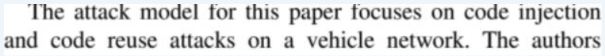
Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)



plus the magnitude of its consequence

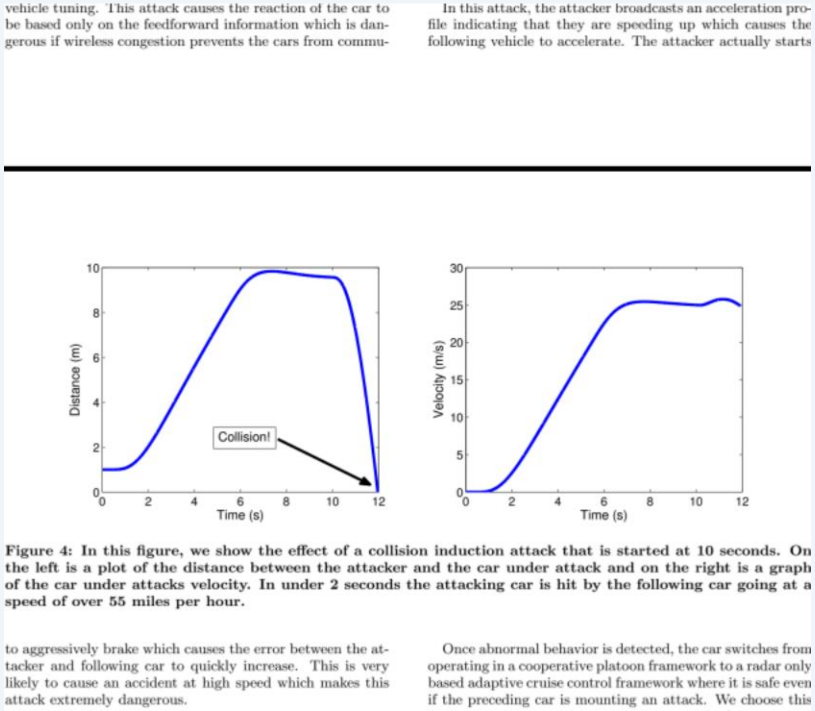
Threat description
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)
(IoT4CPS) Unintended transfer of data (information leakage)
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information

Digital Twins for Dependability Improvement of Autonomous Drivi: 10: 57|196 - 10: 316|296 (0)



The attack model for this paper focuses on code injection and code reuse attacks on a vehicle network. The authors

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)



vehicle tuning. This attack causes the reaction of the car to be based only on the feedforward information which is dangerous if wireless congestion prevents the cars from commu-

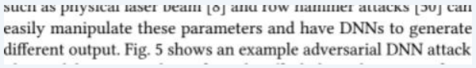
In this attack, the attacker broadcasts an acceleration profile indicating that they are speeding up which causes the following vehicle to accelerate. The attacker actually starts

Figure 4: In this figure, we show the effect of a collision induction attack that is started at 10 seconds. On the left is a plot of the distance between the attacker and the car under attack and on the right is a graph of the car under attacks velocity. In under 2 seconds the attacking car is hit by the following car going at a speed of over 55 miles per hour.

to aggressively brake which causes the error between the attacker and following car to quickly increase. This is very likely to cause an accident at high speed which makes this attack extremely dangerous.

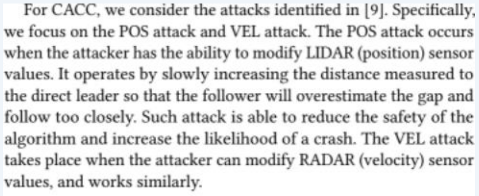
Once abnormal behavior is detected, the car switches from operating in a cooperative platoon framework to a radar only based adaptive cruise control framework where it is safe even if the preceding car is mounting an attack. We choose this

Is your commute driving you crazy a study of misbehavior in veh: 5 - 6 (0)



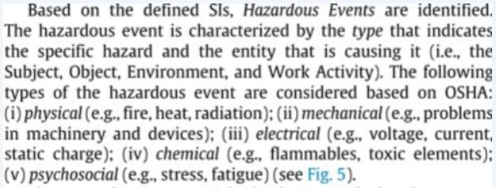
such as physical laser beam [8] and row hammer attacks [30] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



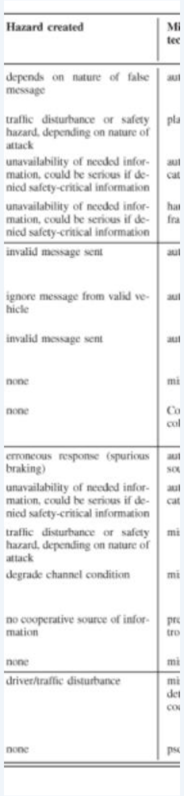
For CACC, we consider the attacks identified in [9]. Specifically, we focus on the POS attack and VEL attack. The POS attack occurs when the attacker has the ability to modify LIDAR (position) sensor values. It operates by slowly increasing the distance measured to the direct leader so that the follower will overestimate the gap and follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack takes place when the attacker can modify RADAR (velocity) sensor values, and works similarly.

Network and system level security in connected vehicle applicat: 3 - 3 (0)



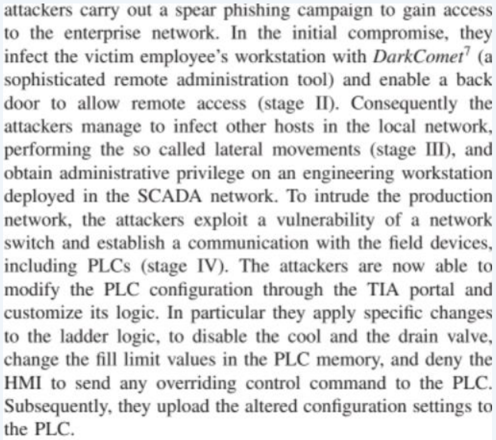
Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates the specific hazard and the entity that is causing it (i.e., the Subject, Object, Environment, and Work Activity). The following types of the hazardous event are considered based on OSHA: (i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current, static charge); (iv) *chemical* (e.g., flammables, toxic elements); (v) *psychosocial* (e.g., stress, fatigue) (see Fig. 5).

Ontology development for run-time safety management methodology: 5 - 5 (0)



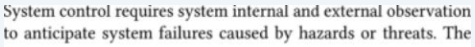
Hazard created	Mi tec
depends on nature of false message	aut
traffic disturbance or safety hazard, depending on nature of attack	pla
unavailability of needed information, could be serious if denied safety-critical information	aut cat
unavailability of needed information, could be serious if denied safety-critical information	ha fra
invalid message sent	aut
ignore message from valid vehicle	aut
invalid message sent	aut
none	mi
none	Co col
erroneous response (spurious braking)	aut sou
unavailability of needed information, could be serious if denied safety-critical information	aut cat
traffic disturbance or safety hazard, depending on nature of attack	mi
degrade channel condition	mi
no cooperative source of information	pro tro
none	mi
driver/traffic disturbance	mi det co
none	ps

Potential cyberattacks on automated vehicles: 8: 410|300 - 8: 500|705 (0)



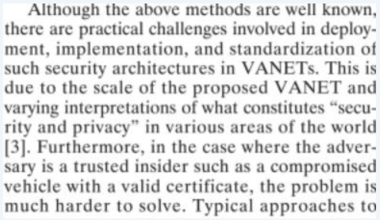
attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with *DarkComet*[7] (a sophisticated remote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral movements (stage III), and obtain administrative privilege on an engineering workstation deployed in the SCADA network. To intrude the production network, the attackers exploit a vulnerability of a network switch and establish a communication with the field devices, including PLCs (stage IV). The attackers are now able to modify the PLC configuration through the TIA portal and customize its logic. In particular they apply specific changes to the ladder logic, to disable the cool and the drain valve, change the fill limit values in the PLC memory, and deny the HMI to send any overriding control command to the PLC. Subsequently, they upload the altered configuration settings to the PLC.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



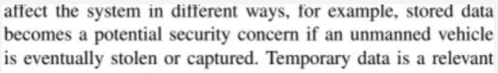
System control requires system internal and external observation to anticipate system failures caused by hazards or threats. The

SARA Security Automotive Risk Analysis Method: 7 - 7 (0)



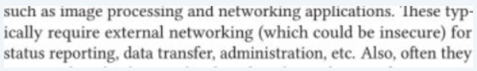
Although the above methods are well known, there are practical challenges involved in deployment, implementation, and standardization of such security architectures in VANETs. This is due to the scale of the proposed VANET and varying interpretations of what constitutes "security and privacy" in various areas of the world [3]. Furthermore, in the case where the adversary is a trusted insider such as a compromised vehicle with a valid certificate, the problem is much harder to solve. Typical approaches to

Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)

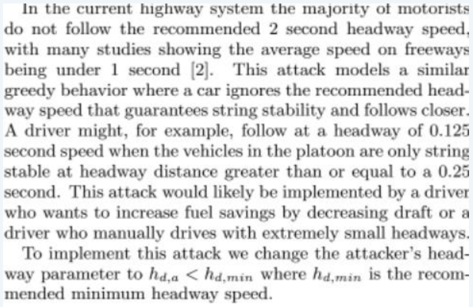
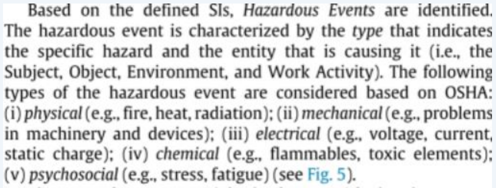


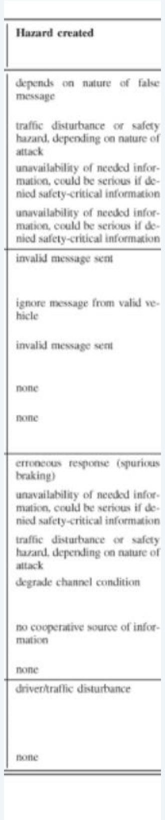
affect the system in different ways, for example, stored data becomes a potential security concern if an unmanned vehicle is eventually stolen or captured. Temporary data is a relevant

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)



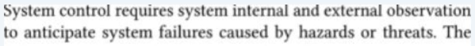
such as image processing and networking applications. These typically require external networking (which could be insecure) for status reporting, data transfer, administration, etc. Also, often they

	VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)
Integration\Safety Hazards\Hazard Source\Internal	 In the current highway system the majority of motorists do not follow the recommended 2 second headway speed, with many studies showing the average speed on freeways being under 1 second [2]. This attack models a similar greedy behavior where a car ignores the recommended headway speed that guarantees string stability and follows closer. A driver might, for example, follow at a headway of 0.125 second speed when the vehicles in the platoon are only string stable at headway distance greater than or equal to a 0.25 second. This attack would likely be implemented by a driver who wants to increase fuel savings by decreasing draft or a driver who manually drives with extremely small headways. To implement this attack we change the attacker's headway parameter to $h_{d,a} < h_{d,min}$ where $h_{d,min}$ is the recommended minimum headway speed. Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)
	 Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates the specific hazard and the entity that is causing it (i.e., the Subject, Object, Environment, and Work Activity). The following types of the hazardous event are considered based on OSHA: (i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current, static charge); (iv) *chemical* (e.g., flammables, toxic elements); (v) *psychosocial* (e.g., stress, fatigue) (see Fig. 5). Ontology development for run-time safety management methodology: 5 - 5 (0)



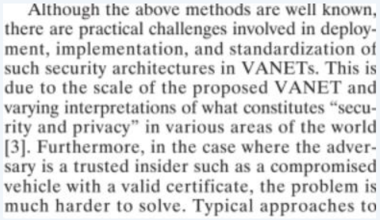
Hazard created
depends on nature of false message
traffic disturbance or safety hazard, depending on nature of attack
unavailability of needed information, could be serious if denied safety-critical information
unavailability of needed information, could be serious if denied safety-critical information
invalid message sent
ignore message from valid vehicle
invalid message sent
none
none
erroneous response (spurious braking)
unavailability of needed information, could be serious if denied safety-critical information
traffic disturbance or safety hazard, depending on nature of attack
degrade channel condition
no cooperative source of information
none
driver/traffic disturbance
none

Potential cyberattacks on automated vehicles: 8: 405|291 - 8: 485|708 (0)



System control requires system internal and external observation to anticipate system failures caused by hazards or threats. The

SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

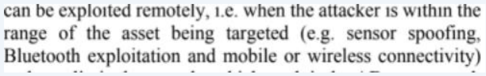


Although the above methods are well known, there are practical challenges involved in deployment, implementation, and standardization of such security architectures in VANETs. This is due to the scale of the proposed VANET and varying interpretations of what constitutes "security and privacy" in various areas of the world [3]. Furthermore, in the case where the adversary is a trusted insider such as a compromised vehicle with a valid certificate, the problem is much harder to solve. Typical approaches to

Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)

s due to a failure

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)



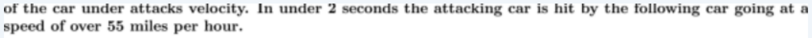
can be exploited remotely, i.e. when the attacker is within the range of the asset being targeted (e.g. sensor spoofing, Bluetooth exploitation and mobile or wireless connectivity)

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

Integration\Safety Hazards\Hazard Cause Old (Deprecated)\Collision with Another Object

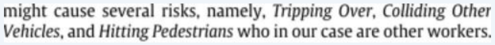
Impact
Collision
Collision
Collision
Dissolved platoon

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 300|693 - 5: 503|739 (0)



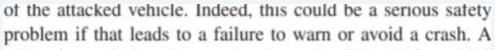
of the car under attacks velocity. In under 2 seconds the attacking car is hit by the following car going at a speed of over 55 miles per hour.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



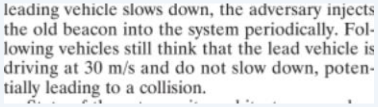
might cause several risks, namely, *Tripping Over*, *Colliding Other Vehicles*, and *Hitting Pedestrians* who in our case are other workers.

Ontology development for run-time safety management methodology: 6 - 6 (0)



of the attacked vehicle. Indeed, this could be a serious safety problem if that leads to a failure to warn or avoid a crash. A

Potential cyberattacks on automated vehicles: 9 - 9 (0)



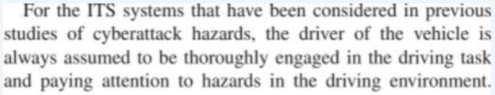
leading vehicle slows down, the adversary injects the old beacon into the system periodically. Following vehicles still think that the lead vehicle is driving at 30 m/s and do not slow down, potentially leading to a collision.

Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)

Integration\Safety Hazards\Hazard Cause Old (Deprecated)\Environmental Conditions

chemical (e.g., flammables, toxic elements)

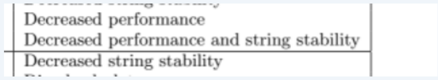
Ontology development for run-time safety management methodology: 5 - 5 (0)



For the ITS systems that have been considered in previous studies of cyberattack hazards, the driver of the vehicle is always assumed to be thoroughly engaged in the driving task and paying attention to hazards in the driving environment.

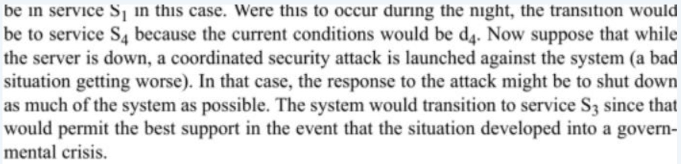
Potential cyberattacks on automated vehicles: 3 - 3 (0)

Integration\Safety Hazards\Hazard Cause Old (Deprecated)\Hardware Failure



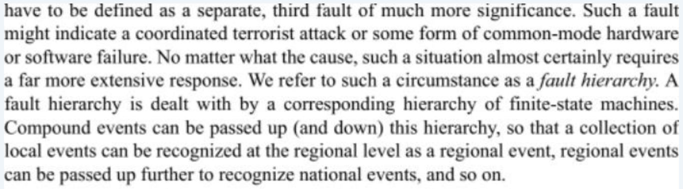
Decreased performance
Decreased performance and string stability
Decreased string stability

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 296|612 - 5: 519|646 (0)



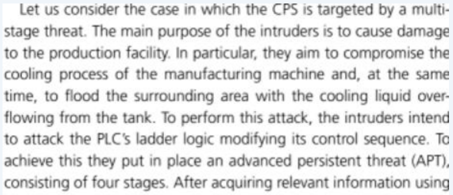
be in service S_1 in this case. Were this to occur during the night, the transition would be to service S_4 because the current conditions would be d_4. Now suppose that while the server is down, a coordinated security attack is launched against the system (a bad situation getting worse). In that case, the response to the attack might be to shut down as much of the system as possible. The system would transition to service S_3 since that would permit the best support in the event that the situation developed into a governmental crisis.

Achieving critical system survivability through software archit: 10 - 10 (0)



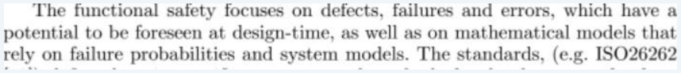
have to be defined as a separate, third fault of much more significance. Such a fault might indicate a coordinated terrorist attack or some form of common-mode hardware or software failure. No matter what the cause, such a situation almost certainly requires a far more extensive response. We refer to such a circumstance as a *fault hierarchy*. A fault hierarchy is dealt with by a corresponding hierarchy of finite-state machines. Compound events can be passed up (and down) this hierarchy, so that a collection of local events can be recognized at the regional level as a regional event, regional events can be passed up further to recognize national events, and so on.

Achieving critical system survivability through software archit: 14 - 14 (0)



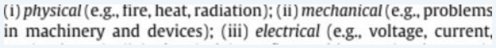
Let us consider the case in which the CPS is targeted by a multistage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using

Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)



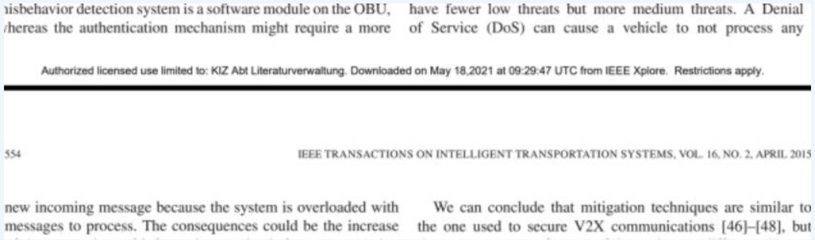
The functional safety focuses on defects, failures and errors, which have a potential to be foreseen at design-time, as well as on mathematical models that rely on failure probabilities and system models. The standards, (e.g. ISO26262

Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



(i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current,

Ontology development for run-time safety management methodology: 5 - 5 (0)



ıisbehavior detection system is a software module on the OBU, /hereas the authentication mechanism might require a more

have fewer low threats but more medium threats. A Denial of Service (DoS) can cause a vehicle to not process any

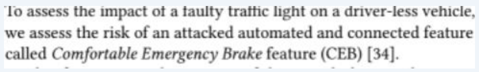
Authorized licensed use limited to: KIZ Abt Literaturverwaltung. Downloaded on May 18,2021 at 09:29:47 UTC from IEEE Xplore. Restrictions apply.

554 IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015

new incoming message because the system is overloaded with messages to process. The consequences could be the increase

We can conclude that mitigation techniques are similar to the one used to secure V2X communications [46]–[48], but

Potential cyberattacks on automated vehicles: 8 - 9 (0)

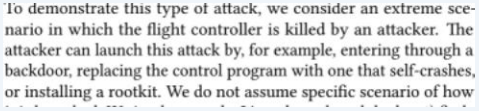


To assess the impact of a faulty traffic light on a driver-less vehicle, we assess the risk of an attacked automated and connected feature called *Comfortable Emergency Brake* feature (CEB) [34].

SARA Security Automotive Risk Analysis Method: 9 - 9 (0)

s due to a failure

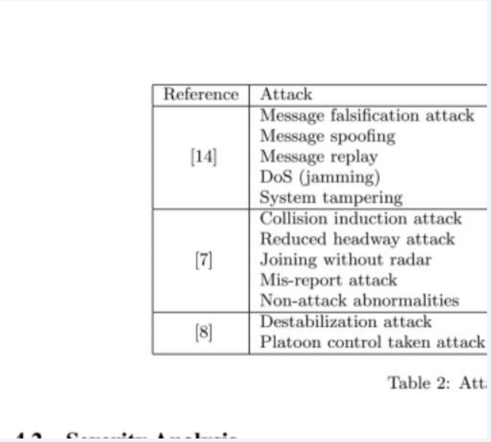
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)



To demonstrate this type of attack, we consider an extreme scenario in which the flight controller is killed by an attacker. The attacker can launch this attack by, for example, entering through a backdoor, replacing the control program with one that self-crashes, or installing a rootkit. We do not assume specific scenario of how

VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

Integration\Safety Hazards\Hazard Cause Old (Deprecated)\External attack

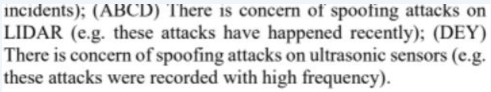


Reference	Attack
[14]	Message falsification attack Message spoofing Message replay DoS (jamming) System tampering
[7]	Collision induction attack Reduced headway attack Joining without radar Mis-report attack Non-attack abnormalities
[8]	Destabilization attack Platoon control taken attack

Table 2: Att

4.2 Security Analysis

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 47|560 - 5: 296|781 (0)

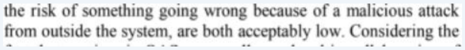


incidents); (ABCD) There is concern of spoofing attacks on LIDAR (e.g. these attacks have happened recently); (DEY) There is concern of spoofing attacks on ultrasonic sensors (e.g. these attacks were recorded with high frequency).

A simplified approach for dynamic security risk management in c: 6 - 6 (0)

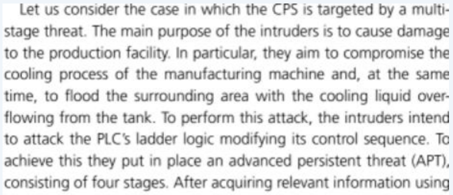
a coordinated terrorist attack

Achieving critical system survivability through software archit: 14 - 14 (0)



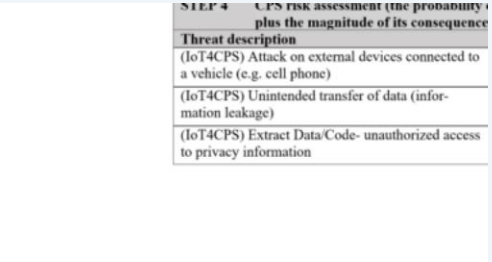
the risk of something going wrong because of a malicious attack from outside the system, are both acceptably low. Considering the

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



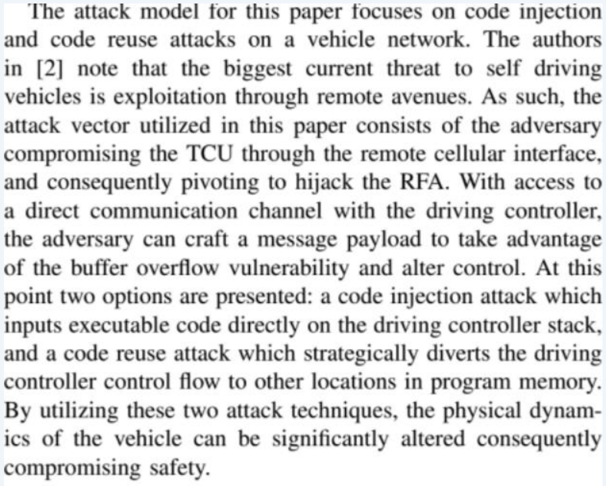
Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat (APT), consisting of four stages. After acquiring relevant information using

Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)



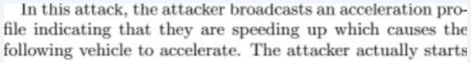
STEP 4	CPS risk assessment (the probability plus the magnitude of its consequence
Threat description	
(IoT4CPS) Attack on external devices connected to a vehicle (e.g. cell phone)	
(IoT4CPS) Unintended transfer of data (information leakage)	
(IoT4CPS) Extract Data/Code- unauthorized access to privacy information	

Digital Twins for Dependability Improvement of Autonomous Drivi: 9: 65|176 - 9: 315|305 (0)



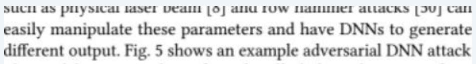
The attack model for this paper focuses on code injection and code reuse attacks on a vehicle network. The authors in [2] note that the biggest current threat to self driving vehicles is exploitation through remote avenues. As such, the attack vector utilized in this paper consists of the adversary compromising the TCU through the remote cellular interface, and consequently pivoting to hijack the RFA. With access to a direct communication channel with the driving controller, the adversary can craft a message payload to take advantage of the buffer overflow vulnerability and alter control. At this point two options are presented: a code injection attack which inputs executable code directly on the driving controller stack, and a code reuse attack which strategically diverts the driving controller control flow to other locations in program memory. By utilizing these two attack techniques, the physical dynamics of the vehicle can be significantly altered consequently compromising safety.

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)



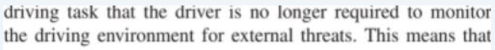
In this attack, the attacker broadcasts an acceleration profile indicating that they are speeding up which causes the following vehicle to accelerate. The attacker actually starts

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



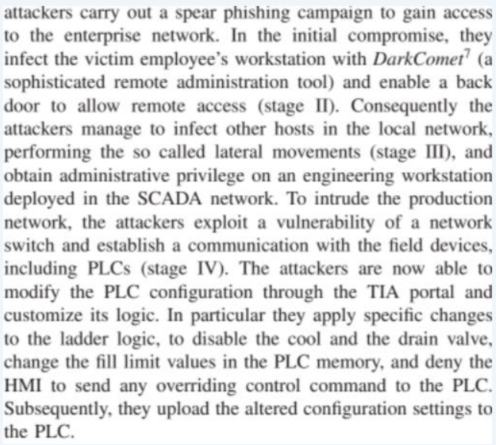
such as physical laser beam [8] and row hammer attacks [30] can easily manipulate these parameters and have DNNs to generate different output. Fig. 5 shows an example adversarial DNN attack

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)



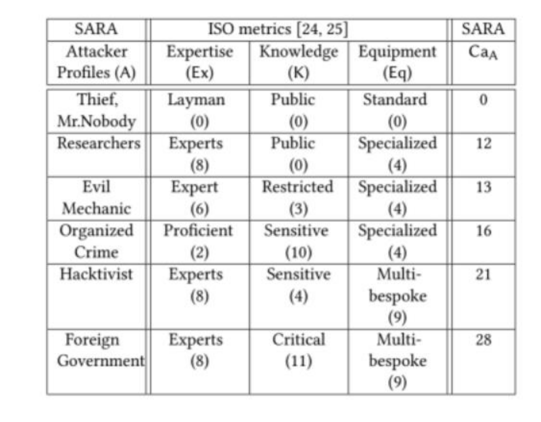
driving task that the driver is no longer required to monitor the driving environment for external threats. This means that

Potential cyberattacks on automated vehicles: 3 - 3 (0)



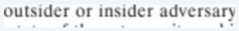
attackers carry out a spear phishing campaign to gain access to the enterprise network. In the initial compromise, they infect the victim employee's workstation with *DarkComet*[7] (a sophisticated remote administration tool) and enable a back door to allow remote access (stage II). Consequently the attackers manage to infect other hosts in the local network, performing the so called lateral movements (stage III), and obtain administrative privilege on an engineering workstation deployed in the SCADA network. To intrude the production network, the attackers exploit a vulnerability of a network switch and establish a communication with the field devices, including PLCs (stage IV). The attackers are now able to modify the PLC configuration through the TIA portal and customize its logic. In particular they apply specific changes to the ladder logic, to disable the cool and the drain valve, change the fill limit values in the PLC memory, and deny the HMI to send any overriding control command to the PLC. Subsequently, they upload the altered configuration settings to the PLC.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



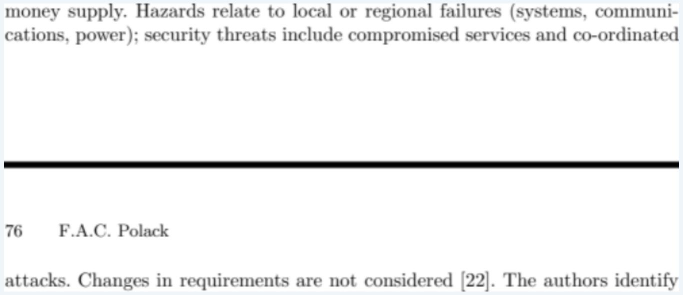
SARA	ISO metrics [24, 25]			SARA
Attacker Profiles (A)	Expertise (Ex)	Knowledge (K)	Equipment (Eq)	Ca_A
Thief, Mr.Nobody	Layman (0)	Public (0)	Standard (0)	0
Researchers	Experts (8)	Public (0)	Specialized (4)	12
Evil Mechanic	Expert (6)	Restricted (3)	Specialized (4)	13
Organized Crime	Proficient (2)	Sensitive (10)	Specialized (4)	16
Hacktivist	Experts (8)	Sensitive (4)	Multi-bespoke (9)	21
Foreign Government	Experts (8)	Critical (11)	Multi-bespoke (9)	28

SARA Security Automotive Risk Analysis Method: 6: 30|348 - 6: 307|561 (0)



outsider or insider adversary

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

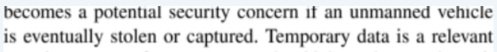


money supply. Hazards relate to local or regional failures (systems, communications, power); security threats include compromised services and co-ordinated

76 F.A.C. Polack

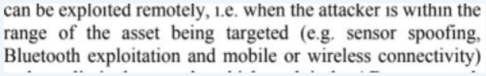
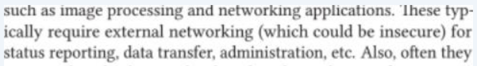
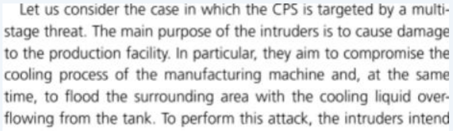
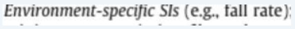
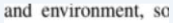
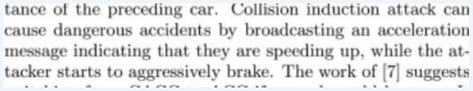
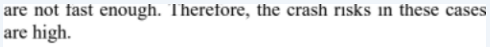
attacks. Changes in requirements are not considered [22]. The authors identify

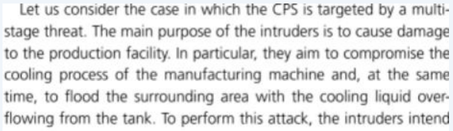
Self-organisation for Survival in Complex Computer Architecture: 10 - 11 (0)



becomes a potential security concern if an unmanned vehicle is eventually stolen or captured. Temporary data is a relevant

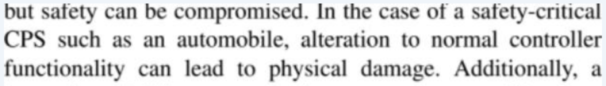
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 4 - 4 (0)

	 can be exploited remotely, i.e. when the attacker is within the range of the asset being targeted (e.g. sensor spoofing, Bluetooth exploitation and mobile or wireless connectivity) TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)
	 such as image processing and networking applications. These typically require external networking (which could be insecure) for status reporting, data transfer, administration, etc. Also, often they VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)
Integration\Safety Hazards\Safety Quality Factors\Environmental	 Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid over-flowing from the tank. To perform this attack, the intruders intend Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)
	 *Environment-specific SIs* (e.g., fall rate); Ontology development for run-time safety management methodology: 5 - 5 (0)
	 and environment, sc STUART ReSilient archiTecture to dynamically manage Unmanned ae: 2 - 2 (0)
Integration\Safety Hazards\Safety Quality Factors\Property	 tance of the preceding car. Collision induction attack can cause dangerous accidents by broadcasting an acceleration message indicating that they are speeding up, while the attacker starts to aggressively brake. The work of [7] suggests A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)
	 are not fast enough. Therefore, the crash risks in these cases are high. A simplified approach for dynamic security risk management in c: 8 - 8 (0)



Let us consider the case in which the CPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend

Countering targeted cyber-physical attacks using anomaly detect: 6 - 6 (0)

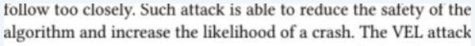


but safety can be compromised. In the case of a safety-critical CPS such as an automobile, alteration to normal controller functionality can lead to physical damage. Additionally, a

Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)

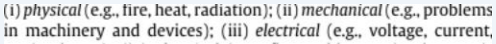
Property Damage

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



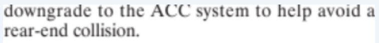
follow too closely. Such attack is able to reduce the safety of the algorithm and increase the likelihood of a crash. The VEL attack

Network and system level security in connected vehicle applicat: 3 - 3 (0)



(i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current,

Ontology development for run-time safety management methodology: 5 - 5 (0)

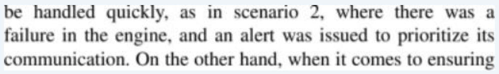


downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)

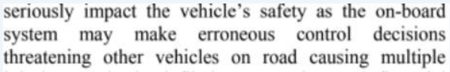
an accident involving two cars. It

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)

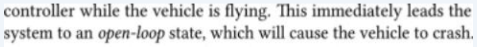


be handled quickly, as in scenario 2, where there was a failure in the engine, and an alert was issued to prioritize its communication. On the other hand, when it comes to ensuring

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)

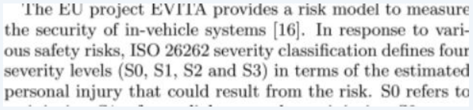

seriously impact the vehicle's safety as the on-board system may make erroneous control decisions threatening other vehicles on road causing multiple

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

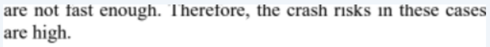

controller while the vehicle is flying. This immediately leads the system to an *open-loop* state, which will cause the vehicle to crash.

VirtualDrone virtual sensing actuation and communication for at: 6 - 6 (0)

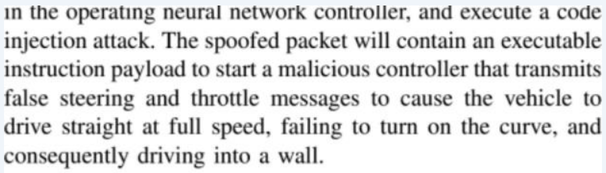
Integration\Safety Hazards\Safety Quality Factors\Health


The EU project EVITA provides a risk model to measure the security of in-vehicle systems [16]. In response to various safety risks, ISO 26262 severity classification defines four severity levels (S0, S1, S2 and S3) in terms of the estimated personal injury that could result from the risk. S0 refers to

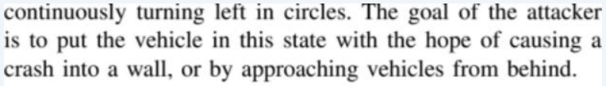
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)


are not fast enough. Therefore, the crash risks in these cases are high.

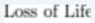
A simplified approach for dynamic security risk management in c: 8 - 8 (0)


in the operating neural network controller, and execute a code injection attack. The spoofed packet will contain an executable instruction payload to start a malicious controller that transmits false steering and throttle messages to cause the vehicle to drive straight at full speed, failing to turn on the curve, and consequently driving into a wall.

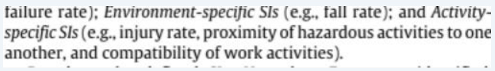
Integrated moving target defense and control reconfiguration fo: 8 - 8 (0)


continuously turning left in circles. The goal of the attacker is to put the vehicle in this state with the hope of causing a crash into a wall, or by approaching vehicles from behind.

Integrated moving target defense and control reconfiguration fo: 9 - 9 (0)

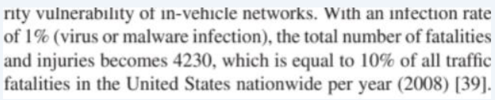

Loss of Life

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



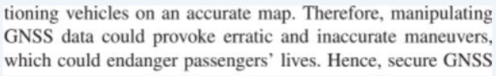
failure rate); *Environment-specific SIs* (e.g., fall rate); and *Activity-specific SIs* (e.g., injury rate, proximity of hazardous activities to one another, and compatibility of work activities).

Ontology development for run-time safety management methodology: 5 - 5 (0)



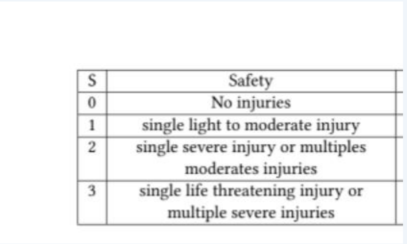
rity vulnerability of in-vehicle networks. With an infection rate of 1% (virus or malware infection), the total number of fatalities and injuries becomes 4230, which is equal to 10% of all traffic fatalities in the United States nationwide per year (2008) [39].

Potential cyberattacks on automated vehicles: 7 - 7 (0)



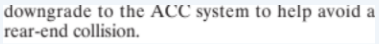
tioning vehicles on an accurate map. Therefore, manipulating GNSS data could provoke erratic and inaccurate maneuvers, which could endanger passengers' lives. Hence, secure GNSS

Potential cyberattacks on automated vehicles: 9 - 9 (0)



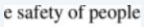
S	Safety
0	No injuries
1	single light to moderate injury
2	single severe injury or multiples moderates injuries
3	single life threatening injury or multiple severe injuries

SARA Security Automotive Risk Analysis Method: 8: 14|600 - 8: 221|720 (0)



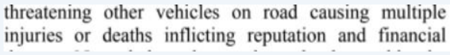
downgrade to the ACC system to help avoid a rear-end collision.

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



e safety of people

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 2 - 2 (0)

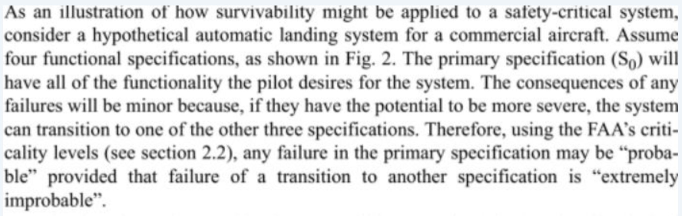


threatening other vehicles on road causing multiple injuries or deaths inflicting reputation and financial

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

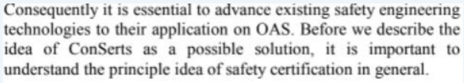
Integration\Integration of Security and Safety\Loosely Integrated

A simplified approach for dynamic security risk management in c: 4: 302|130 - 4: 351|265 (0)



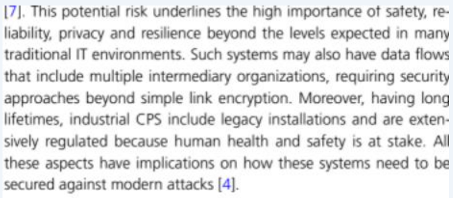
As an illustration of how survivability might be applied to a safety-critical system, consider a hypothetical automatic landing system for a commercial aircraft. Assume four functional specifications, as shown in Fig. 2. The primary specification (S_0) will have all of the functionality the pilot desires for the system. The consequences of any failures will be minor because, if they have the potential to be more severe, the system can transition to one of the other three specifications. Therefore, using the FAA's criticality levels (see section 2.2), any failure in the primary specification may be "probable" provided that failure of a transition to another specification is "extremely improbable".

Achieving critical system survivability through software archit: 11 - 11 (0)



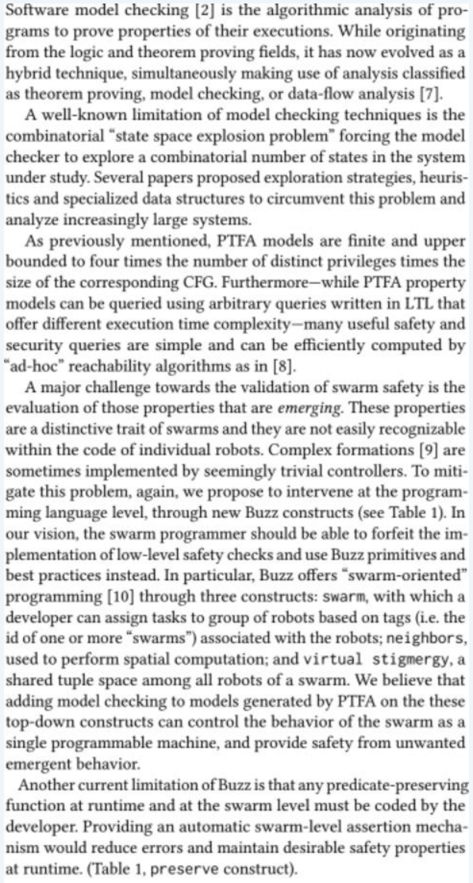
Consequently it is essential to advance existing safety engineering technologies to their application on OAS. Before we describe the idea of ConSerts as a possible solution, it is important to understand the principle idea of safety certification in general.

Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)



[7]. This potential risk underlines the high importance of safety, reliability, privacy and resilience beyond the levels expected in many traditional IT environments. Such systems may also have data flows that include multiple intermediary organizations, requiring security approaches beyond simple link encryption. Moreover, having long lifetimes, industrial CPS include legacy installations and are extensively regulated because human health and safety is at stake. All these aspects have implications on how these systems need to be secured against modern attacks [4].

Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)



Software model checking [2] is the algorithmic analysis of programs to prove properties of their executions. While originating from the logic and theorem proving fields, it has now evolved as a hybrid technique, simultaneously making use of analysis classified as theorem proving, model checking, or data-flow analysis [7].

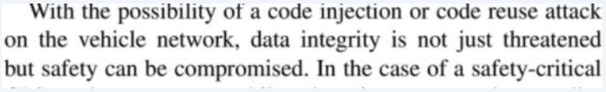
A well-known limitation of model checking techniques is the combinatorial "state space explosion problem" forcing the model checker to explore a combinatorial number of states in the system under study. Several papers proposed exploration strategies, heuristics and specialized data structures to circumvent this problem and analyze increasingly large systems.

As previously mentioned, PTFA models are finite and upper bounded to four times the number of distinct privileges times the size of the corresponding CFG. Furthermore—while PTFA property models can be queried using arbitrary queries written in LTL that offer different execution time complexity—many useful safety and security queries are simple and can be efficiently computed by "ad-hoc" reachability algorithms as in [8].

A major challenge towards the validation of swarm safety is the evaluation of those properties that are *emerging*. These properties are a distinctive trait of swarms and they are not easily recognizable within the code of individual robots. Complex formations [9] are sometimes implemented by seemingly trivial controllers. To mitigate this problem, again, we propose to intervene at the programming language level, through new Buzz constructs (see Table 1). In our vision, the swarm programmer should be able to forfeit the implementation of low-level safety checks and use Buzz primitives and best practices instead. In particular, Buzz offers "swarm-oriented" programming [10] through three constructs: `swarm`, with which a developer can assign tasks to group of robots based on tags (i.e. the id of one or more "swarms") associated with the robots; `neighbors`, used to perform spatial computation; and `virtual stigmergy`, a shared tuple space among all robots of a swarm. We believe that adding model checking to models generated by PTFA on the these top-down constructs can control the behavior of the swarm as a single programmable machine, and provide safety from unwanted emergent behavior.

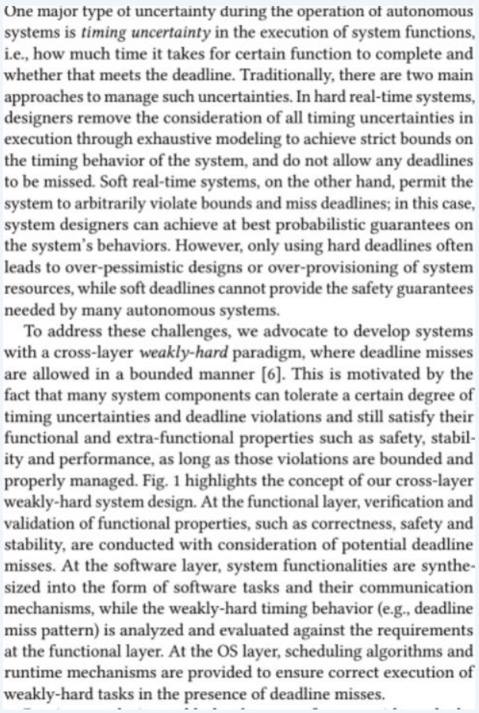
Another current limitation of Buzz is that any predicate-preserving function at runtime and at the swarm level must be coded by the developer. Providing an automatic swarm-level assertion mechanism would reduce errors and maintain desirable safety properties at runtime. (Table 1, `preserve` construct).

Engineering safety in swarm robotics: 4 - 4 (0)



With the possibility of a code injection or code reuse attack on the vehicle network, data integrity is not just threatened but safety can be compromised. In the case of a safety-critical

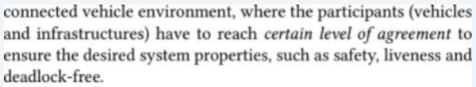
Integrated moving target defense and control reconfiguration fo: 3 - 3 (0)



One major type of uncertainty during the operation of autonomous systems is *timing uncertainty* in the execution of system functions, i.e., how much time it takes for certain function to complete and whether that meets the deadline. Traditionally, there are two main approaches to manage such uncertainties. In hard real-time systems, designers remove the consideration of all timing uncertainties in execution through exhaustive modeling to achieve strict bounds on the timing behavior of the system, and do not allow any deadlines to be missed. Soft real-time systems, on the other hand, permit the system to arbitrarily violate bounds and miss deadlines; in this case, system designers can achieve at best probabilistic guarantees on the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems.

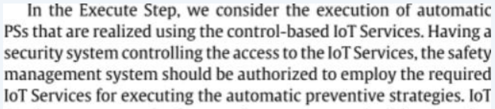
To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses.

Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)



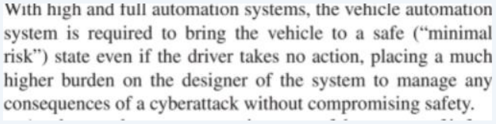
connected vehicle environment, where the participants (vehicles and infrastructures) have to reach *certain level of agreement* to ensure the desired system properties, such as safety, liveness and deadlock-free.

Network and system level security in connected vehicle applicat: 5 - 5 (0)



In the Execute Step, we consider the execution of automatic PSs that are realized using the control-based IoT Services. Having a security system controlling the access to the IoT Services, the safety management system should be authorized to employ the required IoT Services for executing the automatic preventive strategies. IoT

Ontology development for run-time safety management methodology: 6 - 6 (0)



With high and full automation systems, the vehicle automation system is required to bring the vehicle to a safe ("minimal risk") state even if the driver takes no action, placing a much higher burden on the designer of the system to manage any consequences of a cyberattack without compromising safety.

Potential cyberattacks on automated vehicles: 3 - 3 (0)

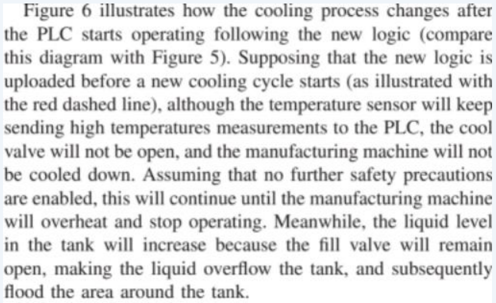
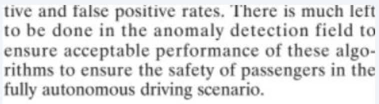


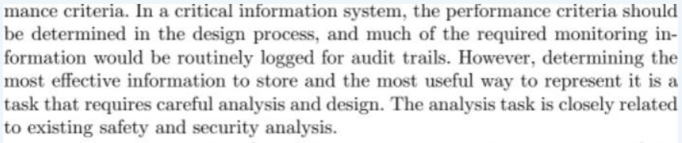
Figure 6 illustrates how the cooling process changes after the PLC starts operating following the new logic (compare this diagram with Figure 5). Supposing that the new logic is uploaded before a new cooling cycle starts (as illustrated with the red dashed line), although the temperature sensor will keep sending high temperatures measurements to the PLC, the cool valve will not be open, and the manufacturing machine will not be cooled down. Assuming that no further safety precautions are enabled, this will continue until the manufacturing machine will overheat and stop operating. Meanwhile, the liquid level in the tank will increase because the fill valve will remain open, making the liquid overflow the tank, and subsequently flood the area around the tank.

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



tive and false positive rates. There is much left to be done in the anomaly detection field to ensure acceptable performance of these algorithms to ensure the safety of passengers in the fully autonomous driving scenario.

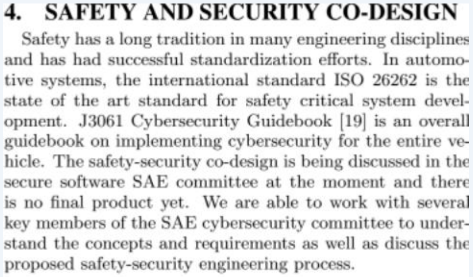
Security vulnerabilities of connected vehicle streams and their: 3 - 3 (0)



mance criteria. In a critical information system, the performance criteria should be determined in the design process, and much of the required monitoring information would be routinely logged for audit trails. However, determining the most effective information to store and the most useful way to represent it is a task that requires careful analysis and design. The analysis task is closely related to existing safety and security analysis.

Self-organisation for Survival in Complex Computer Architecture: 12 - 12 (0)

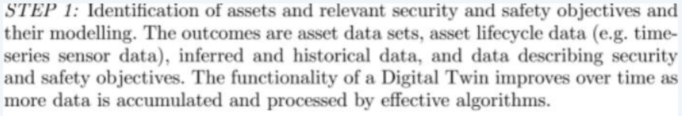
Integration\Integration of Security and Safety\Fully Integrated



4. SAFETY AND SECURITY CO-DESIGN

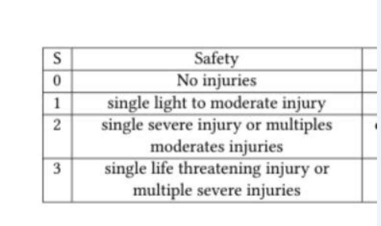
Safety has a long tradition in many engineering disciplines and has had successful standardization efforts. In automotive systems, the international standard ISO 26262 is the state of the art standard for safety critical system development. J3061 Cybersecurity Guidebook [19] is an overall guidebook on implementing cybersecurity for the entire vehicle. The safety-security co-design is being discussed in the secure software SAE committee at the moment and there is no final product yet. We are able to work with several key members of the SAE cybersecurity committee to understand the concepts and requirements as well as discuss the proposed safety-security engineering process.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)



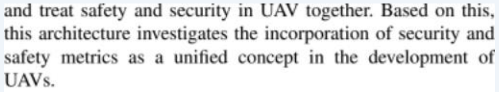
STEP 1: Identification of assets and relevant security and safety objectives and their modelling. The outcomes are asset data sets, asset lifecycle data (e.g. time-series sensor data), inferred and historical data, and data describing security and safety objectives. The functionality of a Digital Twin improves over time as more data is accumulated and processed by effective algorithms.

Digital Twins for Dependability Improvement of Autonomous Drivi: 6 - 6 (0)



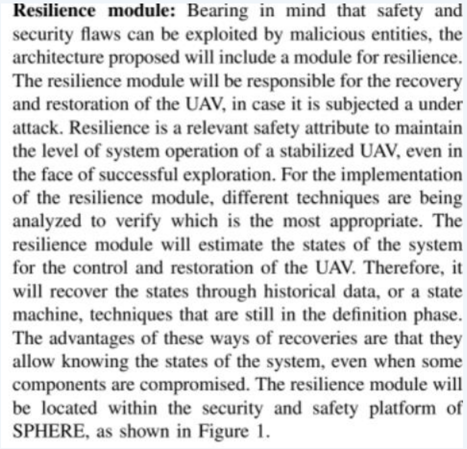
S	Safety
0	No injuries
1	single light to moderate injury
2	single severe injury or multiples moderates injuries
3	single life threatening injury or multiple severe injuries

SARA Security Automotive Risk Analysis Method: 8: 32|597 - 8: 225|710 (0)



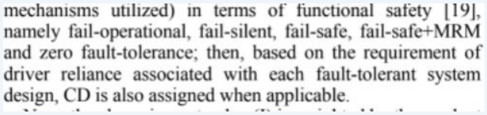
and treat safety and security in UAV together. Based on this, this architecture investigates the incorporation of security and safety metrics as a unified concept in the development of UAVs.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)



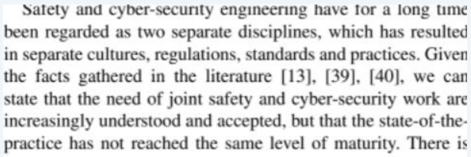
Resilience module: Bearing in mind that safety and security flaws can be exploited by malicious entities, the architecture proposed will include a module for resilience. The resilience module will be responsible for the recovery and restoration of the UAV, in case it is subjected a under attack. Resilience is a relevant safety attribute to maintain the level of system operation of a stabilized UAV, even in the face of successful exploration. For the implementation of the resilience module, different techniques are being analyzed to verify which is the most appropriate. The resilience module will estimate the states of the system for the control and restoration of the UAV. Therefore, it will recover the states through historical data, or a state machine, techniques that are still in the definition phase. The advantages of these ways of recoveries are that they allow knowing the states of the system, even when some components are compromised. The resilience module will be located within the security and safety platform of SPHERE, as shown in Figure 1.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)



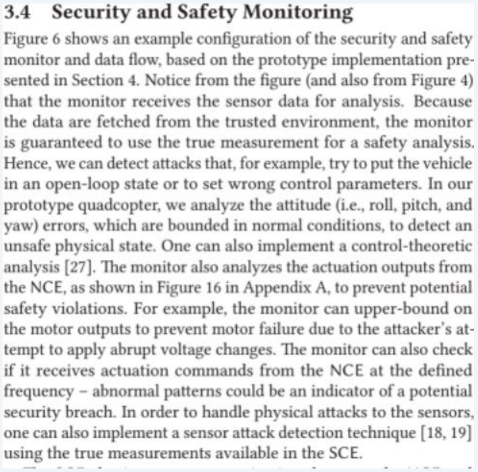
mechanisms utilized) in terms of functional safety [19], namely fail-operational, fail-silent, fail-safe, fail-safe+MRM and zero fault-tolerance; then, based on the requirement of driver reliance associated with each fault-tolerant system design, CD is also assigned when applicable.

TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)



Safety and cyber-security engineering have for a long time been regarded as two separate disciplines, which has resulted in separate cultures, regulations, standards and practices. Given the facts gathered in the literature [13], [39], [40], we can state that the need of joint safety and cyber-security work are increasingly understood and accepted, but that the state-of-the-practice has not reached the same level of maturity. There is

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)

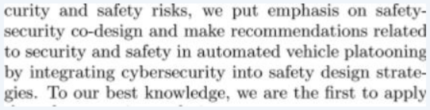


3.4 Security and Safety Monitoring

Figure 6 shows an example configuration of the security and safety monitor and data flow, based on the prototype implementation presented in Section 4. Notice from the figure (and also from Figure 4) that the monitor receives the sensor data for analysis. Because the data are fetched from the trusted environment, the monitor is guaranteed to use the true measurement for a safety analysis. Hence, we can detect attacks that, for example, try to put the vehicle in an open-loop state or to set wrong control parameters. In our prototype quadcopter, we analyze the attitude (i.e., roll, pitch, and yaw) errors, which are bounded in normal conditions, to detect an unsafe physical state. One can also implement a control-theoretic analysis [27]. The monitor also analyzes the actuation outputs from the NCE, as shown in Figure 16 in Appendix A, to prevent potential safety violations. For example, the monitor can upper-bound on the motor outputs to prevent motor failure due to the attacker's attempt to apply abrupt voltage changes. The monitor can also check if it receives actuation commands from the NCE at the defined frequency – abnormal patterns could be an indicator of a potential security breach. In order to handle physical attacks to the sensors, one can also implement a sensor attack detection technique [18, 19] using the true measurements available in the SCE.

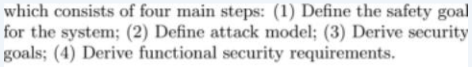
VirtualDrone virtual sensing actuation and communication for at: 5 - 5 (0)

Modeling Approach



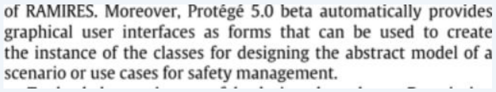
curity and safety risks, we put emphasis on safety-security co-design and make recommendations related to security and safety in automated vehicle platooning by integrating cybersecurity into safety design strategies. To our best knowledge, we are the first to apply

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 2 - 2 (0)



which consists of four main steps: (1) Define the safety goal for the system; (2) Define attack model; (3) Derive security goals; (4) Derive functional security requirements.

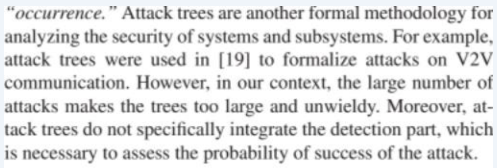
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)



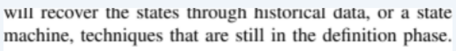
of RAMIRES. Moreover, Protégé 5.0 beta automatically provides graphical user interfaces as forms that can be used to create the instance of the classes for designing the abstract model of a scenario or use cases for safety management.

Implementation

Ontology development for run-time safety management methodology: 12 - 12 (0)

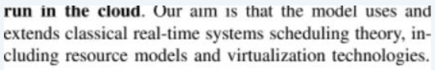

"occurrence." Attack trees are another formal methodology for analyzing the security of systems and subsystems. For example, attack trees were used in [19] to formalize attacks on V2V communication. However, in our context, the large number of attacks makes the trees too large and unwieldy. Moreover, attack trees do not specifically integrate the detection part, which is necessary to assess the probability of success of the attack.

Potential cyberattacks on automated vehicles: 4 - 4 (0)


will recover the states through historical data, or a state machine, techniques that are still in the definition phase.

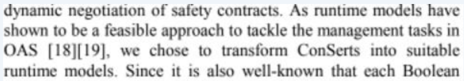
Implementation

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 3 (0)

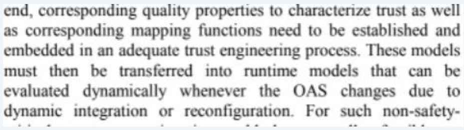

run in the cloud. Our aim is that the model uses and extends classical real-time systems scheduling theory, including resource models and virtualization technologies.

Towards a Framework for Safe and Secure Adaptive Collaborative: 5 - 5 (0)

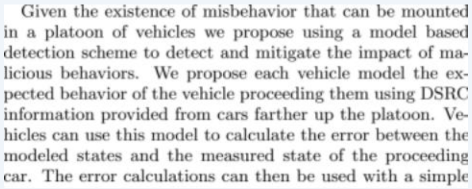
Modeling Approach\Model@Runtime


dynamic negotiation of safety contracts. As runtime models have shown to be a feasible approach to tackle the management tasks in OAS [18][19], we chose to transform ConSerts into suitable runtime models. Since it is also well-known that each Boolean

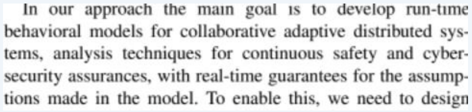
Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)


end, corresponding quality properties to characterize trust as well as corresponding mapping functions need to be established and embedded in an adequate trust engineering process. These models must then be transferred into runtime models that can be evaluated dynamically whenever the OAS changes due to dynamic integration or reconfiguration. For such non-safety-

Approaching runtime trust assurance in open adaptive systems: 6 - 6 (0)

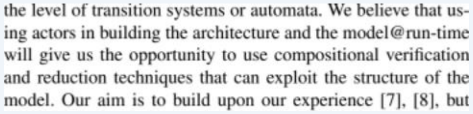

Given the existence of misbehavior that can be mounted in a platoon of vehicles we propose using a model based detection scheme to detect and mitigate the impact of malicious behaviors. We propose each vehicle model the expected behavior of the vehicle proceeding them using DSRC information provided from cars farther up the platoon. Vehicles can use this model to calculate the error between the modeled states and the measured state of the proceeding car. The error calculations can then be used with a simple

Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)



In our approach the main goal is to develop run-time behavioral models for collaborative adaptive distributed systems, analysis techniques for continuous safety and cyber-security assurances, with real-time guarantees for the assumptions made in the model. To enable this, we need to design

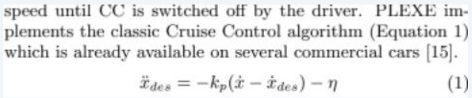
Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)



the level of transition systems or automata. We believe that using actors in building the architecture and the model@run-time will give us the opportunity to use compositional verification and reduction techniques that can exploit the structure of the model. Our aim is to build upon our experience [7], [8], but

Towards a Framework for Safe and Secure Adaptive Collaborative: 2 - 2 (0)

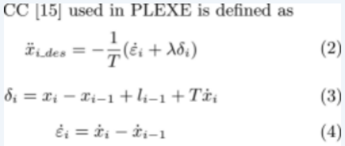
Modeling Approach\Context of the Model\Modeling the System\State and behavior



speed until CC is switched off by the driver. PLEXE implements the classic Cruise Control algorithm (Equation 1) which is already available on several commercial cars [15].

$$\ddot{x}_{des} = -k_p(\dot{x} - \dot{x}_{des}) - \eta \qquad (1)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



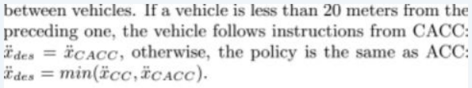
CC [15] used in PLEXE is defined as

$$\ddot{x}_{i\_des} = -\frac{1}{T}(\dot{\varepsilon}_i + \lambda\delta_i) \qquad (2)$$

$$\delta_i = x_i - x_{i-1} + l_{i-1} + T\dot{x}_i \qquad (3)$$

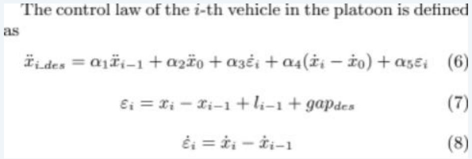
$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (4)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



between vehicles. If a vehicle is less than 20 meters from the preceding one, the vehicle follows instructions from CACC: $\ddot{x}_{des} = \ddot{x}_{CACC}$, otherwise, the policy is the same as ACC: $\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{CACC})$.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



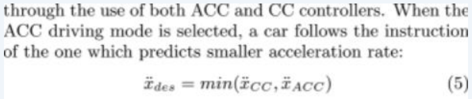
The control law of the i-th vehicle in the platoon is defined as

$$\ddot{x}_{i\_des} = \alpha_1\ddot{x}_{i-1} + \alpha_2\ddot{x}_0 + \alpha_3\dot{\varepsilon}_i + \alpha_4(\dot{x}_i - \dot{x}_0) + \alpha_5\varepsilon_i \qquad (6)$$

$$\varepsilon_i = x_i - x_{i-1} + l_{i-1} + gap_{des} \qquad (7)$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (8)$$

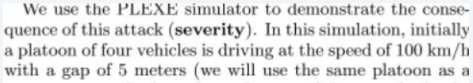
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



through the use of both ACC and CC controllers. When the ACC driving mode is selected, a car follows the instruction of the one which predicts smaller acceleration rate:

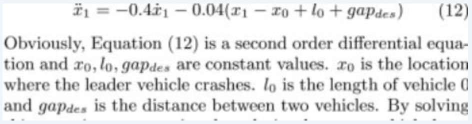
$$\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{ACC}) \quad (5)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



We use the PLEXE simulator to demonstrate the consequence of this attack (**severity**). In this simulation, initially a platoon of four vehicles is driving at the speed of 100 km/h with a gap of 5 meters (we will use the same platoon as a

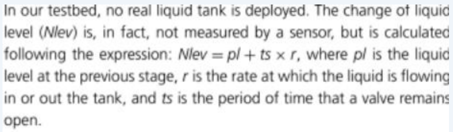
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)



$$\ddot{x}_1 = -0.4\dot{x}_1 - 0.04(x_1 - x_0 + l_0 + gap_{des}) \quad (12)$$

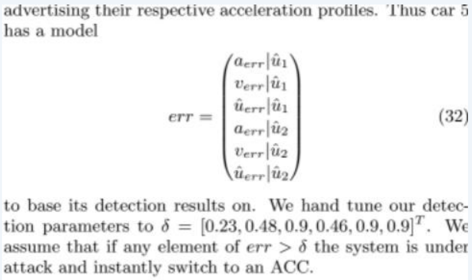
Obviously, Equation (12) is a second order differential equation and x_0, l_0, gap_{des} are constant values. x_0 is the location where the leader vehicle crashes. l_0 is the length of vehicle 0 and gap_{des} is the distance between two vehicles. By solving

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 7 - 7 (0)



In our testbed, no real liquid tank is deployed. The change of liquid level (*Nlev*) is, in fact, not measured by a sensor, but is calculated following the expression: $Nlev = pl + ts \times r$, where *pl* is the liquid level at the previous stage, *r* is the rate at which the liquid is flowing in or out the tank, and *ts* is the period of time that a valve remains open.

Countering targeted cyber-physical attacks using anomaly detect: 5 - 5 (0)

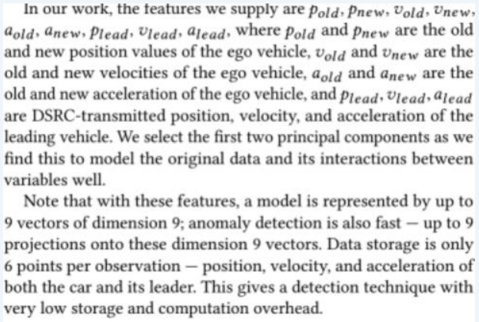


advertising their respective acceleration profiles. Thus car 5 has a model

$$err = \begin{pmatrix} a_{err}|\hat{u}_1 \\ v_{err}|\hat{u}_1 \\ \hat{u}_{err}|\hat{u}_1 \\ a_{err}|\hat{u}_2 \\ v_{err}|\hat{u}_2 \\ \hat{u}_{err}|\hat{u}_2 \end{pmatrix} \quad (32)$$

to base its detection results on. We hand tune our detection parameters to $\delta = [0.23, 0.48, 0.9, 0.46, 0.9, 0.9]^T$. We assume that if any element of $err > \delta$ the system is under attack and instantly switch to an ACC.

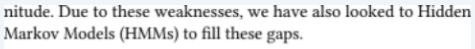
Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



In our work, the features we supply are $p_{old}, p_{new}, v_{old}, v_{new}, a_{old}, a_{new}, p_{lead}, v_{lead}, a_{lead}$, where p_{old} and p_{new} are the old and new position values of the ego vehicle, v_{old} and v_{new} are the old and new velocities of the ego vehicle, a_{old} and a_{new} are the old and new acceleration of the ego vehicle, and $p_{lead}, v_{lead}, a_{lead}$ are DSRC-transmitted position, velocity, and acceleration of the leading vehicle. We select the first two principal components as we find this to model the original data and its interactions between variables well.

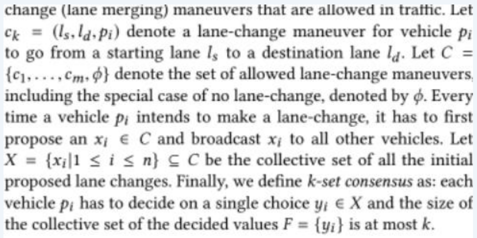
Note that with these features, a model is represented by up to 9 vectors of dimension 9; anomaly detection is also fast — up to 9 projections onto these dimension 9 vectors. Data storage is only 6 points per observation — position, velocity, and acceleration of both the car and its leader. This gives a detection technique with very low storage and computation overhead.

Network and system level security in connected vehicle applicat: 4 - 4 (0)



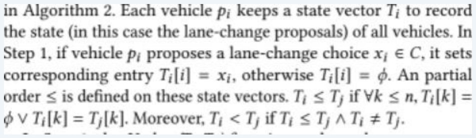
nitude. Due to these weaknesses, we have also looked to Hidden Markov Models (HMMs) to fill these gaps.

Network and system level security in connected vehicle applicat: 4 - 4 (0)



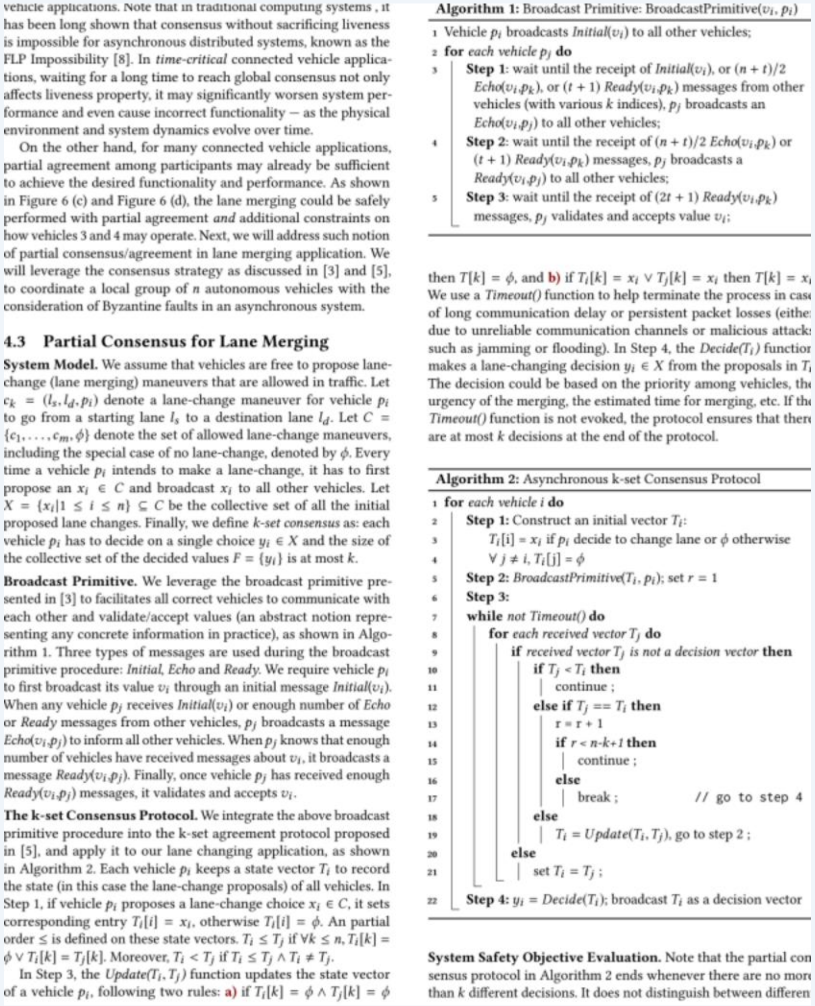
change (lane merging) maneuvers that are allowed in traffic. Let $c_k = (l_s, l_d, p_i)$ denote a lane-change maneuver for vehicle p_i to go from a starting lane l_s to a destination lane l_d. Let $C = \{c_1, \ldots, c_m, \phi\}$ denote the set of allowed lane-change maneuvers, including the special case of no lane-change, denoted by ϕ. Every time a vehicle p_i intends to make a lane-change, it has to first propose an $x_i \in C$ and broadcast x_i to all other vehicles. Let $X = \{x_i | 1 \leq i \leq n\} \subseteq C$ be the collective set of all the initial proposed lane changes. Finally, we define *k-set consensus* as: each vehicle p_i has to decide on a single choice $y_i \in X$ and the size of the collective set of the decided values $F = \{y_i\}$ is at most k.

Network and system level security in connected vehicle applicat: 6 - 6 (0)



in Algorithm 2. Each vehicle p_i keeps a state vector T_i to record the state (in this case the lane-change proposals) of all vehicles. In Step 1, if vehicle p_i proposes a lane-change choice $x_i \in C$, it sets corresponding entry $T_i[i] = x_i$, otherwise $T_i[i] = \phi$. An partial order $\leq$ is defined on these state vectors. $T_i \leq T_j$ if $\forall k \leq n, T_i[k] = \phi \vee T_i[k] = T_j[k]$. Moreover, $T_i < T_j$ if $T_i \leq T_j \wedge T_i \neq T_j$.

Network and system level security in connected vehicle applicat: 6 - 6 (0)



vehicle applications. Note that in traditional computing systems , it has been long shown that consensus without sacrificing liveness is impossible for asynchronous distributed systems, known as the FLP Impossibility [8]. In *time-critical* connected vehicle applications, waiting for a long time to reach global consensus not only affects liveness property, it may significantly worsen system performance and even cause incorrect functionality — as the physical environment and system dynamics evolve over time.

On the other hand, for many connected vehicle applications, partial agreement among participants may already be sufficient to achieve the desired functionality and performance. As shown in Figure 6 (c) and Figure 6 (d), the lane merging could be safely performed with partial agreement *and* additional constraints on how vehicles 3 and 4 may operate. Next, we will address such notion of partial consensus/agreement in lane merging application. We will leverage the consensus strategy as discussed in [3] and [5], to coordinate a local group of n autonomous vehicles with the consideration of Byzantine faults in an asynchronous system.

4.3 Partial Consensus for Lane Merging

System Model. We assume that vehicles are free to propose lane-change (lane merging) maneuvers that are allowed in traffic. Let $c_k = (l_s, l_d, p_i)$ denote a lane-change maneuver for vehicle p_i to go from a starting lane l_s to a destination lane l_d. Let $C = \{c_1, \ldots, c_m, \phi\}$ denote the set of allowed lane-change maneuvers, including the special case of no lane-change, denoted by ϕ. Every time a vehicle p_i intends to make a lane-change, it has to first propose an $x_i \in C$ and broadcast x_i to all other vehicles. Let $X = \{x_i | 1 \leq i \leq n\} \subseteq C$ be the collective set of all the initial proposed lane changes. Finally, we define *k-set consensus* as: each vehicle p_i has to decide on a single choice $y_i \in X$ and the size of the collective set of the decided values $F = \{y_i\}$ is at most k.

Broadcast Primitive. We leverage the broadcast primitive presented in [3] to facilitates all correct vehicles to communicate with each other and validate/accept values (an abstract notion representing any concrete information in practice), as shown in Algorithm 1. Three types of messages are used during the broadcast primitive procedure: *Initial*, *Echo* and *Ready*. We require vehicle p_i to first broadcast its value v_i through an initial message *Initial*(v_i). When any vehicle p_j receives *Initial*(v_i) or enough number of *Echo* or *Ready* messages from other vehicles, p_j broadcasts a message *Echo*(v_i,p_j) to inform all other vehicles. When p_j knows that enough number of vehicles have received messages about v_i, it broadcasts a message *Ready*(v_i,p_j). Finally, once vehicle p_j has received enough *Ready*(v_i,p_j) messages, it validates and accepts v_i.

The k-set Consensus Protocol. We integrate the above broadcast primitive procedure into the k-set agreement protocol proposed in [5], and apply it to our lane changing application, as shown in Algorithm 2. Each vehicle p_i keeps a state vector T_i to record the state (in this case the lane-change proposals) of all vehicles. In Step 1, if vehicle p_i proposes a lane-change choice $x_i \in C$, it sets corresponding entry $T_i[i] = x_i$, otherwise $T_i[i] = \phi$. An partial order $\leq$ is defined on these state vectors. $T_i \leq T_j$ if $\forall k \leq n, T_i[k] = \phi \vee T_i[k] = T_j[k]$. Moreover, $T_i < T_j$ if $T_i \leq T_j \wedge T_i \neq T_j$.

In Step 3, the *Update*(T_i, T_j) function updates the state vector of a vehicle p_i, following two rules: **a)** if $T_i[k] = \phi \wedge T_j[k] = \phi$ then $T[k] = \phi$, and **b)** if $T_i[k] = x_i \vee T_j[k] = x_i$ then $T[k] = x_i$ We use a *Timeout()* function to help terminate the process in cas of long communication delay or persistent packet losses (eithe due to unreliable communication channels or malicious attack such as jamming or flooding). In Step 4, the *Decide*(T_i) functio makes a lane-changing decision $y_i \in X$ from the proposals in T_i The decision could be based on the priority among vehicles, th urgency of the merging, the estimated time for merging, etc. If th *Timeout()* function is not evoked, the protocol ensures that ther are at most k decisions at the end of the protocol.

Algorithm 1: Broadcast Primitive: BroadcastPrimitive(v_i, p_i)

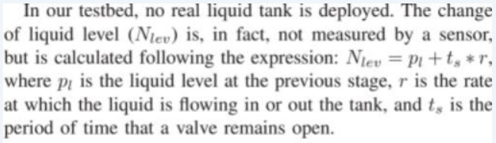
```
Vehicle $p_i$ broadcasts Initial($v_i$) to all other vehicles;
for each vehicle $p_j$ do
    Step 1: wait until the receipt of Initial($v_i$), or $(n + t)/2$
      Echo($v_i$,$p_k$), or $(t + 1)$ Ready($v_i$,$p_k$) messages from other
      vehicles (with various $k$ indices), $p_j$ broadcasts an
      Echo($v_i$,$p_j$) to all other vehicles;
    Step 2: wait until the receipt of $(n + t)/2$ Echo($v_i$,$p_k$) or
      $(t + 1)$ Ready($v_i$,$p_k$) messages, $p_j$ broadcasts a
      Ready($v_i$,$p_j$) to all other vehicles;
    Step 3: wait until the receipt of $(2t + 1)$ Ready($v_i$,$p_k$)
      messages, $p_j$ validates and accepts value $v_i$;
```

Algorithm 2: Asynchronous k-set Consensus Protocol

```
for each vehicle $i$ do
    Step 1: Construct an initial vector $T_i$:
        $T_i[i] = x_i$ if $p_i$ decide to change lane or $\phi$ otherwise
        $\forall j \neq i, T_i[j] = \phi$
    Step 2: BroadcastPrimitive($T_i$, $p_i$); set $r = 1$
    Step 3:
    while not Timeout() do
        for each received vector $T_j$ do
            if received vector $T_j$ is not a decision vector then
                if $T_j < T_i$ then
                    continue ;
                else if $T_j == T_i$ then
                    $r = r + 1$
                    if $r < n-k+1$ then
                        continue ;
                    else
                        break ;              // go to step 4
                else
                    $T_i = Update(T_i, T_j)$, go to step 2 ;
            else
                set $T_i = T_j$ ;
    Step 4: $y_i = Decide(T_i)$; broadcast $T_i$ as a decision vector
```

System Safety Objective Evaluation. Note that the partial con sensus protocol in Algorithm 2 ends whenever there are no mor than k different decisions. It does not distinguish between differen

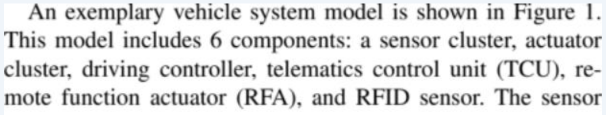
Network and system level security in connected vehicle applicat: 6 - 6 (0)



In our testbed, no real liquid tank is deployed. The change of liquid level (N_{lev}) is, in fact, not measured by a sensor, but is calculated following the expression: $N_{lev} = p_l + t_s * r$, where p_l is the liquid level at the previous stage, r is the rate at which the liquid is flowing in or out the tank, and t_s is the period of time that a valve remains open.

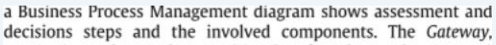
Protecting cyber physical production systems using anomaly dete: 5 - 5 (0)

Modeling Approach\Context of the Model\Modeling the System\For illustration purposes



An exemplary vehicle system model is shown in Figure 1. This model includes 6 components: a sensor cluster, actuator cluster, driving controller, telematics control unit (TCU), remote function actuator (RFA), and RFID sensor. The sensor

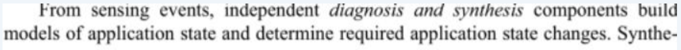
Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)



a Business Process Management diagram shows assessment and decisions steps and the involved components. The *Gateway,*

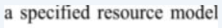
Ontology development for run-time safety management methodology: 8 - 8 (0)

Modeling Approach\Context of the Model\Modeling the System\State



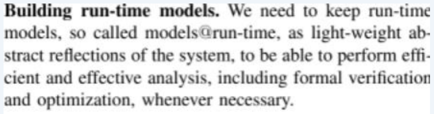
From sensing events, independent *diagnosis and synthesis* components build models of application state and determine required application state changes. Synthe-

Achieving critical system survivability through software archit: 13 - 13 (0)



a specified resource model

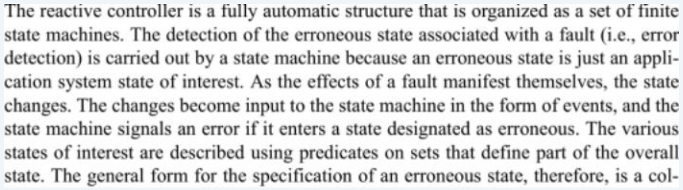
Achieving critical system survivability through software archit: 17 - 17 (0)



Building run-time models. We need to keep run-time models, so called models@run-time, as light-weight abstract reflections of the system, to be able to perform efficient and effective analysis, including formal verification and optimization, whenever necessary.

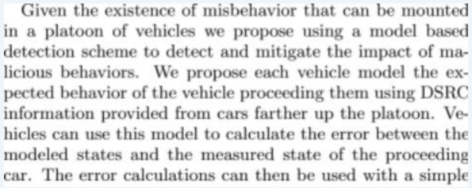
Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)

Modeling Approach\Context of the Model\Modeling the System\Behavior



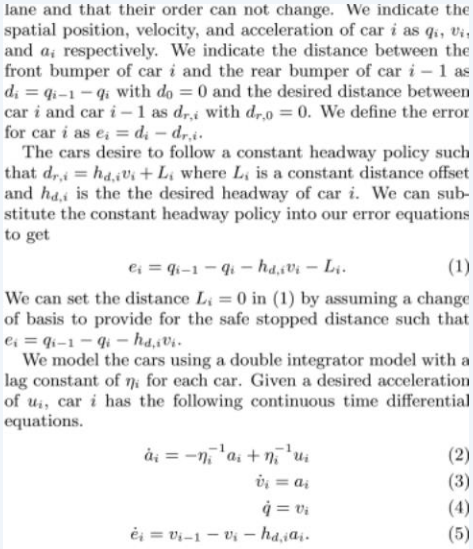
The reactive controller is a fully automatic structure that is organized as a set of finite state machines. The detection of the erroneous state associated with a fault (i.e., error detection) is carried out by a state machine because an erroneous state is just an application system state of interest. As the effects of a fault manifest themselves, the state changes. The changes become input to the state machine in the form of events, and the state machine signals an error if it enters a state designated as erroneous. The various states of interest are described using predicates on sets that define part of the overall state. The general form for the specification of an erroneous state, therefore, is a col-

Achieving critical system survivability through software archit: 14 - 14 (0)



Given the existence of misbehavior that can be mounted in a platoon of vehicles we propose using a model based detection scheme to detect and mitigate the impact of malicious behaviors. We propose each vehicle model the expected behavior of the vehicle proceeding them using DSRC information provided from cars farther up the platoon. Vehicles can use this model to calculate the error between the modeled states and the measured state of the proceeding car. The error calculations can then be used with a simple

Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)



lane and that their order can not change. We indicate the spatial position, velocity, and acceleration of car i as q_i, v_i, and a_i respectively. We indicate the distance between the front bumper of car i and the rear bumper of car $i-1$ as $d_i = q_{i-1} - q_i$ with $d_0 = 0$ and the desired distance between car i and car $i-1$ as $d_{r,i}$ with $d_{r,0} = 0$. We define the error for car i as $e_i = d_i - d_{r,i}$.

The cars desire to follow a constant headway policy such that $d_{r,i} = h_{d,i}v_i + L_i$ where L_i is a constant distance offset and $h_{d,i}$ is the the desired headway of car i. We can substitute the constant headway policy into our error equations to get

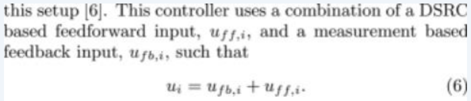
$$e_i = q_{i-1} - q_i - h_{d,i}v_i - L_i. \tag{1}$$

We can set the distance $L_i = 0$ in (1) by assuming a change of basis to provide for the safe stopped distance such that $e_i = q_{i-1} - q_i - h_{d,i}v_i$.

We model the cars using a double integrator model with a lag constant of η_i for each car. Given a desired acceleration of u_i, car i has the following continuous time differential equations.

$$\dot{a}_i = -\eta_i^{-1}a_i + \eta_i^{-1}u_i \tag{2}$$
$$\dot{v}_i = a_i \tag{3}$$
$$\dot{q} = v_i \tag{4}$$
$$\dot{e}_i = v_{i-1} - v_i - h_{d,i}a_i. \tag{5}$$

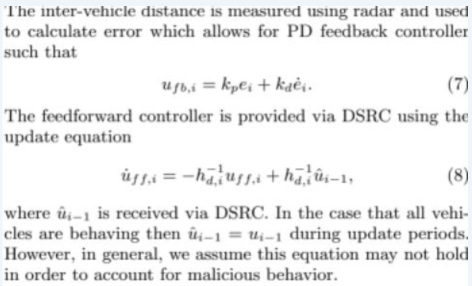
Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



this setup [6]. This controller uses a combination of a DSRC based feedforward input, $u_{ff,i}$, and a measurement based feedback input, $u_{fb,i}$, such that

$$u_i = u_{fb,i} + u_{ff,i}. \tag{6}$$

Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



The inter-vehicle distance is measured using radar and used to calculate error which allows for PD feedback controller such that

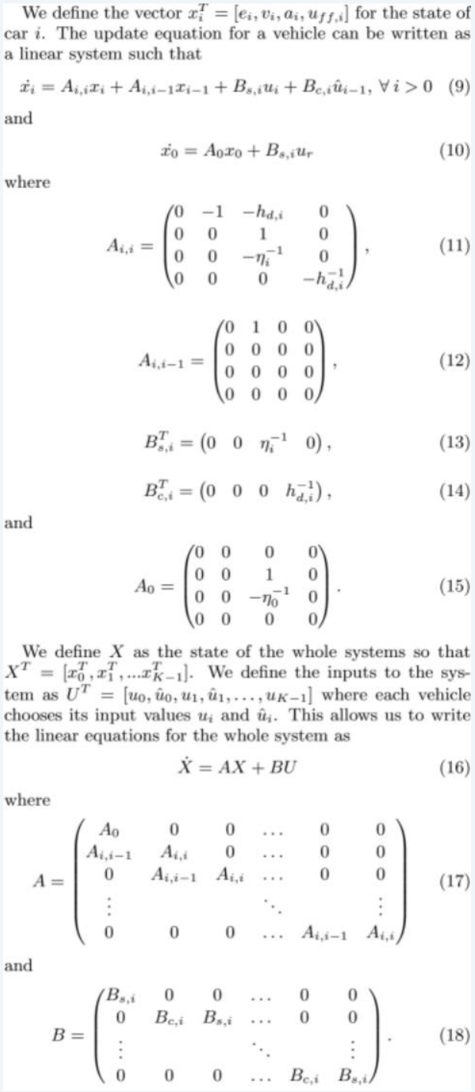
$$u_{fb,i} = k_p e_i + k_d \dot{e}_i. \tag{7}$$

The feedforward controller is provided via DSRC using the update equation

$$\dot{u}_{ff,i} = -h_{d,i}^{-1} u_{ff,i} + h_{d,i}^{-1} \hat{u}_{i-1}, \tag{8}$$

where $\hat{u}_{i-1}$ is received via DSRC. In the case that all vehicles are behaving then $\hat{u}_{i-1} = u_{i-1}$ during update periods. However, in general, we assume this equation may not hold in order to account for malicious behavior.

Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



We define the vector $x_i^T = [e_i, v_i, a_i, u_{ff,i}]$ for the state of car i. The update equation for a vehicle can be written as a linear system such that

$$\dot{x_i} = A_{i,i}x_i + A_{i,i-1}x_{i-1} + B_{s,i}u_i + B_{c,i}\hat{u}_{i-1},\ \forall\, i > 0 \quad (9)$$

and

$$\dot{x_0} = A_0 x_0 + B_{s,i}u_r \quad (10)$$

where

$$A_{i,i} = \begin{pmatrix} 0 & -1 & -h_{d,i} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_i^{-1} & 0 \\ 0 & 0 & 0 & -h_{d,i}^{-1} \end{pmatrix}, \quad (11)$$

$$A_{i,i-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (12)$$

$$B_{s,i}^T = \begin{pmatrix} 0 & 0 & \eta_i^{-1} & 0 \end{pmatrix}, \quad (13)$$

$$B_{c,i}^T = \begin{pmatrix} 0 & 0 & 0 & h_{d,i}^{-1} \end{pmatrix}, \quad (14)$$

and

$$A_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_0^{-1} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (15)$$

We define X as the state of the whole systems so that $X^T = [x_0^T, x_1^T, ...x_{K-1}^T]$. We define the inputs to the system as $U^T = [u_0, \hat{u}_0, u_1, \hat{u}_1, \dots, u_{K-1}]$ where each vehicle chooses its input values u_i and $\hat{u}_i$. This allows us to write the linear equations for the whole system as

$$\dot{X} = AX + BU \quad (16)$$

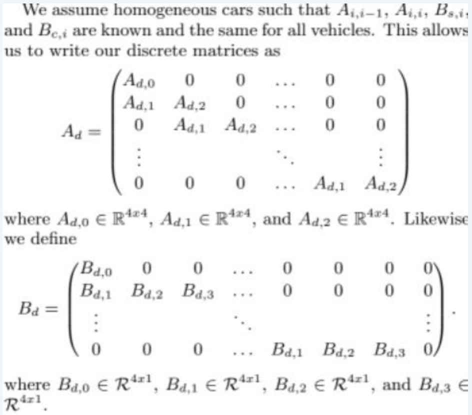
where

$$A = \begin{pmatrix} A_0 & 0 & 0 & \dots & 0 & 0 \\ A_{i,i-1} & A_{i,i} & 0 & \dots & 0 & 0 \\ 0 & A_{i,i-1} & A_{i,i} & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & A_{i,i-1} & A_{i,i} \end{pmatrix} \quad (17)$$

and

$$B = \begin{pmatrix} B_{s,i} & 0 & 0 & \dots & 0 & 0 \\ 0 & B_{c,i} & B_{s,i} & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & B_{c,i} & B_{s,i} \end{pmatrix}. \quad (18)$$

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



We assume homogeneous cars such that $A_{i,i-1}$, $A_{i,i}$, $B_{s,i}$, and $B_{c,i}$ are known and the same for all vehicles. This allows us to write our discrete matrices as

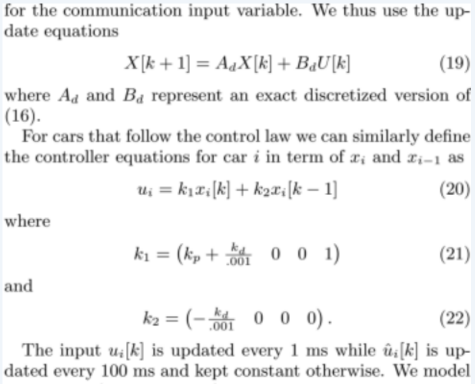
$$A_d = \begin{pmatrix} A_{d,0} & 0 & 0 & \dots & 0 & 0 \\ A_{d,1} & A_{d,2} & 0 & \dots & 0 & 0 \\ 0 & A_{d,1} & A_{d,2} & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & A_{d,1} & A_{d,2} \end{pmatrix}$$

where $A_{d,0} \in \mathbb{R}^{4x4}$, $A_{d,1} \in \mathbb{R}^{4x4}$, and $A_{d,2} \in \mathbb{R}^{4x4}$. Likewise we define

$$B_d = \begin{pmatrix} B_{d,0} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \dots & 0 & 0 & 0 & 0 \\ \vdots & & & \ddots & & & & \vdots \\ 0 & 0 & 0 & \dots & B_{d,1} & B_{d,2} & B_{d,3} & 0 \end{pmatrix}.$$

where $B_{d,0} \in \mathcal{R}^{4x1}$, $B_{d,1} \in \mathcal{R}^{4x1}$, $B_{d,2} \in \mathcal{R}^{4x1}$, and $B_{d,3} \in \mathcal{R}^{4x1}$.

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



for the communication input variable. We thus use the update equations

$$X[k+1] = A_d X[k] + B_d U[k] \tag{19}$$

where A_d and B_d represent an exact discretized version of (16).

For cars that follow the control law we can similarly define the controller equations for car i in term of x_i and x_{i-1} as

$$u_i = k_1 x_i[k] + k_2 x_i[k-1] \tag{20}$$

where

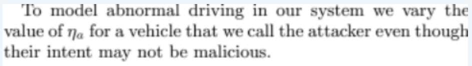
$$k_1 = \begin{pmatrix} k_p + \frac{k_d}{.001} & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

and

$$k_2 = \begin{pmatrix} -\frac{k_d}{.001} & 0 & 0 & 0 \end{pmatrix}. \tag{22}$$

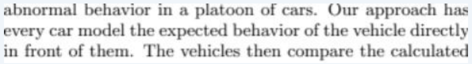
The input $u_i[k]$ is updated every 1 ms while $\hat{u}_i[k]$ is updated every 100 ms and kept constant otherwise. We model

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



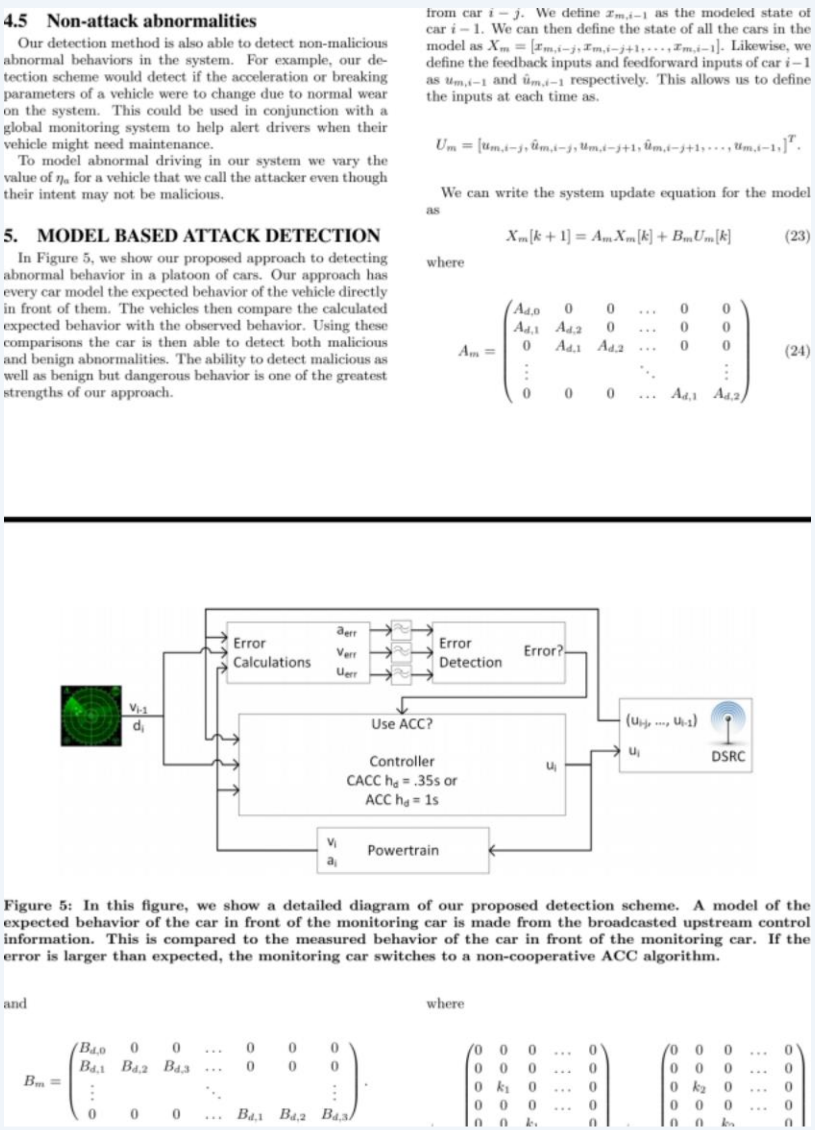
To model abnormal driving in our system we vary the value of η_a for a vehicle that we call the attacker even though their intent may not be malicious.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



abnormal behavior in a platoon of cars. Our approach has every car model the expected behavior of the vehicle directly in front of them. The vehicles then compare the calculated

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



4.5 Non-attack abnormalities

Our detection method is also able to detect non-malicious abnormal behaviors in the system. For example, our detection scheme would detect if the acceleration or breaking parameters of a vehicle were to change due to normal wear on the system. This could be used in conjunction with a global monitoring system to help alert drivers when their vehicle might need maintenance.

To model abnormal driving in our system we vary the value of η_a for a vehicle that we call the attacker even though their intent may not be malicious.

5. MODEL BASED ATTACK DETECTION

In Figure 5, we show our proposed approach to detecting abnormal behavior in a platoon of cars. Our approach has every car model the expected behavior of the vehicle directly in front of them. The vehicles then compare the calculated expected behavior with the observed behavior. Using these comparisons the car is then able to detect both malicious and benign abnormalities. The ability to detect malicious as well as benign but dangerous behavior is one of the greatest strengths of our approach.

from car $i-j$. We define $x_{m,i-1}$ as the modeled state of car $i-1$. We can then define the state of all the cars in the model as $X_m = [x_{m,i-j}, x_{m,i-j+1}, \ldots, x_{m,i-1}]$. Likewise, we define the feedback inputs and feedforward inputs of car $i-1$ as $u_{m,i-1}$ and $\hat{u}_{m,i-1}$ respectively. This allows us to define the inputs at each time as.

$$U_m = [u_{m,i-j}, \hat{u}_{m,i-j}, u_{m,i-j+1}, \hat{u}_{m,i-j+1}, \ldots, u_{m,i-1},]^T.$$

We can write the system update equation for the model as

$$X_m[k+1] = A_m X_m[k] + B_m U_m[k] \tag{23}$$

where

$$A_m = \begin{pmatrix} A_{d,0} & 0 & 0 & \ldots & 0 & 0 \\ A_{d,1} & A_{d,2} & 0 & \ldots & 0 & 0 \\ 0 & A_{d,1} & A_{d,2} & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & A_{d,1} & A_{d,2} \end{pmatrix} \tag{24}$$

Figure 5: In this figure, we show a detailed diagram of our proposed detection scheme. A model of the expected behavior of the car in front of the monitoring car is made from the broadcasted upstream control information. This is compared to the measured behavior of the car in front of the monitoring car. If the error is larger than expected, the monitoring car switches to a non-cooperative ACC algorithm.

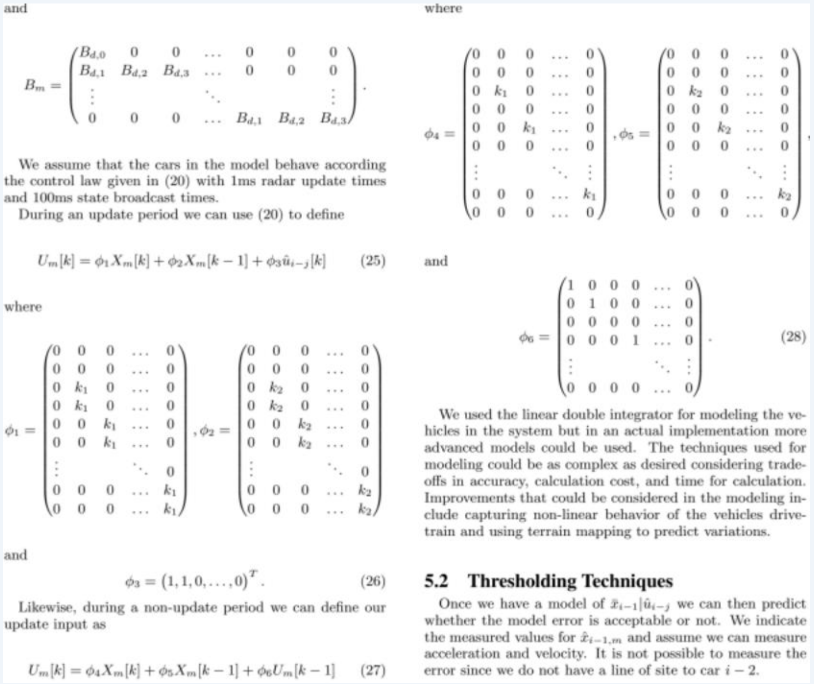
and

$$B_m = \begin{pmatrix} B_{d,0} & 0 & 0 & \ldots & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \ldots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \ldots & B_{d,1} & B_{d,2} & B_{d,3} \end{pmatrix}.$$

where

$$\begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & k_1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & k_1 & & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & k_2 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & k_2 & & 0 \end{pmatrix}$$

Is your commute driving you crazy a study of misbehavior in veh: 6 - 7 (0)



and

$$B_m = \begin{pmatrix} B_{d,0} & 0 & 0 & \dots & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & B_{d,1} & B_{d,2} & B_{d,3} \end{pmatrix}.$$

We assume that the cars in the model behave according the control law given in (20) with 1ms radar update times and 100ms state broadcast times.

During an update period we can use (20) to define

$$U_m[k] = \phi_1 X_m[k] + \phi_2 X_m[k-1] + \phi_3 \hat{u}_{i-j}[k] \qquad (25)$$

where

$$\phi_1 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & k_1 \end{pmatrix}, \phi_2 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & k_2 \end{pmatrix}$$

and

$$\phi_3 = (1, 1, 0, \dots, 0)^T. \qquad (26)$$

Likewise, during a non-update period we can define our update input as

$$U_m[k] = \phi_4 X_m[k] + \phi_5 X_m[k-1] + \phi_6 U_m[k-1] \qquad (27)$$

where

$$\phi_4 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, \phi_5 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix},$$

and

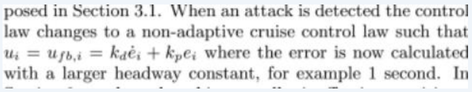
$$\phi_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \qquad (28)$$

We used the linear double integrator for modeling the vehicles in the system but in an actual implementation more advanced models could be used. The techniques used for modeling could be as complex as desired considering tradeoffs in accuracy, calculation cost, and time for calculation. Improvements that could be considered in the modeling include capturing non-linear behavior of the vehicles drivetrain and using terrain mapping to predict variations.

5.2 Thresholding Techniques

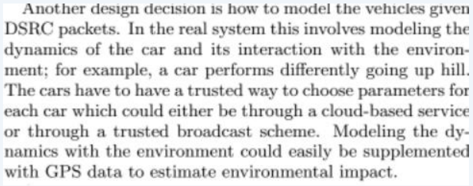
Once we have a model of $\bar{x}_{i-1}|\hat{u}_{i-j}$ we can then predict whether the model error is acceptable or not. We indicate the measured values for $\hat{x}_{i-1,m}$ and assume we can measure acceleration and velocity. It is not possible to measure the error since we do not have a line of site to car $i-2$.

Is your commute driving you crazy a study of misbehavior in veh: 7 - 7 (0)



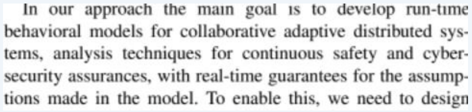
posed in Section 3.1. When an attack is detected the control law changes to a non-adaptive cruise control law such that $u_i = u_{fb,i} = k_d\dot{e}_i + k_p e_i$ where the error is now calculated with a larger headway constant, for example 1 second. In

Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



Another design decision is how to model the vehicles given DSRC packets. In the real system this involves modeling the dynamics of the car and its interaction with the environment; for example, a car performs differently going up hill. The cars have to have a trusted way to choose parameters for each car which could either be through a cloud-based service or through a trusted broadcast scheme. Modeling the dynamics with the environment could easily be supplemented with GPS data to estimate environmental impact.

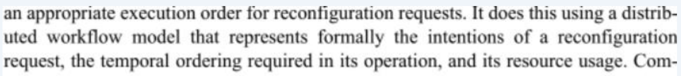
Is your commute driving you crazy a study of misbehavior in veh: 10 - 10 (0)



In our approach the main goal is to develop run-time behavioral models for collaborative adaptive distributed systems, analysis techniques for continuous safety and cyber-security assurances, with real-time guarantees for the assumptions made in the model. To enable this, we need to design

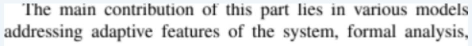
Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)

Modeling Approach\Context of the Model\Adaptation



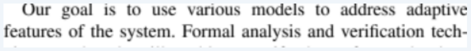
an appropriate execution order for reconfiguration requests. It does this using a distributed workflow model that represents formally the intentions of a reconfiguration request, the temporal ordering required in its operation, and its resource usage. Com-

Achieving critical system survivability through software archit: 17 - 17 (0)



The main contribution of this part lies in various models addressing adaptive features of the system, formal analysis,

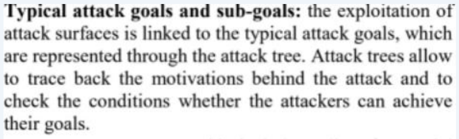
Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)



Our goal is to use various models to address adaptive features of the system. Formal analysis and verification tech-

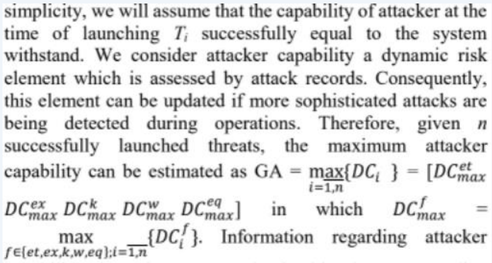
Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)

Modeling Approach\Context of the Model\Security



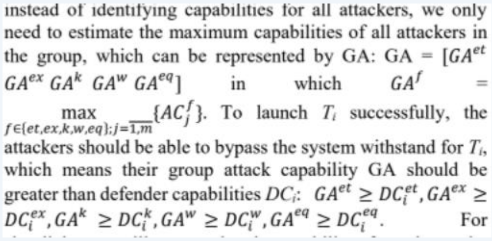
Typical attack goals and sub-goals: the exploitation of attack surfaces is linked to the typical attack goals, which are represented through the attack tree. Attack trees allow to trace back the motivations behind the attack and to check the conditions whether the attackers can achieve their goals.

A simplified approach for dynamic security risk management in c: 4 - 4 (0)



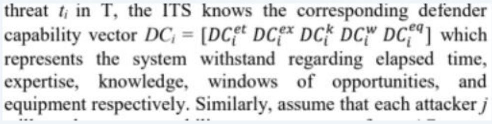
simplicity, we will assume that the capability of attacker at the time of launching T_i successfully equal to the system withstand. We consider attacker capability a dynamic risk element which is assessed by attack records. Consequently, this element can be updated if more sophisticated attacks are being detected during operations. Therefore, given n successfully launched threats, the maximum attacker capability can be estimated as GA = $\underline{\max}_{i=1,n}\{DC_i\} = [DC^{et}_{max}\ DC^{ex}_{max}\ DC^{k}_{max}\ DC^{w}_{max}\ DC^{eq}_{max}]$ in which $DC^{f}_{max} = \max_{f\in\{et,ex,k,w,eq\};i=\overline{1,n}}\{DC^{f}_{i}\}$. Information regarding attacker

A simplified approach for dynamic security risk management in c: 5 - 5 (0)



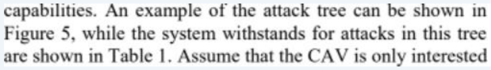
instead of identifying capabilities for all attackers, we only need to estimate the maximum capabilities of all attackers in the group, which can be represented by GA: GA = $[GA^{et}$ GA^{ex} GA^{k} GA^{w} $GA^{eq}]$ in which GA^{f} = $\max_{f\in\{et,ex,k,w,eq\};j=\overline{1,m}}\{AC_j^f\}$. To launch T_i successfully, the attackers should be able to bypass the system withstand for T_i, which means their group attack capability GA should be greater than defender capabilities DC_i: $GA^{et} \geq DC_i^{et}, GA^{ex} \geq DC_i^{ex}, GA^{k} \geq DC_i^{k}, GA^{w} \geq DC_i^{w}, GA^{eq} \geq DC_i^{eq}$. For

A simplified approach for dynamic security risk management in c: 5 - 5 (0)



threat t_i in T, the ITS knows the corresponding defender capability vector $DC_i = [DC_i^{et}\ DC_i^{ex}\ DC_i^{k}\ DC_i^{w}\ DC_i^{eq}]$ which represents the system withstand regarding elapsed time, expertise, knowledge, windows of opportunities, and equipment respectively. Similarly, assume that each attacker j

A simplified approach for dynamic security risk management in c: 5 - 5 (0)



capabilities. An example of the attack tree can be shown in Figure 5, while the system withstands for attacks in this tree are shown in Table 1. Assume that the CAV is only interested

A simplified approach for dynamic security risk management in c: 6 - 6 (0)

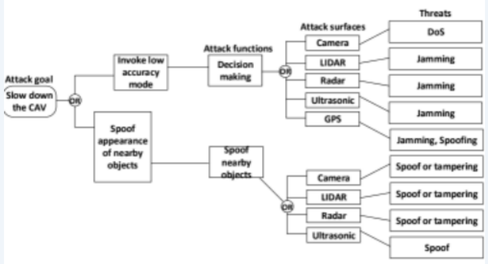
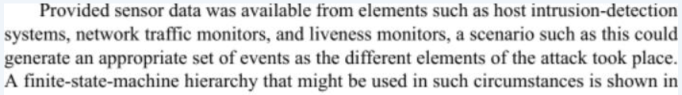


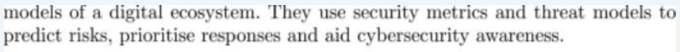
Figure 5. Example of an attack tree regarding the physical sensors

A simplified approach for dynamic security risk management in c: 7 - 7 (0)



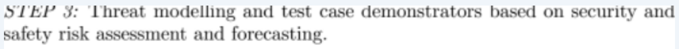
Provided sensor data was available from elements such as host intrusion-detection systems, network traffic monitors, and liveness monitors, a scenario such as this could generate an appropriate set of events as the different elements of the attack took place. A finite-state-machine hierarchy that might be used in such circumstances is shown in

Achieving critical system survivability through software archit: 15 - 15 (0)



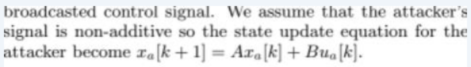
models of a digital ecosystem. They use security metrics and threat models to predict risks, prioritise responses and aid cybersecurity awareness.

Digital Twins for Dependability Improvement of Autonomous Drivi: 4 - 4 (0)



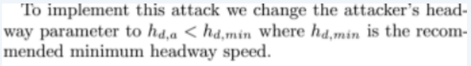
STEP 3: Threat modelling and test case demonstrators based on security and safety risk assessment and forecasting.

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)



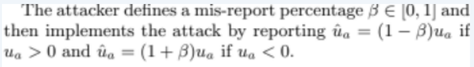
broadcasted control signal. We assume that the attacker's signal is non-additive so the state update equation for the attacker become $x_a[k+1] = Ax_a[k] + Bu_a[k]$.

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



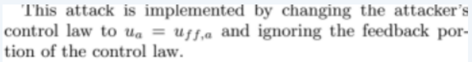
To implement this attack we change the attacker's headway parameter to $h_{d,a} < h_{d,min}$ where $h_{d,min}$ is the recommended minimum headway speed.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



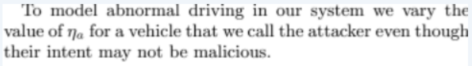
The attacker defines a mis-report percentage $\beta \in [0, 1]$ and then implements the attack by reporting $\hat{u}_a = (1 - \beta)u_a$ if $u_a > 0$ and $\hat{u}_a = (1 + \beta)u_a$ if $u_a < 0$.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



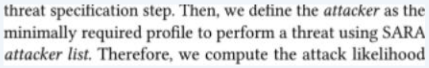
This attack is implemented by changing the attacker's control law to $u_a = u_{ff,a}$ and ignoring the feedback portion of the control law.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



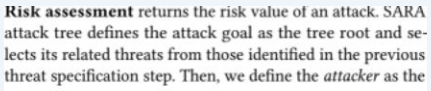
To model abnormal driving in our system we vary the value of η_a for a vehicle that we call the attacker even though their intent may not be malicious.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



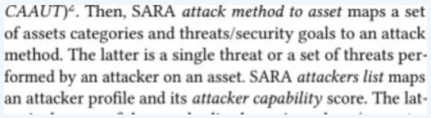
threat specification step. Then, we define the *attacker* as the minimally required profile to perform a threat using SARA *attacker list.* Therefore, we compute the attack likelihood

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



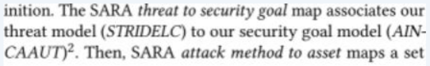
Risk assessment returns the risk value of an attack. SARA attack tree defines the attack goal as the tree root and selects its related threats from those identified in the previous threat specification step. Then, we define the *attacker* as the

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



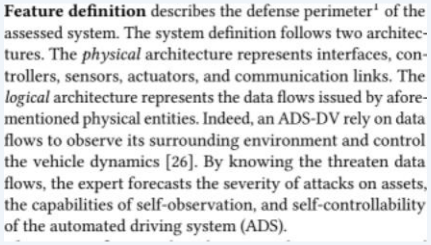
CAAUT)[2]. Then, SARA *attack method to asset* maps a set of assets categories and threats/security goals to an attack method. The latter is a single threat or a set of threats performed by an attacker on an asset. SARA *attackers list* maps an attacker profile and its *attacker capability* score. The lat-

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



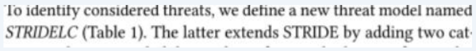
inition. The SARA *threat to security goal* map associates our threat model (*STRIDELC*) to our security goal model (*AINCAAUT*)[2]. Then, SARA *attack method to asset* maps a set

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



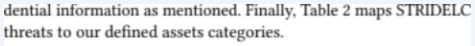
Feature definition describes the defense perimeter[1] of the assessed system. The system definition follows two architectures. The *physical* architecture represents interfaces, controllers, sensors, actuators, and communication links. The *logical* architecture represents the data flows issued by aforementioned physical entities. Indeed, an ADS-DV rely on data flows to observe its surrounding environment and control the vehicle dynamics [26]. By knowing the threaten data flows, the expert forecasts the severity of attacks on assets, the capabilities of self-observation, and self-controllability of the automated driving system (ADS).

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



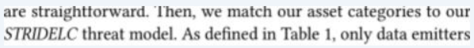
To identify considered threats, we define a new threat model named *STRIDELC* (Table 1). The latter extends STRIDE by adding two cat-

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)



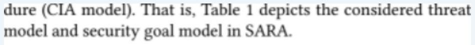
dential information as mentioned. Finally, Table 2 maps STRIDELC threats to our defined assets categories.

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)



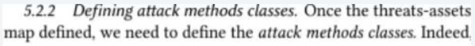
are straightforward. Then, we match our asset categories to our *STRIDELC* threat model. As defined in Table 1, only data emitters

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)



dure (CIA model). That is, Table 1 depicts the considered threat model and security goal model in SARA.

SARA Security Automotive Risk Analysis Method: 4 - 4 (0)



5.2.2 *Defining attack methods classes.* Once the threats-assets map defined, we need to define the *attack methods classes*. Indeed,

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)

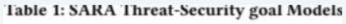
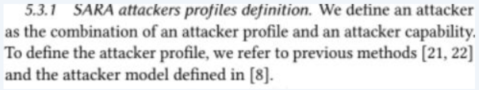


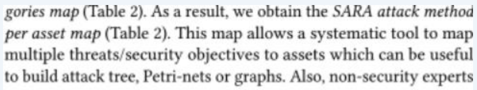
Table 1: SARA Threat-Security goal Models

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



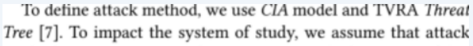
5.3.1 SARA attackers profiles definition. We define an attacker as the combination of an attacker profile and an attacker capability. To define the attacker profile, we refer to previous methods [21, 22] and the attacker model defined in [8].

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



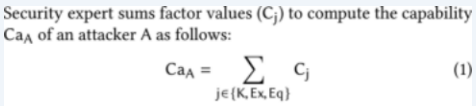
gories map (Table 2). As a result, we obtain the *SARA attack method per asset map* (Table 2). This map allows a systematic tool to map multiple threats/security objectives to assets which can be useful to build attack tree, Petri-nets or graphs. Also, non-security experts

SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



To define attack method, we use *CIA* model and TVRA *Threat Tree* [7]. To impact the system of study, we assume that attack

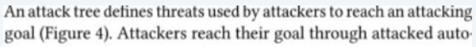
SARA Security Automotive Risk Analysis Method: 5 - 5 (0)



Security expert sums factor values (C_j) to compute the capability Ca_A of an attacker A as follows:

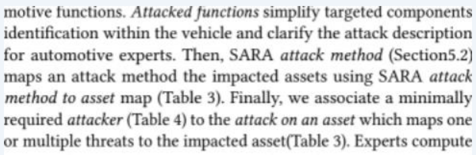
$$Ca_A = \sum_{j\in\{K,Ex,Eq\}} C_j \tag{1}$$

SARA Security Automotive Risk Analysis Method: 6 - 6 (0)



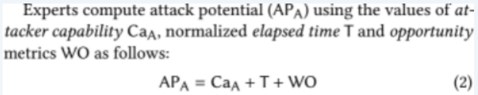
An attack tree defines threats used by attackers to reach an attacking goal (Figure 4). Attackers reach their goal through attacked auto-

SARA Security Automotive Risk Analysis Method: 6 - 6 (0)



motive functions. *Attacked functions* simplify targeted components identification within the vehicle and clarify the attack description for automotive experts. Then, SARA *attack method* (Section5.2) maps an attack method the impacted assets using SARA *attack method to asset* map (Table 3). Finally, we associate a minimally required *attacker* (Table 4) to the *attack on an asset* which maps one or multiple threats to the impacted asset(Table 3). Experts compute

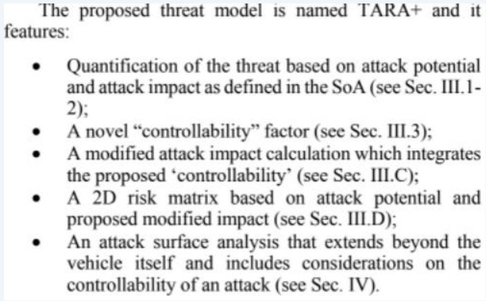
SARA Security Automotive Risk Analysis Method: 6 - 6 (0)



Experts compute attack potential (AP_A) using the values of *attacker capability* Ca_A, normalized *elapsed time* T and *opportunity* metrics WO as follows:

$$AP_A = Ca_A + T + WO \tag{2}$$

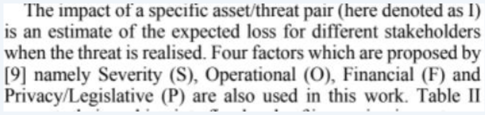
SARA Security Automotive Risk Analysis Method: 7 - 7 (0)



The proposed threat model is named TARA+ and it features:

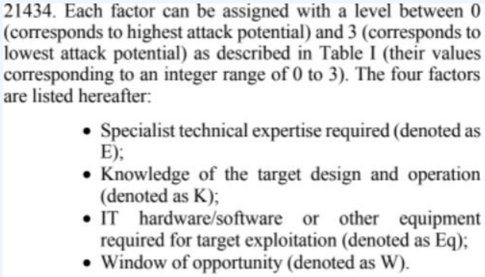
- Quantification of the threat based on attack potential and attack impact as defined in the SoA (see Sec. III.1-2);
- A novel "controllability" factor (see Sec. III.3);
- A modified attack impact calculation which integrates the proposed 'controllability' (see Sec. III.C);
- A 2D risk matrix based on attack potential and proposed modified impact (see Sec. III.D);
- An attack surface analysis that extends beyond the vehicle itself and includes considerations on the controllability of an attack (see Sec. IV).

TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)



The impact of a specific asset/threat pair (here denoted as I) is an estimate of the expected loss for different stakeholders when the threat is realised. Four factors which are proposed by [9] namely Severity (S), Operational (O), Financial (F) and Privacy/Legislative (P) are also used in this work. Table II

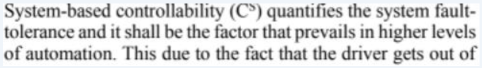
TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)



21434. Each factor can be assigned with a level between 0 (corresponds to highest attack potential) and 3 (corresponds to lowest attack potential) as described in Table I (their values corresponding to an integer range of 0 to 3). The four factors are listed hereafter:

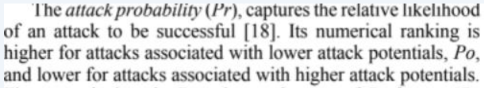
- Specialist technical expertise required (denoted as E);
- Knowledge of the target design and operation (denoted as K);
- IT hardware/software or other equipment required for target exploitation (denoted as Eq);
- Window of opportunity (denoted as W).

TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)



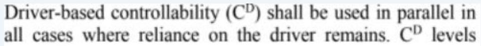
System-based controllability (C^s) quantifies the system fault-tolerance and it shall be the factor that prevails in higher levels of automation. This due to the fact that the driver gets out of

TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)

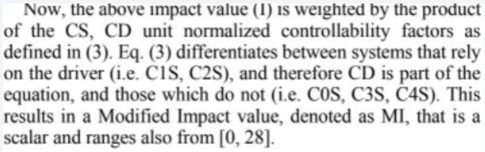


The *attack probability* (*Pr*), captures the relative likelihood of an attack to be successful [18]. Its numerical ranking is higher for attacks associated with lower attack potentials, *Po*, and lower for attacks associated with higher attack potentials.

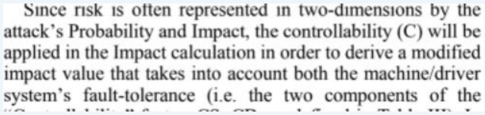
TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)


Driver-based controllability (C^D) shall be used in parallel in all cases where reliance on the driver remains. C^D levels

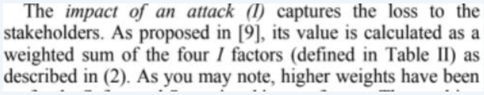
TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)


Now, the above impact value (I) is weighted by the product of the CS, CD unit normalized controllability factors as defined in (3). Eq. (3) differentiates between systems that rely on the driver (i.e. C1S, C2S), and therefore CD is part of the equation, and those which do not (i.e. C0S, C3S, C4S). This results in a Modified Impact value, denoted as MI, that is a scalar and ranges also from [0, 28].

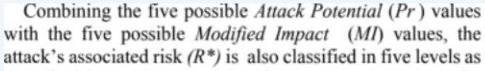
TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)


Since risk is often represented in two-dimensions by the attack's Probability and Impact, the controllability (C) will be applied in the Impact calculation in order to derive a modified impact value that takes into account both the machine/driver system's fault-tolerance (i.e. the two components of the

TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)

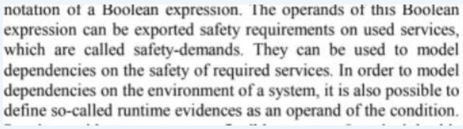

The *impact of an attack (I)* captures the loss to the stakeholders. As proposed in [9], its value is calculated as a weighted sum of the four *I* factors (defined in Table II) as described in (2). As you may note, higher weights have been

TARA Controllability-aware Threat Analysis and Risk Assessment: 3 - 3 (0)

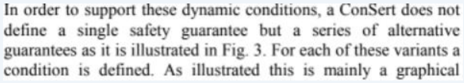

Combining the five possible *Attack Potential* (*Pr*) values with the five possible *Modified Impact* (*MI*) values, the attack's associated risk *(R*)* is also classified in five levels as

TARA Controllability-aware Threat Analysis and Risk Assessment: 4 - 4 (0)

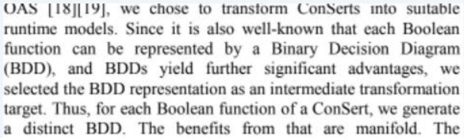
Modeling Approach\Context of the Model\Safety


notation of a Boolean expression. The operands of this Boolean expression can be exported safety requirements on used services, which are called safety-demands. They can be used to model dependencies on the safety of required services. In order to model dependencies on the environment of a system, it is also possible to define so-called runtime evidences as an operand of the condition.

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)

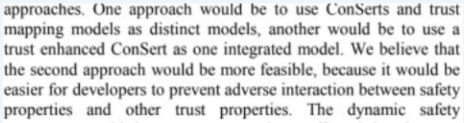

In order to support these dynamic conditions, a ConSert does not define a single safety guarantee but a series of alternative guarantees as it is illustrated in Fig. 3. For each of these variants a condition is defined. As illustrated this is mainly a graphical

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)



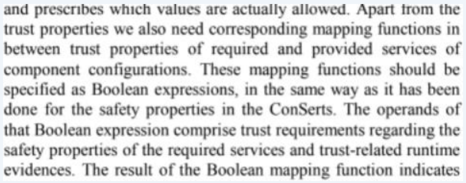
OAS [18][19], we chose to transform ConSerts into suitable runtime models. Since it is also well-known that each Boolean function can be represented by a Binary Decision Diagram (BDD), and BDDs yield further significant advantages, we selected the BDD representation as an intermediate transformation target. Thus, for each Boolean function of a ConSert, we generate a distinct BDD. The benefits from that are manifold. The

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)



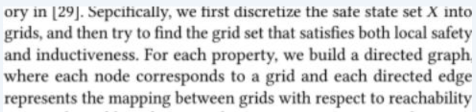
approaches. One approach would be to use ConSerts and trust mapping models as distinct models, another would be to use a trust enhanced ConSert as one integrated model. We believe that the second approach would be more feasible, because it would be easier for developers to prevent adverse interaction between safety properties and other trust properties. The dynamic safety

Approaching runtime trust assurance in open adaptive systems: 5 - 5 (0)



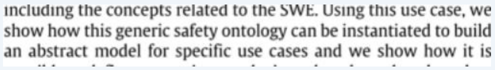
and prescribes which values are actually allowed. Apart from the trust properties we also need corresponding mapping functions in between trust properties of required and provided services of component configurations. These mapping functions should be specified as Boolean expressions, in the same way as it has been done for the safety properties in the ConSerts. The operands of that Boolean expression comprise trust requirements regarding the safety properties of the required services and trust-related runtime evidences. The result of the Boolean mapping function indicates

Approaching runtime trust assurance in open adaptive systems: 5 - 5 (0)



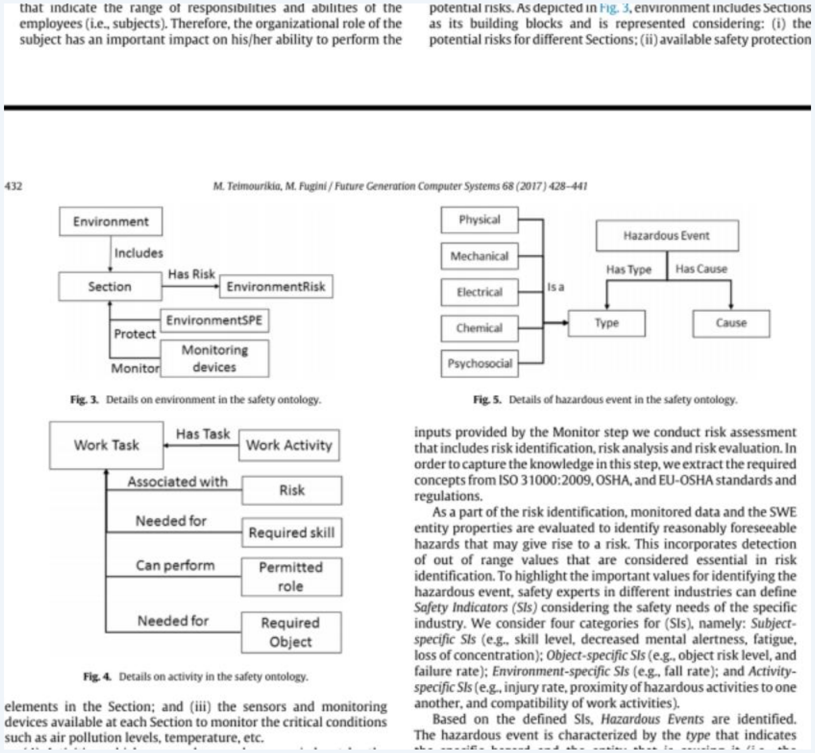
ory in [29]. Sepcifically, we first discretize the safe state set X into grids, and then try to find the grid set that satisfies both local safety and inductiveness. For each property, we build a directed graph where each node corresponds to a grid and each directed edge represents the mapping between grids with respect to reachability

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



including the concepts related to the SWE. Using this use case, we show how this generic safety ontology can be instantiated to build an abstract model for specific use cases and we show how it is

Ontology development for run-time safety management methodology: 2 - 2 (0)



that indicate the range of responsibilities and abilities of the employees (i.e., subjects). Therefore, the organizational role of the subject has an important impact on his/her ability to perform the

potential risks. As depicted in Fig. 3, environment includes Sections as its building blocks and is represented considering: (i) the potential risks for different Sections; (ii) available safety protection

432 M. Teimourikia, M. Fugini / Future Generation Computer Systems 68 (2017) 428–441

Fig. 3. Details on environment in the safety ontology.

Fig. 4. Details on activity in the safety ontology.

elements in the Section; and (iii) the sensors and monitoring devices available at each Section to monitor the critical conditions such as air pollution levels, temperature, etc.

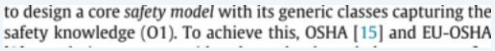
Fig. 5. Details of hazardous event in the safety ontology.

inputs provided by the Monitor step we conduct risk assessment that includes risk identification, risk analysis and risk evaluation. In order to capture the knowledge in this step, we extract the required concepts from ISO 31000:2009, OSHA, and EU-OSHA standards and regulations.

As a part of the risk identification, monitored data and the SWE entity properties are evaluated to identify reasonably foreseeable hazards that may give rise to a risk. This incorporates detection of out of range values that are considered essential in risk identification. To highlight the important values for identifying the hazardous event, safety experts in different industries can define *Safety Indicators (SIs)* considering the safety needs of the specific industry. We consider four categories for (SIs), namely: *Subject-specific SIs* (e.g., skill level, decreased mental alertness, fatigue, loss of concentration); *Object-specific SIs* (e.g., object risk level, and failure rate); *Environment-specific SIs* (e.g., fall rate); and *Activity-specific SIs* (e.g., injury rate, proximity of hazardous activities to one another, and compatibility of work activities).

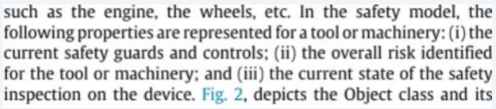
Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates

Ontology development for run-time safety management methodology: 4 - 5 (0)



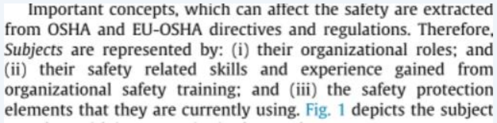
to design a core *safety model* with its generic classes capturing the safety knowledge (O1). To achieve this, OSHA [15] and EU-OSHA

Ontology development for run-time safety management methodology: 4 - 4 (0)



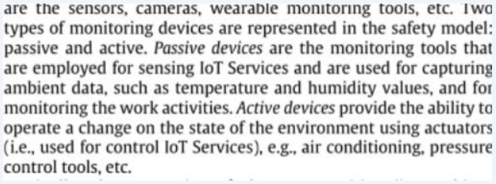
such as the engine, the wheels, etc. In the safety model, the following properties are represented for a tool or machinery: (i) the current safety guards and controls; (ii) the overall risk identified for the tool or machinery; and (iii) the current state of the safety inspection on the device. Fig. 2, depicts the Object class and its

Ontology development for run-time safety management methodology: 4 - 4 (0)



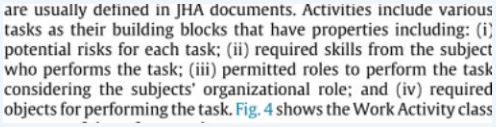
Important concepts, which can affect the safety are extracted from OSHA and EU-OSHA directives and regulations. Therefore, *Subjects* are represented by: (i) their organizational roles; and (ii) their safety related skills and experience gained from organizational safety training; and (iii) the safety protection elements that they are currently using. Fig. 1 depicts the subject

Ontology development for run-time safety management methodology: 4 - 4 (0)



are the sensors, cameras, wearable monitoring tools, etc. Two types of monitoring devices are represented in the safety model: passive and active. *Passive devices* are the monitoring tools that are employed for sensing IoT Services and are used for capturing ambient data, such as temperature and humidity values, and for monitoring the work activities. *Active devices* provide the ability to operate a change on the state of the environment using actuators (i.e., used for control IoT Services), e.g., air conditioning, pressure control tools, etc.

Ontology development for run-time safety management methodology: 5 - 5 (0)



are usually defined in JHA documents. Activities include various tasks as their building blocks that have properties including: (i) potential risks for each task; (ii) required skills from the subject who performs the task; (iii) permitted roles to perform the task considering the subjects' organizational role; and (iv) required objects for performing the task. Fig. 4 shows the Work Activity class

Ontology development for run-time safety management methodology: 5 - 5 (0)

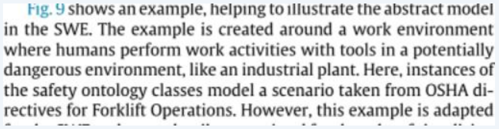
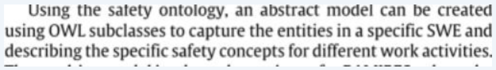


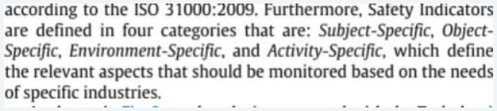
Fig. 9 shows an example, helping to illustrate the abstract model in the SWE. The example is created around a work environment where humans perform work activities with tools in a potentially dangerous environment, like an industrial plant. Here, instances of the safety ontology classes model a scenario taken from OSHA directives for Forklift Operations. However, this example is adapted

Ontology development for run-time safety management methodology: 6 - 6 (0)



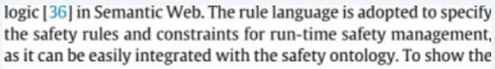
Using the safety ontology, an abstract model can be created using OWL subclasses to capture the entities in a specific SWE and describing the specific safety concepts for different work activities.

Ontology development for run-time safety management methodology: 6 - 6 (0)



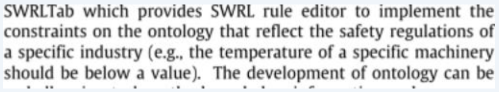
according to the ISO 31000:2009. Furthermore, Safety Indicators are defined in four categories that are: *Subject-Specific*, *Object-Specific*, *Environment-Specific*, and *Activity-Specific*, which define the relevant aspects that should be monitored based on the needs of specific industries.

Ontology development for run-time safety management methodology: 6 - 6 (0)

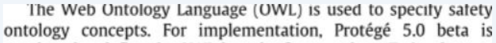


logic [36] in Semantic Web. The rule language is adopted to specify the safety rules and constraints for run-time safety management, as it can be easily integrated with the safety ontology. To show the

Ontology development for run-time safety management methodology: 7 - 7 (0)

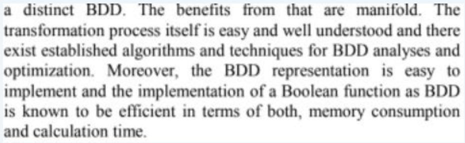

SWRLTab which provides SWRL rule editor to implement the constraints on the ontology that reflect the safety regulations of a specific industry (e.g., the temperature of a specific machinery should be below a value). The development of ontology can be

Ontology development for run-time safety management methodology: 12 - 12 (0)

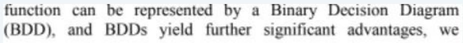

The Web Ontology Language (OWL) is used to specify safety ontology concepts. For implementation, Protégé 5.0 beta is

Ontology development for run-time safety management methodology: 12 - 12 (0)

Modeling Approach\Meta-Model Type / Modeling Language\Logic


a distinct BDD. The benefits from that are manifold. The transformation process itself is easy and well understood and there exist established algorithms and techniques for BDD analyses and optimization. Moreover, the BDD representation is easy to implement and the implementation of a Boolean function as BDD is known to be efficient in terms of both, memory consumption and calculation time.

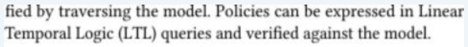
Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)


function can be represented by a Binary Decision Diagram (BDD), and BDDs yield further significant advantages, we

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)

Boolean expressions

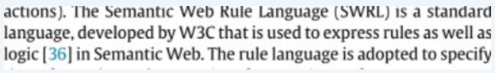
Approaching runtime trust assurance in open adaptive systems: 5 - 5 (0)


fied by traversing the model. Policies can be expressed in Linear Temporal Logic (LTL) queries and verified against the model.

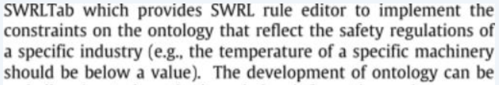
Engineering safety in swarm robotics: 3 - 3 (0)

n LTL

Engineering safety in swarm robotics: 4 - 4 (0)

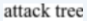

actions). The Semantic Web Rule Language (SWRL) is a standard language, developed by W3C that is used to express rules as well as logic [36] in Semantic Web. The rule language is adopted to specify

Ontology development for run-time safety management methodology: 7 - 7 (0)

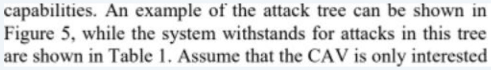

SWRLTab which provides SWRL rule editor to implement the constraints on the ontology that reflect the safety regulations of a specific industry (e.g., the temperature of a specific machinery should be below a value). The development of ontology can be

Ontology development for run-time safety management methodology: 12 - 12 (0)

Modeling Approach\Meta-Model Type / Modeling Language\Functional (Fault/Attack Tree)


attack tree

A simplified approach for dynamic security risk management in c: 4 - 4 (0)


capabilities. An example of the attack tree can be shown in Figure 5, while the system withstands for attacks in this tree are shown in Table 1. Assume that the CAV is only interested

A simplified approach for dynamic security risk management in c: 6 - 6 (0)

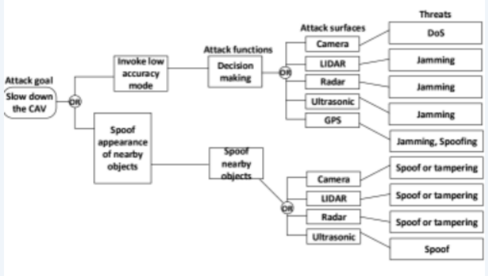
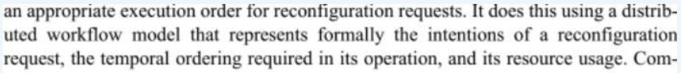


Figure 5. Example of an attack tree regarding the physical sensors

A simplified approach for dynamic security risk management in c: 7 - 7 (0)


an appropriate execution order for reconfiguration requests. It does this using a distributed workflow model that represents formally the intentions of a reconfiguration request, the temporal ordering required in its operation, and its resource usage. Com-

Achieving critical system survivability through software archit: 17 - 17 (0)

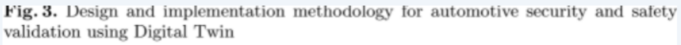
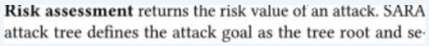
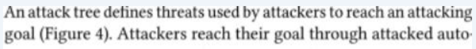
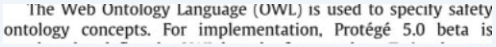
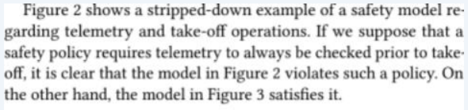
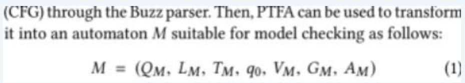
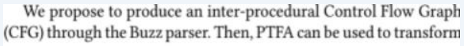
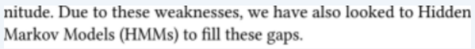
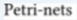


Fig. 3. Design and implementation methodology for automotive security and safety validation using Digital Twin

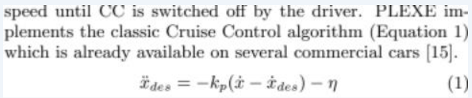
Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)


Risk assessment returns the risk value of an attack. SARA attack tree defines the attack goal as the tree root and se-

SARA Security Automotive Risk Analysis Method: 3 - 3 (0)

	 attack tree SARA Security Automotive Risk Analysis Method: 5 - 5 (0)  An attack tree defines threats used by attackers to reach an attacking goal (Figure 4). Attackers reach their goal through attacked auto- SARA Security Automotive Risk Analysis Method: 6 - 6 (0)
Modeling Approach\Meta-Model Type / Modeling Language\Structural (Component Diagram)	 OWL subclasses to capture the entities in a specific SWE Ontology development for run-time safety management methodology: 6 - 6 (0)  The Web Ontology Language (OWL) is used to specify safety ontology concepts. For implementation, Protégé 5.0 beta is Ontology development for run-time safety management methodology: 12 - 12 (0)
Modeling Approach\Meta-Model Type / Modeling Language\Behavioral (Automata)	 Figure 2 shows a stripped-down example of a safety model regarding telemetry and take-off operations. If we suppose that a safety policy requires telemetry to always be checked prior to take-off, it is clear that the model in Figure 2 violates such a policy. On the other hand, the model in Figure 3 satisfies it. Engineering safety in swarm robotics: 3 - 3 (0)  (CFG) through the Buzz parser. Then, PTFA can be used to transform it into an automaton M suitable for model checking as follows: $M = (Q_M, L_M, T_M, q_0, V_M, G_M, A_M)$ (1) Engineering safety in swarm robotics: 3 - 3 (0)  We propose to produce an inter-procedural Control Flow Graph (CFG) through the Buzz parser. Then, PTFA can be used to transform Engineering safety in swarm robotics: 3 - 3 (0)
	 nitude. Due to these weaknesses, we have also looked to Hidden Markov Models (HMMs) to fill these gaps. Network and system level security in connected vehicle applicat: 4 - 4 (0)
	 Petri-nets SARA Security Automotive Risk Analysis Method: 5 - 5 (0)

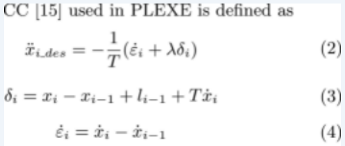
Modeling Approach\Meta-Model Type / Modeling Language\Behavioral (Automata)\Mathematical Model



speed until CC is switched off by the driver. PLEXE implements the classic Cruise Control algorithm (Equation 1) which is already available on several commercial cars [15].

$$\ddot{x}_{des} = -k_p(\dot{x} - \dot{x}_{des}) - \eta \qquad (1)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



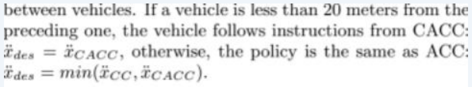
CC [15] used in PLEXE is defined as

$$\ddot{x}_{i\_des} = -\frac{1}{T}(\dot{\varepsilon}_i + \lambda\delta_i) \qquad (2)$$

$$\delta_i = x_i - x_{i-1} + l_{i-1} + T\dot{x}_i \qquad (3)$$

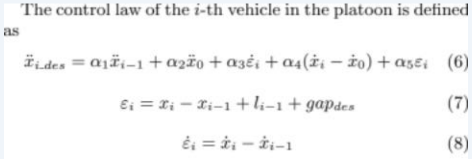
$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (4)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



between vehicles. If a vehicle is less than 20 meters from the preceding one, the vehicle follows instructions from CACC: $\ddot{x}_{des} = \ddot{x}_{CACC}$, otherwise, the policy is the same as ACC: $\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{CACC})$.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



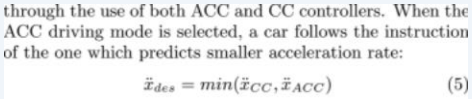
The control law of the i-th vehicle in the platoon is defined as

$$\ddot{x}_{i\_des} = \alpha_1\ddot{x}_{i-1} + \alpha_2\ddot{x}_0 + \alpha_3\dot{\varepsilon}_i + \alpha_4(\dot{x}_i - \dot{x}_0) + \alpha_5\varepsilon_i \qquad (6)$$

$$\varepsilon_i = x_i - x_{i-1} + l_{i-1} + gap_{des} \qquad (7)$$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (8)$$

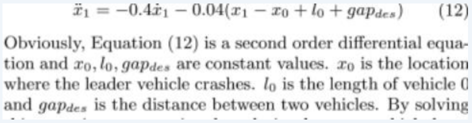
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



through the use of both ACC and CC controllers. When the ACC driving mode is selected, a car follows the instruction of the one which predicts smaller acceleration rate:

$$\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{ACC}) \qquad (5)$$

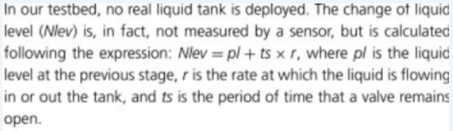
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



$$\ddot{x}_1 = -0.4\dot{x}_1 - 0.04(x_1 - x_0 + l_0 + gap_{des}) \qquad (12)$$

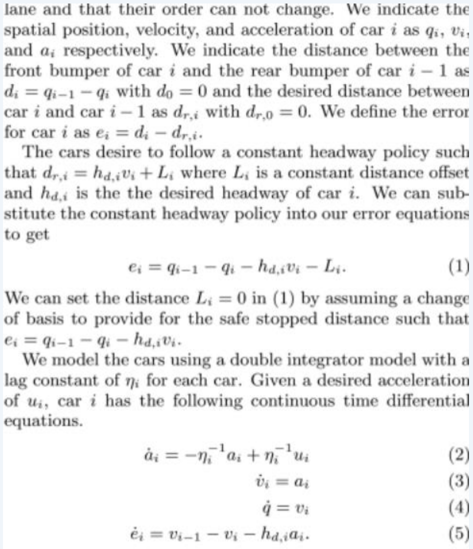
Obviously, Equation (12) is a second order differential equation and x_0, l_0, gap_{des} are constant values. x_0 is the location where the leader vehicle crashes. l_0 is the length of vehicle 0 and gap_{des} is the distance between two vehicles. By solving

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 7 - 7 (0)



In our testbed, no real liquid tank is deployed. The change of liquid level (*Nlev*) is, in fact, not measured by a sensor, but is calculated following the expression: $Nlev = pl + ts \times r$, where *pl* is the liquid level at the previous stage, *r* is the rate at which the liquid is flowing in or out the tank, and *ts* is the period of time that a valve remains open.

Countering targeted cyber-physical attacks using anomaly detect: 5 - 5 (0)



lane and that their order can not change. We indicate the spatial position, velocity, and acceleration of car i as q_i, v_i, and a_i respectively. We indicate the distance between the front bumper of car i and the rear bumper of car $i-1$ as $d_i = q_{i-1} - q_i$ with $d_0 = 0$ and the desired distance between car i and car $i-1$ as $d_{r,i}$ with $d_{r,0} = 0$. We define the error for car i as $e_i = d_i - d_{r,i}$.

The cars desire to follow a constant headway policy such that $d_{r,i} = h_{d,i}v_i + L_i$ where L_i is a constant distance offset and $h_{d,i}$ is the the desired headway of car i. We can substitute the constant headway policy into our error equations to get

$$e_i = q_{i-1} - q_i - h_{d,i}v_i - L_i. \tag{1}$$

We can set the distance $L_i = 0$ in (1) by assuming a change of basis to provide for the safe stopped distance such that $e_i = q_{i-1} - q_i - h_{d,i}v_i$.

We model the cars using a double integrator model with a lag constant of η_i for each car. Given a desired acceleration of u_i, car i has the following continuous time differential equations.

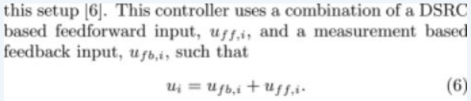
$$\dot{a}_i = -\eta_i^{-1}a_i + \eta_i^{-1}u_i \tag{2}$$

$$\dot{v}_i = a_i \tag{3}$$

$$\dot{q} = v_i \tag{4}$$

$$\dot{e}_i = v_{i-1} - v_i - h_{d,i}a_i. \tag{5}$$

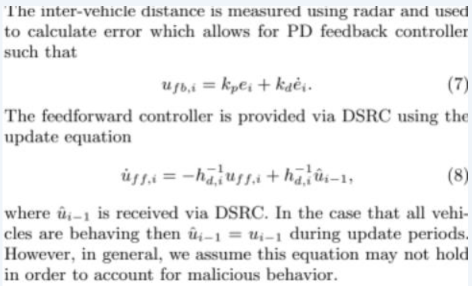
Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



this setup [6]. This controller uses a combination of a DSRC based feedforward input, $u_{ff,i}$, and a measurement based feedback input, $u_{fb,i}$, such that

$$u_i = u_{fb,i} + u_{ff,i}. \tag{6}$$

Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



The inter-vehicle distance is measured using radar and used to calculate error which allows for PD feedback controller such that

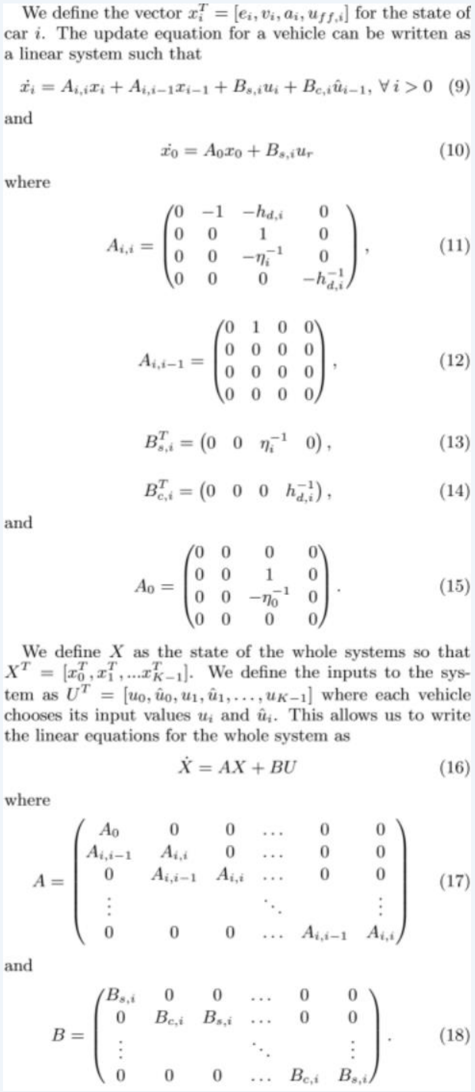
$$u_{fb,i} = k_p e_i + k_d \dot{e}_i. \tag{7}$$

The feedforward controller is provided via DSRC using the update equation

$$\dot{u}_{ff,i} = -h_{d,i}^{-1} u_{ff,i} + h_{d,i}^{-1} \hat{u}_{i-1}, \tag{8}$$

where $\hat{u}_{i-1}$ is received via DSRC. In the case that all vehicles are behaving then $\hat{u}_{i-1} = u_{i-1}$ during update periods. However, in general, we assume this equation may not hold in order to account for malicious behavior.

Is your commute driving you crazy a study of misbehavior in veh: 3 - 3 (0)



We define the vector $x_i^T = [e_i, v_i, a_i, u_{ff,i}]$ for the state of car i. The update equation for a vehicle can be written as a linear system such that

$$\dot{x_i} = A_{i,i}x_i + A_{i,i-1}x_{i-1} + B_{s,i}u_i + B_{c,i}\hat{u}_{i-1},\ \forall\, i > 0 \quad (9)$$

and

$$\dot{x_0} = A_0 x_0 + B_{s,i}u_r \quad (10)$$

where

$$A_{i,i} = \begin{pmatrix} 0 & -1 & -h_{d,i} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_i^{-1} & 0 \\ 0 & 0 & 0 & -h_{d,i}^{-1} \end{pmatrix}, \quad (11)$$

$$A_{i,i-1} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad (12)$$

$$B_{s,i}^T = \begin{pmatrix} 0 & 0 & \eta_i^{-1} & 0 \end{pmatrix}, \quad (13)$$

$$B_{c,i}^T = \begin{pmatrix} 0 & 0 & 0 & h_{d,i}^{-1} \end{pmatrix}, \quad (14)$$

and

$$A_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\eta_0^{-1} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \quad (15)$$

We define X as the state of the whole systems so that $X^T = [x_0^T, x_1^T, ... x_{K-1}^T]$. We define the inputs to the system as $U^T = [u_0, \hat{u}_0, u_1, \hat{u}_1, \ldots, u_{K-1}]$ where each vehicle chooses its input values u_i and $\hat{u}_i$. This allows us to write the linear equations for the whole system as

$$\dot{X} = AX + BU \quad (16)$$

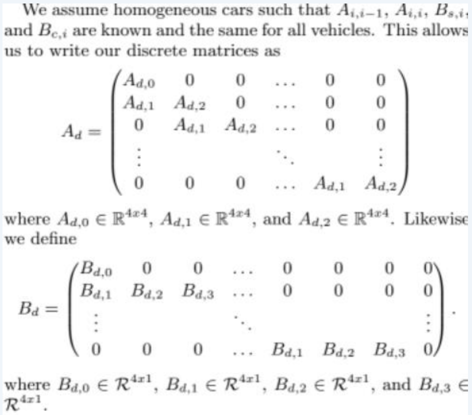
where

$$A = \begin{pmatrix} A_0 & 0 & 0 & \ldots & 0 & 0 \\ A_{i,i-1} & A_{i,i} & 0 & \ldots & 0 & 0 \\ 0 & A_{i,i-1} & A_{i,i} & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & A_{i,i-1} & A_{i,i} \end{pmatrix} \quad (17)$$

and

$$B = \begin{pmatrix} B_{s,i} & 0 & 0 & \ldots & 0 & 0 \\ 0 & B_{c,i} & B_{s,i} & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & B_{c,i} & B_{s,i} \end{pmatrix}. \quad (18)$$

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



We assume homogeneous cars such that $A_{i,i-1}$, $A_{i,i}$, $B_{s,i}$, and $B_{c,i}$ are known and the same for all vehicles. This allows us to write our discrete matrices as

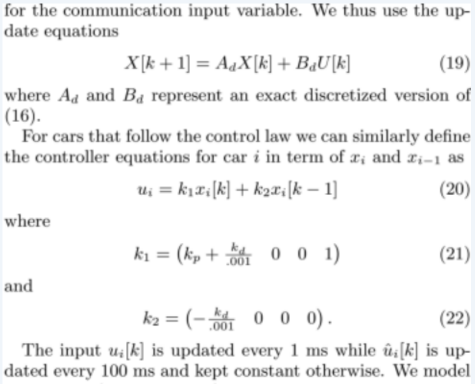
$$A_d = \begin{pmatrix} A_{d,0} & 0 & 0 & \dots & 0 & 0 \\ A_{d,1} & A_{d,2} & 0 & \dots & 0 & 0 \\ 0 & A_{d,1} & A_{d,2} & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & A_{d,1} & A_{d,2} \end{pmatrix}$$

where $A_{d,0} \in \mathbb{R}^{4x4}$, $A_{d,1} \in \mathbb{R}^{4x4}$, and $A_{d,2} \in \mathbb{R}^{4x4}$. Likewise we define

$$B_d = \begin{pmatrix} B_{d,0} & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \dots & 0 & 0 & 0 & 0 \\ \vdots & & & \ddots & & & & \vdots \\ 0 & 0 & 0 & \dots & B_{d,1} & B_{d,2} & B_{d,3} & 0 \end{pmatrix}.$$

where $B_{d,0} \in \mathcal{R}^{4x1}$, $B_{d,1} \in \mathcal{R}^{4x1}$, $B_{d,2} \in \mathcal{R}^{4x1}$, and $B_{d,3} \in \mathcal{R}^{4x1}$.

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



for the communication input variable. We thus use the update equations

$$X[k+1] = A_d X[k] + B_d U[k] \tag{19}$$

where A_d and B_d represent an exact discretized version of (16).

For cars that follow the control law we can similarly define the controller equations for car i in term of x_i and x_{i-1} as

$$u_i = k_1 x_i[k] + k_2 x_i[k-1] \tag{20}$$

where

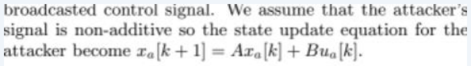
$$k_1 = \begin{pmatrix} k_p + \frac{k_d}{.001} & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

and

$$k_2 = \begin{pmatrix} -\frac{k_d}{.001} & 0 & 0 & 0 \end{pmatrix}. \tag{22}$$

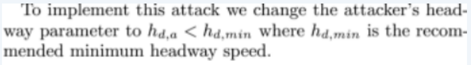
The input $u_i[k]$ is updated every 1 ms while $\hat{u}_i[k]$ is updated every 100 ms and kept constant otherwise. We model

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



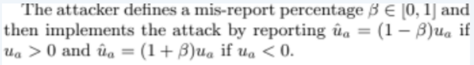
broadcasted control signal. We assume that the attacker's signal is non-additive so the state update equation for the attacker become $x_a[k+1] = Ax_a[k] + Bu_a[k]$.

Is your commute driving you crazy a study of misbehavior in veh: 4 - 4 (0)



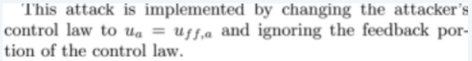
To implement this attack we change the attacker's headway parameter to $h_{d,a} < h_{d,min}$ where $h_{d,min}$ is the recommended minimum headway speed.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



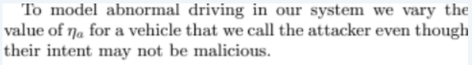
The attacker defines a mis-report percentage $\beta \in [0, 1]$ and then implements the attack by reporting $\hat{u}_a = (1 - \beta)u_a$ if $u_a > 0$ and $\hat{u}_a = (1 + \beta)u_a$ if $u_a < 0$.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



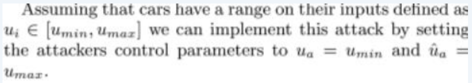
This attack is implemented by changing the attacker's control law to $u_a = u_{ff,a}$ and ignoring the feedback portion of the control law.

Is your commute driving you crazy a study of misbehavior in veh: 5 - 5 (0)



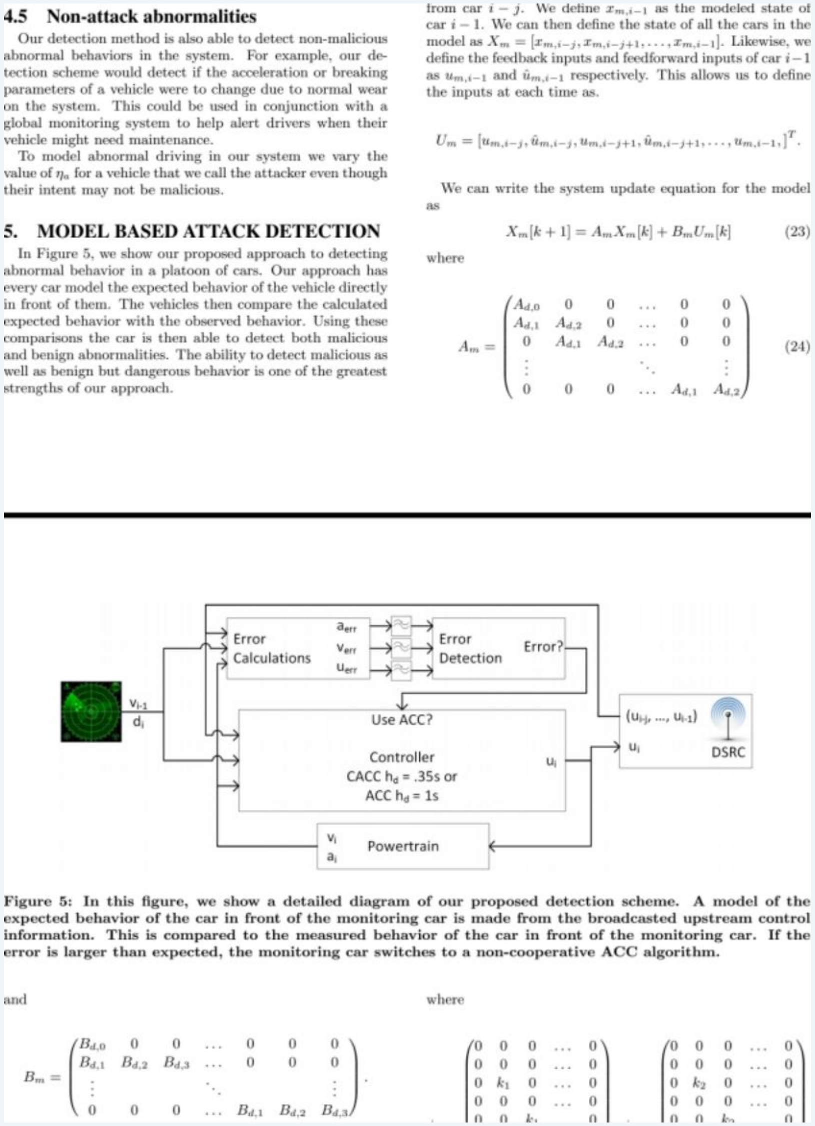
To model abnormal driving in our system we vary the value of η_a for a vehicle that we call the attacker even though their intent may not be malicious.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



Assuming that cars have a range on their inputs defined as $u_i \in [u_{min}, u_{max}]$ we can implement this attack by setting the attackers control parameters to $u_a = u_{min}$ and $\hat{u}_a = u_{max}$.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



4.5 Non-attack abnormalities

Our detection method is also able to detect non-malicious abnormal behaviors in the system. For example, our detection scheme would detect if the acceleration or breaking parameters of a vehicle were to change due to normal wear on the system. This could be used in conjunction with a global monitoring system to help alert drivers when their vehicle might need maintenance.

To model abnormal driving in our system we vary the value of η_a for a vehicle that we call the attacker even though their intent may not be malicious.

5. MODEL BASED ATTACK DETECTION

In Figure 5, we show our proposed approach to detecting abnormal behavior in a platoon of cars. Our approach has every car model the expected behavior of the vehicle directly in front of them. The vehicles then compare the calculated expected behavior with the observed behavior. Using these comparisons the car is then able to detect both malicious and benign abnormalities. The ability to detect malicious as well as benign but dangerous behavior is one of the greatest strengths of our approach.

from car $i-j$. We define $x_{m,i-1}$ as the modeled state of car $i-1$. We can then define the state of all the cars in the model as $X_m = [x_{m,i-j}, x_{m,i-j+1}, \dots, x_{m,i-1}]$. Likewise, we define the feedback inputs and feedforward inputs of car $i-1$ as $u_{m,i-1}$ and $\hat{u}_{m,i-1}$ respectively. This allows us to define the inputs at each time as.

$$U_m = [u_{m,i-j}, \hat{u}_{m,i-j}, u_{m,i-j+1}, \hat{u}_{m,i-j+1}, \dots, u_{m,i-1},]^T.$$

We can write the system update equation for the model as

$$X_m[k+1] = A_m X_m[k] + B_m U_m[k] \tag{23}$$

where

$$A_m = \begin{pmatrix} A_{d,0} & 0 & 0 & \dots & 0 & 0 \\ A_{d,1} & A_{d,2} & 0 & \dots & 0 & 0 \\ 0 & A_{d,1} & A_{d,2} & \dots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & A_{d,1} & A_{d,2} \end{pmatrix} \tag{24}$$

Figure 5: In this figure, we show a detailed diagram of our proposed detection scheme. A model of the expected behavior of the car in front of the monitoring car is made from the broadcasted upstream control information. This is compared to the measured behavior of the car in front of the monitoring car. If the error is larger than expected, the monitoring car switches to a non-cooperative ACC algorithm.

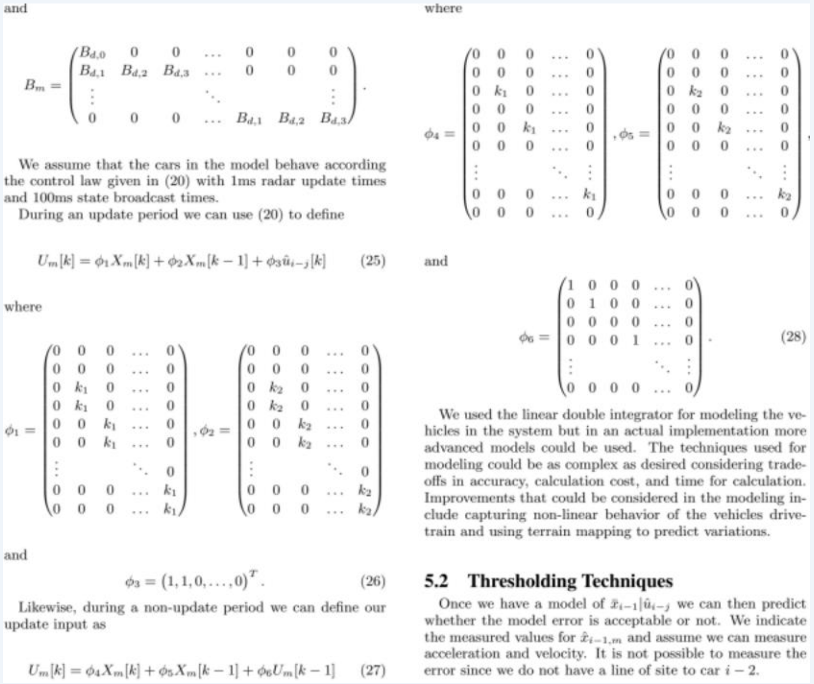
and

$$B_m = \begin{pmatrix} B_{d,0} & 0 & 0 & \dots & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & B_{d,1} & B_{d,2} & B_{d,3} \end{pmatrix}.$$

where

$$\begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_1 & & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_2 & & 0 \end{pmatrix}$$

Is your commute driving you crazy a study of misbehavior in veh: 6 - 7 (0)



and

$$B_m = \begin{pmatrix} B_{d,0} & 0 & 0 & \dots & 0 & 0 & 0 \\ B_{d,1} & B_{d,2} & B_{d,3} & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & B_{d,1} & B_{d,2} & B_{d,3} \end{pmatrix}.$$

We assume that the cars in the model behave according the control law given in (20) with 1ms radar update times and 100ms state broadcast times.

During an update period we can use (20) to define

$$U_m[k] = \phi_1 X_m[k] + \phi_2 X_m[k-1] + \phi_3 \hat{u}_{i-j}[k] \qquad (25)$$

where

$$\phi_1 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & k_1 \end{pmatrix}, \phi_2 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & k_2 \end{pmatrix}$$

and

$$\phi_3 = (1, 1, 0, \dots, 0)^T. \qquad (26)$$

Likewise, during a non-update period we can define our update input as

$$U_m[k] = \phi_4 X_m[k] + \phi_5 X_m[k-1] + \phi_6 U_m[k-1] \qquad (27)$$

where

$$\phi_4 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_1 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}, \phi_5 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & k_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & k_2 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & k_2 \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix},$$

and

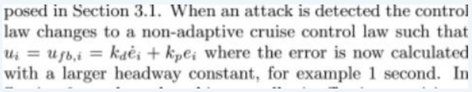
$$\phi_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \qquad (28)$$

We used the linear double integrator for modeling the vehicles in the system but in an actual implementation more advanced models could be used. The techniques used for modeling could be as complex as desired considering trade-offs in accuracy, calculation cost, and time for calculation. Improvements that could be considered in the modeling include capturing non-linear behavior of the vehicles drive-train and using terrain mapping to predict variations.

5.2 Thresholding Techniques

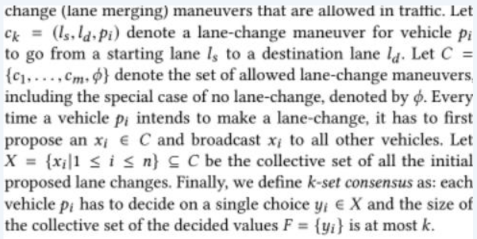
Once we have a model of $\bar{x}_{i-1}|\hat{u}_{i-j}$ we can then predict whether the model error is acceptable or not. We indicate the measured values for $\hat{x}_{i-1,m}$ and assume we can measure acceleration and velocity. It is not possible to measure the error since we do not have a line of site to car $i-2$.

Is your commute driving you crazy a study of misbehavior in veh: 7 - 7 (0)



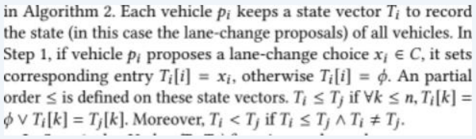
posed in Section 3.1. When an attack is detected the control law changes to a non-adaptive cruise control law such that $u_i = u_{fb,i} = k_d\dot{e}_i + k_p e_i$ where the error is now calculated with a larger headway constant, for example 1 second. In

Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



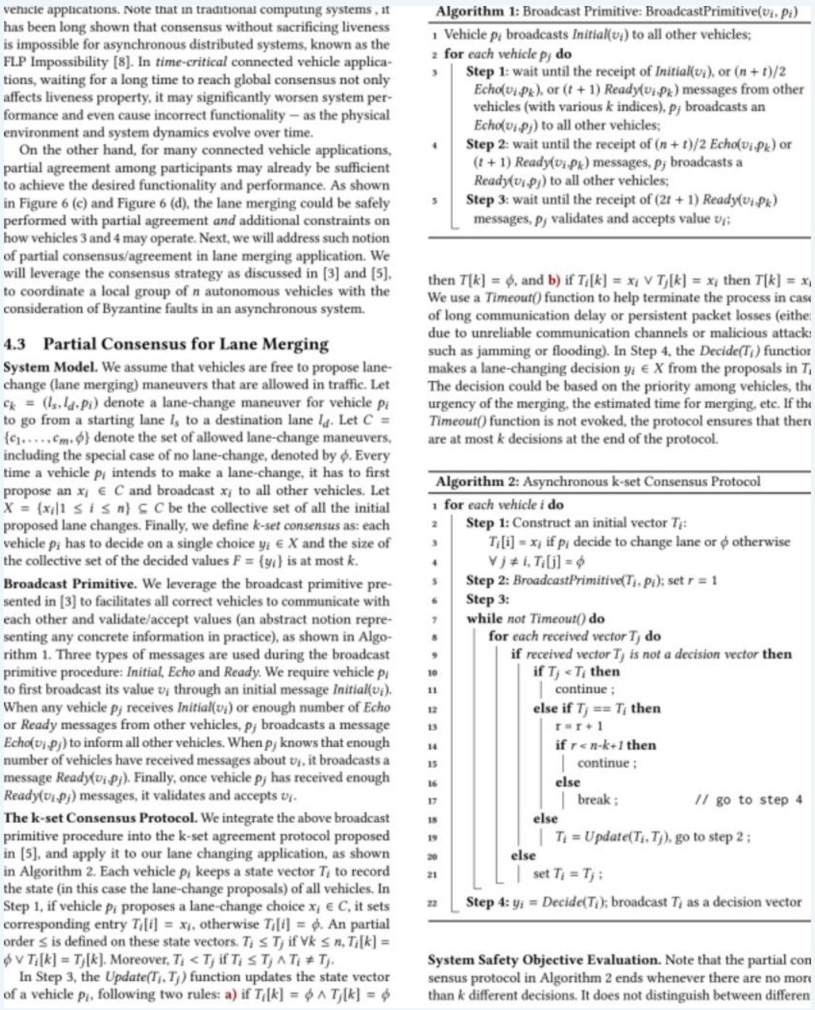
change (lane merging) maneuvers that are allowed in traffic. Let $c_k = (l_s, l_d, p_i)$ denote a lane-change maneuver for vehicle p_i to go from a starting lane l_s to a destination lane l_d. Let $C = \{c_1, \dots, c_m, \phi\}$ denote the set of allowed lane-change maneuvers, including the special case of no lane-change, denoted by ϕ. Every time a vehicle p_i intends to make a lane-change, it has to first propose an $x_i \in C$ and broadcast x_i to all other vehicles. Let $X = \{x_i | 1 \le i \le n\} \subseteq C$ be the collective set of all the initial proposed lane changes. Finally, we define *k-set consensus* as: each vehicle p_i has to decide on a single choice $y_i \in X$ and the size of the collective set of the decided values $F = \{y_i\}$ is at most k.

Network and system level security in connected vehicle applicat: 6 - 6 (0)



in Algorithm 2. Each vehicle p_i keeps a state vector T_i to record the state (in this case the lane-change proposals) of all vehicles. In Step 1, if vehicle p_i proposes a lane-change choice $x_i \in C$, it sets corresponding entry $T_i[i] = x_i$, otherwise $T_i[i] = \phi$. An partial order $\leq$ is defined on these state vectors. $T_i \leq T_j$ if $\forall k \leq n, T_i[k] = \phi \vee T_i[k] = T_j[k]$. Moreover, $T_i < T_j$ if $T_i \leq T_j \wedge T_i \neq T_j$.

Network and system level security in connected vehicle applicat: 6 - 6 (0)



vehicle applications. Note that in traditional computing systems , it has been long shown that consensus without sacrificing liveness is impossible for asynchronous distributed systems, known as the FLP Impossibility [8]. In *time-critical* connected vehicle applications, waiting for a long time to reach global consensus not only affects liveness property, it may significantly worsen system performance and even cause incorrect functionality – as the physical environment and system dynamics evolve over time.

On the other hand, for many connected vehicle applications, partial agreement among participants may already be sufficient to achieve the desired functionality and performance. As shown in Figure 6 (c) and Figure 6 (d), the lane merging could be safely performed with partial agreement *and* additional constraints on how vehicles 3 and 4 may operate. Next, we will address such notion of partial consensus/agreement in lane merging application. We will leverage the consensus strategy as discussed in [3] and [5], to coordinate a local group of n autonomous vehicles with the consideration of Byzantine faults in an asynchronous system.

4.3 Partial Consensus for Lane Merging

System Model. We assume that vehicles are free to propose lane-change (lane merging) maneuvers that are allowed in traffic. Let $c_k = (l_s, l_d, p_i)$ denote a lane-change maneuver for vehicle p_i to go from a starting lane l_s to a destination lane l_d. Let $C = \{c_1, \ldots, c_m, \phi\}$ denote the set of allowed lane-change maneuvers, including the special case of no lane-change, denoted by ϕ. Every time a vehicle p_i intends to make a lane-change, it has to first propose an $x_i \in C$ and broadcast x_i to all other vehicles. Let $X = \{x_i | 1 \leq i \leq n\} \subseteq C$ be the collective set of all the initial proposed lane changes. Finally, we define *k-set consensus* as: each vehicle p_i has to decide on a single choice $y_i \in X$ and the size of the collective set of the decided values $F = \{y_i\}$ is at most k.

Broadcast Primitive. We leverage the broadcast primitive presented in [3] to facilitates all correct vehicles to communicate with each other and validate/accept values (an abstract notion representing any concrete information in practice), as shown in Algorithm 1. Three types of messages are used during the broadcast primitive procedure: *Initial*, *Echo* and *Ready*. We require vehicle p_i to first broadcast its value v_i through an initial message *Initial*(v_i). When any vehicle p_j receives *Initial*(v_i) or enough number of *Echo* or *Ready* messages from other vehicles, p_j broadcasts a message *Echo*(v_i,p_j) to inform all other vehicles. When p_j knows that enough number of vehicles have received messages about v_i, it broadcasts a message *Ready*(v_i,p_j). Finally, once vehicle p_j has received enough *Ready*(v_i,p_j) messages, it validates and accepts v_i.

The k-set Consensus Protocol. We integrate the above broadcast primitive procedure into the k-set agreement protocol proposed in [5], and apply it to our lane changing application, as shown in Algorithm 2. Each vehicle p_i keeps a state vector T_i to record the state (in this case the lane-change proposals) of all vehicles. In Step 1, if vehicle p_i proposes a lane-change choice $x_i \in C$, it sets corresponding entry $T_i[i] = x_i$, otherwise $T_i[i] = \phi$. An partial order $\leq$ is defined on these state vectors. $T_i \leq T_j$ if $\forall k \leq n, T_i[k] = \phi \vee T_i[k] = T_j[k]$. Moreover, $T_i < T_j$ if $T_i \leq T_j \wedge T_i \neq T_j$.

In Step 3, the *Update*(T_i, T_j) function updates the state vector of a vehicle p_i, following two rules: **a)** if $T_i[k] = \phi \wedge T_j[k] = \phi$ then $T[k] = \phi$, and **b)** if $T_i[k] = x_i \vee T_j[k] = x_i$ then $T[k] = x_i$ We use a *Timeout()* function to help terminate the process in cas of long communication delay or persistent packet losses (eithe due to unreliable communication channels or malicious attack such as jamming or flooding). In Step 4, the *Decide*(T_i) functior makes a lane-changing decision $y_i \in X$ from the proposals in T_i The decision could be based on the priority among vehicles, th urgency of the merging, the estimated time for merging, etc. If th *Timeout()* function is not evoked, the protocol ensures that ther are at most k decisions at the end of the protocol.

Algorithm 1: Broadcast Primitive: BroadcastPrimitive(v_i, p_i)

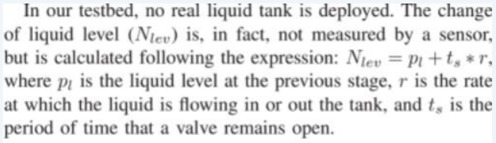
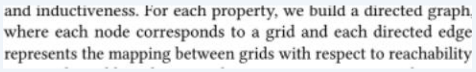
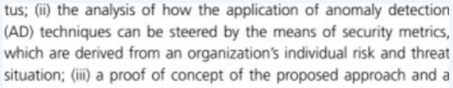
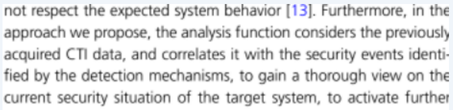
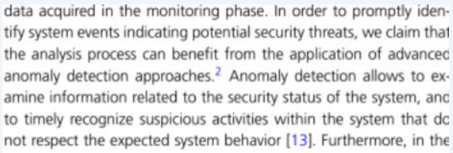
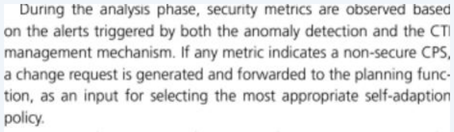
```
Vehicle $p_i$ broadcasts Initial($v_i$) to all other vehicles;
for each vehicle $p_j$ do
    Step 1: wait until the receipt of Initial($v_i$), or $(n + t)/2$
      Echo($v_i$,$p_k$), or $(t + 1)$ Ready($v_i$,$p_k$) messages from other
      vehicles (with various $k$ indices), $p_j$ broadcasts an
      Echo($v_i$,$p_j$) to all other vehicles;
    Step 2: wait until the receipt of $(n + t)/2$ Echo($v_i$,$p_k$) or
      $(t + 1)$ Ready($v_i$,$p_k$) messages, $p_j$ broadcasts a
      Ready($v_i$,$p_j$) to all other vehicles;
    Step 3: wait until the receipt of $(2t + 1)$ Ready($v_i$,$p_k$)
      messages, $p_j$ validates and accepts value $v_i$;
```

Algorithm 2: Asynchronous k-set Consensus Protocol

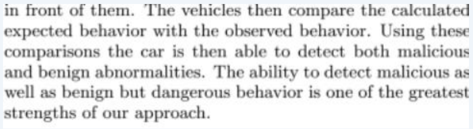
```
for each vehicle i do
    Step 1: Construct an initial vector $T_i$:
        $T_i[i] = x_i$ if $p_i$ decide to change lane or $\phi$ otherwise
        $\forall j \neq i, T_i[j] = \phi$
    Step 2: BroadcastPrimitive($T_i$, $p_i$); set $r = 1$
    Step 3:
    while not Timeout() do
        for each received vector $T_j$ do
            if received vector $T_j$ is not a decision vector then
                if $T_j < T_i$ then
                    continue ;
                else if $T_j == T_i$ then
                    r = r + 1
                    if $r < n-k+1$ then
                        continue ;
                    else
                        break ;            // go to step 4
                else
                    $T_i = Update(T_i, T_j)$, go to step 2 ;
            else
                set $T_i = T_j$ ;
    Step 4: $y_i = Decide(T_i)$; broadcast $T_i$ as a decision vector
```

System Safety Objective Evaluation. Note that the partial con sensus protocol in Algorithm 2 ends whenever there are no mor than k different decisions. It does not distinguish between differen

Network and system level security in connected vehicle applicat: 6 - 6 (0)

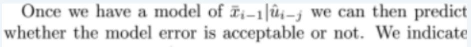
	 In our testbed, no real liquid tank is deployed. The change of liquid level (N_{lev}) is, in fact, not measured by a sensor, but is calculated following the expression: $N_{lev} = p_l + t_s * r$, where p_l is the liquid level at the previous stage, r is the rate at which the liquid is flowing in or out the tank, and t_s is the period of time that a valve remains open. Protecting cyber physical production systems using anomaly dete: 5 - 5 (0)
Modeling Approach\Meta-Model Type / Modeling Language\Other	 and inductiveness. For each property, we build a directed graph, where each node corresponds to a grid and each directed edge represents the mapping between grids with respect to reachability Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)
	 graphs SARA Security Automotive Risk Analysis Method: 5 - 5 (0)
Modeling Approach\Analysis Objectives\Detection	 tus; (ii) the analysis of how the application of anomaly detection (AD) techniques can be steered by the means of security metrics, which are derived from an organization's individual risk and threat situation; (iii) a proof of concept of the proposed approach and a Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)
	 not respect the expected system behavior [13]. Furthermore, in the approach we propose, the analysis function considers the previously acquired CTI data, and correlates it with the security events identified by the detection mechanisms, to gain a thorough view on the current security situation of the target system, to activate further Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)
	 data acquired in the monitoring phase. In order to promptly identify system events indicating potential security threats, we claim that the analysis process can benefit from the application of advanced anomaly detection approaches.[2] Anomaly detection allows to examine information related to the security status of the system, and to timely recognize suspicious activities within the system that do not respect the expected system behavior [13]. Furthermore, in the Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)
	 During the analysis phase, security metrics are observed based on the alerts triggered by both the anomaly detection and the CTI management mechanism. If any metric indicates a non-secure CPS, a change request is generated and forwarded to the planning function, as an input for selecting the most appropriate self-adaption policy.

Countering targeted cyber-physical attacks using anomaly detect: 3 - 3 (0)



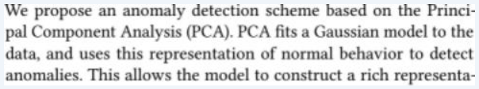
in front of them. The vehicles then compare the calculated expected behavior with the observed behavior. Using these comparisons the car is then able to detect both malicious and benign abnormalities. The ability to detect malicious as well as benign but dangerous behavior is one of the greatest strengths of our approach.

Is your commute driving you crazy a study of misbehavior in veh: 6 - 6 (0)



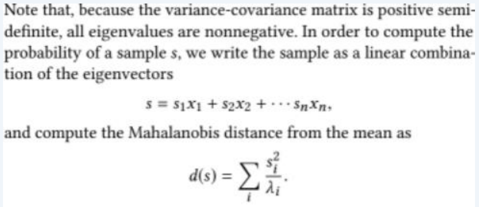
Once we have a model of $\bar{x}_{i-1}|\hat{u}_{i-j}$ we can then predict whether the model error is acceptable or not. We indicate

Is your commute driving you crazy a study of misbehavior in veh: 7 - 7 (0)



We propose an anomaly detection scheme based on the Principal Component Analysis (PCA). PCA fits a Gaussian model to the data, and uses this representation of normal behavior to detect anomalies. This allows the model to construct a rich representa-

Network and system level security in connected vehicle applicat: 3 - 3 (0)



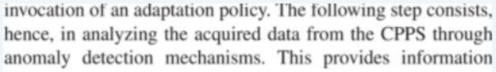
Note that, because the variance-covariance matrix is positive semidefinite, all eigenvalues are nonnegative. In order to compute the probability of a sample s, we write the sample as a linear combination of the eigenvectors

$$s = s_1 x_1 + s_2 x_2 + \cdots s_n x_n,$$

and compute the Mahalanobis distance from the mean as

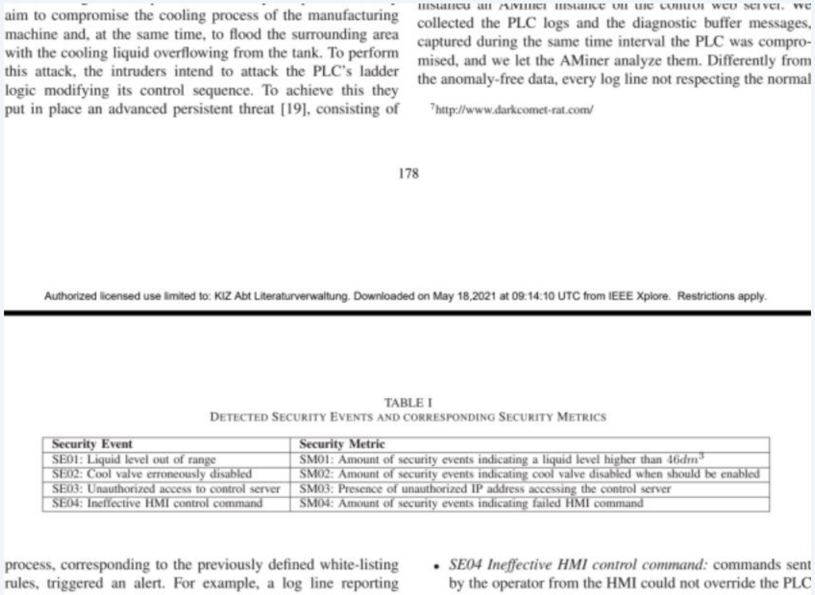
$$d(s) = \sum_i \frac{s_i^2}{\lambda_i}.$$

Network and system level security in connected vehicle applicat: 4 - 4 (0)



invocation of an adaptation policy. The following step consists, hence, in analyzing the acquired data from the CPPS through anomaly detection mechanisms. This provides information

Protecting cyber physical production systems using anomaly dete: 3 - 3 (0)



aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they put in place an advanced persistent threat [19], consisting of

installed an AMiner instance on the control web server. We collected the PLC logs and the diagnostic buffer messages, captured during the same time interval the PLC was compromised, and we let the AMiner analyze them. Differently from the anomaly-free data, every log line not respecting the normal

[7]http://www.darkcomet-rat.com/

178

Authorized licensed use limited to: KIZ Abt Literaturverwaltung. Downloaded on May 18,2021 at 09:14:10 UTC from IEEE Xplore. Restrictions apply.

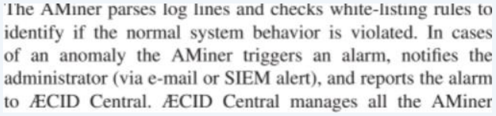
TABLE I
DETECTED SECURITY EVENTS AND CORRESPONDING SECURITY METRICS

Security Event	Security Metric
SE01: Liquid level out of range	SM01: Amount of security events indicating a liquid level higher than $46dm^3$
SE02: Cool valve erroneously disabled	SM02: Amount of security events indicating cool valve disabled when should be enabled
SE03: Unauthorized access to control server	SM03: Presence of unauthorized IP address accessing the control server
SE04: Ineffective HMI control command	SM04: Amount of security events indicating failed HMI command

process, corresponding to the previously defined white-listing rules, triggered an alert. For example, a log line reporting

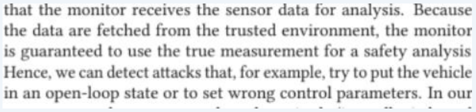
- *SE04 Ineffective HMI control command:* commands sent by the operator from the HMI could not override the PLC

Protecting cyber physical production systems using anomaly dete: 6 - 7 (0)



The AMiner parses log lines and checks white-listing rules to identify if the normal system behavior is violated. In cases of an anomaly the AMiner triggers an alarm, notifies the administrator (via e-mail or SIEM alert), and reports the alarm to ÆCID Central. ÆCID Central manages all the AMiner

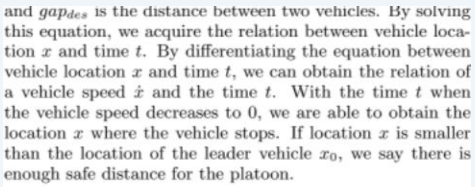
Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



that the monitor receives the sensor data for analysis. Because the data are fetched from the trusted environment, the monitor is guaranteed to use the true measurement for a safety analysis Hence, we can detect attacks that, for example, try to put the vehicle in an open-loop state or to set wrong control parameters. In our

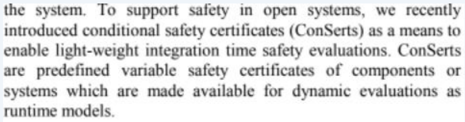
VirtualDrone virtual sensing actuation and communication for at: 5 - 5 (0)

Modeling Approach\Analysis Objectives\Requirements



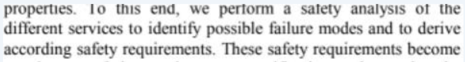
and gap_{des} is the distance between two vehicles. By solving this equation, we acquire the relation between vehicle location x and time t. By differentiating the equation between vehicle location x and time t, we can obtain the relation of a vehicle speed $\dot{x}$ and the time t. With the time t when the vehicle speed decreases to 0, we are able to obtain the location x where the vehicle stops. If location x is smaller than the location of the leader vehicle x_0, we say there is enough safe distance for the platoon.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 7 - 7 (0)



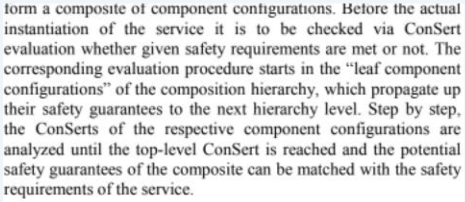
the system. To support safety in open systems, we recently introduced conditional safety certificates (ConSerts) as a means to enable light-weight integration time safety evaluations. ConSerts are predefined variable safety certificates of components or systems which are made available for dynamic evaluations as runtime models.

Approaching runtime trust assurance in open adaptive systems: 1 - 1 (0)



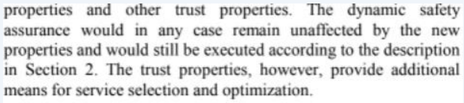
properties. To this end, we perform a safety analysis of the different services to identify possible failure modes and to derive according safety requirements. These safety requirements become

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)



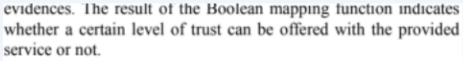
form a composite of component configurations. Before the actual instantiation of the service it is to be checked via ConSert evaluation whether given safety requirements are met or not. The corresponding evaluation procedure starts in the "leaf component configurations" of the composition hierarchy, which propagate up their safety guarantees to the next hierarchy level. Step by step, the ConSerts of the respective component configurations are analyzed until the top-level ConSert is reached and the potential safety guarantees of the composite can be matched with the safety requirements of the service.

Approaching runtime trust assurance in open adaptive systems: 3 - 3 (0)



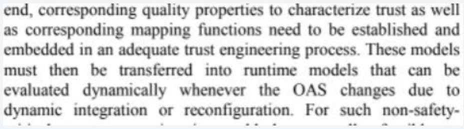
properties and other trust properties. The dynamic safety assurance would in any case remain unaffected by the new properties and would still be executed according to the description in Section 2. The trust properties, however, provide additional means for service selection and optimization.

Approaching runtime trust assurance in open adaptive systems: 5 - 5 (0)



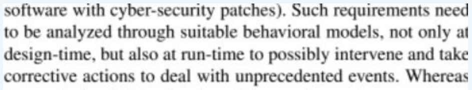
evidences. The result of the Boolean mapping function indicates whether a certain level of trust can be offered with the provided service or not.

Approaching runtime trust assurance in open adaptive systems: 5 - 5 (0)



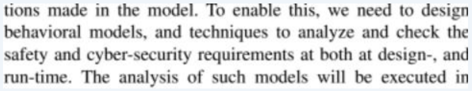
end, corresponding quality properties to characterize trust as well as corresponding mapping functions need to be established and embedded in an adequate trust engineering process. These models must then be transferred into runtime models that can be evaluated dynamically whenever the OAS changes due to dynamic integration or reconfiguration. For such non-safety-

Approaching runtime trust assurance in open adaptive systems: 6 - 6 (0)



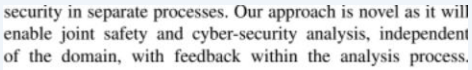
software with cyber-security patches). Such requirements need to be analyzed through suitable behavioral models, not only at design-time, but also at run-time to possibly intervene and take corrective actions to deal with unprecedented events. Whereas

Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)



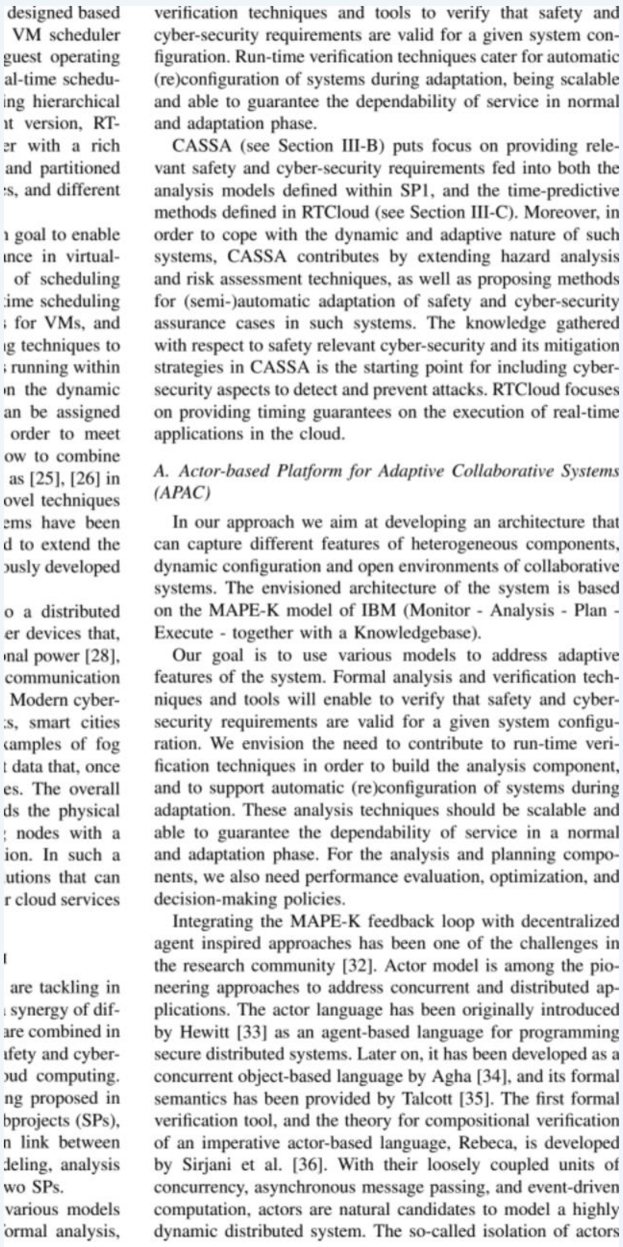
tions made in the model. To enable this, we need to design behavioral models, and techniques to analyze and check the safety and cyber-security requirements at both at design-, and run-time. The analysis of such models will be executed in

Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)



security in separate processes. Our approach is novel as it will enable joint safety and cyber-security analysis, independent of the domain, with feedback within the analysis process.

Towards a Framework for Safe and Secure Adaptive Collaborative: 2 - 2 (0)



designed based
VM scheduler
guest operating
al-time schedu-
ing hierarchical
nt version, RT-
er with a rich
and partitioned
s, and different

n goal to enable
ance in virtual-
of scheduling
time scheduling
s for VMs, and
ng techniques to
s running within
n the dynamic
an be assigned
order to meet
ow to combine
as [25], [26] in
ovel techniques
ems have been
d to extend the
ously developed

o a distributed
er devices that,
nal power [28],
communication
Modern cyber-
s, smart cities
xamples of fog
t data that, once
es. The overall
ds the physical
nodes with a
ion. In such a
utions that can
r cloud services

I

are tackling in
synergy of dif-
are combined in
ifety and cyber-
oud computing.
ng proposed in
bprojects (SPs),
n link between
deling, analysis
wo SPs.
various models
ormal analysis,

verification techniques and tools to verify that safety and cyber-security requirements are valid for a given system configuration. Run-time verification techniques cater for automatic (re)configuration of systems during adaptation, being scalable and able to guarantee the dependability of service in normal and adaptation phase.

CASSA (see Section III-B) puts focus on providing relevant safety and cyber-security requirements fed into both the analysis models defined within SP1, and the time-predictive methods defined in RTCloud (see Section III-C). Moreover, in order to cope with the dynamic and adaptive nature of such systems, CASSA contributes by extending hazard analysis and risk assessment techniques, as well as proposing methods for (semi-)automatic adaptation of safety and cyber-security assurance cases in such systems. The knowledge gathered with respect to safety relevant cyber-security and its mitigation strategies in CASSA is the starting point for including cyber-security aspects to detect and prevent attacks. RTCloud focuses on providing timing guarantees on the execution of real-time applications in the cloud.

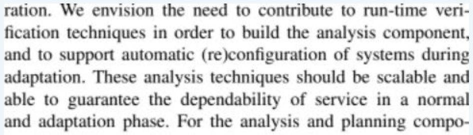
A. Actor-based Platform for Adaptive Collaborative Systems (APAC)

In our approach we aim at developing an architecture that can capture different features of heterogeneous components, dynamic configuration and open environments of collaborative systems. The envisioned architecture of the system is based on the MAPE-K model of IBM (Monitor - Analysis - Plan - Execute - together with a Knowledgebase).

Our goal is to use various models to address adaptive features of the system. Formal analysis and verification techniques and tools will enable to verify that safety and cyber-security requirements are valid for a given system configuration. We envision the need to contribute to run-time verification techniques in order to build the analysis component, and to support automatic (re)configuration of systems during adaptation. These analysis techniques should be scalable and able to guarantee the dependability of service in a normal and adaptation phase. For the analysis and planning components, we also need performance evaluation, optimization, and decision-making policies.

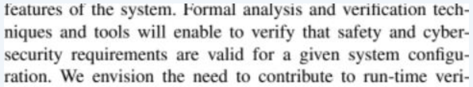
Integrating the MAPE-K feedback loop with decentralized agent inspired approaches has been one of the challenges in the research community [32]. Actor model is among the pioneering approaches to address concurrent and distributed applications. The actor language has been originally introduced by Hewitt [33] as an agent-based language for programming secure distributed systems. Later on, it has been developed as a concurrent object-based language by Agha [34], and its formal semantics has been provided by Talcott [35]. The first formal verification tool, and the theory for compositional verification of an imperative actor-based language, Rebeca, is developed by Sirjani et al. [36]. With their loosely coupled units of concurrency, asynchronous message passing, and event-driven computation, actors are natural candidates to model a highly dynamic distributed system. The so-called isolation of actors

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)



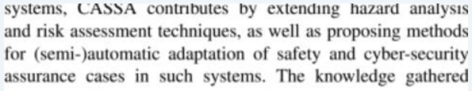
ration. We envision the need to contribute to run-time verification techniques in order to build the analysis component, and to support automatic (re)configuration of systems during adaptation. These analysis techniques should be scalable and able to guarantee the dependability of service in a normal and adaptation phase. For the analysis and planning compo-

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)



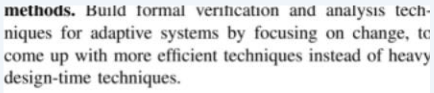
features of the system. Formal analysis and verification techniques and tools will enable to verify that safety and cyber-security requirements are valid for a given system configuration. We envision the need to contribute to run-time veri-

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)



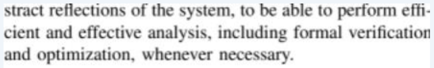
systems, CASSA contributes by extending hazard analysis and risk assessment techniques, as well as proposing methods for (semi-)automatic adaptation of safety and cyber-security assurance cases in such systems. The knowledge gathered

Towards a Framework for Safe and Secure Adaptive Collaborative: 3 - 3 (0)



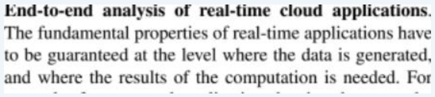
methods. Build formal verification and analysis techniques for adaptive systems by focusing on change, to come up with more efficient techniques instead of heavy design-time techniques.

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)



stract reflections of the system, to be able to perform efficient and effective analysis, including formal verification and optimization, whenever necessary.

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)



End-to-end analysis of real-time cloud applications. The fundamental properties of real-time applications have to be guaranteed at the level where the data is generated, and where the results of the computation is needed. For

Towards a Framework for Safe and Secure Adaptive Collaborative: 5 - 5 (0)

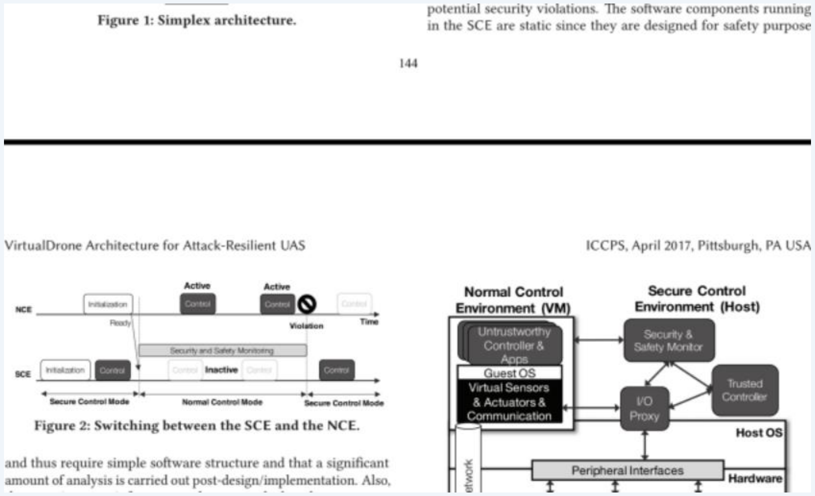


Figure 1: Simplex architecture.

potential security violations. The software components running in the SCE are static since they are designed for safety purpose

144

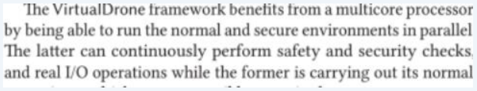
VirtualDrone Architecture for Attack-Resilient UAS ICCPS, April 2017, Pittsburgh, PA USA

Figure 2: Switching between the SCE and the NCE.

and thus require simple software structure and that a significant amount of analysis is carried out post-design/implementation. Also,

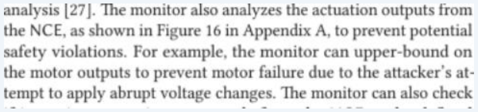
Normal Control Environment (VM)
Secure Control Environment (Host)
Untrustworthy Controller & Apps
Guest OS
Virtual Sensors & Actuators & Communication
Security & Safety Monitor
I/O Proxy
Trusted Controller
Host OS
Peripheral Interfaces
Hardware
etwork

VirtualDrone virtual sensing actuation and communication for at: 2 - 3 (0)



The VirtualDrone framework benefits from a multicore processor by being able to run the normal and secure environments in parallel The latter can continuously perform safety and security checks and real I/O operations while the former is carrying out its normal

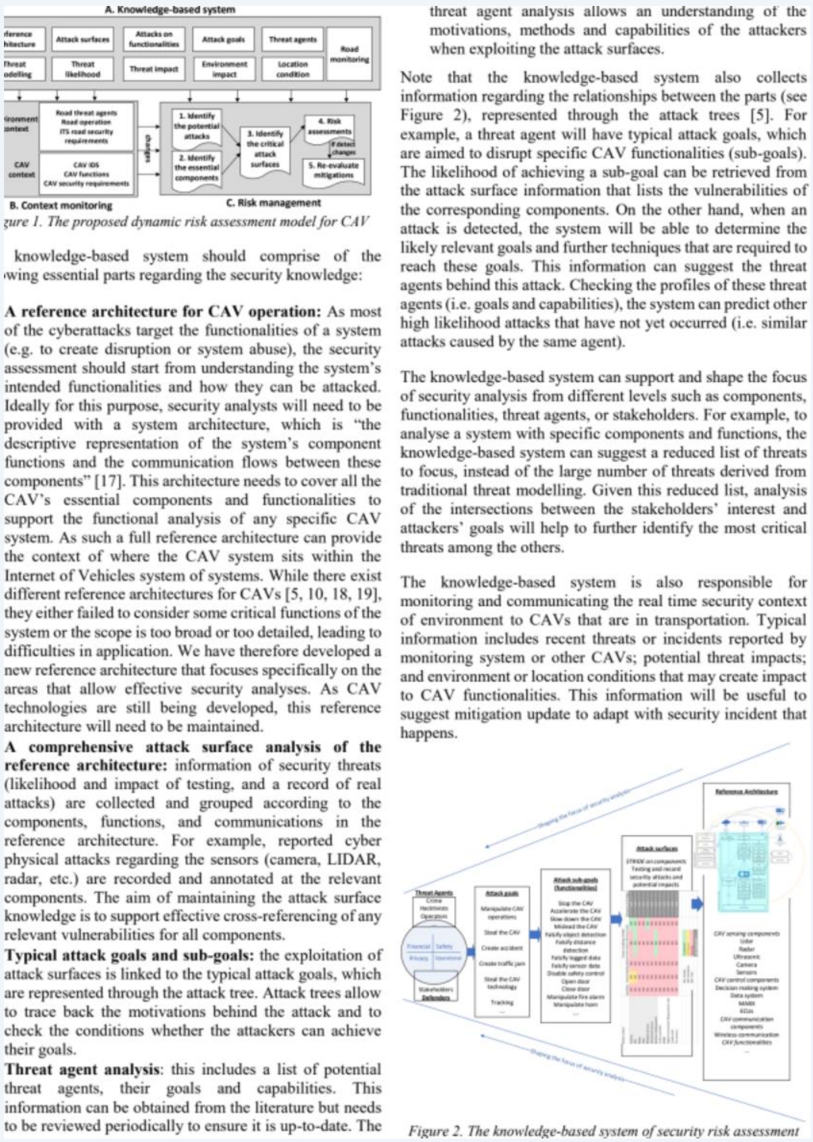
VirtualDrone virtual sensing actuation and communication for at: 3 - 3 (0)



analysis [27]. The monitor also analyzes the actuation outputs from the NCE, as shown in Figure 16 in Appendix A, to prevent potential safety violations. For example, the monitor can upper-bound on the motor outputs to prevent motor failure due to the attacker's attempt to apply abrupt voltage changes. The monitor can also check

VirtualDrone virtual sensing actuation and communication for at: 5 - 5 (0)

Modeling Approach\Analysis Objectives\Risk and Threat Analysis



gure 1. The proposed dynamic risk assessment model for CAV

knowledge-based system should comprise of the wing essential parts regarding the security knowledge:

A reference architecture for CAV operation: As most of the cyberattacks target the functionalities of a system (e.g. to create disruption or system abuse), the security assessment should start from understanding the system's intended functionalities and how they can be attacked. Ideally for this purpose, security analysts will need to be provided with a system architecture, which is "the descriptive representation of the system's component functions and the communication flows between these components" [17]. This architecture needs to cover all the CAV's essential components and functionalities to support the functional analysis of any specific CAV system. As such a full reference architecture can provide the context of where the CAV system sits within the Internet of Vehicles system of systems. While there exist different reference architectures for CAVs [5, 10, 18, 19], they either failed to consider some critical functions of the system or the scope is too broad or too detailed, leading to difficulties in application. We have therefore developed a new reference architecture that focuses specifically on the areas that allow effective security analyses. As CAV technologies are still being developed, this reference architecture will need to be maintained.

A comprehensive attack surface analysis of the reference architecture: information of security threats (likelihood and impact of testing, and a record of real attacks) are collected and grouped according to the components, functions, and communications in the reference architecture. For example, reported cyber physical attacks regarding the sensors (camera, LIDAR, radar, etc.) are recorded and annotated at the relevant components. The aim of maintaining the attack surface knowledge is to support effective cross-referencing of any relevant vulnerabilities for all components.

Typical attack goals and sub-goals: the exploitation of attack surfaces is linked to the typical attack goals, which are represented through the attack tree. Attack trees allow to trace back the motivations behind the attack and to check the conditions whether the attackers can achieve their goals.

Threat agent analysis: this includes a list of potential threat agents, their goals and capabilities. This information can be obtained from the literature but needs to be reviewed periodically to ensure it is up-to-date. The threat agent analysis allows an understanding of the motivations, methods and capabilities of the attackers when exploiting the attack surfaces.

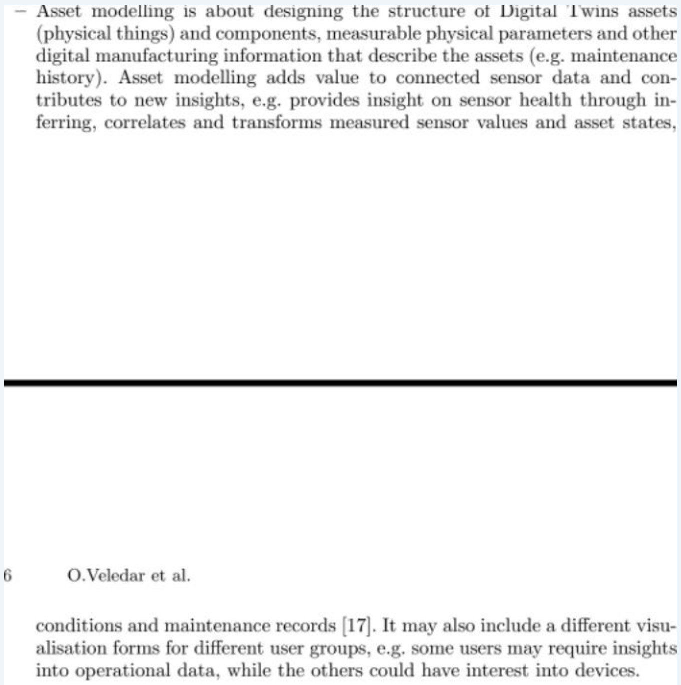
Note that the knowledge-based system also collects information regarding the relationships between the parts (see Figure 2), represented through the attack trees [5]. For example, a threat agent will have typical attack goals, which are aimed to disrupt specific CAV functionalities (sub-goals). The likelihood of achieving a sub-goal can be retrieved from the attack surface information that lists the vulnerabilities of the corresponding components. On the other hand, when an attack is detected, the system will be able to determine the likely relevant goals and further techniques that are required to reach these goals. This information can suggest the threat agents behind this attack. Checking the profiles of these threat agents (i.e. goals and capabilities), the system can predict other high likelihood attacks that have not yet occurred (i.e. similar attacks caused by the same agent).

The knowledge-based system can support and shape the focus of security analysis from different levels such as components, functionalities, threat agents, or stakeholders. For example, to analyse a system with specific components and functions, the knowledge-based system can suggest a reduced list of threats to focus, instead of the large number of threats derived from traditional threat modelling. Given this reduced list, analysis of the intersections between the stakeholders' interest and attackers' goals will help to further identify the most critical threats among the others.

The knowledge-based system is also responsible for monitoring and communicating the real time security context of environment to CAVs that are in transportation. Typical information includes recent threats or incidents reported by monitoring system or other CAVs; potential threat impacts; and environment or location conditions that may create impact to CAV functionalities. This information will be useful to suggest mitigation update to adapt with security incident that happens.

Figure 2. The knowledge-based system of security risk assessment

A simplified approach for dynamic security risk management in c: 4 - 4 (0)

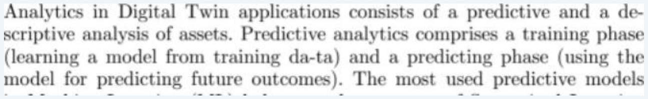


– Asset modelling is about designing the structure of Digital Twins assets (physical things) and components, measurable physical parameters and other digital manufacturing information that describe the assets (e.g. maintenance history). Asset modelling adds value to connected sensor data and contributes to new insights, e.g. provides insight on sensor health through inferring, correlates and transforms measured sensor values and asset states,

6 O.Veledar et al.

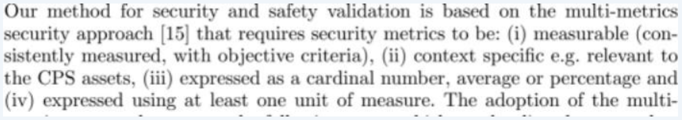
conditions and maintenance records [17]. It may also include a different visualisation forms for different user groups, e.g. some users may require insights into operational data, while the others could have interest into devices.

Digital Twins for Dependability Improvement of Autonomous Drivi: 6 - 7 (0)



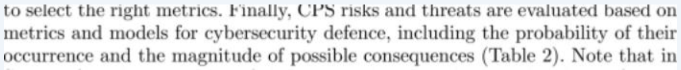
Analytics in Digital Twin applications consists of a predictive and a descriptive analysis of assets. Predictive analytics comprises a training phase (learning a model from training da-ta) and a predicting phase (using the model for predicting future outcomes). The most used predictive models

Digital Twins for Dependability Improvement of Autonomous Drivi: 7 - 7 (0)



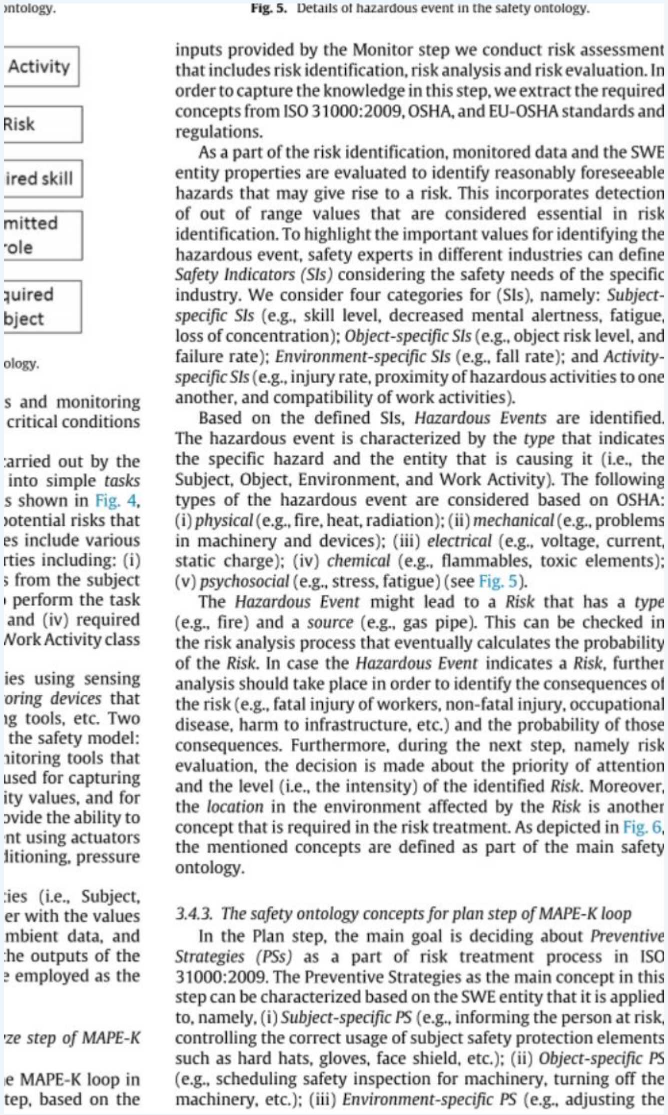
Our method for security and safety validation is based on the multi-metrics security approach [15] that requires security metrics to be: (i) measurable (consistently measured, with objective criteria), (ii) context specific e.g. relevant to the CPS assets, (iii) expressed as a cardinal number, average or percentage and (iv) expressed using at least one unit of measure. The adoption of the multi-

Digital Twins for Dependability Improvement of Autonomous Drivi: 8 - 8 (0)



to select the right metrics. Finally, CPS risks and threats are evaluated based on metrics and models for cybersecurity defence, including the probability of their occurrence and the magnitude of possible consequences (Table 2). Note that in

Digital Twins for Dependability Improvement of Autonomous Drivi: 8 - 8 (0)



ontology.

Activity
Risk
ired skill
mitted role
quired bject

ology.

s and monitoring
critical conditions

arried out by the
into simple *tasks*
s shown in Fig. 4,
potential risks that
es include various
rties including: (i)
s from the subject
perform the task
and (iv) required
Work Activity class

ies using sensing
oring devices that
ng tools, etc. Two
the safety model:
nitoring tools that
used for capturing
ity values, and for
ovide the ability to
nt using actuators
litioning, pressure

ties (i.e., Subject,
er with the values
mbient data, and
the outputs of the
e employed as the

ze step of MAPE-K

e MAPE-K loop in
tep, based on the

Fig. 5. Details of hazardous event in the safety ontology.

inputs provided by the Monitor step we conduct risk assessment that includes risk identification, risk analysis and risk evaluation. In order to capture the knowledge in this step, we extract the required concepts from ISO 31000:2009, OSHA, and EU-OSHA standards and regulations.

As a part of the risk identification, monitored data and the SWE entity properties are evaluated to identify reasonably foreseeable hazards that may give rise to a risk. This incorporates detection of out of range values that are considered essential in risk identification. To highlight the important values for identifying the hazardous event, safety experts in different industries can define *Safety Indicators (SIs)* considering the safety needs of the specific industry. We consider four categories for (SIs), namely: *Subject-specific SIs* (e.g., skill level, decreased mental alertness, fatigue, loss of concentration); *Object-specific SIs* (e.g., object risk level, and failure rate); *Environment-specific SIs* (e.g., fall rate); and *Activity-specific SIs* (e.g., injury rate, proximity of hazardous activities to one another, and compatibility of work activities).

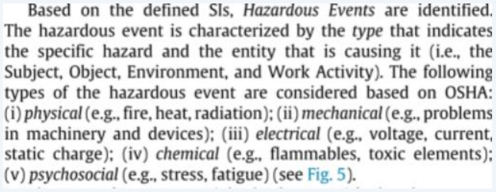
Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates the specific hazard and the entity that is causing it (i.e., the Subject, Object, Environment, and Work Activity). The following types of the hazardous event are considered based on OSHA: (i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current, static charge); (iv) *chemical* (e.g., flammables, toxic elements); (v) *psychosocial* (e.g., stress, fatigue) (see Fig. 5).

The *Hazardous Event* might lead to a *Risk* that has a *type* (e.g., fire) and a *source* (e.g., gas pipe). This can be checked in the risk analysis process that eventually calculates the probability of the *Risk*. In case the *Hazardous Event* indicates a *Risk*, further analysis should take place in order to identify the consequences of the risk (e.g., fatal injury of workers, non-fatal injury, occupational disease, harm to infrastructure, etc.) and the probability of those consequences. Furthermore, during the next step, namely risk evaluation, the decision is made about the priority of attention and the level (i.e., the intensity) of the identified *Risk*. Moreover, the *location* in the environment affected by the *Risk* is another concept that is required in the risk treatment. As depicted in Fig. 6, the mentioned concepts are defined as part of the main safety ontology.

3.4.3. The safety ontology concepts for plan step of MAPE-K loop

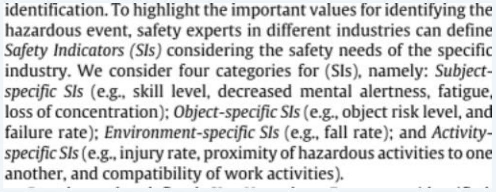
In the Plan step, the main goal is deciding about *Preventive Strategies (PSs)* as a part of risk treatment process in ISO 31000:2009. The Preventive Strategies as the main concept in this step can be characterized based on the SWE entity that it is applied to, namely, (i) *Subject-specific PS* (e.g., informing the person at risk, controlling the correct usage of subject safety protection elements such as hard hats, gloves, face shield, etc.); (ii) *Object-specific PS* (e.g., scheduling safety inspection for machinery, turning off the machinery, etc.); (iii) *Environment-specific PS* (e.g., adjusting the

Ontology development for run-time safety management methodology: 5 - 5 (0)



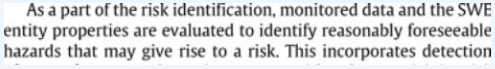
Based on the defined SIs, *Hazardous Events* are identified. The hazardous event is characterized by the *type* that indicates the specific hazard and the entity that is causing it (i.e., the Subject, Object, Environment, and Work Activity). The following types of the hazardous event are considered based on OSHA: (i) *physical* (e.g., fire, heat, radiation); (ii) *mechanical* (e.g., problems in machinery and devices); (iii) *electrical* (e.g., voltage, current, static charge); (iv) *chemical* (e.g., flammables, toxic elements); (v) *psychosocial* (e.g., stress, fatigue) (see Fig. 5).

Ontology development for run-time safety management methodology: 5 - 5 (0)



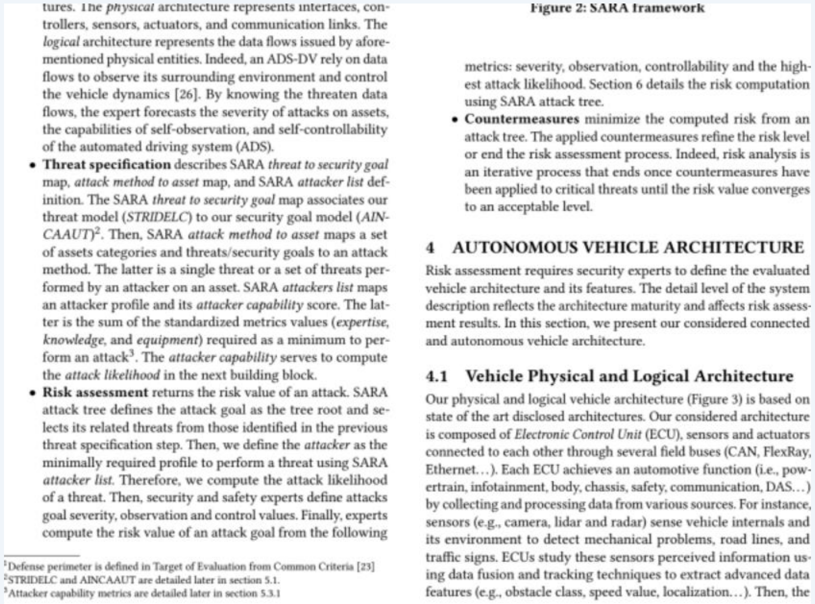
identification. To highlight the important values for identifying the hazardous event, safety experts in different industries can define *Safety Indicators (SIs)* considering the safety needs of the specific industry. We consider four categories for (SIs), namely: *Subject-specific SIs* (e.g., skill level, decreased mental alertness, fatigue, loss of concentration); *Object-specific SIs* (e.g., object risk level, and failure rate); *Environment-specific SIs* (e.g., fall rate); and *Activity-specific SIs* (e.g., injury rate, proximity of hazardous activities to one another, and compatibility of work activities).

Ontology development for run-time safety management methodology: 5 - 5 (0)



As a part of the risk identification, monitored data and the SWE entity properties are evaluated to identify reasonably foreseeable hazards that may give rise to a risk. This incorporates detection

Ontology development for run-time safety management methodology: 5 - 5 (0)



tures. The *physical* architecture represents interfaces, controllers, sensors, actuators, and communication links. The *logical* architecture represents the data flows issued by aforementioned physical entities. Indeed, an ADS-DV rely on data flows to observe its surrounding environment and control the vehicle dynamics [26]. By knowing the threaten data flows, the expert forecasts the severity of attacks on assets, the capabilities of self-observation, and self-controllability of the automated driving system (ADS).

- **Threat specification** describes SARA *threat to security goal* map, *attack method to asset* map, and SARA *attacker list* definition. The SARA *threat to security goal* map associates our threat model (*STRIDELC*) to our security goal model (*AINCAAUT*)[2]. Then, SARA *attack method to asset* maps a set of assets categories and threats/security goals to an attack method. The latter is a single threat or a set of threats performed by an attacker on an asset. SARA *attackers list* maps an attacker profile and its *attacker capability* score. The latter is the sum of the standardized metrics values (*expertise*, *knowledge*, and *equipment*) required as a minimum to perform an attack[3]. The *attacker capability* serves to compute the *attack likelihood* in the next building block.
- **Risk assessment** returns the risk value of an attack. SARA attack tree defines the attack goal as the tree root and selects its related threats from those identified in the previous threat specification step. Then, we define the *attacker* as the minimally required profile to perform a threat using SARA *attacker list*. Therefore, we compute the attack likelihood of a threat. Then, security and safety experts define attacks goal severity, observation and control values. Finally, experts compute the risk value of an attack goal from the following

Figure 2: SARA framework

metrics: severity, observation, controllability and the highest attack likelihood. Section 6 details the risk computation using SARA attack tree.

- **Countermeasures** minimize the computed risk from an attack tree. The applied countermeasures refine the risk level or end the risk assessment process. Indeed, risk analysis is an iterative process that ends once countermeasures have been applied to critical threats until the risk value converges to an acceptable level.

4 AUTONOMOUS VEHICLE ARCHITECTURE

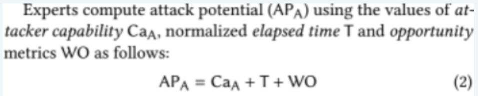
Risk assessment requires security experts to define the evaluated vehicle architecture and its features. The detail level of the system description reflects the architecture maturity and affects risk assessment results. In this section, we present our considered connected and autonomous vehicle architecture.

4.1 Vehicle Physical and Logical Architecture

Our physical and logical vehicle architecture (Figure 3) is based on state of the art disclosed architectures. Our considered architecture is composed of *Electronic Control Unit* (ECU), sensors and actuators connected to each other through several field buses (CAN, FlexRay, Ethernet…). Each ECU achieves an automotive function (i.e., powertrain, infotainment, body, chassis, safety, communication, DAS…) by collecting and processing data from various sources. For instance, sensors (e.g., camera, lidar and radar) sense vehicle internals and its environment to detect mechanical problems, road lines, and traffic signs. ECUs study these sensors perceived information using data fusion and tracking techniques to extract advanced data features (e.g., obstacle class, speed value, localization…). Then, the

[1] Defense perimeter is defined in Target of Evaluation from Common Criteria [23]
[2] STRIDELC and AINCAAUT are detailed later in section 5.1.
[3] Attacker capability metrics are detailed later in section 5.3.1

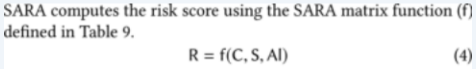
SARA Security Automotive Risk Analysis Method: 3 - 3 (0)



Experts compute attack potential (AP_A) using the values of *attacker capability* Ca_A, normalized *elapsed time* T and *opportunity* metrics WO as follows:

$$AP_A = Ca_A + T + WO \qquad (2)$$

SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

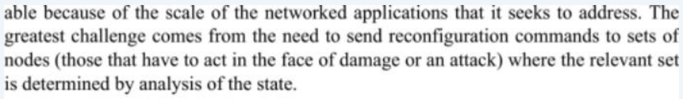


SARA computes the risk score using the SARA matrix function (f) defined in Table 9.

$$R = f(C, S, AI) \qquad (4)$$

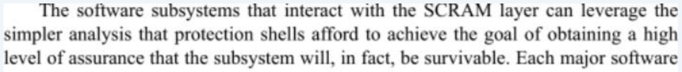
SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

Modeling Approach\Analysis Objectives\Adaptation



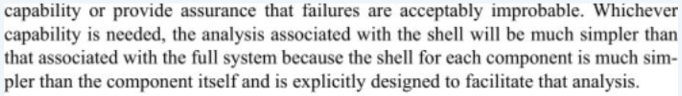
able because of the scale of the networked applications that it seeks to address. The greatest challenge comes from the need to send reconfiguration commands to sets of nodes (those that have to act in the face of damage or an attack) where the relevant set is determined by analysis of the state.

Achieving critical system survivability through software archit: 15 - 15 (0)



The software subsystems that interact with the SCRAM layer can leverage the simpler analysis that protection shells afford to achieve the goal of obtaining a high level of assurance that the subsystem will, in fact, be survivable. Each major software

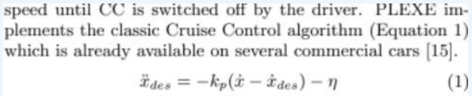
Achieving critical system survivability through software archit: 19 - 19 (0)



capability or provide assurance that failures are acceptably improbable. Whichever capability is needed, the analysis associated with the shell will be much simpler than that associated with the full system because the shell for each component is much simpler than the component itself and is explicitly designed to facilitate that analysis.

Achieving critical system survivability through software archit: 19 - 19 (0)

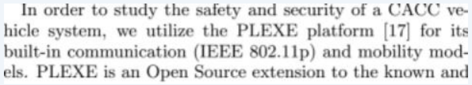
Modeling Approach\Analysis Objectives\Evaluation



speed until CC is switched off by the driver. PLEXE implements the classic Cruise Control algorithm (Equation 1) which is already available on several commercial cars [15].

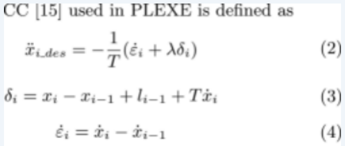
$$\ddot{x}_{des} = -k_p(\dot{x} - \dot{x}_{des}) - \eta \qquad (1)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



In order to study the safety and security of a CACC vehicle system, we utilize the PLEXE platform [17] for its built-in communication (IEEE 802.11p) and mobility models. PLEXE is an Open Source extension to the known and

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



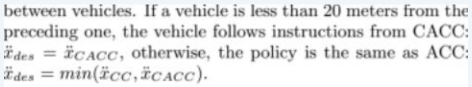
CC [15] used in PLEXE is defined as

$$\ddot{x}_{i\_des} = -\frac{1}{T}(\dot{\varepsilon}_i + \lambda\delta_i) \qquad (2)$$

$$\delta_i = x_i - x_{i-1} + l_{i-1} + T\dot{x}_i \qquad (3)$$

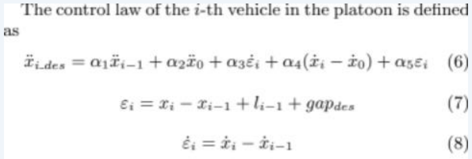
$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (4)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



between vehicles. If a vehicle is less than 20 meters from the preceding one, the vehicle follows instructions from CACC: $\ddot{x}_{des} = \ddot{x}_{CACC}$, otherwise, the policy is the same as ACC: $\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{CACC})$.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)



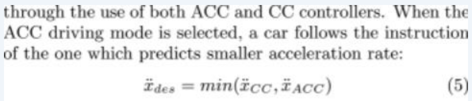
The control law of the i-th vehicle in the platoon is defined as

$$\ddot{x}_{i\_des} = \alpha_1\ddot{x}_{i-1} + \alpha_2\ddot{x}_0 + \alpha_3\dot{\varepsilon}_i + \alpha_4(\dot{x}_i - \dot{x}_0) + \alpha_5\varepsilon_i \qquad (6)$$

$$\varepsilon_i = x_i - x_{i-1} + l_{i-1} + gap_{des} \qquad (7)$$

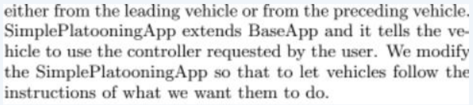
$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1} \qquad (8)$$

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)

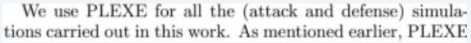

through the use of both ACC and CC controllers. When the ACC driving mode is selected, a car follows the instruction of the one which predicts smaller acceleration rate:

$$\ddot{x}_{des} = min(\ddot{x}_{CC}, \ddot{x}_{ACC}) \qquad (5)$$

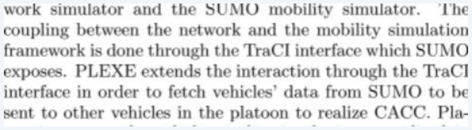
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 3 - 3 (0)


either from the leading vehicle or from the preceding vehicle. SimplePlatooningApp extends BaseApp and it tells the vehicle to use the controller requested by the user. We modify the SimplePlatooningApp so that to let vehicles follow the instructions of what we want them to do.

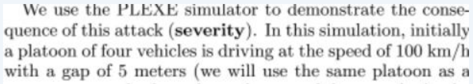
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)


We use PLEXE for all the (attack and defense) simulations carried out in this work. As mentioned earlier, PLEXE

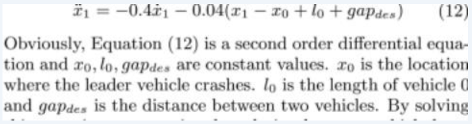
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)


work simulator and the SUMO mobility simulator. The coupling between the network and the mobility simulation framework is done through the TraCI interface which SUMO exposes. PLEXE extends the interaction through the TraCI interface in order to fetch vehicles' data from SUMO to be sent to other vehicles in the platoon to realize CACC. Pla-

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)

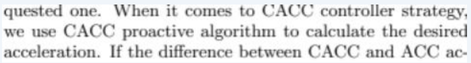

We use the PLEXE simulator to demonstrate the consequence of this attack (**severity**). In this simulation, initially a platoon of four vehicles is driving at the speed of 100 km/h with a gap of 5 meters (we will use the same platoon as a

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)

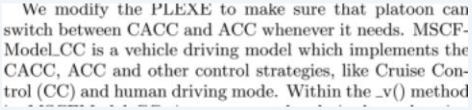

$$\ddot{x}_1 = -0.4\dot{x}_1 - 0.04(x_1 - x_0 + l_0 + gap_{des}) \qquad (12)$$

Obviously, Equation (12) is a second order differential equation and x_0, l_0, gap_{des} are constant values. x_0 is the location where the leader vehicle crashes. l_0 is the length of vehicle 0 and gap_{des} is the distance between two vehicles. By solving

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 7 - 7 (0)

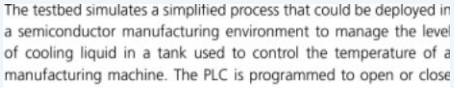

quested one. When it comes to CACC controller strategy, we use CACC proactive algorithm to calculate the desired acceleration. If the difference between CACC and ACC ac-

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 8 - 8 (0)



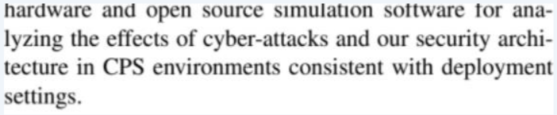
We modify the PLEXE to make sure that platoon can switch between CACC and ACC whenever it needs. MSCFModel_CC is a vehicle driving model which implements the CACC, ACC and other control strategies, like Cruise Control (CC) and human driving mode. Within the _v() method

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 8 - 8 (0)



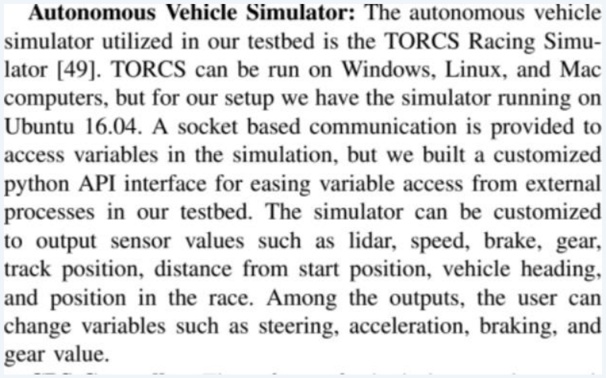
The testbed simulates a simplified process that could be deployed in a semiconductor manufacturing environment to manage the level of cooling liquid in a tank used to control the temperature of a manufacturing machine. The PLC is programmed to open or close

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



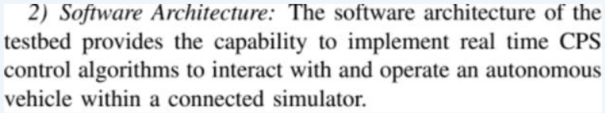
hardware and open source simulation software for analyzing the effects of cyber-attacks and our security architecture in CPS environments consistent with deployment settings.

Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)



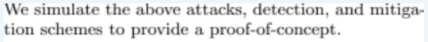
Autonomous Vehicle Simulator: The autonomous vehicle simulator utilized in our testbed is the TORCS Racing Simulator [49]. TORCS can be run on Windows, Linux, and Mac computers, but for our setup we have the simulator running on Ubuntu 16.04. A socket based communication is provided to access variables in the simulation, but we built a customized python API interface for easing variable access from external processes in our testbed. The simulator can be customized to output sensor values such as lidar, speed, brake, gear, track position, distance from start position, vehicle heading, and position in the race. Among the outputs, the user can change variables such as steering, acceleration, braking, and gear value.

Integrated moving target defense and control reconfiguration fo: 7 - 7 (0)

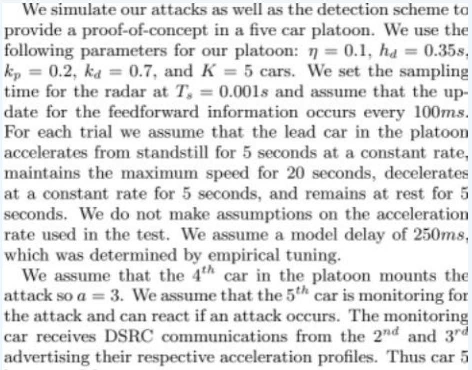


2) *Software Architecture:* The software architecture of the testbed provides the capability to implement real time CPS control algorithms to interact with and operate an autonomous vehicle within a connected simulator.

Integrated moving target defense and control reconfiguration fo: 7 - 7 (0)

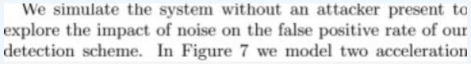

We simulate the above attacks, detection, and mitigation schemes to provide a proof-of-concept.

Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)

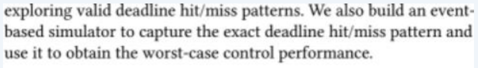

We simulate our attacks as well as the detection scheme to provide a proof-of-concept in a five car platoon. We use the following parameters for our platoon: $\eta = 0.1$, $h_d = 0.35s$, $k_p = 0.2$, $k_d = 0.7$, and $K = 5$ cars. We set the sampling time for the radar at $T_s = 0.001s$ and assume that the update for the feedforward information occurs every $100ms$. For each trial we assume that the lead car in the platoon accelerates from standstill for 5 seconds at a constant rate, maintains the maximum speed for 20 seconds, decelerates at a constant rate for 5 seconds, and remains at rest for 5 seconds. We do not make assumptions on the acceleration rate used in the test. We assume a model delay of $250ms$, which was determined by empirical tuning.

We assume that the 4^{th} car in the platoon mounts the attack so $a = 3$. We assume that the 5^{th} car is monitoring for the attack and can react if an attack occurs. The monitoring car receives DSRC communications from the 2^{nd} and 3^{rd} advertising their respective acceleration profiles. Thus car 5

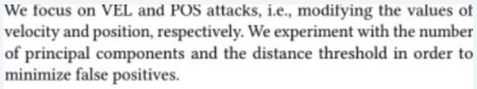
Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)


We simulate the system without an attacker present to explore the impact of noise on the false positive rate of our detection scheme. In Figure 7 we model two acceleration

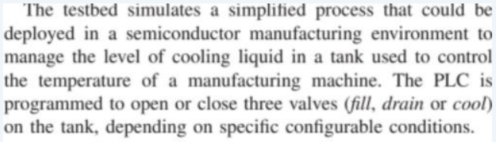
Is your commute driving you crazy a study of misbehavior in veh: 9 - 9 (0)


exploring valid deadline hit/miss patterns. We also build an event-based simulator to capture the exact deadline hit/miss pattern and use it to obtain the worst-case control performance.

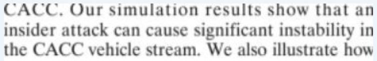
Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)


We focus on VEL and POS attacks, i.e., modifying the values of velocity and position, respectively. We experiment with the number of principal components and the distance threshold in order to minimize false positives.

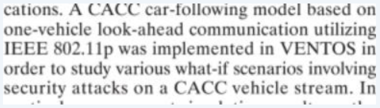
Network and system level security in connected vehicle applicat: 4 - 4 (0)


The testbed simulates a simplified process that could be deployed in a semiconductor manufacturing environment to manage the level of cooling liquid in a tank used to control the temperature of a manufacturing machine. The PLC is programmed to open or close three valves (*fill*, *drain* or *cool*) on the tank, depending on specific configurable conditions.

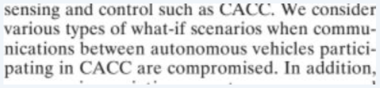
Protecting cyber physical production systems using anomaly dete: 5 - 5 (0)


CACC. Our simulation results show that an insider attack can cause significant instability in the CACC vehicle stream. We also illustrate how

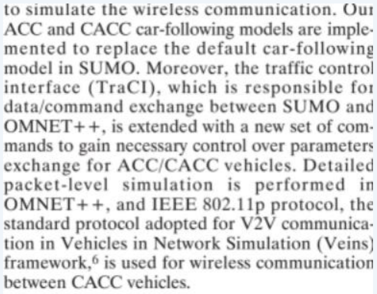
Security vulnerabilities of connected vehicle streams and their: 1 - 1 (0)


cations. A CACC car-following model based on one-vehicle look-ahead communication utilizing IEEE 802.11p was implemented in VENTOS in order to study various what-if scenarios involving security attacks on a CACC vehicle stream. In

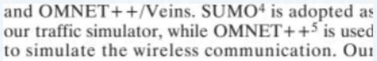
Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)


sensing and control such as CACC. We consider various types of what-if scenarios when communications between autonomous vehicles participating in CACC are compromised. In addition,

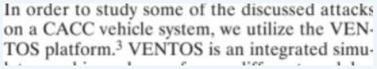
Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)


to simulate the wireless communication. Our ACC and CACC car-following models are implemented to replace the default car-following model in SUMO. Moreover, the traffic control interface (TraCI), which is responsible for data/command exchange between SUMO and OMNET++, is extended with a new set of commands to gain necessary control over parameters exchange for ACC/CACC vehicles. Detailed packet-level simulation is performed in OMNET++, and IEEE 802.11p protocol, the standard protocol adopted for V2V communication in Vehicles in Network Simulation (Veins) framework,[6] is used for wireless communication between CACC vehicles.

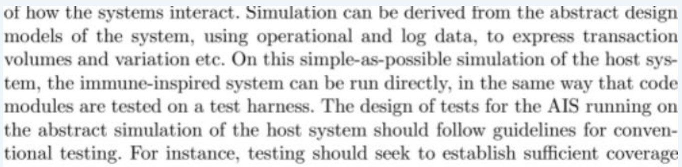
Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)


and OMNET++/Veins. SUMO[4] is adopted as our traffic simulator, while OMNET++[5] is used to simulate the wireless communication. Our

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)


In order to study some of the discussed attacks on a CACC vehicle system, we utilize the VENTOS platform.[3] VENTOS is an integrated simu-

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)

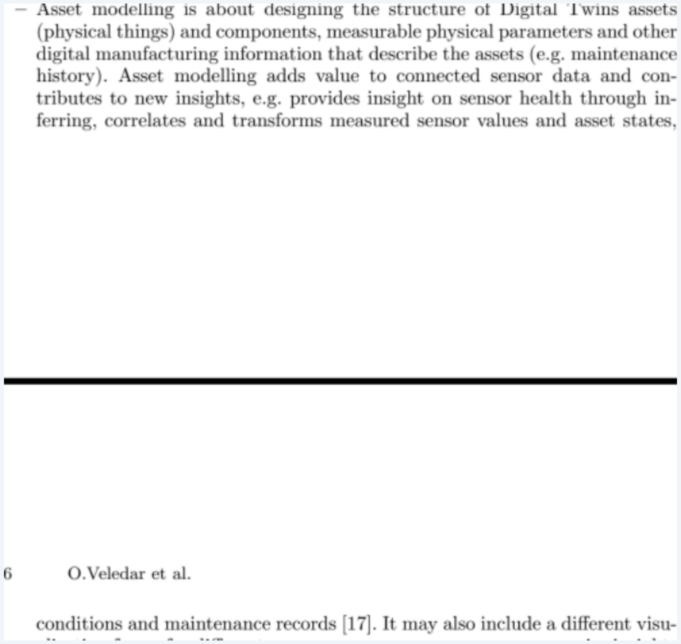


of how the systems interact. Simulation can be derived from the abstract design models of the system, using operational and log data, to express transaction volumes and variation etc. On this simple-as-possible simulation of the host system, the immune-inspired system can be run directly, in the same way that code modules are tested on a test harness. The design of tests for the AIS running on the abstract simulation of the host system should follow guidelines for conventional testing. For instance, testing should seek to establish sufficient coverage

Only discusses the usage of simulation

Self-organisation for Survival in Complex Computer Architecture: 15 - 15 (0)

Modeling Approach\Analysis Objectives\Other

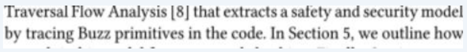


– Asset modelling is about designing the structure of Digital Twins assets (physical things) and components, measurable physical parameters and other digital manufacturing information that describe the assets (e.g. maintenance history). Asset modelling adds value to connected sensor data and contributes to new insights, e.g. provides insight on sensor health through inferring, correlates and transforms measured sensor values and asset states,

6 O.Veledar et al.

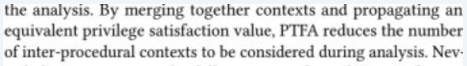
conditions and maintenance records [17]. It may also include a different visu-

Digital Twins for Dependability Improvement of Autonomous Drivi: 6 - 7 (0)



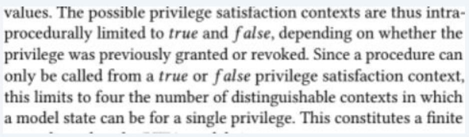
Traversal Flow Analysis [8] that extracts a safety and security model by tracing Buzz primitives in the code. In Section 5, we outline how

Engineering safety in swarm robotics: 2 - 2 (0)



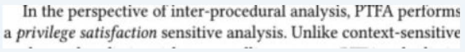
the analysis. By merging together contexts and propagating an equivalent privilege satisfaction value, PTFA reduces the number of inter-procedural contexts to be considered during analysis. Nev-

Engineering safety in swarm robotics: 3 - 3 (0)



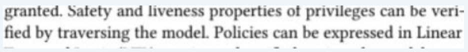
values. The possible privilege satisfaction contexts are thus intra-procedurally limited to *true* and *false*, depending on whether the privilege was previously granted or revoked. Since a procedure can only be called from a *true* or *false* privilege satisfaction context, this limits to four the number of distinguishable contexts in which a model state can be for a single privilege. This constitutes a finite

Engineering safety in swarm robotics: 3 - 3 (0)



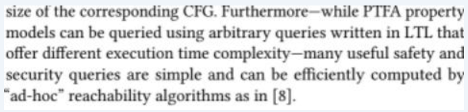
In the perspective of inter-procedural analysis, PTFA performs a *privilege satisfaction* sensitive analysis. Unlike context-sensitive

Engineering safety in swarm robotics: 3 - 3 (0)



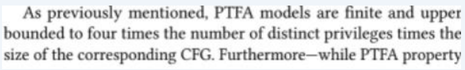
granted. Safety and liveness properties of privileges can be verified by traversing the model. Policies can be expressed in Linear

Engineering safety in swarm robotics: 3 - 3 (0)



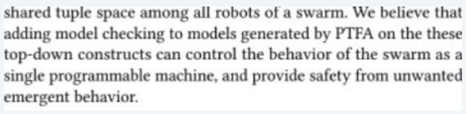
size of the corresponding CFG. Furthermore—while PTFA property models can be queried using arbitrary queries written in LTL that offer different execution time complexity—many useful safety and security queries are simple and can be efficiently computed by "ad-hoc" reachability algorithms as in [8].

Engineering safety in swarm robotics: 4 - 4 (0)



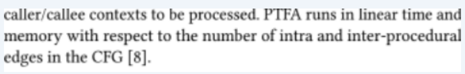
As previously mentioned, PTFA models are finite and upper bounded to four times the number of distinct privileges times the size of the corresponding CFG. Furthermore—while PTFA property

Engineering safety in swarm robotics: 4 - 4 (0)



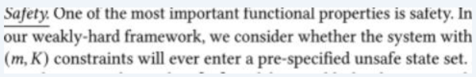
shared tuple space among all robots of a swarm. We believe that adding model checking to models generated by PTFA on the these top-down constructs can control the behavior of the swarm as a single programmable machine, and provide safety from unwanted emergent behavior.

Engineering safety in swarm robotics: 4 - 4 (0)



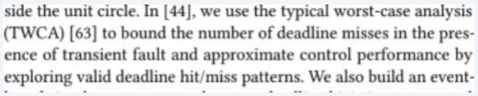
caller/callee contexts to be processed. PTFA runs in linear time and memory with respect to the number of intra and inter-procedural edges in the CFG [8].

Engineering safety in swarm robotics: 4 - 4 (0)



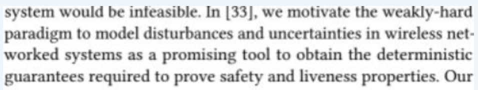
Safety. One of the most important functional properties is safety. In our weakly-hard framework, we consider whether the system with (m, K) constraints will ever enter a pre-specified unsafe state set.

Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)



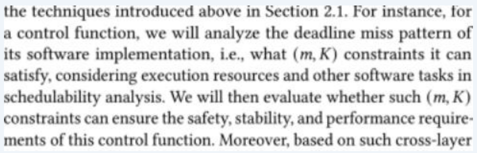
side the unit circle. In [44], we use the typical worst-case analysis (TWCA) [63] to bound the number of deadline misses in the presence of transient fault and approximate control performance by exploring valid deadline hit/miss patterns. We also build an event-

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



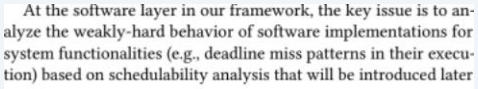
system would be infeasible. In [33], we motivate the weakly-hard paradigm to model disturbances and uncertainties in wireless networked systems as a promising tool to obtain the deterministic guarantees required to prove safety and liveness properties. Our

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



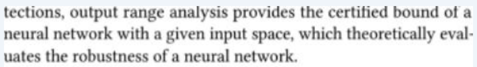
the techniques introduced above in Section 2.1. For instance, for a control function, we will analyze the deadline miss pattern of its software implementation, i.e., what (m, K) constraints it can satisfy, considering execution resources and other software tasks in schedulability analysis. We will then evaluate whether such (m, K) constraints can ensure the safety, stability, and performance requirements of this control function. Moreover, based on such cross-layer

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



At the software layer in our framework, the key issue is to analyze the weakly-hard behavior of software implementations for system functionalities (e.g., deadline miss patterns in their execution) based on schedulability analysis that will be introduced later

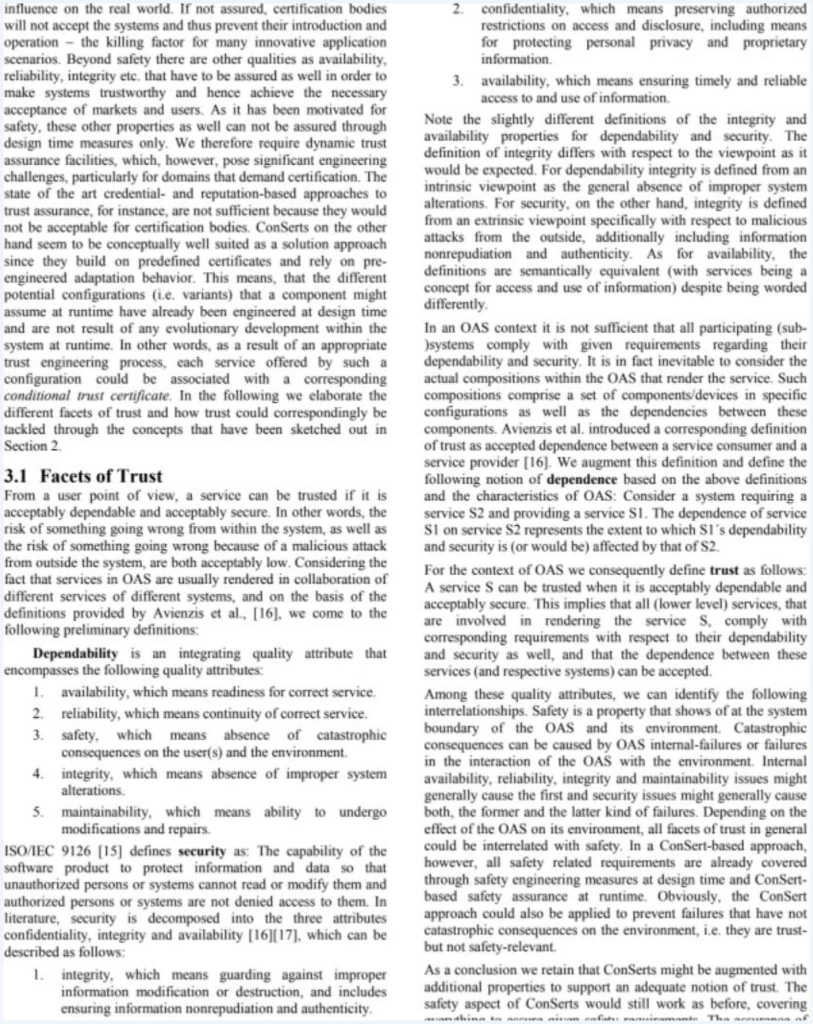
Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



tections, output range analysis provides the certified bound of a neural network with a given input space, which theoretically evaluates the robustness of a neural network.

Know the unknowns addressing disturbances and uncertainties in: 6 - 6 (0)

Modeling Approach\Adaptation Property Checked



influence on the real world. If not assured, certification bodies will not accept the systems and thus prevent their introduction and operation – the killing factor for many innovative application scenarios. Beyond safety there are other qualities as availability, reliability, integrity etc. that have to be assured as well in order to make systems trustworthy and hence achieve the necessary acceptance of markets and users. As it has been motivated for safety, these other properties as well can not be assured through design time measures only. We therefore require dynamic trust assurance facilities, which, however, pose significant engineering challenges, particularly for domains that demand certification. The state of the art credential- and reputation-based approaches to trust assurance, for instance, are not sufficient because they would not be acceptable for certification bodies. ConSerts on the other hand seem to be conceptually well suited as a solution approach since they build on predefined certificates and rely on pre-engineered adaptation behavior. This means, that the different potential configurations (i.e. variants) that a component might assume at runtime have already been engineered at design time and are not result of any evolutionary development within the system at runtime. In other words, as a result of an appropriate trust engineering process, each service offered by such a configuration could be associated with a corresponding *conditional trust certificate*. In the following we elaborate the different facets of trust and how trust could correspondingly be tackled through the concepts that have been sketched out in Section 2.

3.1 Facets of Trust

From a user point of view, a service can be trusted if it is acceptably dependable and acceptably secure. In other words, the risk of something going wrong from within the system, as well as the risk of something going wrong because of a malicious attack from outside the system, are both acceptably low. Considering the fact that services in OAS are usually rendered in collaboration of different services of different systems, and on the basis of the definitions provided by Avienzis et al., [16], we come to the following preliminary definitions:

Dependability is an integrating quality attribute that encompasses the following quality attributes:

1. availability, which means readiness for correct service.
2. reliability, which means continuity of correct service.
3. safety, which means absence of catastrophic consequences on the user(s) and the environment.
4. integrity, which means absence of improper system alterations.
5. maintainability, which means ability to undergo modifications and repairs.

ISO/IEC 9126 [15] defines **security** as: The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them. In literature, security is decomposed into the three attributes confidentiality, integrity and availability [16][17], which can be described as follows:

1. integrity, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity.
2. confidentiality, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information.
3. availability, which means ensuring timely and reliable access to and use of information.

Note the slightly different definitions of the integrity and availability properties for dependability and security. The definition of integrity differs with respect to the viewpoint as it would be expected. For dependability integrity is defined from an intrinsic viewpoint as the general absence of improper system alterations. For security, on the other hand, integrity is defined from an extrinsic viewpoint specifically with respect to malicious attacks from the outside, additionally including information nonrepudiation and authenticity. As for availability, the definitions are semantically equivalent (with services being a concept for access and use of information) despite being worded differently.

In an OAS context it is not sufficient that all participating (sub-)systems comply with given requirements regarding their dependability and security. It is in fact inevitable to consider the actual compositions within the OAS that render the service. Such compositions comprise a set of components/devices in specific configurations as well as the dependencies between these components. Avienzis et al. introduced a corresponding definition of trust as accepted dependence between a service consumer and a service provider [16]. We augment this definition and define the following notion of **dependence** based on the above definitions and the characteristics of OAS: Consider a system requiring a service S2 and providing a service S1. The dependence of service S1 on service S2 represents the extent to which S1´s dependability and security is (or would be) affected by that of S2.

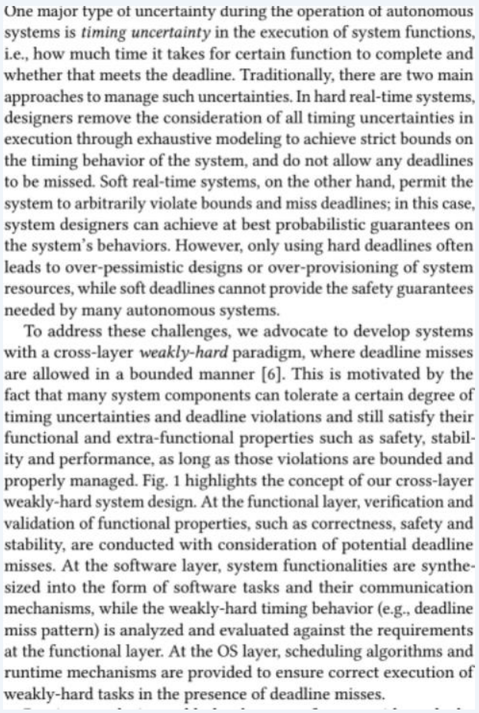
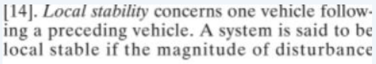
For the context of OAS we consequently define **trust** as follows: A service S can be trusted when it is acceptably dependable and acceptably secure. This implies that all (lower level) services, that are involved in rendering the service S, comply with corresponding requirements with respect to their dependability and security as well, and that the dependence between these services (and respective systems) can be accepted.

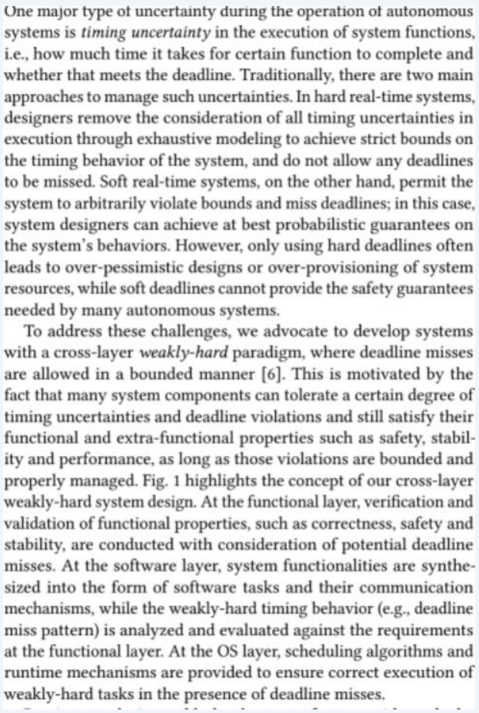
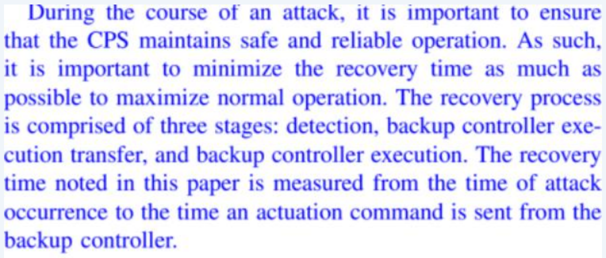
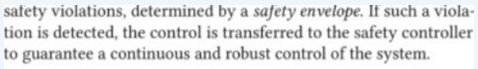
Among these quality attributes, we can identify the following interrelationships. Safety is a property that shows of at the system boundary of the OAS and its environment. Catastrophic consequences can be caused by OAS internal-failures or failures in the interaction of the OAS with the environment. Internal availability, reliability, integrity and maintainability issues might generally cause the first and security issues might generally cause both, the former and the latter kind of failures. Depending on the effect of the OAS on its environment, all facets of trust in general could be interrelated with safety. In a ConSert-based approach, however, all safety related requirements are already covered through safety engineering measures at design time and ConSert-based safety assurance at runtime. Obviously, the ConSert approach could also be applied to prevent failures that have not catastrophic consequences on the environment, i.e. they are trust- but not safety-relevant.

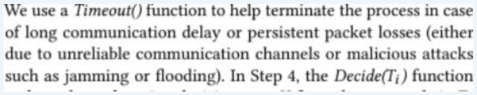
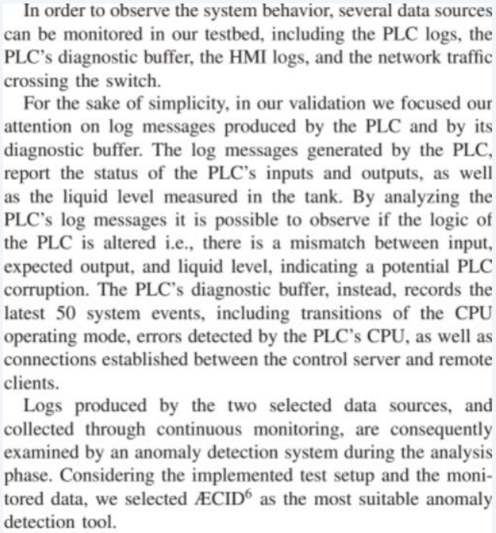
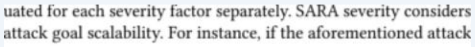
As a conclusion we retain that ConSerts might be augmented with additional properties to support an adequate notion of trust. The safety aspect of ConSerts would still work as before, covering everything to assure given safety requirements. The assurance of

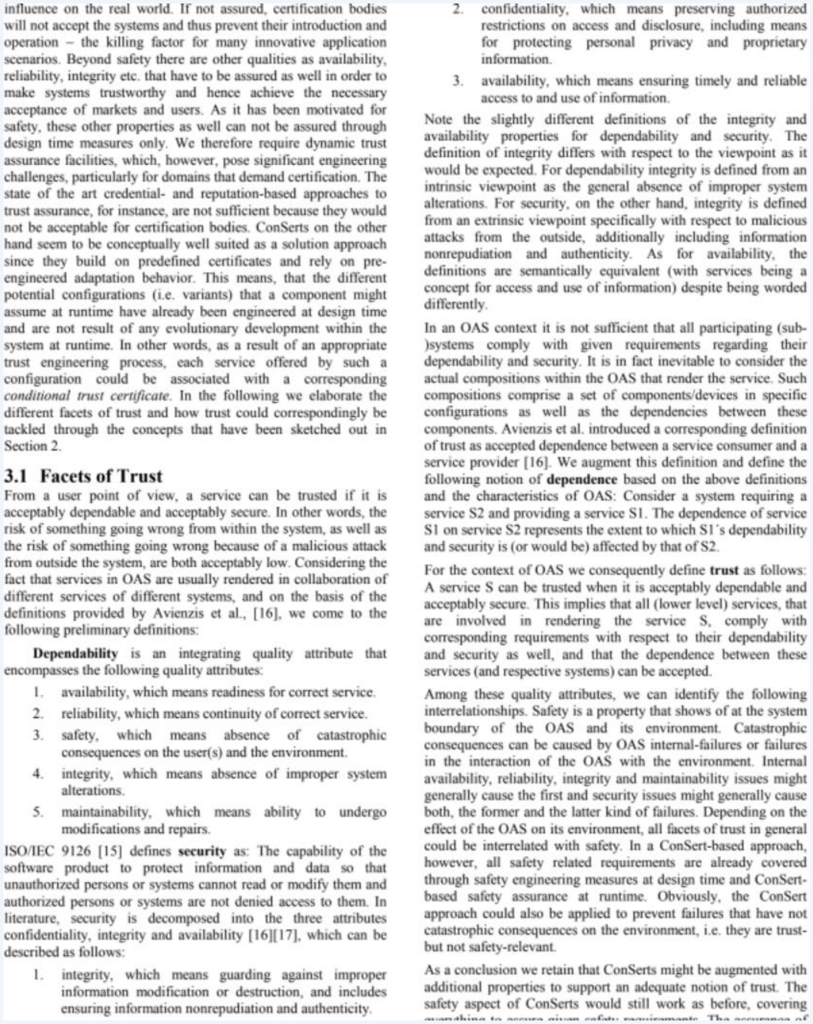
Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)

Modeling Approach\Adaptation Property Checked\Stability

	 One major type of uncertainty during the operation of autonomous systems is *timing uncertainty* in the execution of system functions, i.e., how much time it takes for certain function to complete and whether that meets the deadline. Traditionally, there are two main approaches to manage such uncertainties. In hard real-time systems, designers remove the consideration of all timing uncertainties in execution through exhaustive modeling to achieve strict bounds on the timing behavior of the system, and do not allow any deadlines to be missed. Soft real-time systems, on the other hand, permit the system to arbitrarily violate bounds and miss deadlines; in this case, system designers can achieve at best probabilistic guarantees on the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems. To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses. Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)
	 [14]. *Local stability* concerns one vehicle following a preceding vehicle. A system is said to be local stable if the magnitude of disturbance Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)
Modeling Approach\Adaptation Property Checked\Accuracy	

	 One major type of uncertainty during the operation of autonomous systems is *timing uncertainty* in the execution of system functions, i.e., how much time it takes for certain function to complete and whether that meets the deadline. Traditionally, there are two main approaches to manage such uncertainties. In hard real-time systems, designers remove the consideration of all timing uncertainties in execution through exhaustive modeling to achieve strict bounds on the timing behavior of the system, and do not allow any deadlines to be missed. Soft real-time systems, on the other hand, permit the system to arbitrarily violate bounds and miss deadlines; in this case, system designers can achieve at best probabilistic guarantees on the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems. To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses. Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)
Modeling Approach\Adaptation Property Checked\Settling Time	 During the course of an attack, it is important to ensure that the CPS maintains safe and reliable operation. As such, it is important to minimize the recovery time as much as possible to maximize normal operation. The recovery process is comprised of three stages: detection, backup controller execution transfer, and backup controller execution. The recovery time noted in this paper is measured from the time of attack occurrence to the time an actuation command is sent from the backup controller. Integrated moving target defense and control reconfiguration fo: 6 - 6 (0)
Modeling Approach\Adaptation Property Checked\Robustness	 safety violations, determined by a *safety envelope*. If such a violation is detected, the control is transferred to the safety controller to guarantee a continuous and robust control of the system.

	VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)
Modeling Approach\Adaptation Property Checked\Termination	 We use a *Timeout()* function to help terminate the process in case of long communication delay or persistent packet losses (either due to unreliable communication channels or malicious attacks such as jamming or flooding). In Step 4, the *Decide*(T_i) function Network and system level security in connected vehicle applicat: 6 - 6 (0)
Modeling Approach\Adaptation Property Checked\Consistency	 In order to observe the system behavior, several data sources can be monitored in our testbed, including the PLC logs, the PLC's diagnostic buffer, the HMI logs, and the network traffic crossing the switch. For the sake of simplicity, in our validation we focused our attention on log messages produced by the PLC and by its diagnostic buffer. The log messages generated by the PLC, report the status of the PLC's inputs and outputs, as well as the liquid level measured in the tank. By analyzing the PLC's log messages it is possible to observe if the logic of the PLC is altered i.e., there is a mismatch between input, expected output, and liquid level, indicating a potential PLC corruption. The PLC's diagnostic buffer, instead, records the latest 50 system events, including transitions of the CPU operating mode, errors detected by the PLC's CPU, as well as connections established between the control server and remote clients. Logs produced by the two selected data sources, and collected through continuous monitoring, are consequently examined by an anomaly detection system during the analysis phase. Considering the implemented test setup and the monitored data, we selected ÆCID[6] as the most suitable anomaly detection tool. Protecting cyber physical production systems using anomaly dete: 5 - 5 (0)
Modeling Approach\Adaptation Property Checked\Scalability	 uated for each severity factor separately. SARA severity considers attack goal scalability. For instance, if the aforementioned attack SARA Security Automotive Risk Analysis Method: 7 - 7 (0)
Modeling Approach\Adaptation Property Checked\Security	



influence on the real world. If not assured, certification bodies will not accept the systems and thus prevent their introduction and operation – the killing factor for many innovative application scenarios. Beyond safety there are other qualities as availability, reliability, integrity etc. that have to be assured as well in order to make systems trustworthy and hence achieve the necessary acceptance of markets and users. As it has been motivated for safety, these other properties as well can not be assured through design time measures only. We therefore require dynamic trust assurance facilities, which, however, pose significant engineering challenges, particularly for domains that demand certification. The state of the art credential- and reputation-based approaches to trust assurance, for instance, are not sufficient because they would not be acceptable for certification bodies. ConSerts on the other hand seem to be conceptually well suited as a solution approach since they build on predefined certificates and rely on pre-engineered adaptation behavior. This means, that the different potential configurations (i.e. variants) that a component might assume at runtime have already been engineered at design time and are not result of any evolutionary development within the system at runtime. In other words, as a result of an appropriate trust engineering process, each service offered by such a configuration could be associated with a corresponding *conditional trust certificate*. In the following we elaborate the different facets of trust and how trust could correspondingly be tackled through the concepts that have been sketched out in Section 2.

3.1 Facets of Trust

From a user point of view, a service can be trusted if it is acceptably dependable and acceptably secure. In other words, the risk of something going wrong from within the system, as well as the risk of something going wrong because of a malicious attack from outside the system, are both acceptably low. Considering the fact that services in OAS are usually rendered in collaboration of different services of different systems, and on the basis of the definitions provided by Avienzis et al., [16], we come to the following preliminary definitions:

Dependability is an integrating quality attribute that encompasses the following quality attributes:

1. availability, which means readiness for correct service.
2. reliability, which means continuity of correct service.
3. safety, which means absence of catastrophic consequences on the user(s) and the environment.
4. integrity, which means absence of improper system alterations.
5. maintainability, which means ability to undergo modifications and repairs.

ISO/IEC 9126 [15] defines **security** as: The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them. In literature, security is decomposed into the three attributes confidentiality, integrity and availability [16][17], which can be described as follows:

1. integrity, which means guarding against improper information modification or destruction, and includes ensuring information nonrepudiation and authenticity.
2. confidentiality, which means preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information.
3. availability, which means ensuring timely and reliable access to and use of information.

Note the slightly different definitions of the integrity and availability properties for dependability and security. The definition of integrity differs with respect to the viewpoint as it would be expected. For dependability integrity is defined from an intrinsic viewpoint as the general absence of improper system alterations. For security, on the other hand, integrity is defined from an extrinsic viewpoint specifically with respect to malicious attacks from the outside, additionally including information nonrepudiation and authenticity. As for availability, the definitions are semantically equivalent (with services being a concept for access and use of information) despite being worded differently.

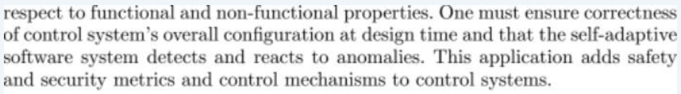
In an OAS context it is not sufficient that all participating (sub-)systems comply with given requirements regarding their dependability and security. It is in fact inevitable to consider the actual compositions within the OAS that render the service. Such compositions comprise a set of components/devices in specific configurations as well as the dependencies between these components. Avienzis et al. introduced a corresponding definition of trust as accepted dependence between a service consumer and a service provider [16]. We augment this definition and define the following notion of **dependence** based on the above definitions and the characteristics of OAS: Consider a system requiring a service S2 and providing a service S1. The dependence of service S1 on service S2 represents the extent to which S1´s dependability and security is (or would be) affected by that of S2.

For the context of OAS we consequently define **trust** as follows: A service S can be trusted when it is acceptably dependable and acceptably secure. This implies that all (lower level) services, that are involved in rendering the service S, comply with corresponding requirements with respect to their dependability and security as well, and that the dependence between these services (and respective systems) can be accepted.

Among these quality attributes, we can identify the following interrelationships. Safety is a property that shows of at the system boundary of the OAS and its environment. Catastrophic consequences can be caused by OAS internal-failures or failures in the interaction of the OAS with the environment. Internal availability, reliability, integrity and maintainability issues might generally cause the first and security issues might generally cause both, the former and the latter kind of failures. Depending on the effect of the OAS on its environment, all facets of trust in general could be interrelated with safety. In a ConSert-based approach, however, all safety related requirements are already covered through safety engineering measures at design time and ConSert-based safety assurance at runtime. Obviously, the ConSert approach could also be applied to prevent failures that have not catastrophic consequences on the environment, i.e. they are trust- but not safety-relevant.

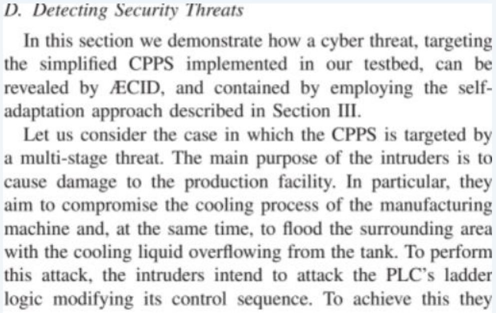
As a conclusion we retain that ConSerts might be augmented with additional properties to support an adequate notion of trust. The safety aspect of ConSerts would still work as before, covering everything to assure given safety requirements. The assurance of

Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)



respect to functional and non-functional properties. One must ensure correctness of control system's overall configuration at design time and that the self-adaptive software system detects and reacts to anomalies. This application adds safety and security metrics and control mechanisms to control systems.

Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)

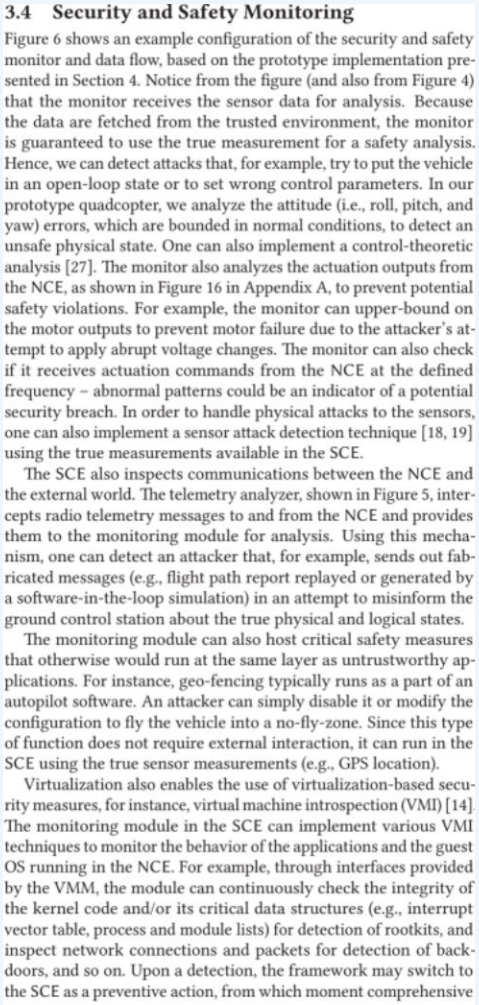


D. Detecting Security Threats

In this section we demonstrate how a cyber threat, targeting the simplified CPPS implemented in our testbed, can be revealed by ÆCID, and contained by employing the self-adaptation approach described in Section III.

Let us consider the case in which the CPPS is targeted by a multi-stage threat. The main purpose of the intruders is to cause damage to the production facility. In particular, they aim to compromise the cooling process of the manufacturing machine and, at the same time, to flood the surrounding area with the cooling liquid overflowing from the tank. To perform this attack, the intruders intend to attack the PLC's ladder logic modifying its control sequence. To achieve this they

Protecting cyber physical production systems using anomaly dete: 6 - 6 (0)



3.4 Security and Safety Monitoring

Figure 6 shows an example configuration of the security and safety monitor and data flow, based on the prototype implementation presented in Section 4. Notice from the figure (and also from Figure 4) that the monitor receives the sensor data for analysis. Because the data are fetched from the trusted environment, the monitor is guaranteed to use the true measurement for a safety analysis. Hence, we can detect attacks that, for example, try to put the vehicle in an open-loop state or to set wrong control parameters. In our prototype quadcopter, we analyze the attitude (i.e., roll, pitch, and yaw) errors, which are bounded in normal conditions, to detect an unsafe physical state. One can also implement a control-theoretic analysis [27]. The monitor also analyzes the actuation outputs from the NCE, as shown in Figure 16 in Appendix A, to prevent potential safety violations. For example, the monitor can upper-bound on the motor outputs to prevent motor failure due to the attacker's attempt to apply abrupt voltage changes. The monitor can also check if it receives actuation commands from the NCE at the defined frequency – abnormal patterns could be an indicator of a potential security breach. In order to handle physical attacks to the sensors, one can also implement a sensor attack detection technique [18, 19] using the true measurements available in the SCE.

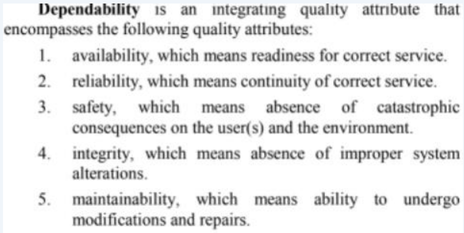
The SCE also inspects communications between the NCE and the external world. The telemetry analyzer, shown in Figure 5, intercepts radio telemetry messages to and from the NCE and provides them to the monitoring module for analysis. Using this mechanism, one can detect an attacker that, for example, sends out fabricated messages (e.g., flight path report replayed or generated by a software-in-the-loop simulation) in an attempt to misinform the ground control station about the true physical and logical states.

The monitoring module can also host critical safety measures that otherwise would run at the same layer as untrustworthy applications. For instance, geo-fencing typically runs as a part of an autopilot software. An attacker can simply disable it or modify the configuration to fly the vehicle into a no-fly-zone. Since this type of function does not require external interaction, it can run in the SCE using the true sensor measurements (e.g., GPS location).

Virtualization also enables the use of virtualization-based security measures, for instance, virtual machine introspection (VMI) [14]. The monitoring module in the SCE can implement various VMI techniques to monitor the behavior of the applications and the guest OS running in the NCE. For example, through interfaces provided by the VMM, the module can continuously check the integrity of the kernel code and/or its critical data structures (e.g., interrupt vector table, process and module lists) for detection of rootkits, and inspect network connections and packets for detection of backdoors, and so on. Upon a detection, the framework may switch to the SCE as a preventive action, from which moment comprehensive

VirtualDrone virtual sensing actuation and communication for at: 5 - 5 (0)

Modeling Approach\Adaptation
Property Checked\Dependability

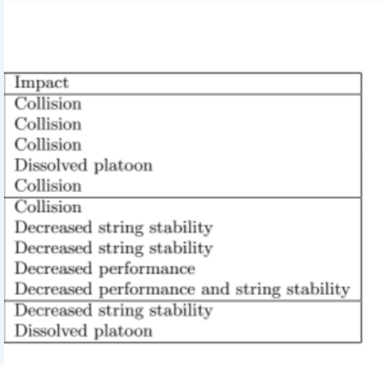


Dependability is an integrating quality attribute that encompasses the following quality attributes:

1. availability, which means readiness for correct service.
2. reliability, which means continuity of correct service.
3. safety, which means absence of catastrophic consequences on the user(s) and the environment.
4. integrity, which means absence of improper system alterations.
5. maintainability, which means ability to undergo modifications and repairs.

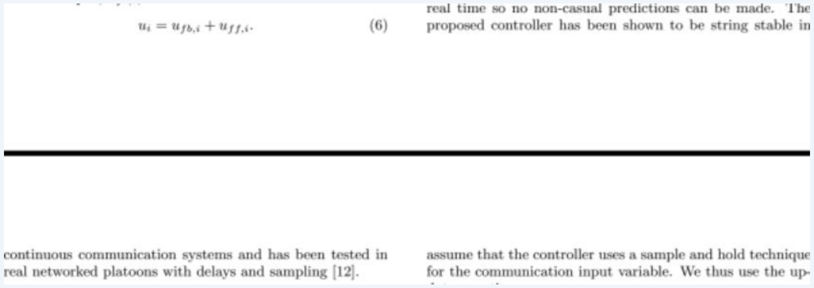
Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)

Modeling Approach\Adaptation Property Checked\String Stability



Impact
Collision
Collision
Collision
Dissolved platoon
Collision
Collision
Decreased string stability
Decreased string stability
Decreased performance
Decreased performance and string stability
Decreased string stability
Dissolved platoon

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5: 301|592 - 5: 497|775 (0)



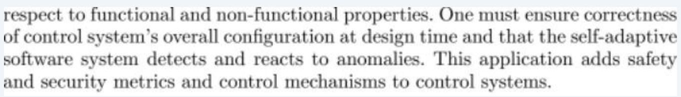
$$u_i = u_{fb,i} + u_{ff,i}. \quad (6)$$

real time so no non-casual predictions can be made. The proposed controller has been shown to be string stable in

continuous communication systems and has been tested in real networked platoons with delays and sampling [12].

assume that the controller uses a sample and hold technique for the communication input variable. We thus use the up-

Is your commute driving you crazy a study of misbehavior in veh: 3 - 4 (0)

Modeling Approach\Adaptation Property Checked\Controllability	 respect to functional and non-functional properties. One must ensure correctness of control system's overall configuration at design time and that the self-adaptive software system detects and reacts to anomalies. This application adds safety and security metrics and control mechanisms to control systems. Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)

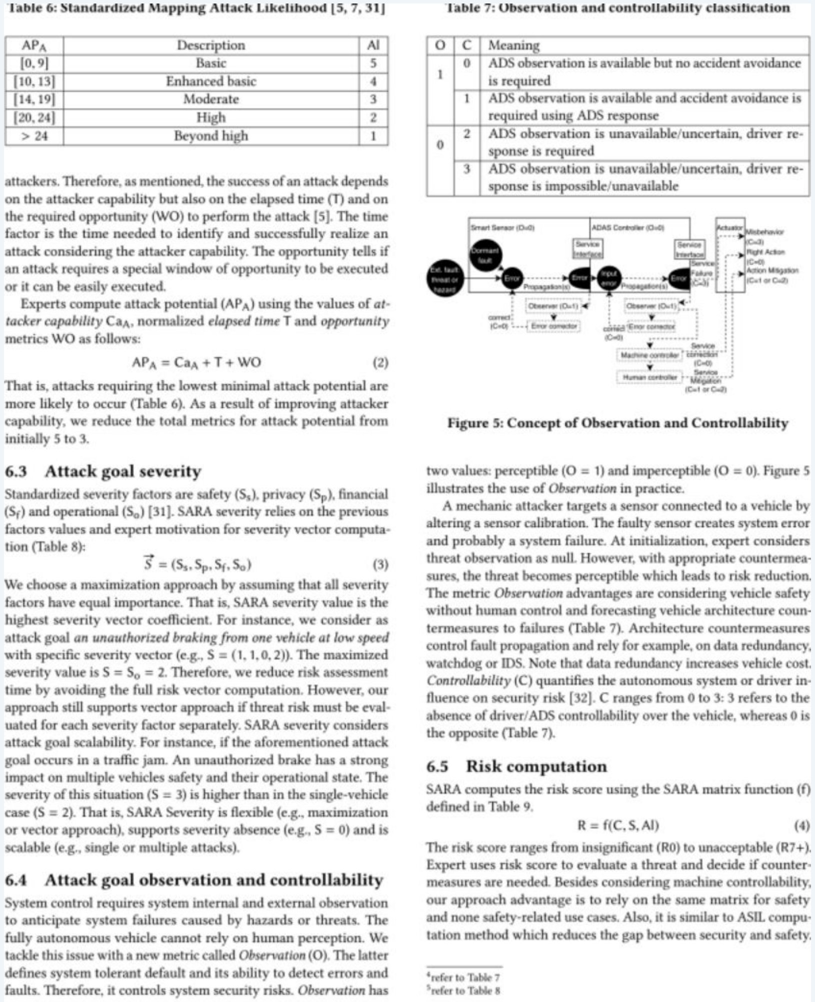


Table 6: Standardized Mapping Attack Likelihood [5, 7, 31]

AP_A	Description	AI
[0, 9]	Basic	5
[10, 13]	Enhanced basic	4
[14, 19]	Moderate	3
[20, 24]	High	2
> 24	Beyond high	1

attackers. Therefore, as mentioned, the success of an attack depends on the attacker capability but also on the elapsed time (T) and on the required opportunity (WO) to perform the attack [5]. The time factor is the time needed to identify and successfully realize an attack considering the attacker capability. The opportunity tells if an attack requires a special window of opportunity to be executed or it can be easily executed.

Experts compute attack potential (AP_A) using the values of *attacker capability* Ca_A, normalized *elapsed time* T and *opportunity* metrics WO as follows:

$$AP_A = Ca_A + T + WO \quad (2)$$

That is, attacks requiring the lowest minimal attack potential are more likely to occur (Table 6). As a result of improving attacker capability, we reduce the total metrics for attack potential from initially 5 to 3.

6.3 Attack goal severity

Standardized severity factors are safety (S_s), privacy (S_p), financial (S_f) and operational (S_o) [31]. SARA severity relies on the previous factors values and expert motivation for severity vector computation (Table 8):

$$\vec{S} = (S_s, S_p, S_f, S_o) \quad (3)$$

We choose a maximization approach by assuming that all severity factors have equal importance. That is, SARA severity value is the highest severity vector coefficient. For instance, we consider as attack goal *an unauthorized braking from one vehicle at low speed* with specific severity vector (e.g., $S = (1, 1, 0, 2)$). The maximized severity value is $S = S_o = 2$. Therefore, we reduce risk assessment time by avoiding the full risk vector computation. However, our approach still supports vector approach if threat risk must be evaluated for each severity factor separately. SARA severity considers attack goal scalability. For instance, if the aforementioned attack goal occurs in a traffic jam. An unauthorized brake has a strong impact on multiple vehicles safety and their operational state. The severity of this situation ($S = 3$) is higher than in the single-vehicle case ($S = 2$). That is, SARA Severity is flexible (e.g., maximization or vector approach), supports severity absence (e.g., $S = 0$) and is scalable (e.g., single or multiple attacks).

6.4 Attack goal observation and controllability

System control requires system internal and external observation to anticipate system failures caused by hazards or threats. The fully autonomous vehicle cannot rely on human perception. We tackle this issue with a new metric called *Observation* (O). The latter defines system tolerant default and its ability to detect errors and faults. Therefore, it controls system security risks. *Observation* has two values: perceptible ($O = 1$) and imperceptible ($O = 0$). Figure 5 illustrates the use of *Observation* in practice.

Table 7: Observation and controllability classification

O	C	Meaning
1	0	ADS observation is available but no accident avoidance is required
	1	ADS observation is available and accident avoidance is required using ADS response
0	2	ADS observation is unavailable/uncertain, driver response is required
	3	ADS observation is unavailable/uncertain, driver response is impossible/unavailable

Figure 5: Concept of Observation and Controllability

A mechanic attacker targets a sensor connected to a vehicle by altering a sensor calibration. The faulty sensor creates system error and probably a system failure. At initialization, expert considers threat observation as null. However, with appropriate countermeasures, the threat becomes perceptible which leads to risk reduction. The metric *Observation* advantages are considering vehicle safety without human control and forecasting vehicle architecture countermeasures to failures (Table 7). Architecture countermeasures control fault propagation and rely for example, on data redundancy, watchdog or IDS. Note that data redundancy increases vehicle cost. *Controllability* (C) quantifies the autonomous system or driver influence on security risk [32]. C ranges from 0 to 3: 3 refers to the absence of driver/ADS controllability over the vehicle, whereas 0 is the opposite (Table 7).

6.5 Risk computation

SARA computes the risk score using the SARA matrix function (f) defined in Table 9.

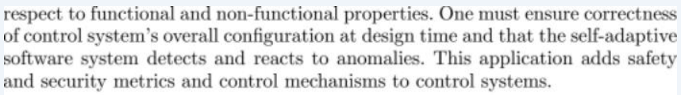
$$R = f(C, S, AI) \quad (4)$$

The risk score ranges from insignificant (R0) to unacceptable (R7+). Expert uses risk score to evaluate a threat and decide if countermeasures are needed. Besides considering machine controllability, our approach advantage is to rely on the same matrix for safety and none safety-related use cases. Also, it is similar to ASIL computation method which reduces the gap between security and safety.

[4] refer to Table 7
[5] refer to Table 8

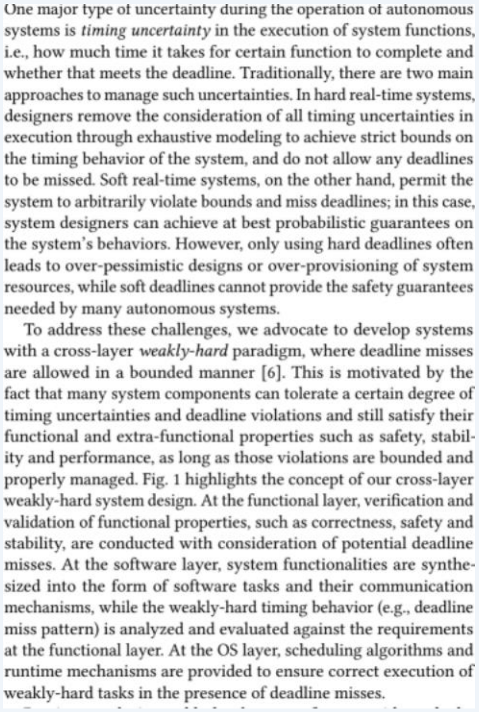
SARA Security Automotive Risk Analysis Method: 7 - 7 (0)

Modeling Approach\Adaptation
Property Checked\Safety



respect to functional and non-functional properties. One must ensure correctness of control system's overall configuration at design time and that the self-adaptive software system detects and reacts to anomalies. This application adds safety and security metrics and control mechanisms to control systems.

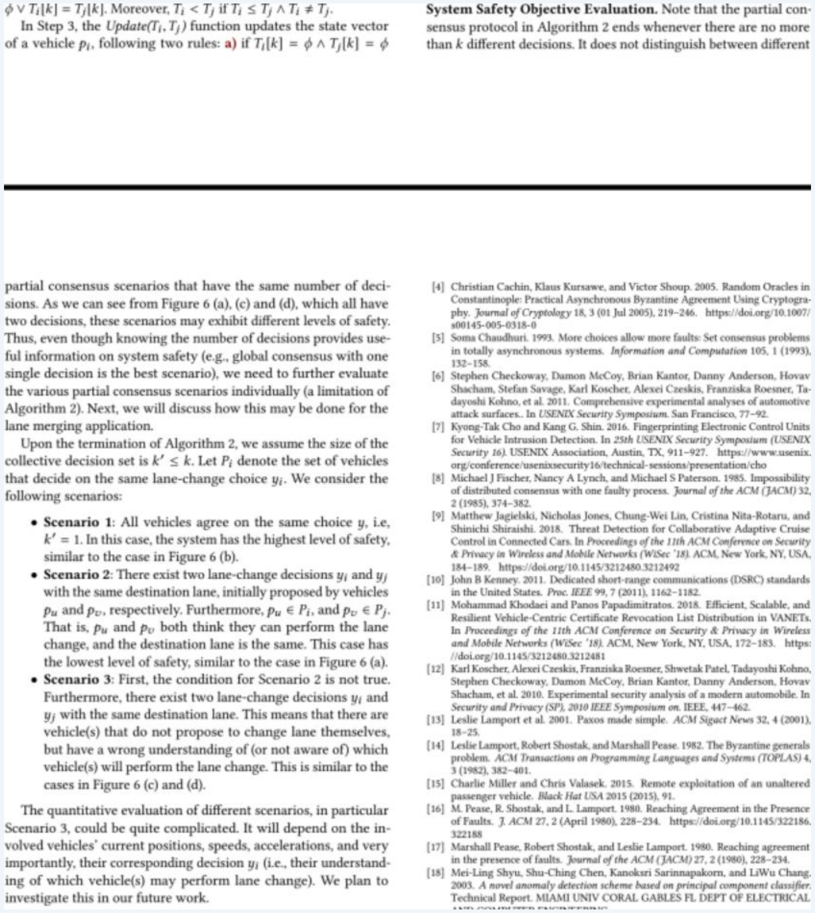
Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



One major type of uncertainty during the operation of autonomous systems is *timing uncertainty* in the execution of system functions, i.e., how much time it takes for certain function to complete and whether that meets the deadline. Traditionally, there are two main approaches to manage such uncertainties. In hard real-time systems, designers remove the consideration of all timing uncertainties in execution through exhaustive modeling to achieve strict bounds on the timing behavior of the system, and do not allow any deadlines to be missed. Soft real-time systems, on the other hand, permit the system to arbitrarily violate bounds and miss deadlines; in this case, system designers can achieve at best probabilistic guarantees on the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems.

To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses.

Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)



$\phi \vee T_i[k] = T_j[k]$. Moreover, $T_i < T_j$ if $T_i \leq T_j \wedge T_i \neq T_j$.

In Step 3, the *Update*(T_i, T_j) function updates the state vector of a vehicle p_i, following two rules: **a)** if $T_i[k] = \phi \wedge T_j[k] = \phi$

System Safety Objective Evaluation. Note that the partial consensus protocol in Algorithm 2 ends whenever there are no more than k different decisions. It does not distinguish between different

partial consensus scenarios that have the same number of decisions. As we can see from Figure 6 (a), (c) and (d), which all have two decisions, these scenarios may exhibit different levels of safety. Thus, even though knowing the number of decisions provides useful information on system safety (e.g., global consensus with one single decision is the best scenario), we need to further evaluate the various partial consensus scenarios individually (a limitation of Algorithm 2). Next, we will discuss how this may be done for the lane merging application.

Upon the termination of Algorithm 2, we assume the size of the collective decision set is $k' \leq k$. Let P_i denote the set of vehicles that decide on the same lane-change choice y_i. We consider the following scenarios:

- **Scenario 1**: All vehicles agree on the same choice y, i.e, $k' = 1$. In this case, the system has the highest level of safety, similar to the case in Figure 6 (b).
- **Scenario 2**: There exist two lane-change decisions y_i and y_j with the same destination lane, initially proposed by vehicles p_u and p_v, respectively. Furthermore, $p_u \in P_i$, and $p_v \in P_j$. That is, p_u and p_v both think they can perform the lane change, and the destination lane is the same. This case has the lowest level of safety, similar to the case in Figure 6 (a).
- **Scenario 3**: First, the condition for Scenario 2 is not true. Furthermore, there exist two lane-change decisions y_i and y_j with the same destination lane. This means that there are vehicle(s) that do not propose to change lane themselves, but have a wrong understanding of (or not aware of) which vehicle(s) will perform the lane change. This is similar to the cases in Figure 6 (c) and (d).

The quantitative evaluation of different scenarios, in particular Scenario 3, could be quite complicated. It will depend on the involved vehicles' current positions, speeds, accelerations, and very importantly, their corresponding decision y_i (i.e., their understanding of which vehicle(s) may perform lane change). We plan to investigate this in our future work.

[4] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography. *Journal of Cryptology* 18, 3 (01 Jul 2005), 219–246. https://doi.org/10.1007/s00145-005-0318-0

[5] Soma Chaudhuri. 1993. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation* 105, 1 (1993), 132–158.

[6] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. 2011. Comprehensive experimental analyses of automotive attack surfaces.. In *USENIX Security Symposium*. San Francisco, 77–92.

[7] Kyong-Tak Cho and Kang G. Shin. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 911–927. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho

[8] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. 1985. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)* 32, 2 (1985), 374–382.

[9] Matthew Jagielski, Nicholas Jones, Chung-Wei Lin, Cristina Nita-Rotaru, and Shinichi Shiraishi. 2018. Threat Detection for Collaborative Adaptive Cruise Control in Connected Cars. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '18)*. ACM, New York, NY, USA, 184–189. https://doi.org/10.1145/3212480.3212492

[10] John B Kenney. 2011. Dedicated short-range communications (DSRC) standards in the United States. *Proc. IEEE* 99, 7 (2011), 1162–1182.

[11] Mohammad Khodaei and Panos Papadimitratos. 2018. Efficient, Scalable, and Resilient Vehicle-Centric Certificate Revocation List Distribution in VANETs. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '18)*. ACM, New York, NY, USA, 172–183. https://doi.org/10.1145/3212480.3212481

[12] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. 2010. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 447–462.

[13] Leslie Lamport et al. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.

[14] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4, 3 (1982), 382–401.

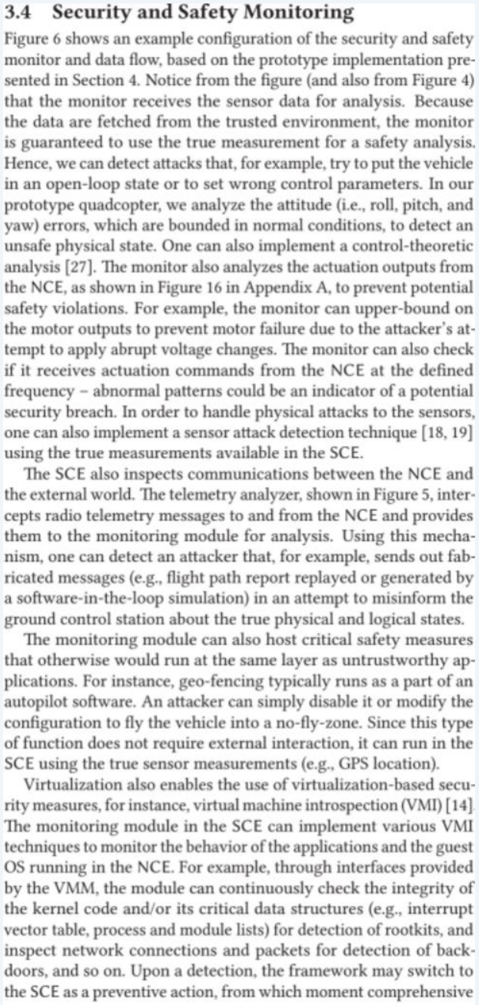
[15] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* 2015 (2015), 91.

[16] M. Pease, R. Shostak, and L. Lamport. 1980. Reaching Agreement in the Presence of Faults. *J. ACM* 27, 2 (April 1980), 228–234. https://doi.org/10.1145/322186.322188

[17] Marshall Pease, Robert Shostak, and Leslie Lamport. 1980. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)* 27, 2 (1980), 228–234.

[18] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. 2003. *A novel anomaly detection scheme based on principal component classifier.* Technical Report. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL

Network and system level security in connected vehicle applicat: 6 - 7 (0)



3.4 Security and Safety Monitoring

Figure 6 shows an example configuration of the security and safety monitor and data flow, based on the prototype implementation presented in Section 4. Notice from the figure (and also from Figure 4) that the monitor receives the sensor data for analysis. Because the data are fetched from the trusted environment, the monitor is guaranteed to use the true measurement for a safety analysis. Hence, we can detect attacks that, for example, try to put the vehicle in an open-loop state or to set wrong control parameters. In our prototype quadcopter, we analyze the attitude (i.e., roll, pitch, and yaw) errors, which are bounded in normal conditions, to detect an unsafe physical state. One can also implement a control-theoretic analysis [27]. The monitor also analyzes the actuation outputs from the NCE, as shown in Figure 16 in Appendix A, to prevent potential safety violations. For example, the monitor can upper-bound on the motor outputs to prevent motor failure due to the attacker's attempt to apply abrupt voltage changes. The monitor can also check if it receives actuation commands from the NCE at the defined frequency – abnormal patterns could be an indicator of a potential security breach. In order to handle physical attacks to the sensors, one can also implement a sensor attack detection technique [18, 19] using the true measurements available in the SCE.

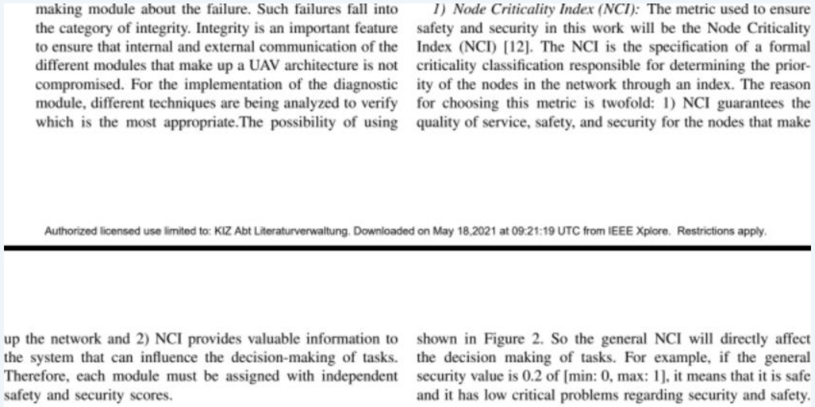
The SCE also inspects communications between the NCE and the external world. The telemetry analyzer, shown in Figure 5, intercepts radio telemetry messages to and from the NCE and provides them to the monitoring module for analysis. Using this mechanism, one can detect an attacker that, for example, sends out fabricated messages (e.g., flight path report replayed or generated by a software-in-the-loop simulation) in an attempt to misinform the ground control station about the true physical and logical states.

The monitoring module can also host critical safety measures that otherwise would run at the same layer as untrustworthy applications. For instance, geo-fencing typically runs as a part of an autopilot software. An attacker can simply disable it or modify the configuration to fly the vehicle into a no-fly-zone. Since this type of function does not require external interaction, it can run in the SCE using the true sensor measurements (e.g., GPS location).

Virtualization also enables the use of virtualization-based security measures, for instance, virtual machine introspection (VMI) [14]. The monitoring module in the SCE can implement various VMI techniques to monitor the behavior of the applications and the guest OS running in the NCE. For example, through interfaces provided by the VMM, the module can continuously check the integrity of the kernel code and/or its critical data structures (e.g., interrupt vector table, process and module lists) for detection of rootkits, and inspect network connections and packets for detection of backdoors, and so on. Upon a detection, the framework may switch to the SCE as a preventive action, from which moment comprehensive

VirtualDrone virtual sensing actuation and communication for at: 5 - 5 (0)

Modeling Approach\Adaptation Property Checked\Node Criticality Index



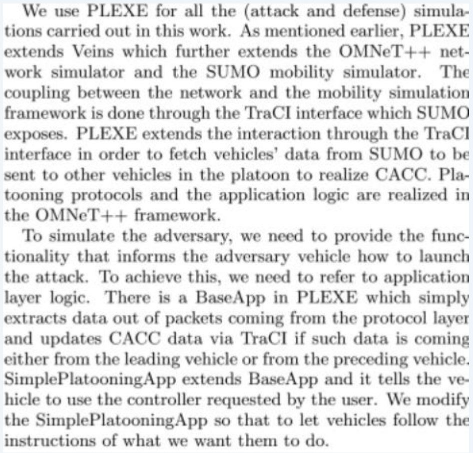
making module about the failure. Such failures fall into the category of integrity. Integrity is an important feature to ensure that internal and external communication of the different modules that make up a UAV architecture is not compromised. For the implementation of the diagnostic module, different techniques are being analyzed to verify which is the most appropriate.The possibility of using

1) Node Criticality Index (NCI): The metric used to ensure safety and security in this work will be the Node Criticality Index (NCI) [12]. The NCI is the specification of a formal criticality classification responsible for determining the priority of the nodes in the network through an index. The reason for choosing this metric is twofold: 1) NCI guarantees the quality of service, safety, and security for the nodes that make up the network and 2) NCI provides valuable information to the system that can influence the decision-making of tasks. Therefore, each module must be assigned with independent safety and security scores.

shown in Figure 2. So the general NCI will directly affect the decision making of tasks. For example, if the general security value is 0.2 of [min: 0, max: 1], it means that it is safe and it has low critical problems regarding security and safety.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 3 - 4 (0)

Modeling Approach\Verification and Validation Technique\Testing



We use PLEXE for all the (attack and defense) simulations carried out in this work. As mentioned earlier, PLEXE extends Veins which further extends the OMNeT++ network simulator and the SUMO mobility simulator. The coupling between the network and the mobility simulation framework is done through the TraCI interface which SUMO exposes. PLEXE extends the interaction through the TraCI interface in order to fetch vehicles' data from SUMO to be sent to other vehicles in the platoon to realize CACC. Platooning protocols and the application logic are realized in the OMNeT++ framework.

To simulate the adversary, we need to provide the functionality that informs the adversary vehicle how to launch the attack. To achieve this, we need to refer to application layer logic. There is a BaseApp in PLEXE which simply extracts data out of packets coming from the protocol layer and updates CACC data via TraCI if such data is coming either from the leading vehicle or from the preceding vehicle. SimplePlatooningApp extends BaseApp and it tells the vehicle to use the controller requested by the user. We modify the SimplePlatooningApp so that to let vehicles follow the instructions of what we want them to do.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)

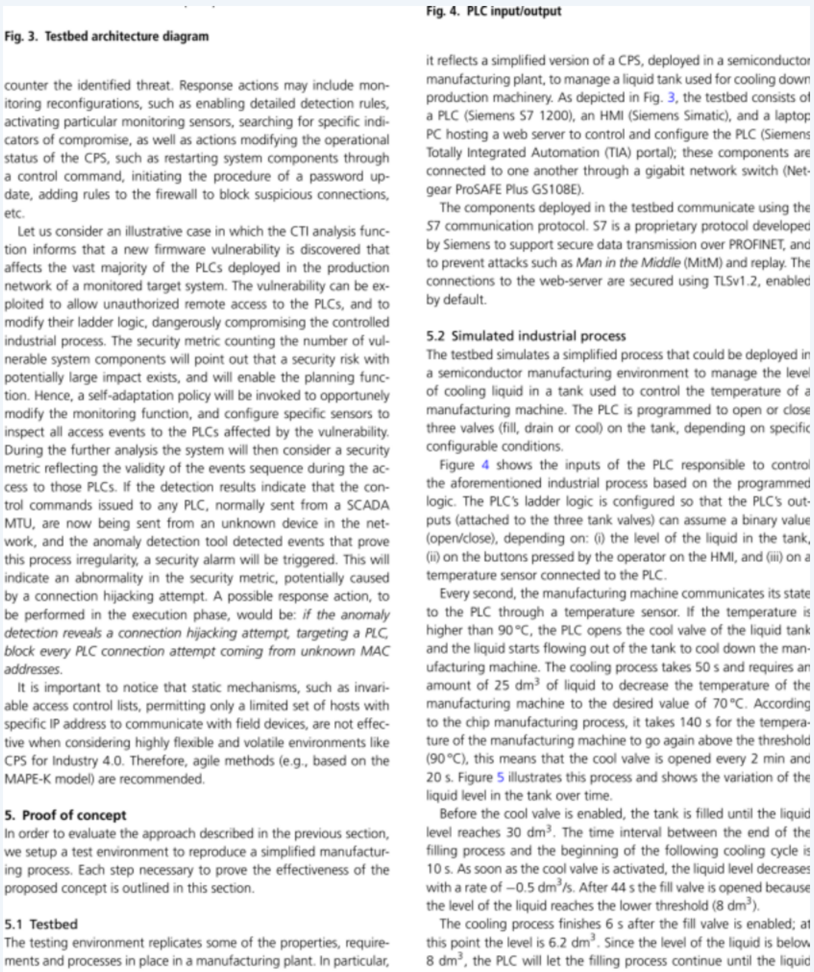


Fig. 3. Testbed architecture diagram

counter the identified threat. Response actions may include monitoring reconfigurations, such as enabling detailed detection rules, activating particular monitoring sensors, searching for specific indicators of compromise, as well as actions modifying the operational status of the CPS, such as restarting system components through a control command, initiating the procedure of a password update, adding rules to the firewall to block suspicious connections, etc.

Let us consider an illustrative case in which the CTI analysis function informs that a new firmware vulnerability is discovered that affects the vast majority of the PLCs deployed in the production network of a monitored target system. The vulnerability can be exploited to allow unauthorized remote access to the PLCs, and to modify their ladder logic, dangerously compromising the controlled industrial process. The security metric counting the number of vulnerable system components will point out that a security risk with potentially large impact exists, and will enable the planning function. Hence, a self-adaptation policy will be invoked to opportunely modify the monitoring function, and configure specific sensors to inspect all access events to the PLCs affected by the vulnerability. During the further analysis the system will then consider a security metric reflecting the validity of the events sequence during the access to those PLCs. If the detection results indicate that the control commands issued to any PLC, normally sent from a SCADA MTU, are now being sent from an unknown device in the network, and the anomaly detection tool detected events that prove this process irregularity, a security alarm will be triggered. This will indicate an abnormality in the security metric, potentially caused by a connection hijacking attempt. A possible response action, to be performed in the execution phase, would be: *if the anomaly detection reveals a connection hijacking attempt, targeting a PLC, block every PLC connection attempt coming from unknown MAC addresses.*

It is important to notice that static mechanisms, such as invariable access control lists, permitting only a limited set of hosts with specific IP address to communicate with field devices, are not effective when considering highly flexible and volatile environments like CPS for Industry 4.0. Therefore, agile methods (e.g., based on the MAPE-K model) are recommended.

5. Proof of concept

In order to evaluate the approach described in the previous section, we setup a test environment to reproduce a simplified manufacturing process. Each step necessary to prove the effectiveness of the proposed concept is outlined in this section.

5.1 Testbed

The testing environment replicates some of the properties, requirements and processes in place in a manufacturing plant. In particular,

Fig. 4. PLC input/output

it reflects a simplified version of a CPS, deployed in a semiconductor manufacturing plant, to manage a liquid tank used for cooling down production machinery. As depicted in Fig. 3, the testbed consists of a PLC (Siemens S7 1200), an HMI (Siemens Simatic), and a laptop PC hosting a web server to control and configure the PLC (Siemens Totally Integrated Automation (TIA) portal); these components are connected to one another through a gigabit network switch (Netgear ProSAFE Plus GS108E).

The components deployed in the testbed communicate using the *S7* communication protocol. S7 is a proprietary protocol developed by Siemens to support secure data transmission over PROFINET, and to prevent attacks such as *Man in the Middle* (MitM) and replay. The connections to the web-server are secured using TLSv1.2, enabled by default.

5.2 Simulated industrial process

The testbed simulates a simplified process that could be deployed in a semiconductor manufacturing environment to manage the level of cooling liquid in a tank used to control the temperature of a manufacturing machine. The PLC is programmed to open or close three valves (fill, drain or cool) on the tank, depending on specific configurable conditions.

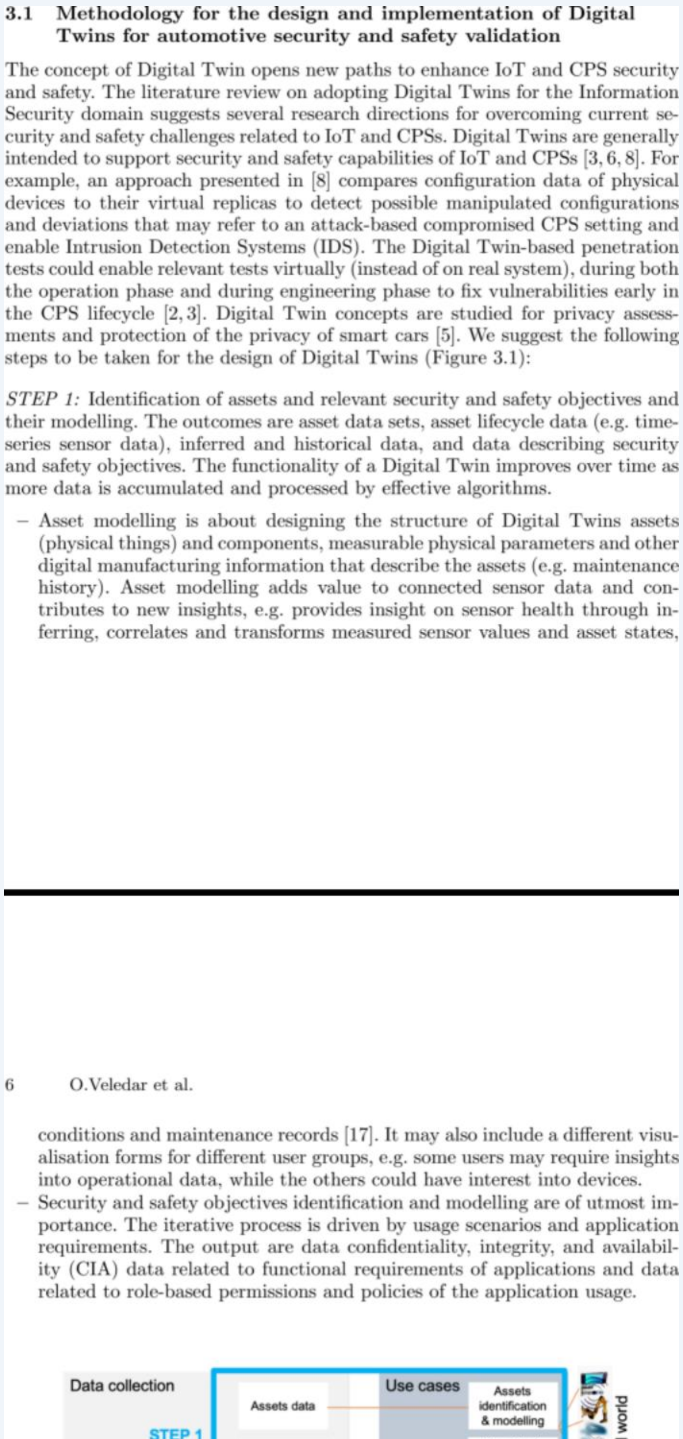
Figure 4 shows the inputs of the PLC responsible to control the aforementioned industrial process based on the programmed logic. The PLC's ladder logic is configured so that the PLC's outputs (attached to the three tank valves) can assume a binary value (open/close), depending on: (i) the level of the liquid in the tank, (ii) on the buttons pressed by the operator on the HMI, and (iii) on a temperature sensor connected to the PLC.

Every second, the manufacturing machine communicates its state to the PLC through a temperature sensor. If the temperature is higher than 90 °C, the PLC opens the cool valve of the liquid tank and the liquid starts flowing out of the tank to cool down the manufacturing machine. The cooling process takes 50 s and requires an amount of 25 dm^3 of liquid to decrease the temperature of the manufacturing machine to the desired value of 70 °C. According to the chip manufacturing process, it takes 140 s for the temperature of the manufacturing machine to go again above the threshold (90 °C), this means that the cool valve is opened every 2 min and 20 s. Figure 5 illustrates this process and shows the variation of the liquid level in the tank over time.

Before the cool valve is enabled, the tank is filled until the liquid level reaches 30 dm^3. The time interval between the end of the filling process and the beginning of the following cooling cycle is 10 s. As soon as the cool valve is activated, the liquid level decreases with a rate of −0.5 dm^3/s. After 44 s the fill valve is opened because the level of the liquid reaches the lower threshold (8 dm^3).

The cooling process finishes 6 s after the fill valve is enabled; at this point the level is 6.2 dm^3. Since the level of the liquid is below 8 dm^3, the PLC will let the filling process continue until the liquid

Countering targeted cyber-physical attacks using anomaly detect: 4 - 4 (0)



3.1 Methodology for the design and implementation of Digital Twins for automotive security and safety validation

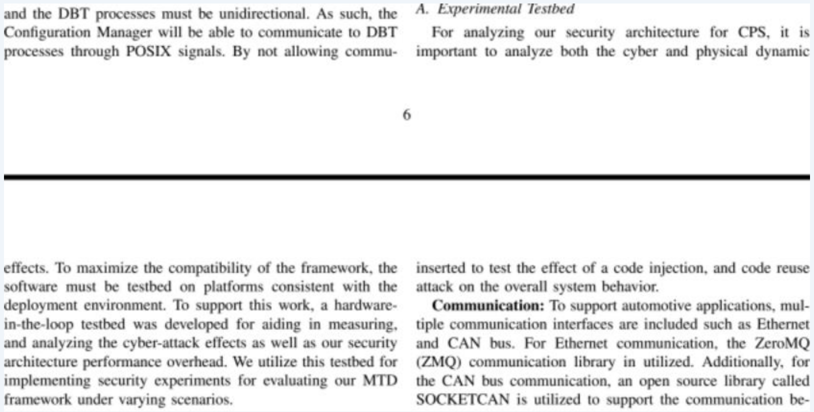
The concept of Digital Twin opens new paths to enhance IoT and CPS security and safety. The literature review on adopting Digital Twins for the Information Security domain suggests several research directions for overcoming current security and safety challenges related to IoT and CPSs. Digital Twins are generally intended to support security and safety capabilities of IoT and CPSs [3, 6, 8]. For example, an approach presented in [8] compares configuration data of physical devices to their virtual replicas to detect possible manipulated configurations and deviations that may refer to an attack-based compromised CPS setting and enable Intrusion Detection Systems (IDS). The Digital Twin-based penetration tests could enable relevant tests virtually (instead of on real system), during both the operation phase and during engineering phase to fix vulnerabilities early in the CPS lifecycle [2, 3]. Digital Twin concepts are studied for privacy assessments and protection of the privacy of smart cars [5]. We suggest the following steps to be taken for the design of Digital Twins (Figure 3.1):

STEP 1: Identification of assets and relevant security and safety objectives and their modelling. The outcomes are asset data sets, asset lifecycle data (e.g. time-series sensor data), inferred and historical data, and data describing security and safety objectives. The functionality of a Digital Twin improves over time as more data is accumulated and processed by effective algorithms.

- Asset modelling is about designing the structure of Digital Twins assets (physical things) and components, measurable physical parameters and other digital manufacturing information that describe the assets (e.g. maintenance history). Asset modelling adds value to connected sensor data and contributes to new insights, e.g. provides insight on sensor health through inferring, correlates and transforms measured sensor values and asset states,

conditions and maintenance records [17]. It may also include a different visualisation forms for different user groups, e.g. some users may require insights into operational data, while the others could have interest into devices.
- Security and safety objectives identification and modelling are of utmost importance. The iterative process is driven by usage scenarios and application requirements. The output are data confidentiality, integrity, and availability (CIA) data related to functional requirements of applications and data related to role-based permissions and policies of the application usage.

Digital Twins for Dependability Improvement of Autonomous Drivi: 6 - 8 (0)



and the DBT processes must be unidirectional. As such, the Configuration Manager will be able to communicate to DBT processes through POSIX signals. By not allowing commu-

A. Experimental Testbed

For analyzing our security architecture for CPS, it is important to analyze both the cyber and physical dynamic

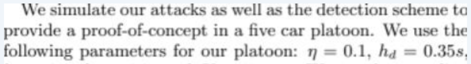
6

effects. To maximize the compatibility of the framework, the software must be testbed on platforms consistent with the deployment environment. To support this work, a hardware-in-the-loop testbed was developed for aiding in measuring, and analyzing the cyber-attack effects as well as our security architecture performance overhead. We utilize this testbed for implementing security experiments for evaluating our MTD framework under varying scenarios.

inserted to test the effect of a code injection, and code reuse attack on the overall system behavior.

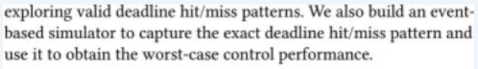
Communication: To support automotive applications, multiple communication interfaces are included such as Ethernet and CAN bus. For Ethernet communication, the ZeroMQ (ZMQ) communication library in utilized. Additionally, for the CAN bus communication, an open source library called SOCKETCAN is utilized to support the communication be-

Integrated moving target defense and control reconfiguration fo: 6 - 7 (0)



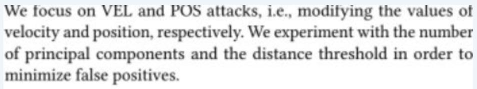
We simulate our attacks as well as the detection scheme to provide a proof-of-concept in a five car platoon. We use the following parameters for our platoon: $\eta = 0.1$, $h_d = 0.35s$,

Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



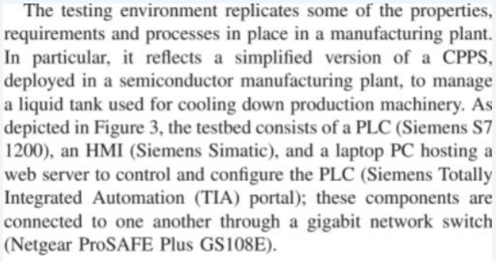
exploring valid deadline hit/miss patterns. We also build an event-based simulator to capture the exact deadline hit/miss pattern and use it to obtain the worst-case control performance.

Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)



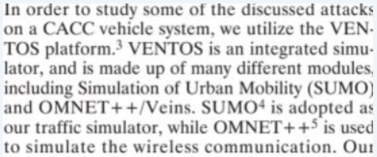
We focus on VEL and POS attacks, i.e., modifying the values of velocity and position, respectively. We experiment with the number of principal components and the distance threshold in order to minimize false positives.

Network and system level security in connected vehicle applicat: 4 - 4 (0)



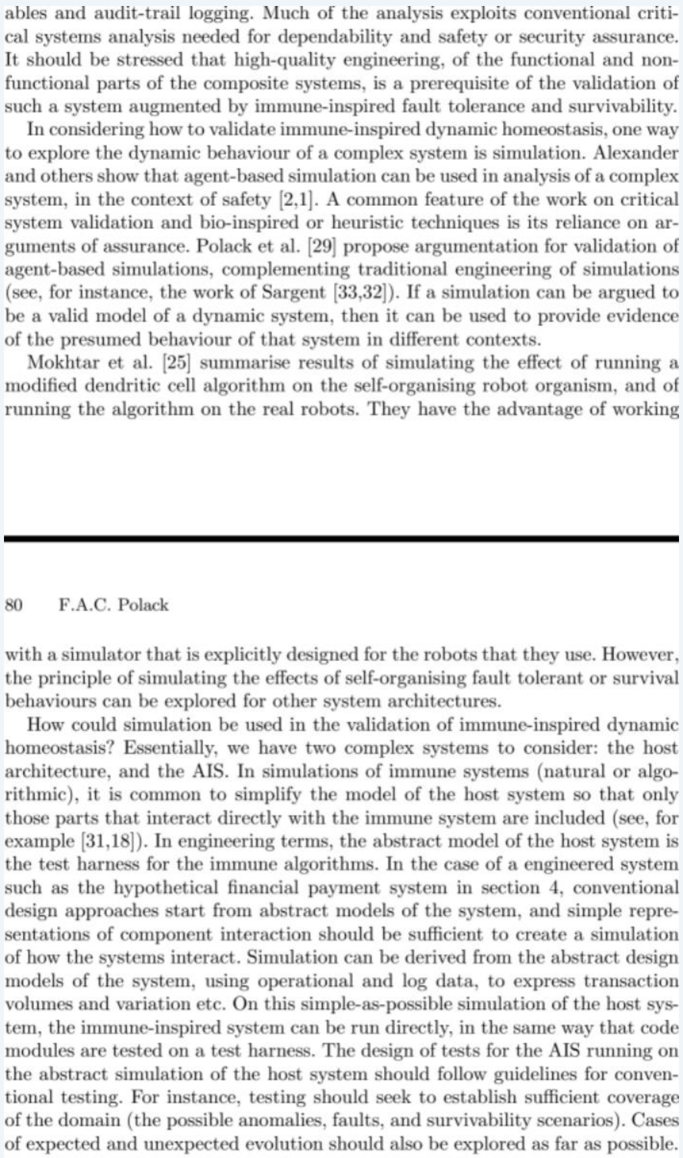
The testing environment replicates some of the properties, requirements and processes in place in a manufacturing plant. In particular, it reflects a simplified version of a CPPS, deployed in a semiconductor manufacturing plant, to manage a liquid tank used for cooling down production machinery. As depicted in Figure 3, the testbed consists of a PLC (Siemens S7 1200), an HMI (Siemens Simatic), and a laptop PC hosting a web server to control and configure the PLC (Siemens Totally Integrated Automation (TIA) portal); these components are connected to one another through a gigabit network switch (Netgear ProSAFE Plus GS108E).

Protecting cyber physical production systems using anomaly dete: 4 - 4 (0)



In order to study some of the discussed attacks on a CACC vehicle system, we utilize the VENTOS platform.[3] VENTOS is an integrated simulator, and is made up of many different modules, including Simulation of Urban Mobility (SUMO) and OMNET++/Veins. SUMO[4] is adopted as our traffic simulator, while OMNET++[5] is used to simulate the wireless communication. Our

Security vulnerabilities of connected vehicle streams and their: 4 - 4 (0)



ables and audit-trail logging. Much of the analysis exploits conventional critical systems analysis needed for dependability and safety or security assurance. It should be stressed that high-quality engineering, of the functional and non-functional parts of the composite systems, is a prerequisite of the validation of such a system augmented by immune-inspired fault tolerance and survivability.

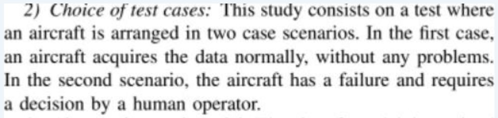
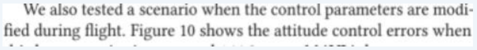
In considering how to validate immune-inspired dynamic homeostasis, one way to explore the dynamic behaviour of a complex system is simulation. Alexander and others show that agent-based simulation can be used in analysis of a complex system, in the context of safety [2,1]. A common feature of the work on critical system validation and bio-inspired or heuristic techniques is its reliance on arguments of assurance. Polack et al. [29] propose argumentation for validation of agent-based simulations, complementing traditional engineering of simulations (see, for instance, the work of Sargent [33,32]). If a simulation can be argued to be a valid model of a dynamic system, then it can be used to provide evidence of the presumed behaviour of that system in different contexts.

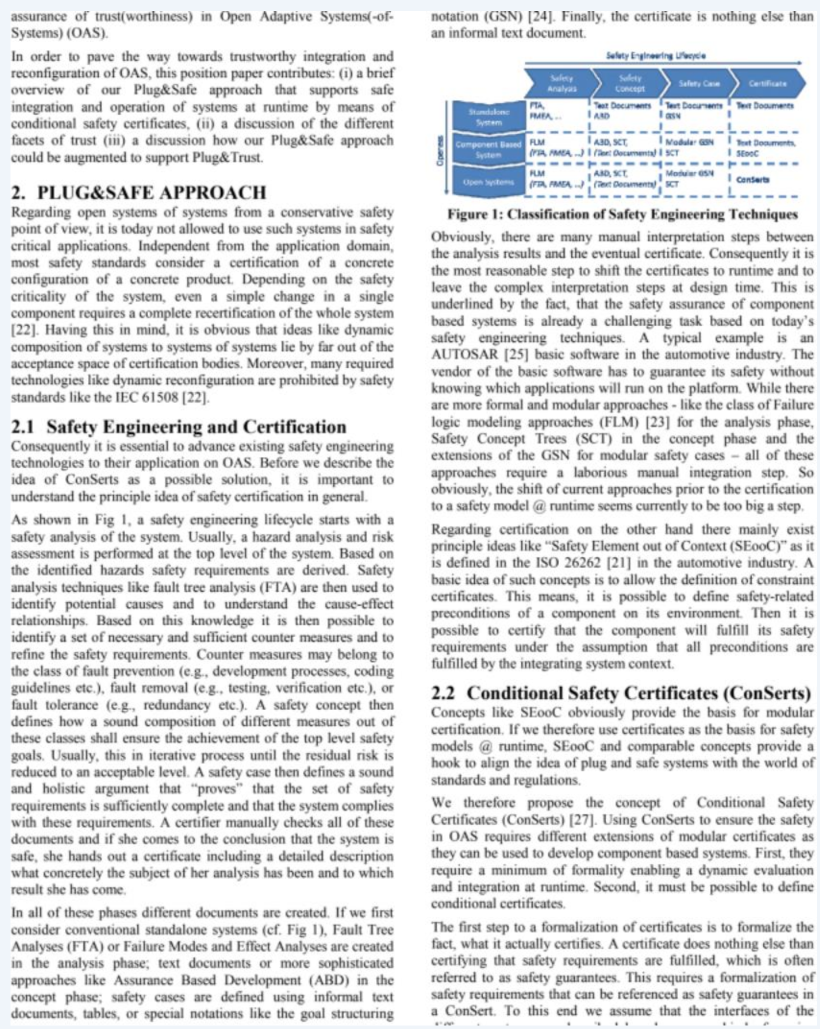
Mokhtar et al. [25] summarise results of simulating the effect of running a modified dendritic cell algorithm on the self-organising robot organism, and of running the algorithm on the real robots. They have the advantage of working

with a simulator that is explicitly designed for the robots that they use. However, the principle of simulating the effects of self-organising fault tolerant or survival behaviours can be explored for other system architectures.

How could simulation be used in the validation of immune-inspired dynamic homeostasis? Essentially, we have two complex systems to consider: the host architecture, and the AIS. In simulations of immune systems (natural or algorithmic), it is common to simplify the model of the host system so that only those parts that interact directly with the immune system are included (see, for example [31,18]). In engineering terms, the abstract model of the host system is the test harness for the immune algorithms. In the case of a engineered system such as the hypothetical financial payment system in section 4, conventional design approaches start from abstract models of the system, and simple representations of component interaction should be sufficient to create a simulation of how the systems interact. Simulation can be derived from the abstract design models of the system, using operational and log data, to express transaction volumes and variation etc. On this simple-as-possible simulation of the host system, the immune-inspired system can be run directly, in the same way that code modules are tested on a test harness. The design of tests for the AIS running on the abstract simulation of the host system should follow guidelines for conventional testing. For instance, testing should seek to establish sufficient coverage of the domain (the possible anomalies, faults, and survivability scenarios). Cases of expected and unexpected evolution should also be explored as far as possible.

Self-organisation for Survival in Complex Computer Architecture: 14 - 15 (0)

	 *2) Choice of test cases:* This study consists on a test where an aircraft is arranged in two case scenarios. In the first case, an aircraft acquires the data normally, without any problems. In the second scenario, the aircraft has a failure and requires a decision by a human operator. STUART ReSilient archiTecture to dynamically manage Unmanned ae: 5 - 5 (0)
	 We also tested a scenario when the control parameters are modified during flight. Figure 10 shows the attitude control errors when VirtualDrone virtual sensing actuation and communication for at: 7 - 7 (0)
Modeling Approach\Verification and Validation Technique\Formal Verification	



assurance of trust(worthiness) in Open Adaptive Systems(-of-Systems) (OAS).

In order to pave the way towards trustworthy integration and reconfiguration of OAS, this position paper contributes: (i) a brief overview of our Plug&Safe approach that supports safe integration and operation of systems at runtime by means of conditional safety certificates, (ii) a discussion of the different facets of trust (iii) a discussion how our Plug&Safe approach could be augmented to support Plug&Trust.

2. PLUG&SAFE APPROACH

Regarding open systems of systems from a conservative safety point of view, it is today not allowed to use such systems in safety critical applications. Independent from the application domain, most safety standards consider a certification of a concrete configuration of a concrete product. Depending on the safety criticality of the system, even a simple change in a single component requires a complete recertification of the whole system [22]. Having this in mind, it is obvious that ideas like dynamic composition of systems to systems of systems lie by far out of the acceptance space of certification bodies. Moreover, many required technologies like dynamic reconfiguration are prohibited by safety standards like the IEC 61508 [22].

2.1 Safety Engineering and Certification

Consequently it is essential to advance existing safety engineering technologies to their application on OAS. Before we describe the idea of ConSerts as a possible solution, it is important to understand the principle idea of safety certification in general.

As shown in Fig 1, a safety engineering lifecycle starts with a safety analysis of the system. Usually, a hazard analysis and risk assessment is performed at the top level of the system. Based on the identified hazards safety requirements are derived. Safety analysis techniques like fault tree analysis (FTA) are then used to identify potential causes and to understand the cause-effect relationships. Based on this knowledge it is then possible to identify a set of necessary and sufficient counter measures and to refine the safety requirements. Counter measures may belong to the class of fault prevention (e.g., development processes, coding guidelines etc.), fault removal (e.g., testing, verification etc.), or fault tolerance (e.g., redundancy etc.). A safety concept then defines how a sound composition of different measures out of these classes shall ensure the achievement of the top level safety goals. Usually, this in iterative process until the residual risk is reduced to an acceptable level. A safety case then defines a sound and holistic argument that "proves" that the set of safety requirements is sufficiently complete and that the system complies with these requirements. A certifier manually checks all of these documents and if she comes to the conclusion that the system is safe, she hands out a certificate including a detailed description what concretely the subject of her analysis has been and to which result she has come.

In all of these phases different documents are created. If we first consider conventional standalone systems (cf. Fig 1), Fault Tree Analyses (FTA) or Failure Modes and Effect Analyses are created in the analysis phase; text documents or more sophisticated approaches like Assurance Based Development (ABD) in the concept phase; safety cases are defined using informal text documents, tables, or special notations like the goal structuring notation (GSN) [24]. Finally, the certificate is nothing else than an informal text document.

Safety Engineering Lifecycle

Openness	Safety Analysis	Safety Concept	Safety Case	Certificate
Standalone System	FTA, FMEA, ...	Text Documents ABD	Text Documents GSN	Text Documents
Component Based System	FLM (FTA, FMEA, ...)	ABD, SCT, (Text Documents)	Modular GSN SCT	Text Documents, SEooC
Open Systems	FLM (FTA, FMEA, ...)	ABD, SCT, (Text Documents)	Modular GSN SCT	ConSerts

Figure 1: Classification of Safety Engineering Techniques

Obviously, there are many manual interpretation steps between the analysis results and the eventual certificate. Consequently it is the most reasonable step to shift the certificates to runtime and to leave the complex interpretation steps at design time. This is underlined by the fact, that the safety assurance of component based systems is already a challenging task based on today's safety engineering techniques. A typical example is an AUTOSAR [25] basic software in the automotive industry. The vendor of the basic software has to guarantee its safety without knowing which applications will run on the platform. While there are more formal and modular approaches - like the class of Failure logic modeling approaches (FLM) [23] for the analysis phase, Safety Concept Trees (SCT) in the concept phase and the extensions of the GSN for modular safety cases – all of these approaches require a laborious manual integration step. So obviously, the shift of current approaches prior to the certification to a safety model @ runtime seems currently to be too big a step.

Regarding certification on the other hand there mainly exist principle ideas like "Safety Element out of Context (SEooC)" as it is defined in the ISO 26262 [21] in the automotive industry. A basic idea of such concepts is to allow the definition of constraint certificates. This means, it is possible to define safety-related preconditions of a component on its environment. Then it is possible to certify that the component will fulfill its safety requirements under the assumption that all preconditions are fulfilled by the integrating system context.

2.2 Conditional Safety Certificates (ConSerts)

Concepts like SEooC obviously provide the basis for modular certification. If we therefore use certificates as the basis for safety models @ runtime, SEooC and comparable concepts provide a hook to align the idea of plug and safe systems with the world of standards and regulations.

We therefore propose the concept of Conditional Safety Certificates (ConSerts) [27]. Using ConSerts to ensure the safety in OAS requires different extensions of modular certificates as they can be used to develop component based systems. First, they require a minimum of formality enabling a dynamic evaluation and integration at runtime. Second, it must be possible to define conditional certificates.

The first step to a formalization of certificates is to formalize the fact, what it actually certifies. A certificate does nothing else than certifying that safety requirements are fulfilled, which is often referred to as safety guarantees. This requires a formalization of safety requirements that can be referenced as safety guarantees in a ConSert. To this end we assume that the interfaces of the

Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)

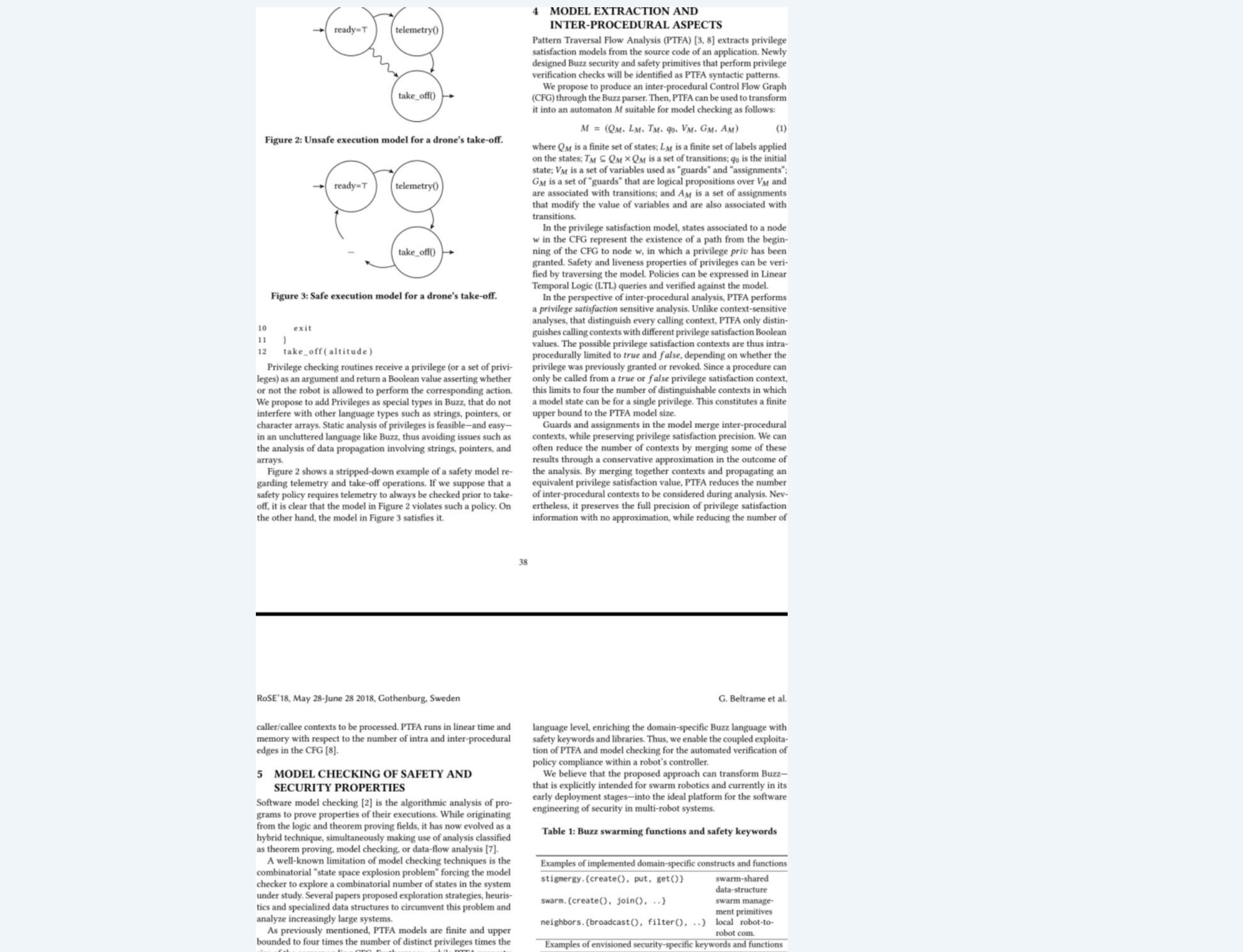


Figure 2: Unsafe execution model for a drone's take-off.

Figure 3: Safe execution model for a drone's take-off.

```
        exit
    }
    take_off(altitude)
```

Privilege checking routines receive a privilege (or a set of privileges) as an argument and return a Boolean value asserting whether or not the robot is allowed to perform the corresponding action. We propose to add Privileges as special types in Buzz, that do not interfere with other language types such as strings, pointers, or character arrays. Static analysis of privileges is feasible—and easy—in an uncluttered language like Buzz, thus avoiding issues such as the analysis of data propagation involving strings, pointers, and arrays.

Figure 2 shows a stripped-down example of a safety model regarding telemetry and take-off operations. If we suppose that a safety policy requires telemetry to always be checked prior to take-off, it is clear that the model in Figure 2 violates such a policy. On the other hand, the model in Figure 3 satisfies it.

4 MODEL EXTRACTION AND INTER-PROCEDURAL ASPECTS

Pattern Traversal Flow Analysis (PTFA) [3, 8] extracts privilege satisfaction models from the source code of an application. Newly designed Buzz security and safety primitives that perform privilege verification checks will be identified as PTFA syntactic patterns.

We propose to produce an inter-procedural Control Flow Graph (CFG) through the Buzz parser. Then, PTFA can be used to transform it into an automaton M suitable for model checking as follows:

$$M = (Q_M, L_M, T_M, q_0, V_M, G_M, A_M) \tag{1}$$

where Q_M is a finite set of states; L_M is a finite set of labels applied on the states; $T_M \subseteq Q_M \times Q_M$ is a set of transitions; q_0 is the initial state; V_M is a set of variables used as "guards" and "assignments"; G_M is a set of "guards" that are logical propositions over V_M and are associated with transitions; and A_M is a set of assignments that modify the value of variables and are also associated with transitions.

In the privilege satisfaction model, states associated to a node w in the CFG represent the existence of a path from the beginning of the CFG to node w, in which a privilege $priv$ has been granted. Safety and liveness properties of privileges can be verified by traversing the model. Policies can be expressed in Linear Temporal Logic (LTL) queries and verified against the model.

In the perspective of inter-procedural analysis, PTFA performs a *privilege satisfaction* sensitive analysis. Unlike context-sensitive analyses, that distinguish every calling context, PTFA only distinguishes calling contexts with different privilege satisfaction Boolean values. The possible privilege satisfaction contexts are thus intra-procedurally limited to $true$ and $false$, depending on whether the privilege was previously granted or revoked. Since a procedure can only be called from a $true$ or $false$ privilege satisfaction context, this limits to four the number of distinguishable contexts in which a model state can be for a single privilege. This constitutes a finite upper bound to the PTFA model size.

Guards and assignments in the model merge inter-procedural contexts, while preserving privilege satisfaction precision. We can often reduce the number of contexts by merging some of these results through a conservative approximation in the outcome of the analysis. By merging together contexts and propagating an equivalent privilege satisfaction value, PTFA reduces the number of inter-procedural contexts to be considered during analysis. Nevertheless, it preserves the full precision of privilege satisfaction information with no approximation, while reducing the number of

caller/callee contexts to be processed. PTFA runs in linear time and memory with respect to the number of intra and inter-procedural edges in the CFG [8].

5 MODEL CHECKING OF SAFETY AND SECURITY PROPERTIES

Software model checking [2] is the algorithmic analysis of programs to prove properties of their executions. While originating from the logic and theorem proving fields, it has now evolved as a hybrid technique, simultaneously making use of analysis classified as theorem proving, model checking, or data-flow analysis [7].

A well-known limitation of model checking techniques is the combinatorial "state space explosion problem" forcing the model checker to explore a combinatorial number of states in the system under study. Several papers proposed exploration strategies, heuristics and specialized data structures to circumvent this problem and analyze increasingly large systems.

As previously mentioned, PTFA models are finite and upper bounded to four times the number of distinct privileges times the

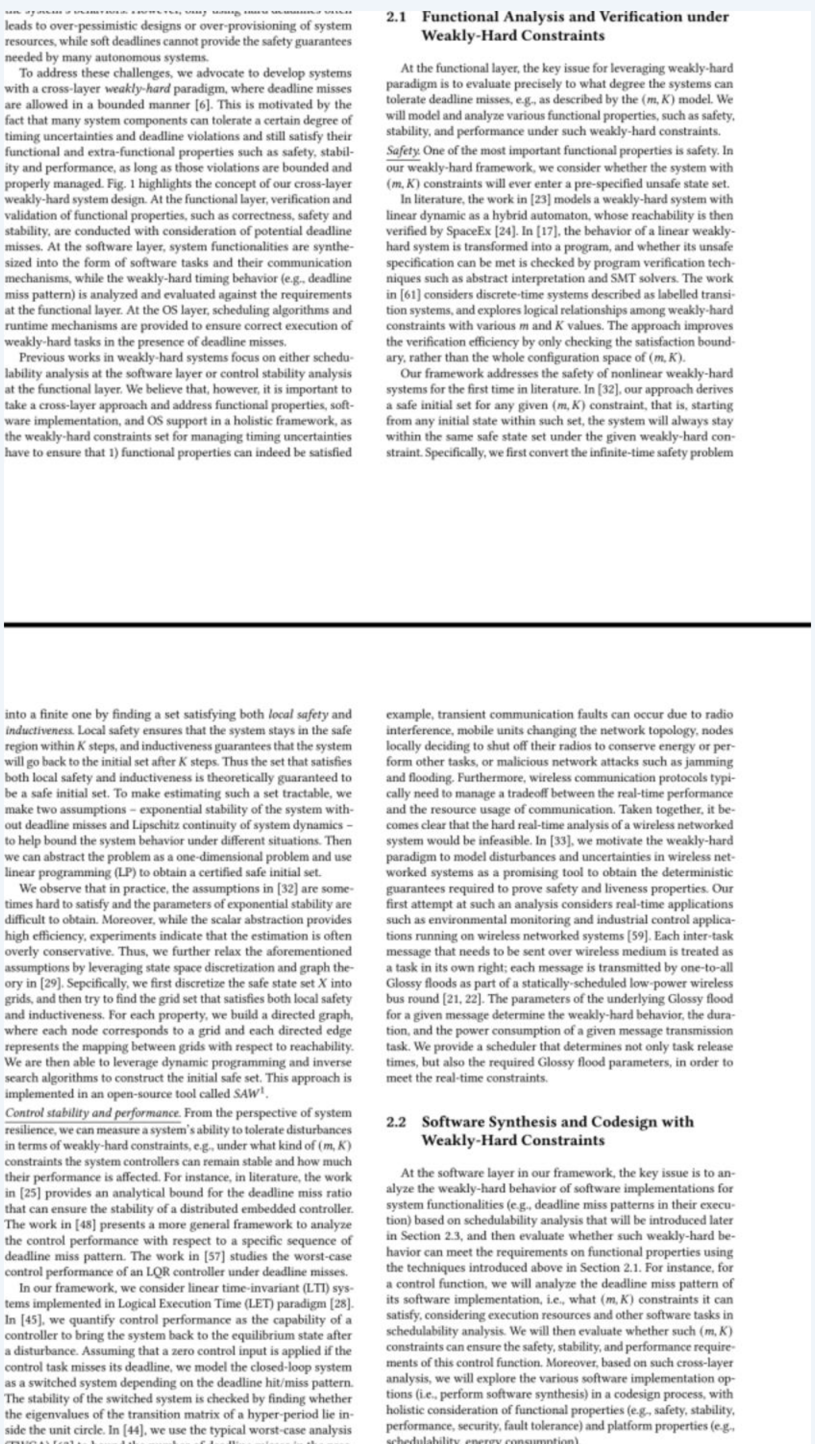
language level, enriching the domain-specific Buzz language with safety keywords and libraries. Thus, we enable the coupled exploitation of PTFA and model checking for the automated verification of policy compliance within a robot's controller.

We believe that the proposed approach can transform Buzz—that is explicitly intended for swarm robotics and currently in its early deployment stages—into the ideal platform for the software engineering of security in multi-robot systems.

Table 1: Buzz swarming functions and safety keywords

Examples of implemented domain-specific constructs and functions	
`stigmergy.{create(), put, get()}`	swarm-shared data-structure
`swarm.{create(), join(), ..}`	swarm management primitives
`neighbors.{broadcast(), filter(), ..}`	local robot-to-robot com.
Examples of envisioned security-specific keywords and functions	

	Engineering safety in swarm robotics: 3 - 4 (0)



the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems.

To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the fact that many system components can tolerate a certain degree of timing uncertainties and deadline violations and still satisfy their functional and extra-functional properties such as safety, stability and performance, as long as those violations are bounded and properly managed. Fig. 1 highlights the concept of our cross-layer weakly-hard system design. At the functional layer, verification and validation of functional properties, such as correctness, safety and stability, are conducted with consideration of potential deadline misses. At the software layer, system functionalities are synthesized into the form of software tasks and their communication mechanisms, while the weakly-hard timing behavior (e.g., deadline miss pattern) is analyzed and evaluated against the requirements at the functional layer. At the OS layer, scheduling algorithms and runtime mechanisms are provided to ensure correct execution of weakly-hard tasks in the presence of deadline misses.

Previous works in weakly-hard systems focus on either schedulability analysis at the software layer or control stability analysis at the functional layer. We believe that, however, it is important to take a cross-layer approach and address functional properties, software implementation, and OS support in a holistic framework, as the weakly-hard constraints set for managing timing uncertainties have to ensure that 1) functional properties can indeed be satisfied

2.1 Functional Analysis and Verification under Weakly-Hard Constraints

At the functional layer, the key issue for leveraging weakly-hard paradigm is to evaluate precisely to what degree the systems can tolerate deadline misses, e.g., as described by the (m, K) model. We will model and analyze various functional properties, such as safety, stability, and performance under such weakly-hard constraints.

Safety. One of the most important functional properties is safety. In our weakly-hard framework, we consider whether the system with (m, K) constraints will ever enter a pre-specified unsafe state set.

In literature, the work in [23] models a weakly-hard system with linear dynamic as a hybrid automaton, whose reachability is then verified by SpaceEx [24]. In [17], the behavior of a linear weakly-hard system is transformed into a program, and whether its unsafe specification can be met is checked by program verification techniques such as abstract interpretation and SMT solvers. The work in [61] considers discrete-time systems described as labelled transition systems, and explores logical relationships among weakly-hard constraints with various m and K values. The approach improves the verification efficiency by only checking the satisfaction boundary, rather than the whole configuration space of (m, K).

Our framework addresses the safety of nonlinear weakly-hard systems for the first time in literature. In [32], our approach derives a safe initial set for any given (m, K) constraint, that is, starting from any initial state within such set, the system will always stay within the same safe state set under the given weakly-hard constraint. Specifically, we first convert the infinite-time safety problem

into a finite one by finding a set satisfying both *local safety* and *inductiveness*. Local safety ensures that the system stays in the safe region within K steps, and inductiveness guarantees that the system will go back to the initial set after K steps. Thus the set that satisfies both local safety and inductiveness is theoretically guaranteed to be a safe initial set. To make estimating such a set tractable, we make two assumptions – exponential stability of the system without deadline misses and Lipschitz continuity of system dynamics – to help bound the system behavior under different situations. Then we can abstract the problem as a one-dimensional problem and use linear programming (LP) to obtain a certified safe initial set.

We observe that in practice, the assumptions in [32] are sometimes hard to satisfy and the parameters of exponential stability are difficult to obtain. Moreover, while the scalar abstraction provides high efficiency, experiments indicate that the estimation is often overly conservative. Thus, we further relax the aforementioned assumptions by leveraging state space discretization and graph theory in [29]. Sepcifically, we first discretize the safe state set X into grids, and then try to find the grid set that satisfies both local safety and inductiveness. For each property, we build a directed graph, where each node corresponds to a grid and each directed edge represents the mapping between grids with respect to reachability. We are then able to leverage dynamic programming and inverse search algorithms to construct the initial safe set. This approach is implemented in an open-source tool called *SAW*[1].

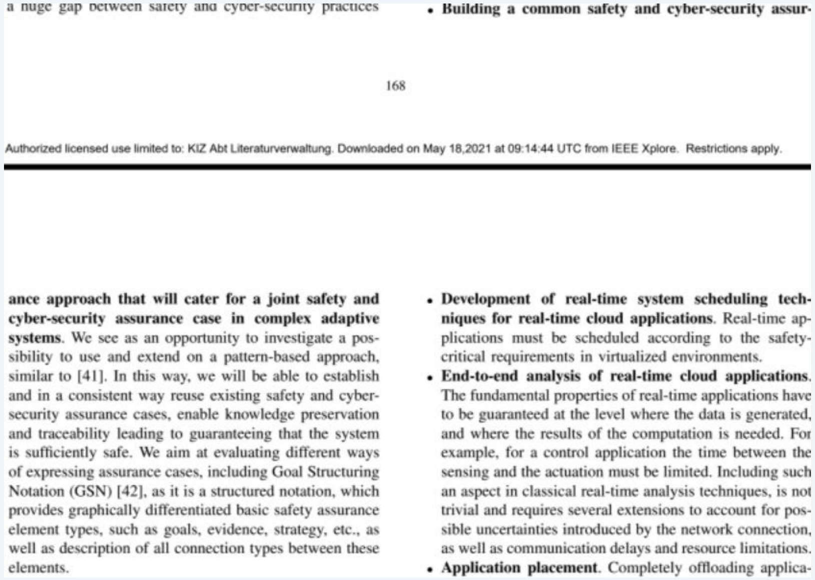
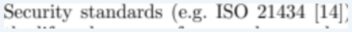
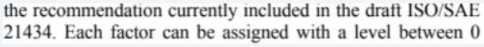
Control stability and performance. From the perspective of system resilience, we can measure a system's ability to tolerate disturbances in terms of weakly-hard constraints, e.g., under what kind of (m, K) constraints the system controllers can remain stable and how much their performance is affected. For instance, in literature, the work in [25] provides an analytical bound for the deadline miss ratio that can ensure the stability of a distributed embedded controller. The work in [48] presents a more general framework to analyze the control performance with respect to a specific sequence of deadline miss pattern. The work in [57] studies the worst-case control performance of an LQR controller under deadline misses.

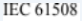
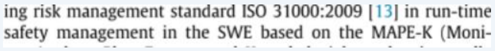
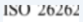
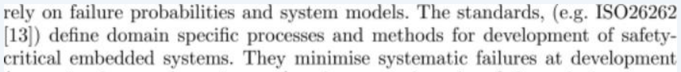
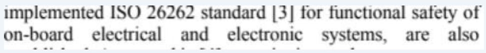
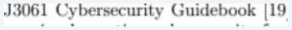
In our framework, we consider linear time-invariant (LTI) systems implemented in Logical Execution Time (LET) paradigm [28]. In [45], we quantify control performance as the capability of a controller to bring the system back to the equilibrium state after a disturbance. Assuming that a zero control input is applied if the control task misses its deadline, we model the closed-loop system as a switched system depending on the deadline hit/miss pattern. The stability of the switched system is checked by finding whether the eigenvalues of the transition matrix of a hyper-period lie inside the unit circle. In [44], we use the typical worst-case analysis (TWCA) [63] to bound the number of deadline misses in the

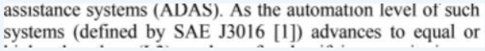
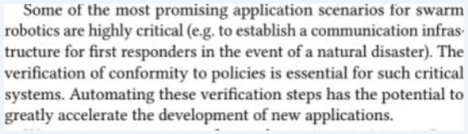
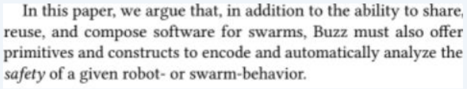
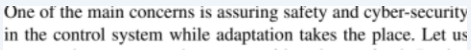
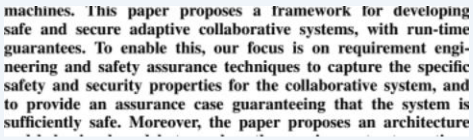
example, transient communication faults can occur due to radio interference, mobile units changing the network topology, nodes locally deciding to shut off their radios to conserve energy or perform other tasks, or malicious network attacks such as jamming and flooding. Furthermore, wireless communication protocols typically need to manage a tradeoff between the real-time performance and the resource usage of communication. Taken together, it becomes clear that the hard real-time analysis of a wireless networked system would be infeasible. In [33], we motivate the weakly-hard paradigm to model disturbances and uncertainties in wireless networked systems as a promising tool to obtain the deterministic guarantees required to prove safety and liveness properties. Our first attempt at such an analysis considers real-time applications such as environmental monitoring and industrial control applications running on wireless networked systems [59]. Each inter-task message that needs to be sent over wireless medium is treated as a task in its own right; each message is transmitted by one-to-all Glossy floods as part of a statically-scheduled low-power wireless bus round [21, 22]. The parameters of the underlying Glossy flood for a given message determine the weakly-hard behavior, the duration, and the power consumption of a given message transmission task. We provide a scheduler that determines not only task release times, but also the required Glossy flood parameters, in order to meet the real-time constraints.

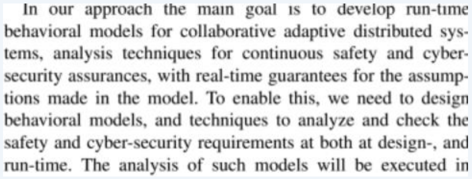
2.2 Software Synthesis and Codesign with Weakly-Hard Constraints

At the software layer in our framework, the key issue is to analyze the weakly-hard behavior of software implementations for system functionalities (e.g., deadline miss patterns in their execution) based on schedulability analysis that will be introduced later in Section 2.3, and then evaluate whether such weakly-hard behavior can meet the requirements on functional properties using the techniques introduced above in Section 2.1. For instance, for a control function, we will analyze the deadline miss pattern of its software implementation, i.e., what (m, K) constraints it can satisfy, considering execution resources and other software tasks in schedulability analysis. We will then evaluate whether such (m, K) constraints can ensure the safety, stability, and performance requirements of this control function. Moreover, based on such cross-layer analysis, we will explore the various software implementation options (i.e., perform software synthesis) in a codesign process, with holistic consideration of functional properties (e.g., safety, stability, performance, security, fault tolerance) and platform properties (e.g., schedulability, energy consumption).

	Know the unknowns addressing disturbances and uncertainties in: 2 - 7 (0)
	 a huge gap between safety and cyber-security practices • **Building a common safety and cyber-security assurance approach that will cater for a joint safety and cyber-security assurance case in complex adaptive systems**. We see as an opportunity to investigate a possibility to use and extend on a pattern-based approach, similar to [41]. In this way, we will be able to establish and in a consistent way reuse existing safety and cyber-security assurance cases, enable knowledge preservation and traceability leading to guaranteeing that the system is sufficiently safe. We aim at evaluating different ways of expressing assurance cases, including Goal Structuring Notation (GSN) [42], as it is a structured notation, which provides graphically differentiated basic safety assurance element types, such as goals, evidence, strategy, etc., as well as description of all connection types between these elements. • **Development of real-time system scheduling techniques for real-time cloud applications**. Real-time applications must be scheduled according to the safety-critical requirements in virtualized environments. • **End-to-end analysis of real-time cloud applications**. The fundamental properties of real-time applications have to be guaranteed at the level where the data is generated, and where the results of the computation is needed. For example, for a control application the time between the sensing and the actuation must be limited. Including such an aspect in classical real-time analysis techniques, is not trivial and requires several extensions to account for possible uncertainties introduced by the network connection, as well as communication delays and resource limitations. • **Application placement**. Completely offloading applica- 168 Authorized licensed use limited to: KIZ Abt Literaturverwaltung. Downloaded on May 18,2021 at 09:14:44 UTC from IEEE Xplore. Restrictions apply. Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 5 (0)
Standards\Security\ISO/IEC 9126	 SO/IEC 9126 Approaching runtime trust assurance in open adaptive systems: 4 - 4 (0)
Standards\Security\ISO/SAE 21434	 Security standards (e.g. ISO 21434 [14]) Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)
	 the recommendation currently included in the draft ISO/SAE 21434. Each factor can be assigned with a level between 0 TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)
Standards\Security\ISO/IEC18045	 ISO/IEC18045 TARA Controllability-aware Threat Analysis and Risk Assessment: 2 - 2 (0)

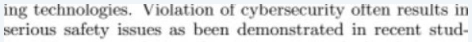
Standards\Safety\IEC 61508	 IEC 61508 Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)
Standards\Safety\ISO 31000:2009	 ing risk management standard ISO 31000:2009 [13] in run-time safety management in the SWE based on the MAPE-K (Moni- Ontology development for run-time safety management methodology: 2 - 2 (0)
Standards\Safety\ISO 26262	 ISO 26262 A Functional Co-Design towards Safe and Secure Vehicle Platooni: 5 - 5 (0)
	 ISO 26262 Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)
	 rely on failure probabilities and system models. The standards, (e.g. ISO26262 [13]) define domain specific processes and methods for development of safety-critical embedded systems. They minimise systematic failures at development Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)
	 e ISO 26262 Potential cyberattacks on automated vehicles: 3 - 3 (0)
	 h ISO26262 SARA Security Automotive Risk Analysis Method: 2 - 2 (0)
	 implemented ISO 26262 standard [3] for functional safety of on-board electrical and electronic systems, are also TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)
Standards\Agnostic\SAE J3016	 J3061 Cybersecurity Guidebook [19 A Functional Co-Design towards Safe and Secure Vehicle Platooni: 4 - 4 (0)

	SAE J3016 Potential cyberattacks on automated vehicles: 3 - 3 (0) SAEJ3061 SARA Security Automotive Risk Analysis Method: 2 - 2 (0)  assistance systems (ADAS). As the automation level of such systems (defined by SAE J3016 [1]) advances to equal or TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)
Challenges\Addressed\Verification	 Some of the most promising application scenarios for swarm robotics are highly critical (e.g. to establish a communication infrastructure for first responders in the event of a natural disaster). The verification of conformity to policies is essential for such critical systems. Automating these verification steps has the potential to greatly accelerate the development of new applications. Engineering safety in swarm robotics: 2 - 2 (0)
Challenges\Addressed\Verification\Addressed by	 In this paper, we argue that, in addition to the ability to share, reuse, and compose software for swarms, Buzz must also offer primitives and constructs to encode and automatically analyze the *safety* of a given robot- or swarm-behavior. Engineering safety in swarm robotics: 2 - 2 (0)
Challenges\Addressed\Adaptation	 One of the main concerns is assuring safety and cyber-security in the control system while adaptation takes the place. Let us Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)
Challenges\Addressed\Adaptation\Addressed by	 **machines. This paper proposes a framework for developing safe and secure adaptive collaborative systems, with run-time guarantees. To enable this, our focus is on requirement engineering and safety assurance techniques to capture the specific safety and security properties for the collaborative system, and to provide an assurance case guaranteeing that the system is sufficiently safe. Moreover, the paper proposes an architecture** Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)

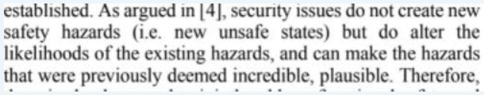

In our approach the main goal is to develop run-time behavioral models for collaborative adaptive distributed systems, analysis techniques for continuous safety and cyber-security assurances, with real-time guarantees for the assumptions made in the model. To enable this, we need to design behavioral models, and techniques to analyze and check the safety and cyber-security requirements at both at design-, and run-time. The analysis of such models will be executed in

Towards a Framework for Safe and Secure Adaptive Collaborative: 1 - 1 (0)

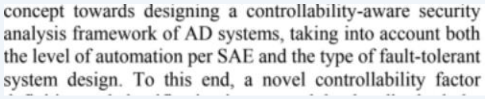
Challenges\Addressed\Attack


ing technologies. Violation of cybersecurity often results in serious safety issues as been demonstrated in recent stud-

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 1 - 1 (0)

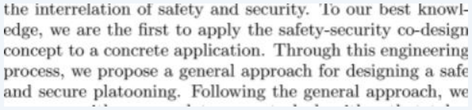

established. As argued in [4], security issues do not create new safety hazards (i.e. new unsafe states) but do alter the likelihoods of the existing hazards, and can make the hazards that were previously deemed incredible, plausible. Therefore,

TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)

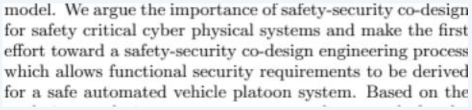

concept towards designing a controllability-aware security analysis framework of AD systems, taking into account both the level of automation per SAE and the type of fault-tolerant system design. To this end, a novel controllability factor

TARA Controllability-aware Threat Analysis and Risk Assessment: 5 - 5 (0)

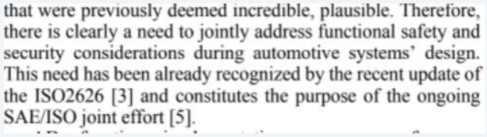
Challenges\Addressed\Attack\Addressed by


the interrelation of safety and security. To our best knowledge, we are the first to apply the safety-security co-design concept to a concrete application. Through this engineering process, we propose a general approach for designing a safe and secure platooning. Following the general approach, we

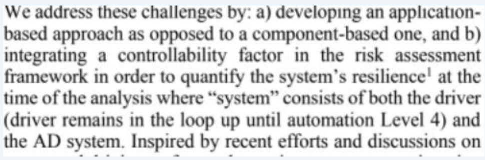
A Functional Co-Design towards Safe and Secure Vehicle Platooni: 1 - 1 (0)


model. We argue the importance of safety-security co-design for safety critical cyber physical systems and make the first effort toward a safety-security co-design engineering process which allows functional security requirements to be derived for a safe automated vehicle platoon system. Based on the

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 9 - 9 (0)

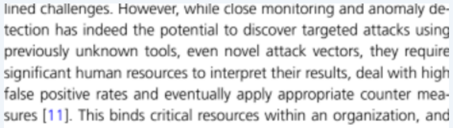

that were previously deemed incredible, plausible. Therefore, there is clearly a need to jointly address functional safety and security considerations during automotive systems' design. This need has been already recognized by the recent update of the ISO2626 [3] and constitutes the purpose of the ongoing SAE/ISO joint effort [5].

TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)

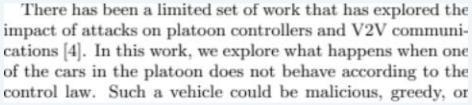

We address these challenges by: a) developing an application-based approach as opposed to a component-based one, and b) integrating a controllability factor in the risk assessment framework in order to quantify the system's resilience[1] at the time of the analysis where "system" consists of both the driver (driver remains in the loop up until automation Level 4) and the AD system. Inspired by recent efforts and discussions on

TARA Controllability-aware Threat Analysis and Risk Assessment: 1 - 1 (0)

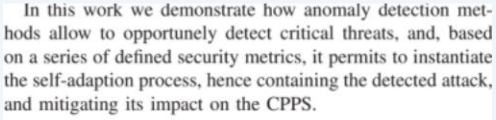
Challenges\Addressed\Detecting Defects


lined challenges. However, while close monitoring and anomaly detection has indeed the potential to discover targeted attacks using previously unknown tools, even novel attack vectors, they require significant human resources to interpret their results, deal with high false positive rates and eventually apply appropriate counter measures [11]. This binds critical resources within an organization, and

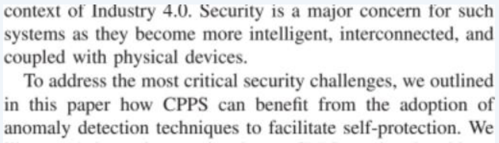
Countering targeted cyber-physical attacks using anomaly detect: 1 - 1 (0)


There has been a limited set of work that has explored the impact of attacks on platoon controllers and V2V communications [4]. In this work, we explore what happens when one of the cars in the platoon does not behave according to the control law. Such a vehicle could be malicious, greedy, or

Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)


In this work we demonstrate how anomaly detection methods allow to opportunely detect critical threats, and, based on a series of defined security metrics, it permits to instantiate the self-adaption process, hence containing the detected attack, and mitigating its impact on the CPPS.

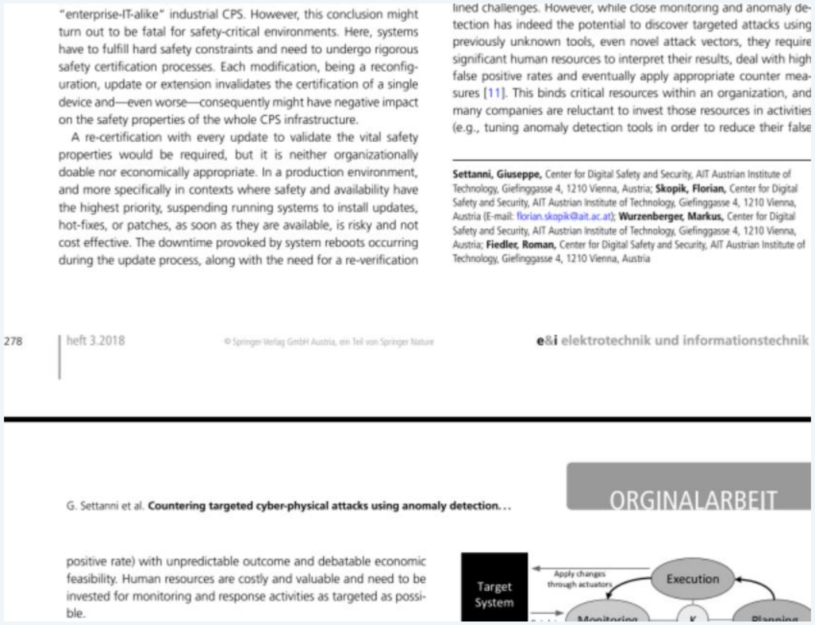
Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)


context of Industry 4.0. Security is a major concern for such systems as they become more intelligent, interconnected, and coupled with physical devices.

To address the most critical security challenges, we outlined in this paper how CPPS can benefit from the adoption of anomaly detection techniques to facilitate self-protection. We

Protecting cyber physical production systems using anomaly dete: 8 - 8 (0)

Challenges\Addressed\Detecting Defects\Addressed by



"enterprise-IT-alike" industrial CPS. However, this conclusion might turn out to be fatal for safety-critical environments. Here, systems have to fulfill hard safety constraints and need to undergo rigorous safety certification processes. Each modification, being a reconfiguration, update or extension invalidates the certification of a single device and—even worse—consequently might have negative impact on the safety properties of the whole CPS infrastructure.

A re-certification with every update to validate the vital safety properties would be required, but it is neither organizationally doable nor economically appropriate. In a production environment, and more specifically in contexts where safety and availability have the highest priority, suspending running systems to install updates, hot-fixes, or patches, as soon as they are available, is risky and not cost effective. The downtime provoked by system reboots occurring during the update process, along with the need for a re-verification

lined challenges. However, while close monitoring and anomaly detection has indeed the potential to discover targeted attacks using previously unknown tools, even novel attack vectors, they require significant human resources to interpret their results, deal with high false positive rates and eventually apply appropriate counter measures [11]. This binds critical resources within an organization, and many companies are reluctant to invest those resources in activities (e.g., tuning anomaly detection tools in order to reduce their false

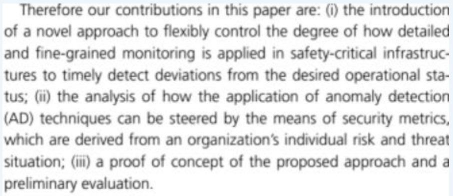
Settanni, Giuseppe, Center for Digital Safety and Security, AIT Austrian Institute of Technology, Giefinggasse 4, 1210 Vienna, Austria; **Skopik, Florian,** Center for Digital Safety and Security, AIT Austrian Institute of Technology, Giefinggasse 4, 1210 Vienna, Austria (E-mail: florian.skopik@ait.ac.at); **Wurzenberger, Markus,** Center for Digital Safety and Security, AIT Austrian Institute of Technology, Giefinggasse 4, 1210 Vienna, Austria; **Fiedler, Roman,** Center for Digital Safety and Security, AIT Austrian Institute of Technology, Giefinggasse 4, 1210 Vienna, Austria

278 | heft 3.2018 © Springer-Verlag GmbH Austria, ein Teil von Springer Nature **e&i** elektrotechnik und informationstechnik

G. Settanni et al. **Countering targeted cyber-physical attacks using anomaly detection...** ORGINALARBEIT

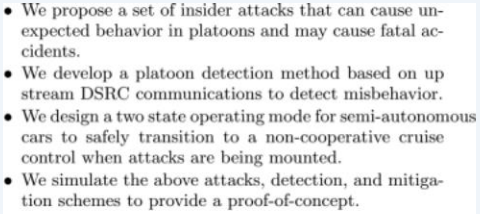
positive rate) with unpredictable outcome and debatable economic feasibility. Human resources are costly and valuable and need to be invested for monitoring and response activities as targeted as possible.

Countering targeted cyber-physical attacks using anomaly detect: 1 - 2 (0)



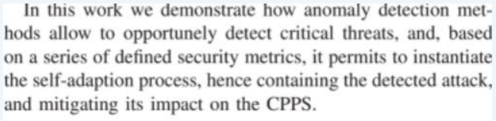
Therefore our contributions in this paper are: (i) the introduction of a novel approach to flexibly control the degree of how detailed and fine-grained monitoring is applied in safety-critical infrastructures to timely detect deviations from the desired operational status; (ii) the analysis of how the application of anomaly detection (AD) techniques can be steered by the means of security metrics, which are derived from an organization's individual risk and threat situation; (iii) a proof of concept of the proposed approach and a preliminary evaluation.

Countering targeted cyber-physical attacks using anomaly detect: 2 - 2 (0)



- We propose a set of insider attacks that can cause unexpected behavior in platoons and may cause fatal accidents.
- We develop a platoon detection method based on up stream DSRC communications to detect misbehavior.
- We design a two state operating mode for semi-autonomous cars to safely transition to a non-cooperative cruise control when attacks are being mounted.
- We simulate the above attacks, detection, and mitigation schemes to provide a proof-of-concept.

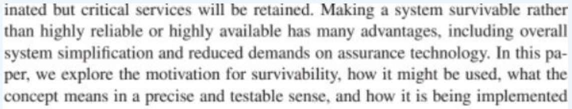
Is your commute driving you crazy a study of misbehavior in veh: 2 - 2 (0)



In this work we demonstrate how anomaly detection methods allow to opportunely detect critical threats, and, based on a series of defined security metrics, it permits to instantiate the self-adaption process, hence containing the detected attack, and mitigating its impact on the CPPS.

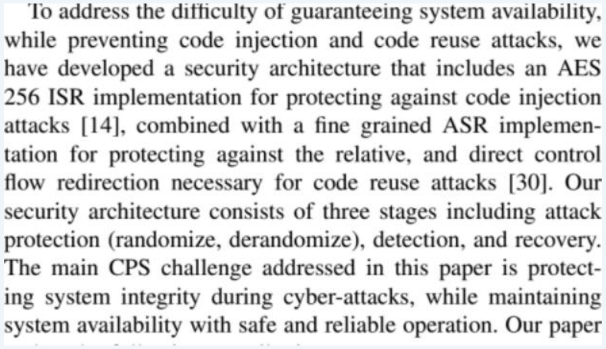
Protecting cyber physical production systems using anomaly dete: 2 - 2 (0)

Challenges\Addressed\Survivability and Resilience



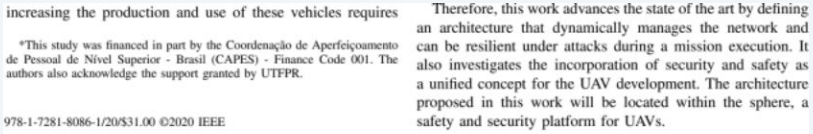
inated but critical services will be retained. Making a system survivable rather than highly reliable or highly available has many advantages, including overall system simplification and reduced demands on assurance technology. In this paper, we explore the motivation for survivability, how it might be used, what the concept means in a precise and testable sense, and how it is being implemented

Achieving critical system survivability through software archit: 1 - 1 (0)



To address the difficulty of guaranteeing system availability, while preventing code injection and code reuse attacks, we have developed a security architecture that includes an AES 256 ISR implementation for protecting against code injection attacks [14], combined with a fine grained ASR implementation for protecting against the relative, and direct control flow redirection necessary for code reuse attacks [30]. Our security architecture consists of three stages including attack protection (randomize, derandomize), detection, and recovery. The main CPS challenge addressed in this paper is protecting system integrity during cyber-attacks, while maintaining system availability with safe and reliable operation. Our paper

Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)



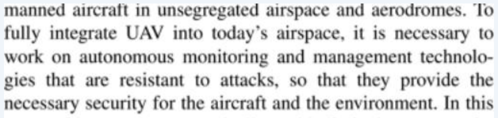
increasing the production and use of these vehicles requires

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors also acknowledge the support granted by UTFPR.

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

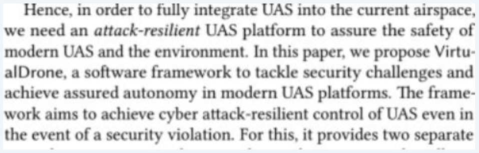
Therefore, this work advances the state of the art by defining an architecture that dynamically manages the network and can be resilient under attacks during a mission execution. It also investigates the incorporation of security and safety as a unified concept for the UAV development. The architecture proposed in this work will be located within the sphere, a safety and security platform for UAVs.

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)



manned aircraft in unsegregated airspace and aerodromes. To fully integrate UAV into today's airspace, it is necessary to work on autonomous monitoring and management technologies that are resistant to attacks, so that they provide the necessary security for the aircraft and the environment. In this

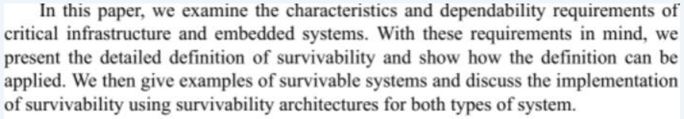
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)



Hence, in order to fully integrate UAS into the current airspace, we need an *attack-resilient* UAS platform to assure the safety of modern UAS and the environment. In this paper, we propose VirtualDrone, a software framework to tackle security challenges and achieve assured autonomy in modern UAS platforms. The framework aims to achieve cyber attack-resilient control of UAS even in the event of a security violation. For this, it provides two separate

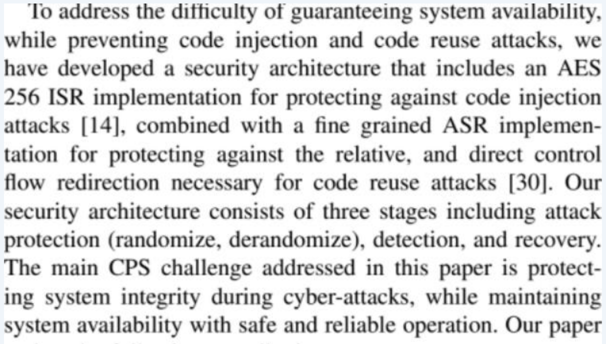
VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)

Challenges\Addressed\Survivability and Resilience\Addressed by



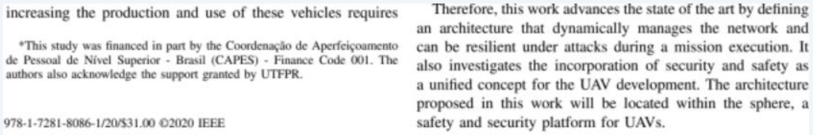
In this paper, we examine the characteristics and dependability requirements of critical infrastructure and embedded systems. With these requirements in mind, we present the detailed definition of survivability and show how the definition can be applied. We then give examples of survivable systems and discuss the implementation of survivability using survivability architectures for both types of system.

Achieving critical system survivability through software archit: 2 - 2 (0)



To address the difficulty of guaranteeing system availability, while preventing code injection and code reuse attacks, we have developed a security architecture that includes an AES 256 ISR implementation for protecting against code injection attacks [14], combined with a fine grained ASR implementation for protecting against the relative, and direct control flow redirection necessary for code reuse attacks [30]. Our security architecture consists of three stages including attack protection (randomize, derandomize), detection, and recovery. The main CPS challenge addressed in this paper is protecting system integrity during cyber-attacks, while maintaining system availability with safe and reliable operation. Our paper

Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)

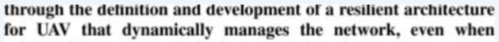

increasing the production and use of these vehicles requires

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. The authors also acknowledge the support granted by UTFPR.

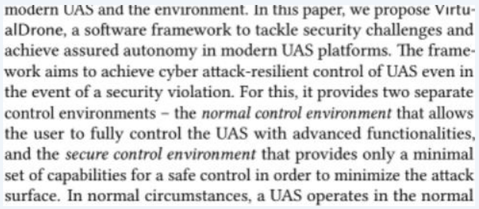
978-1-7281-8086-1/20/$31.00 ©2020 IEEE

Therefore, this work advances the state of the art by defining an architecture that dynamically manages the network and can be resilient under attacks during a mission execution. It also investigates the incorporation of security and safety as a unified concept for the UAV development. The architecture proposed in this work will be located within the sphere, a safety and security platform for UAVs.

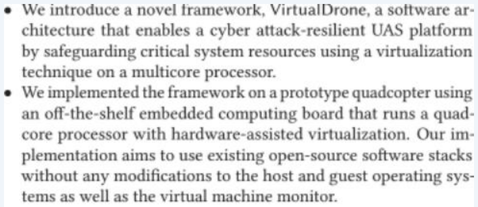
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)


through the definition and development of a resilient architecture for UAV that dynamically manages the network, even when

STUART ReSilient archiTecture to dynamically manage Unmanned ae: 1 - 1 (0)

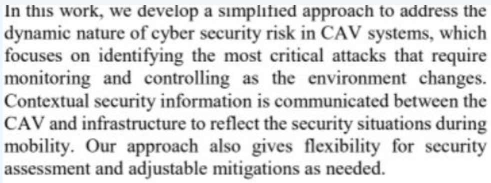

modern UAS and the environment. In this paper, we propose VirtualDrone, a software framework to tackle security challenges and achieve assured autonomy in modern UAS platforms. The framework aims to achieve cyber attack-resilient control of UAS even in the event of a security violation. For this, it provides two separate control environments – the *normal control environment* that allows the user to fully control the UAS with advanced functionalities, and the *secure control environment* that provides only a minimal set of capabilities for a safe control in order to minimize the attack surface. In normal circumstances, a UAS operates in the normal

VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)

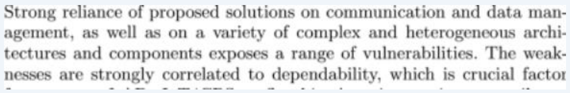

- We introduce a novel framework, VirtualDrone, a software architecture that enables a cyber attack-resilient UAS platform by safeguarding critical system resources using a virtualization technique on a multicore processor.
- We implemented the framework on a prototype quadcopter using an off-the-shelf embedded computing board that runs a quad-core processor with hardware-assisted virtualization. Our implementation aims to use existing open-source software stacks without any modifications to the host and guest operating systems as well as the virtual machine monitor.

VirtualDrone virtual sensing actuation and communication for at: 2 - 2 (0)

Challenges\Addressed\System and Environment

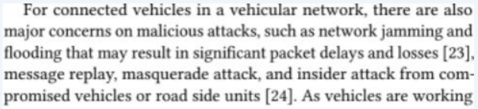

In this work, we develop a simplified approach to address the dynamic nature of cyber security risk in CAV systems, which focuses on identifying the most critical attacks that require monitoring and controlling as the environment changes. Contextual security information is communicated between the CAV and infrastructure to reflect the security situations during mobility. Our approach also gives flexibility for security assessment and adjustable mitigations as needed.

A simplified approach for dynamic security risk management in c: 2 - 2 (0)



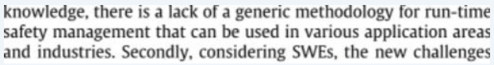
Strong reliance of proposed solutions on communication and data management, as well as on a variety of complex and heterogeneous architectures and components exposes a range of vulnerabilities. The weaknesses are strongly correlated to dependability, which is crucial factor

Digital Twins for Dependability Improvement of Autonomous Drivi: 2 - 2 (0)



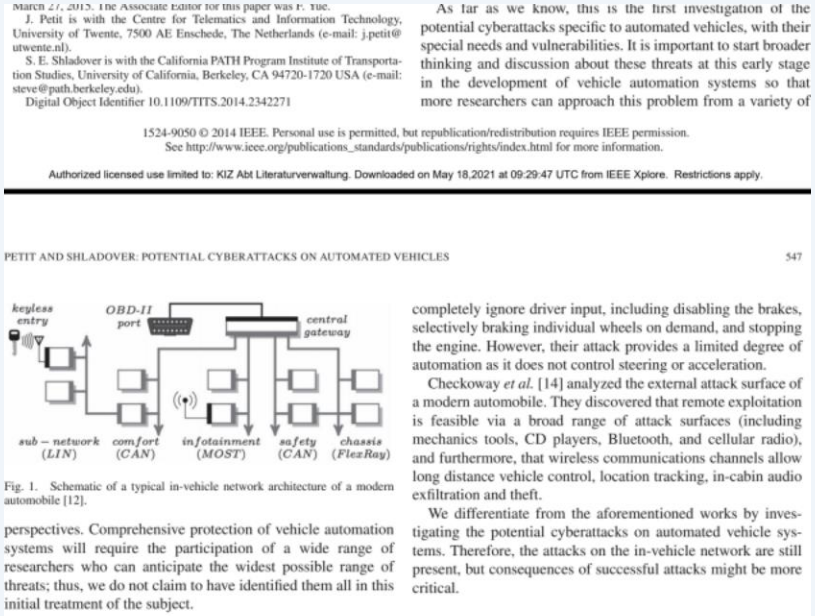
For connected vehicles in a vehicular network, there are also major concerns on malicious attacks, such as network jamming and flooding that may result in significant packet delays and losses [23], message replay, masquerade attack, and insider attack from compromised vehicles or road side units [24]. As vehicles are working

Network and system level security in connected vehicle applicat: 1 - 1 (0)



knowledge, there is a lack of a generic methodology for run-time safety management that can be used in various application areas and industries. Secondly, considering SWEs, the new challenges

Ontology development for run-time safety management methodology: 3 - 3 (0)



keyless entry
OBD-II port
central gateway
sub – network (LIN)
comfort (CAN)
infotainment (MOST)
safety (CAN)
chassis (FlexRay)

As far as we know, this is the first investigation of the potential cyberattacks specific to automated vehicles, with their special needs and vulnerabilities. It is important to start broader thinking and discussion about these threats at this early stage in the development of vehicle automation systems so that more researchers can approach this problem from a variety of perspectives. Comprehensive protection of vehicle automation systems will require the participation of a wide range of researchers who can anticipate the widest possible range of threats; thus, we do not claim to have identified them all in this initial treatment of the subject.

PETIT AND SHLADOVER: POTENTIAL CYBERATTACKS ON AUTOMATED VEHICLES 547

Fig. 1. Schematic of a typical in-vehicle network architecture of a modern automobile [12].

completely ignore driver input, including disabling the brakes, selectively braking individual wheels on demand, and stopping the engine. However, their attack provides a limited degree of automation as it does not control steering or acceleration.

Checkoway *et al.* [14] analyzed the external attack surface of a modern automobile. They discovered that remote exploitation is feasible via a broad range of attack surfaces (including mechanics tools, CD players, Bluetooth, and cellular radio), and furthermore, that wireless communications channels allow long distance vehicle control, location tracking, in-cabin audio exfiltration and theft.

We differentiate from the aforementioned works by investigating the potential cyberattacks on automated vehicle systems. Therefore, the attacks on the in-vehicle network are still present, but consequences of successful attacks might be more critical.

March 27, 2015. The Associate Editor for this paper was F. Yue.

J. Petit is with the Centre for Telematics and Information Technology, University of Twente, 7500 AE Enschede, The Netherlands (e-mail: j.petit@utwente.nl).

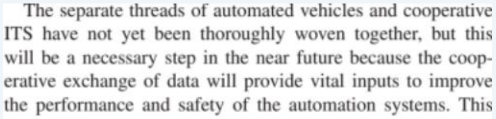
S. E. Shladover is with the California PATH Program Institute of Transportation Studies, University of California, Berkeley, CA 94720-1720 USA (e-mail: steve@path.berkeley.edu).

Digital Object Identifier 10.1109/TITS.2014.2342271

1524-9050 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

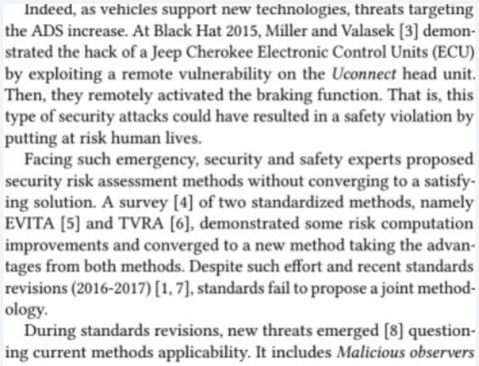
Authorized licensed use limited to: KIZ Abt Literaturverwaltung. Downloaded on May 18,2021 at 09:29:47 UTC from IEEE Xplore. Restrictions apply.

Potential cyberattacks on automated vehicles: 1 - 2 (0)



The separate threads of automated vehicles and cooperative ITS have not yet been thoroughly woven together, but this will be a necessary step in the near future because the cooperative exchange of data will provide vital inputs to improve the performance and safety of the automation systems. This

Potential cyberattacks on automated vehicles: 1 - 1 (0)

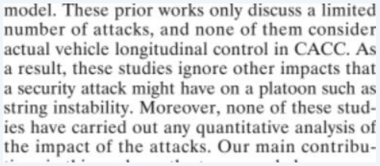


Indeed, as vehicles support new technologies, threats targeting the ADS increase. At Black Hat 2015, Miller and Valasek [3] demonstrated the hack of a Jeep Cherokee Electronic Control Units (ECU) by exploiting a remote vulnerability on the *Uconnect* head unit. Then, they remotely activated the braking function. That is, this type of security attacks could have resulted in a safety violation by putting at risk human lives.

Facing such emergency, security and safety experts proposed security risk assessment methods without converging to a satisfying solution. A survey [4] of two standardized methods, namely EVITA [5] and TVRA [6], demonstrated some risk computation improvements and converged to a new method taking the advantages from both methods. Despite such effort and recent standards revisions (2016-2017) [1, 7], standards fail to propose a joint methodology.

During standards revisions, new threats emerged [8] questioning current methods applicability. It includes *Malicious observers*

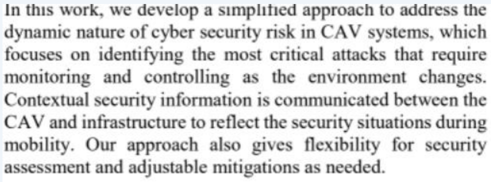
SARA Security Automotive Risk Analysis Method: 1 - 1 (0)



model. These prior works only discuss a limited number of attacks, and none of them consider actual vehicle longitudinal control in CACC. As a result, these studies ignore other impacts that a security attack might have on a platoon such as string instability. Moreover, none of these studies have carried out any quantitative analysis of the impact of the attacks. Our main contribu-

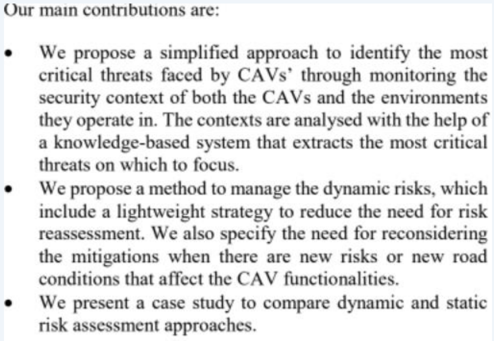
Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

Challenges\Addressed\System and Environment\Addressed by



In this work, we develop a simplified approach to address the dynamic nature of cyber security risk in CAV systems, which focuses on identifying the most critical attacks that require monitoring and controlling as the environment changes. Contextual security information is communicated between the CAV and infrastructure to reflect the security situations during mobility. Our approach also gives flexibility for security assessment and adjustable mitigations as needed.

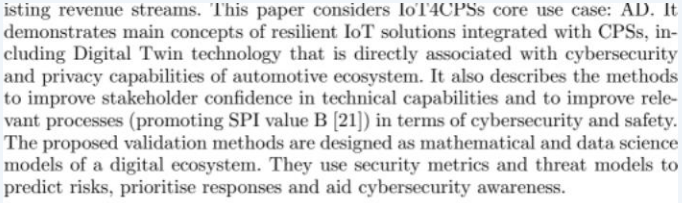
A simplified approach for dynamic security risk management in c: 2 - 2 (0)



Our main contributions are:

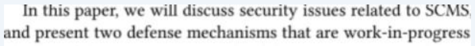
- We propose a simplified approach to identify the most critical threats faced by CAVs' through monitoring the security context of both the CAVs and the environments they operate in. The contexts are analysed with the help of a knowledge-based system that extracts the most critical threats on which to focus.
- We propose a method to manage the dynamic risks, which include a lightweight strategy to reduce the need for risk reassessment. We also specify the need for reconsidering the mitigations when there are new risks or new road conditions that affect the CAV functionalities.
- We present a case study to compare dynamic and static risk assessment approaches.

A simplified approach for dynamic security risk management in c: 2 - 2 (0)



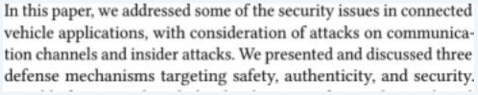
isting revenue streams. This paper considers IoT4CPSs core use case: AD. It demonstrates main concepts of resilient IoT solutions integrated with CPSs, including Digital Twin technology that is directly associated with cybersecurity and privacy capabilities of automotive ecosystem. It also describes the methods to improve stakeholder confidence in technical capabilities and to improve relevant processes (promoting SPI value B [21]) in terms of cybersecurity and safety. The proposed validation methods are designed as mathematical and data science models of a digital ecosystem. They use security metrics and threat models to predict risks, prioritise responses and aid cybersecurity awareness.

Digital Twins for Dependability Improvement of Autonomous Drivi: 4 - 4 (0)



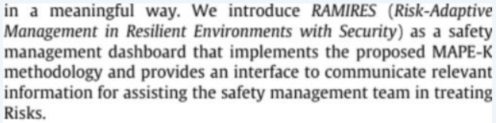
In this paper, we will discuss security issues related to SCMS, and present two defense mechanisms that are work-in-progress

Network and system level security in connected vehicle applicat: 2 - 2 (0)



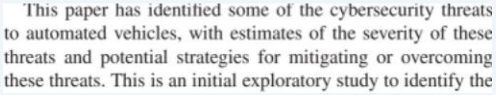
In this paper, we addressed some of the security issues in connected vehicle applications, with consideration of attacks on communication channels and insider attacks. We presented and discussed three defense mechanisms targeting safety, authenticity, and security.

Network and system level security in connected vehicle applicat: 7 - 7 (0)

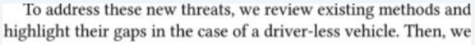


in a meaningful way. We introduce *RAMIRES* (*Risk-Adaptive Management in Resilient Environments with Security*) as a safety management dashboard that implements the proposed MAPE-K methodology and provides an interface to communicate relevant information for assisting the safety management team in treating Risks.

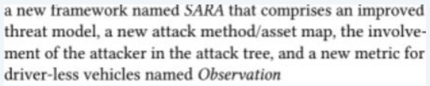
Ontology development for run-time safety management methodology: 2 - 2 (0)


This paper has identified some of the cybersecurity threats to automated vehicles, with estimates of the severity of these threats and potential strategies for mitigating or overcoming these threats. This is an initial exploratory study to identify the

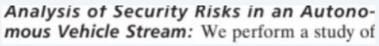
Potential cyberattacks on automated vehicles: 9 - 9 (0)


To address these new threats, we review existing methods and highlight their gaps in the case of a driver-less vehicle. Then, we

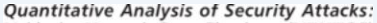
SARA Security Automotive Risk Analysis Method: 2 - 2 (0)


a new framework named *SARA* that comprises an improved threat model, a new attack method/asset map, the involvement of the attacker in the attack tree, and a new metric for driver-less vehicles named *Observation*

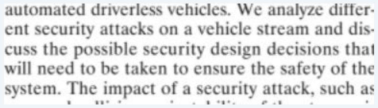
SARA Security Automotive Risk Analysis Method: 2 - 2 (0)


Analysis of Security Risks in an Autonomous Vehicle Stream: We perform a study of

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)

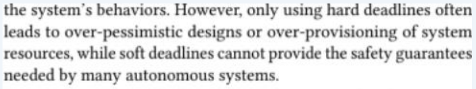

Quantitative Analysis of Security Attacks:

Security vulnerabilities of connected vehicle streams and their: 2 - 2 (0)


automated driverless vehicles. We analyze different security attacks on a vehicle stream and discuss the possible security design decisions that will need to be taken to ensure the safety of the system. The impact of a security attack, such as

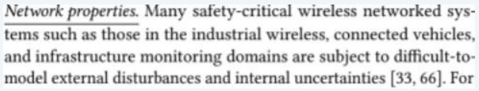
Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)

Challenges\Addressed\Other

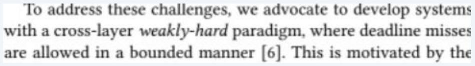
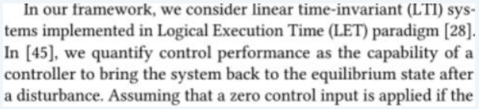
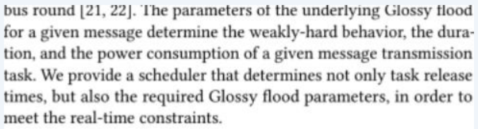
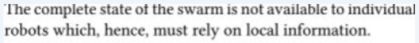
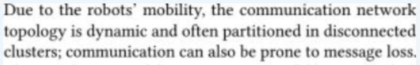
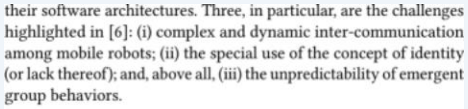
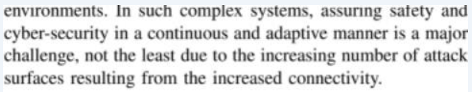

the system's behaviors. However, only using hard deadlines often leads to over-pessimistic designs or over-provisioning of system resources, while soft deadlines cannot provide the safety guarantees needed by many autonomous systems.

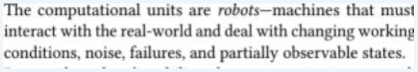
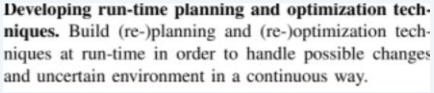
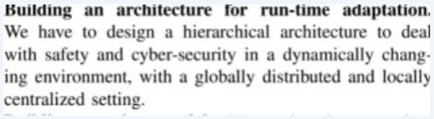
Timing uncertainty

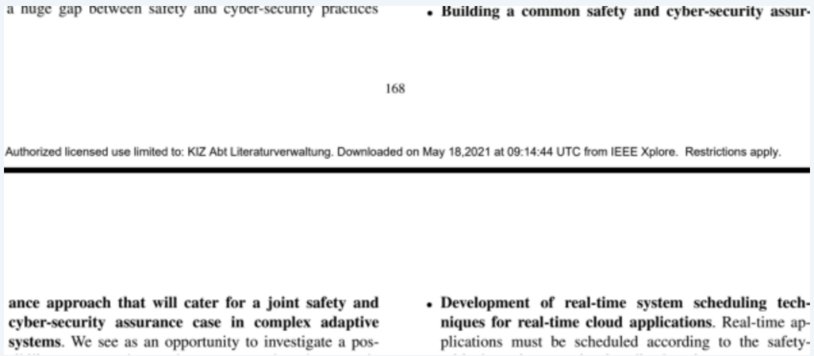
Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)


Network properties. Many safety-critical wireless networked systems such as those in the industrial wireless, connected vehicles, and infrastructure monitoring domains are subject to difficult-to-model external disturbances and internal uncertainties [33, 66]. For

Network properties

	Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)
Challenges\Addressed\Other\Addressed by	 To address these challenges, we advocate to develop systems with a cross-layer *weakly-hard* paradigm, where deadline misses are allowed in a bounded manner [6]. This is motivated by the Know the unknowns addressing disturbances and uncertainties in: 2 - 2 (0)  In our framework, we consider linear time-invariant (LTI) systems implemented in Logical Execution Time (LET) paradigm [28]. In [45], we quantify control performance as the capability of a controller to bring the system back to the equilibrium state after a disturbance. Assuming that a zero control input is applied if the Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)  bus round [21, 22]. The parameters of the underlying Glossy flood for a given message determine the weakly-hard behavior, the duration, and the power consumption of a given message transmission task. We provide a scheduler that determines not only task release times, but also the required Glossy flood parameters, in order to meet the real-time constraints. Know the unknowns addressing disturbances and uncertainties in: 3 - 3 (0)
Challenges\Open\System	 The complete state of the swarm is not available to individual robots which, hence, must rely on local information. Engineering safety in swarm robotics: 1 - 1 (0)  Due to the robots' mobility, the communication network topology is dynamic and often partitioned in disconnected clusters; communication can also be prone to message loss. Engineering safety in swarm robotics: 1 - 1 (0)  their software architectures. Three, in particular, are the challenges highlighted in [6]: (i) complex and dynamic inter-communication among mobile robots; (ii) the special use of the concept of identity (or lack thereof); and, above all, (iii) the unpredictability of emergent group behaviors. Engineering safety in swarm robotics: 2 - 2 (0)
	 environments. In such complex systems, assuring safety and cyber-security in a continuous and adaptive manner is a major challenge, not the least due to the increasing number of attack surfaces resulting from the increased connectivity. Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)

Challenges\Open\Environment	 The computational units are *robots*—machines that must interact with the real-world and deal with changing working conditions, noise, failures, and partially observable states. Engineering safety in swarm robotics: 1 - 1 (0) needs. Moreover, safety management incorporates various steps based on the standards and directives (i.e., OSHA) that should be clarified and adopted in the introduced methodology, specifically considering the SWE requirements. Finally, incorporating the Ontology development for run-time safety management methodology: 3 - 3 (0) and industries. Secondly, considering SWEs, the new challenges that they introduce and their new requirements regarding safety, a methodology should be designed to tackle these challenges and needs. Moreover, safety management incorporates various steps Ontology development for run-time safety management methodology: 3 - 3 (0)  **Developing run-time planning and optimization techniques.** Build (re-)planning and (re-)optimization techniques at run-time in order to handle possible changes and uncertain environment in a continuous way. Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)  **Building an architecture for run-time adaptation.** We have to design a hierarchical architecture to deal with safety and cyber-security in a dynamically changing environment, with a globally distributed and locally centralized setting. Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)
Challenges\Open\Safety and Security	considering the SWE requirements. Finally, incorporating the balance between safety and security is a challenging task and needs to be tackled with care as explained in what follows. Ontology development for run-time safety management methodology: 3 - 3 (0)



a huge gap between safety and cyber-security practices

- **Building a common safety and cyber-security assur-**

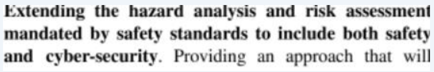
168

Authorized licensed use limited to: KIZ Abt Literaturverwaltung. Downloaded on May 18,2021 at 09:14:44 UTC from IEEE Xplore. Restrictions apply.

ance approach that will cater for a joint safety and cyber-security assurance case in complex adaptive systems. We see as an opportunity to investigate a pos-

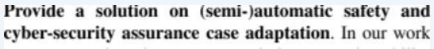
- **Development of real-time system scheduling techniques for real-time cloud applications**. Real-time applications must be scheduled according to the safety-

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 5 (0)



Extending the hazard analysis and risk assessment mandated by safety standards to include both safety and cyber-security. Providing an approach that will

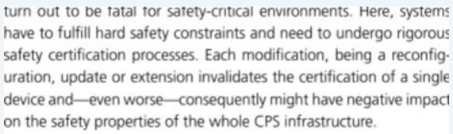
Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)



Provide a solution on (semi-)automatic safety and cyber-security assurance case adaptation. In our work

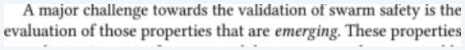
Towards a Framework for Safe and Secure Adaptive Collaborative: 5 - 5 (0)

Challenges\Open\Adaptation



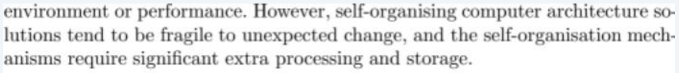
turn out to be fatal for safety-critical environments. Here, systems have to fulfill hard safety constraints and need to undergo rigorous safety certification processes. Each modification, being a reconfiguration, update or extension invalidates the certification of a single device and—even worse—consequently might have negative impact on the safety properties of the whole CPS infrastructure.

Countering targeted cyber-physical attacks using anomaly detect: 1 - 1 (0)



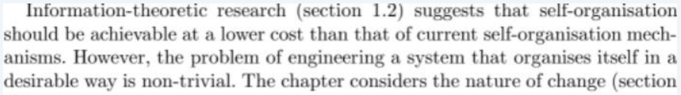
A major challenge towards the validation of swarm safety is the evaluation of those properties that are *emerging*. These properties

Engineering safety in swarm robotics: 4 - 4 (0)



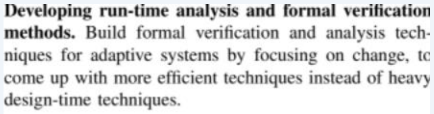
environment or performance. However, self-organising computer architecture solutions tend to be fragile to unexpected change, and the self-organisation mechanisms require significant extra processing and storage.

Self-organisation for Survival in Complex Computer Architecture: 1 - 1 (0)



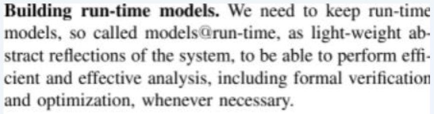
Information-theoretic research (section 1.2) suggests that self-organisation should be achievable at a lower cost than that of current self-organisation mechanisms. However, the problem of engineering a system that organises itself in a desirable way is non-trivial. The chapter considers the nature of change (section

Self-organisation for Survival in Complex Computer Architecture: 2 - 2 (0)



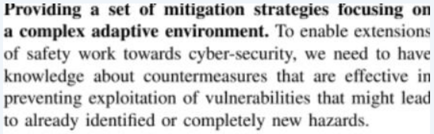
Developing run-time analysis and formal verification methods. Build formal verification and analysis techniques for adaptive systems by focusing on change, to come up with more efficient techniques instead of heavy design-time techniques.

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)



Building run-time models. We need to keep run-time models, so called models@run-time, as light-weight abstract reflections of the system, to be able to perform efficient and effective analysis, including formal verification and optimization, whenever necessary.

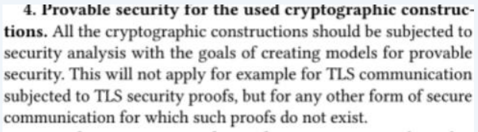
Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)



Providing a set of mitigation strategies focusing on a complex adaptive environment. To enable extensions of safety work towards cyber-security, we need to have knowledge about countermeasures that are effective in preventing exploitation of vulnerabilities that might lead to already identified or completely new hazards.

Towards a Framework for Safe and Secure Adaptive Collaborative: 4 - 4 (0)

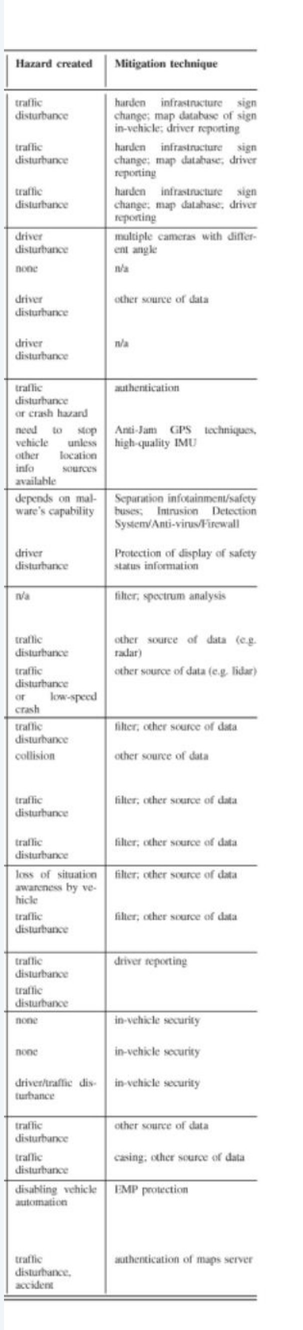
Challenges\Open\Other



4. Provable security for the used cryptographic constructions. All the cryptographic constructions should be subjected to security analysis with the goals of creating models for provable security. This will not apply for example for TLS communication subjected to TLS security proofs, but for any other form of secure communication for which such proofs do not exist.

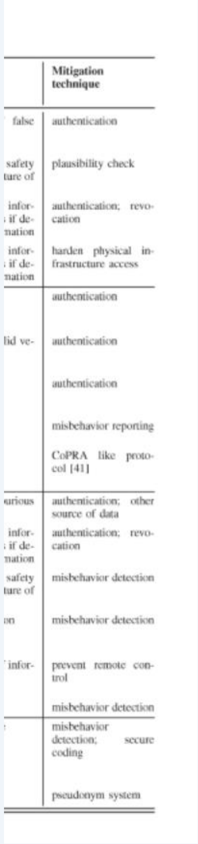
Network and system level security in connected vehicle applicat: 3 - 3 (0)

Treatment



Hazard created	Mitigation technique
traffic disturbance	harden infrastructure sign change; map database of sign in-vehicle; driver reporting
traffic disturbance	harden infrastructure sign change; map database; driver reporting
traffic disturbance	harden infrastructure sign change; map database; driver reporting
driver disturbance	multiple cameras with different angle
none	n/a
driver disturbance	other source of data
driver disturbance	n/a
traffic disturbance or crash hazard	authentication
need to stop vehicle unless other location info sources available	Anti-Jam GPS techniques, high-quality IMU
depends on malware's capability	Separation infotainment/safety buses; Intrusion Detection System/Anti-virus/Firewall
driver disturbance	Protection of display of safety status information
n/a	filter; spectrum analysis
traffic disturbance	other source of data (e.g. radar)
traffic disturbance or low-speed crash	other source of data (e.g. lidar)
traffic disturbance	filter; other source of data
collision	other source of data
traffic disturbance	filter; other source of data
traffic disturbance	filter; other source of data
loss of situation awareness by vehicle	filter; other source of data
traffic disturbance	filter; other source of data
traffic disturbance	driver reporting
traffic disturbance	
none	in-vehicle security
none	in-vehicle security
driver/traffic disturbance	in-vehicle security
traffic disturbance	other source of data
traffic disturbance	casing; other source of data
disabling vehicle automation	EMP protection
traffic disturbance, accident	authentication of maps server

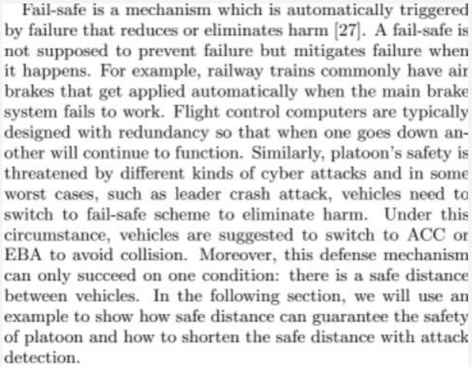
Potential cyberattacks on automated vehicles: 5: 422|45 - 5: 575|724 (0)



	Mitigation technique
false	authentication
safety ture of	plausibility check
infor- if de- nation	authentication; revocation
infor- if de- nation	harden physical infrastructure access
	authentication
lid ve-	authentication
	authentication
	misbehavior reporting
	CoPRA like protocol [41]
urious	authentication; other source of data
infor- if de- nation	authentication; revocation
safety ture of	misbehavior detection
on	misbehavior detection
infor-	prevent remote control
	misbehavior detection
	misbehavior detection; secure coding
	pseudonym system

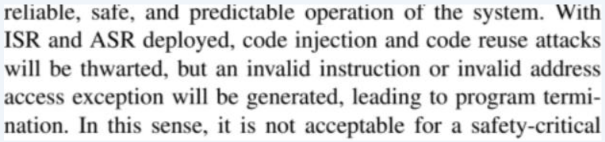
Potential cyberattacks on automated vehicles: 8: 469|299 - 8: 568|728 (0)

Treatment\Hazard Elimination



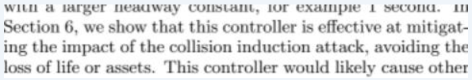
Fail-safe is a mechanism which is automatically triggered by failure that reduces or eliminates harm [27]. A fail-safe is not supposed to prevent failure but mitigates failure when it happens. For example, railway trains commonly have air brakes that get applied automatically when the main brake system fails to work. Flight control computers are typically designed with redundancy so that when one goes down another will continue to function. Similarly, platoon's safety is threatened by different kinds of cyber attacks and in some worst cases, such as leader crash attack, vehicles need to switch to fail-safe scheme to eliminate harm. Under this circumstance, vehicles are suggested to switch to ACC or EBA to avoid collision. Moreover, this defense mechanism can only succeed on one condition: there is a safe distance between vehicles. In the following section, we will use an example to show how safe distance can guarantee the safety of platoon and how to shorten the safe distance with attack detection.

A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)



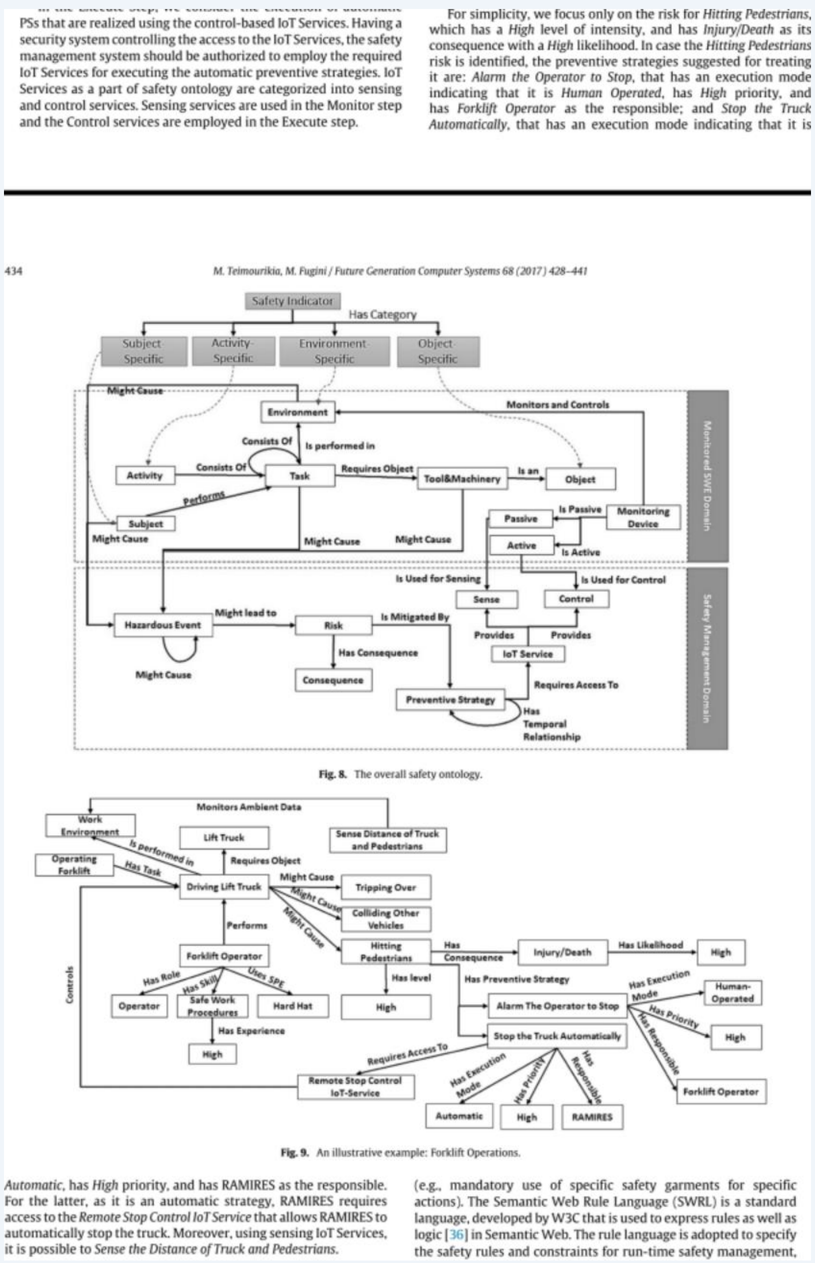
reliable, safe, and predictable operation of the system. With ISR and ASR deployed, code injection and code reuse attacks will be thwarted, but an invalid instruction or invalid address access exception will be generated, leading to program termination. In this sense, it is not acceptable for a safety-critical

Integrated moving target defense and control reconfiguration fo: 2 - 2 (0)



with a larger headway constant, for example 1 second. In Section 6, we show that this controller is effective at mitigating the impact of the collision induction attack, avoiding the loss of life or assets. This controller would likely cause other

Is your commute driving you crazy a study of misbehavior in veh: 8 - 8 (0)



In the Execute step, we consider the execution of automatic PSs that are realized using the control-based IoT Services. Having a security system controlling the access to the IoT Services, the safety management system should be authorized to employ the required IoT Services for executing the automatic preventive strategies. IoT Services as a part of safety ontology are categorized into sensing and control services. Sensing services are used in the Monitor step and the Control services are employed in the Execute step.

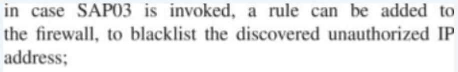
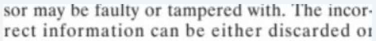
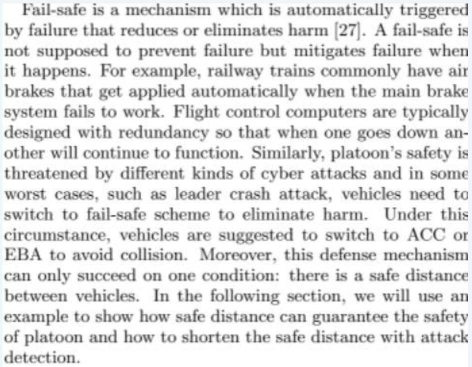
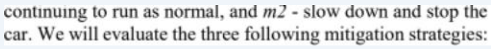
For simplicity, we focus only on the risk for *Hitting Pedestrians*, which has a *High* level of intensity, and has *Injury/Death* as its consequence with a *High* likelihood. In case the *Hitting Pedestrians* risk is identified, the preventive strategies suggested for treating it are: *Alarm the Operator to Stop*, that has an execution mode indicating that it is *Human Operated*, has *High* priority, and has *Forklift Operator* as the responsible; and *Stop the Truck Automatically*, that has an execution mode indicating that it is

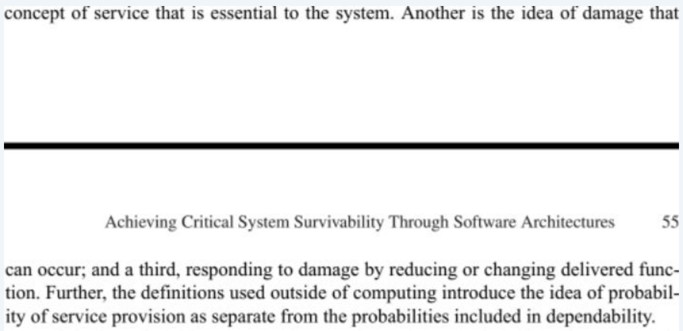
Fig. 8. The overall safety ontology.

Fig. 9. An illustrative example: Forklift Operations.

Automatic, has *High* priority, and has RAMIRES as the responsible. For the latter, as it is an automatic strategy, RAMIRES requires access to the *Remote Stop Control IoT Service* that allows RAMIRES to automatically stop the truck. Moreover, using sensing IoT Services, it is possible to *Sense the Distance of Truck and Pedestrians*.

(e.g., mandatory use of specific safety garments for specific actions). The Semantic Web Rule Language (SWRL) is a standard language, developed by W3C that is used to express rules as well as logic [36] in Semantic Web. The rule language is adopted to specify the safety rules and constraints for run-time safety management,

	Ontology development for run-time safety management methodology: 6 - 7 (0)
	 in case SAP03 is invoked, a rule can be added to the firewall, to blacklist the discovered unauthorized IP address; Protecting cyber physical production systems using anomaly dete: 8 - 8 (0)
	 sor may be faulty or tampered with. The incorrect information can be either discarded or Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)
Treatment\Hazard Reduction	 Fail-safe is a mechanism which is automatically triggered by failure that reduces or eliminates harm [27]. A fail-safe is not supposed to prevent failure but mitigates failure when it happens. For example, railway trains commonly have air brakes that get applied automatically when the main brake system fails to work. Flight control computers are typically designed with redundancy so that when one goes down another will continue to function. Similarly, platoon's safety is threatened by different kinds of cyber attacks and in some worst cases, such as leader crash attack, vehicles need to switch to fail-safe scheme to eliminate harm. Under this circumstance, vehicles are suggested to switch to ACC or EBA to avoid collision. Moreover, this defense mechanism can only succeed on one condition: there is a safe distance between vehicles. In the following section, we will use an example to show how safe distance can guarantee the safety of platoon and how to shorten the safe distance with attack detection. A Functional Co-Design towards Safe and Secure Vehicle Platooni: 6 - 6 (0)
	 continuing to run as normal, and *m2* - slow down and stop the car. We will evaluate the three following mitigation strategies: A simplified approach for dynamic security risk management in c: 7 - 7 (0)

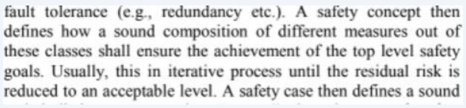


concept of service that is essential to the system. Another is the idea of damage that

Achieving Critical System Survivability Through Software Architectures 55

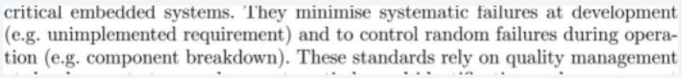
can occur; and a third, responding to damage by reducing or changing delivered function. Further, the definitions used outside of computing introduce the idea of probability of service provision as separate from the probabilities included in dependability.

Achieving critical system survivability through software archit: 4 - 5 (0)



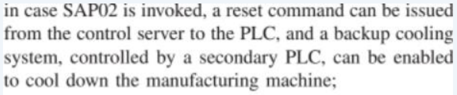
fault tolerance (e.g., redundancy etc.). A safety concept then defines how a sound composition of different measures out of these classes shall ensure the achievement of the top level safety goals. Usually, this in iterative process until the residual risk is reduced to an acceptable level. A safety case then defines a sound

Approaching runtime trust assurance in open adaptive systems: 2 - 2 (0)



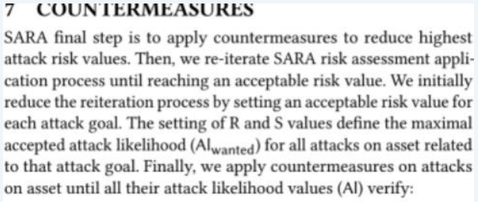
critical embedded systems. They minimise systematic failures at development (e.g. unimplemented requirement) and to control random failures during operation (e.g. component breakdown). These standards rely on quality management

Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



in case SAP02 is invoked, a reset command can be issued from the control server to the PLC, and a backup cooling system, controlled by a secondary PLC, can be enabled to cool down the manufacturing machine;

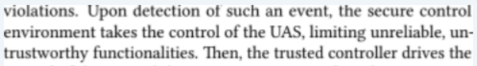
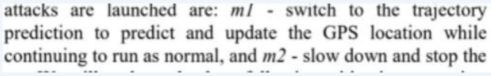
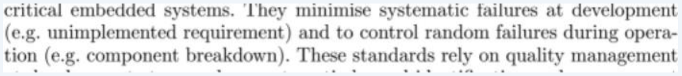
Protecting cyber physical production systems using anomaly dete: 8 - 8 (0)

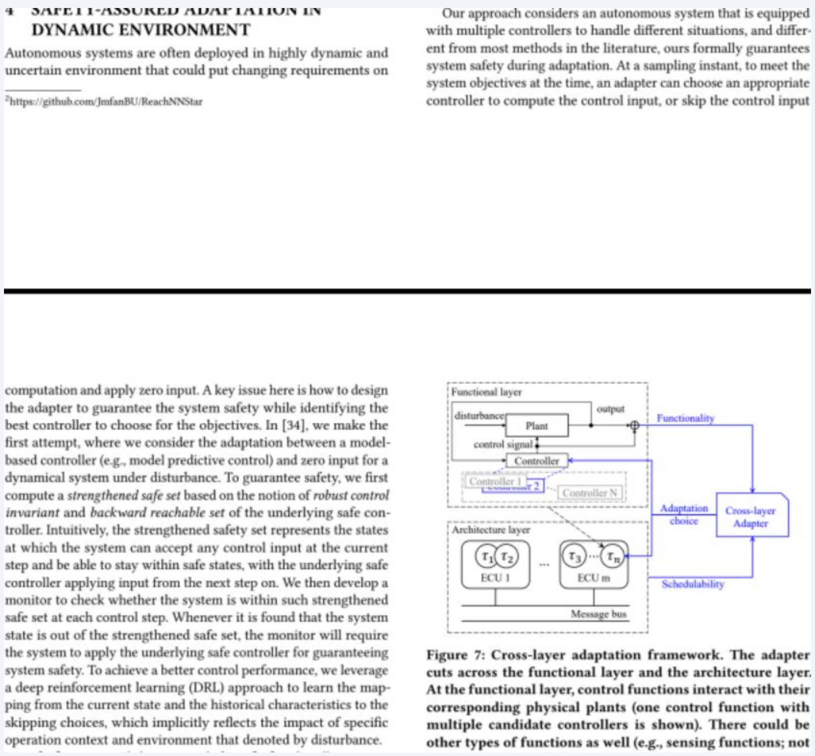


7 COUNTERMEASURES

SARA final step is to apply countermeasures to reduce highest attack risk values. Then, we re-iterate SARA risk assessment application process until reaching an acceptable risk value. We initially reduce the reiteration process by setting an acceptable risk value for each attack goal. The setting of R and S values define the maximal accepted attack likelihood (Al_{wanted}) for all attacks on asset related to that attack goal. Finally, we apply countermeasures on attacks on asset until all their attack likelihood values (Al) verify:

SARA Security Automotive Risk Analysis Method: 10 - 10 (0)

	 violations. Upon detection of such an event, the secure control environment takes the control of the UAS, limiting unreliable, untrustworthy functionalities. Then, the trusted controller drives the VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)
Treatment\Hazard Control	 attacks are launched are: *m1* - switch to the trajectory prediction to predict and update the GPS location while continuing to run as normal, and *m2* - slow down and stop the A simplified approach for dynamic security risk management in c: 7 - 7 (0)
	 critical embedded systems. They minimise systematic failures at development (e.g. unimplemented requirement) and to control random failures during operation (e.g. component breakdown). These standards rely on quality management Digital Twins for Dependability Improvement of Autonomous Drivi: 11 - 11 (0)



4 SAFETY-ASSURED ADAPTATION IN DYNAMIC ENVIRONMENT

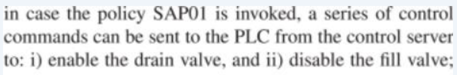
Autonomous systems are often deployed in highly dynamic and uncertain environment that could put changing requirements on

[2]https://github.com/JmfanBU/ReachNNStar

Our approach considers an autonomous system that is equipped with multiple controllers to handle different situations, and different from most methods in the literature, ours formally guarantees system safety during adaptation. At a sampling instant, to meet the system objectives at the time, an adapter can choose an appropriate controller to compute the control input, or skip the control input computation and apply zero input. A key issue here is how to design the adapter to guarantee the system safety while identifying the best controller to choose for the objectives. In [34], we make the first attempt, where we consider the adaptation between a model-based controller (e.g., model predictive control) and zero input for a dynamical system under disturbance. To guarantee safety, we first compute a *strengthened safe set* based on the notion of *robust control invariant* and *backward reachable set* of the underlying safe controller. Intuitively, the strengthened safety set represents the states at which the system can accept any control input at the current step and be able to stay within safe states, with the underlying safe controller applying input from the next step on. We then develop a monitor to check whether the system is within such strengthened safe set at each control step. Whenever it is found that the system state is out of the strengthened safe set, the monitor will require the system to apply the underlying safe controller for guaranteeing system safety. To achieve a better control performance, we leverage a deep reinforcement learning (DRL) approach to learn the mapping from the current state and the historical characteristics to the skipping choices, which implicitly reflects the impact of specific operation context and environment that denoted by disturbance.

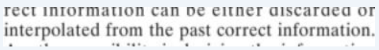
Figure 7: Cross-layer adaptation framework. The adapter cuts across the functional layer and the architecture layer. At the functional layer, control functions interact with their corresponding physical plants (one control function with multiple candidate controllers is shown). There could be other types of functions as well (e.g., sensing functions; not

Know the unknowns addressing disturbances and uncertainties in: 7 - 8 (0)



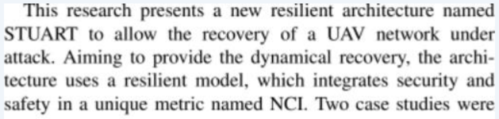
in case the policy SAP01 is invoked, a series of control commands can be sent to the PLC from the control server to: i) enable the drain valve, and ii) disable the fill valve;

Protecting cyber physical production systems using anomaly dete: 8 - 8 (0)



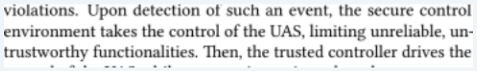
rect information can be either discarded or interpolated from the past correct information.

Security vulnerabilities of connected vehicle streams and their: 6 - 6 (0)



This research presents a new resilient architecture named STUART to allow the recovery of a UAV network under attack. Aiming to provide the dynamical recovery, the architecture uses a resilient model, which integrates security and safety in a unique metric named NCI. Two case studies were

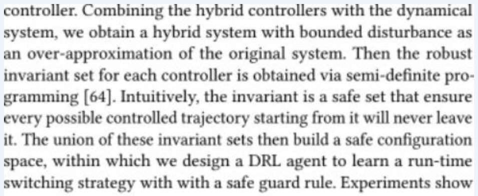
STUART ReSilient archiTecture to dynamically manage Unmanned ae: 6 - 6 (0)



violations. Upon detection of such an event, the secure control environment takes the control of the UAS, limiting unreliable, untrustworthy functionalities. Then, the trusted controller drives the

VirtualDrone virtual sensing actuation and communication for at: 1 - 1 (0)

Other



controller. Combining the hybrid controllers with the dynamical system, we obtain a hybrid system with bounded disturbance as an over-approximation of the original system. Then the robust invariant set for each controller is obtained via semi-definite programming [64]. Intuitively, the invariant is a safe set that ensure every possible controlled trajectory starting from it will never leave it. The union of these invariant sets then build a safe configuration space, within which we design a DRL agent to learn a run-time switching strategy with with a safe guard rule. Experiments show

Know the unknowns addressing disturbances and uncertainties in: 8 - 8 (0)