A Model for Automated Cybersecurity Threat Remediation and Sharing

Francesco Settanni, Leonardo Regano, Cataldo Basile, and Antonio Lioy
Dipartimento di Automatica e Informatica
Politecnico di Torino
Torino, Italy
name.surname@polito.it

***Abstract*—This paper presents an approach to the automatic remediation of threats reported by Cyber Threat Intelligence. Remediation strategies, named Recipes, are expressed in a close-to-natural language for easy validation. Thanks to the developed models, they are interpreted, contextualized, and then translated into CACAO Security playbooks, a standard format ready for automatic enforcement, without human intervention. The presented approach also allows sharing of remediation procedures on threat-sharing platforms (e.g. MISP) which improves the overall security posture. The effectiveness of the approach has been tested in the context of two EC-funded projects.**

***Index Terms*—Network Functions Virtualization, Automated Risk Remediation, Threat Sharing**

I. Introduction

The threat landscape has evolved over time, and today's threats are far more sophisticated and complex than previously seen. The number of new threats in the wild is increasing as never before. Commoditized exploits expand the base of actors able to use complex techniques and tools to perform highly targeted attacks. Organizations are quickly transitioning to full-fledged distributed environments, driven by the advent of flexible networking paradigms, like cloud computing and software networks. While offering many management and cost advantages, this transition expands the attack surface.

In the meanwhile, research has improved the accuracy of monitoring systems. Threat Intelligence (TI) tools, which are outside the scope of this paper, monitor the information system and use advanced AI techniques to detect threats. Discovered threats are passed as security bulletins, named TI Reports (TIRs), broadcast to security operators or tools devoted to mitigating the risks associated with detected threats. Threats could be constantly monitored, and TI could be used for establishing security prevention and detection measures, thus adopting a proactive stance.

However, this information cannot be properly and quickly consumed because tool support currently lacks [1]. Having highly skilled security operators handling cybersecurity risks and TIRs within organizations is imperative. Still, the industry is plagued by a persistent personnel shortage which hinders efforts to improve and advance all aspects related to cybersecurity management [2], [3].

The need for automation in all IT security management is underscored from these premises. Many daily security operations, such as alert detection, triage, and incident response, can be synthesized as a set of procedures, consisting of repetitive actions that, once validated and approved, could be easily automated, thus freeing security operators from specific duties and guaranteeing human supervision for those operations in which liability is required.

This research aims to design a Remediation Module (ReM), a tool that recommends remediation Recipes, which are sequences of actions to mitigate risks from network-centred threats discovered by TI. The actions include:

- information gathering (e.g., discovering the security controls in the operation landscape, exploring network connectivity);
- modifying the operational environment (e.g., adding new security controls to shield threats, changing the routes to force inspection of malicious traffic); and
- changing (network) security policies (e.g., adding new filtering rules to allow or block attackers' packets).

Recipes and their actions are represented in an abstract domain-specific language for reuse in different contexts and operational environments. They use a human-readable syntax to be easily validated and integrated into low-code settings while, at the same time, retaining consistency with commonly used security workflow constructs and unambiguity. The ReM tool interprets TIRs and proposes recipes based on their estimated effectiveness. Once selected, automatically or after explicit approval by security administrators, abstract recipes are instantiated on the operational environment as a workflow of concrete security actions. Both abstract recipes and concrete workflows are suitable for sharing, enhancing the automation of threat response with respect to classical TI feeds, often limited to basic low-level artifacts to be analysed.

To this aim, the contributions of this paper are:

- a *meta-model* for specifying Recipes and the abstract actions they entail,
- a *playbook model* to represent actionable remediation workflows and actions extending the taxonomy of the CACAO Security Playbook standard,
- an *Interpreter* that refines the abstract Recipes and produces actionable playbooks based on the operational environment (in our case a network), and
- a *Deployment engine* that actually applies the concrete actions according to the playbook, enforcing new policies

and security features, or modifying the landscape,
- then we expand the MISP objects data model with the *Remediation* object, allowing for the sharing of actionable remediation procedures through the MISP platform.

The remainder of this work is organized as follows. In Section II, we sketch the landscape of threat remediation and sharing, illustrating some related works. Section III describes formal models behind the remediations and Section IV shows our approach for sharing actionable playbooks. Section V describes the preliminary evaluation. Finally, in Section VI, we draw conclusions and highlight future works.

II. Background

Our work relies on parallel efforts emerging both at the standardization and research level on collaborative cyber threat intelligence sharing. The following subsections will expand upon the concepts on which our research results are based.

A. Cyber Threat Intelligence Sharing

One of the leading efforts on CTI sharing is the MISP threat-sharing platform [4]. The project's main objective is fostering and improving the way in which cyber threat intelligence is used throughout security activities, from threat hunting, vulnerability and risk assessments to incident handling.

MISP serves as a hub for gathering and sharing of CTI data, and its holistic approach includes automated data processing throughout the entire lifecycle, from creation to utilization for activities like threat detection, mitigation, and correlation.

Adopting common data schemas is crucial for the successful implementation of CTI sharing. To this end, MISP provides a flexible data schema for representing, categorizing, and contextualizing information stored and shared through the platform. It features multiple data models: *Core Format*, *Objects*, *Taxonomies*, and *Galaxies*. The last two can be used to aggregate and model diverse data, for example, based on national or internal company classifications. The Core Format defines schemas for events and attributes, for representing individual threat related data points. MISP Objects expand on this by allowing the representation of complex relationships between attributes, providing a detailed and structured description of CTI information. The data models are flexible since they can be expanded with external templates.

B. Remediation procedures

Efforts related to the sharing of threat remediation procedures, up until now, remained particularly fragmented. This is due to the lack of commonly agreed formats and schemas both for the sharing and for the actual actionability of the information shared. Big vendors compete on closed solutions in the field of incident response and have no incentive to stick to a commonly agreed approach. Many smaller-scale solutions are beginning to appear, pushing for standardised methods for the sharing and usage of such information.

Various proposals have been made for effective ways to document, process, share, and operationalize security pipelines and incident response. One of those approaches is *Playbooks* [5]. As Koulouris et al. state [6], playbooks provide a systematic way to formalize incident response strategies, removing the guesswork from even seasoned analysts. They enable strategies to be documented, communicated and shared between teams and individual analysts in an organized fashion, facilitating better evaluation (and development) of techniques and more effective knowledge transfer. Shaked et al. proposed a formal, model-based approach for the design of cybersecurity incident response playbooks [7] and provided a tool prototype demonstrating how it can be leveraged to model and examine security playbooks. The CACAO Security Playbook standard is one such step in the right direction. In our view, although some improvements are still required, it is one of those formats suitable to the aforementioned requirements. At this end, Mavroeidis et al. proposed a MISP object template to enable the sharing of security playbooks which directly maps to the meta-data of the CACAO playbook template [8].

C. Modelling security controls features

This work relies on *Security Capability Model (SCM)*, a formal model of the features security controls provide for helping enforce security policies proposed by Basile et al. [9]. This model, which expands the work of the IETF I2NSF WG, aims to provide a standard way to describe security controls based on their abstract security capabilities (e.g., the conditions and actions they provide for writing configurations regardless of the language syntax). Starting from the security capabilities associated with security controls, it is possible to generate their abstract configuration languages. Then, SCM also provides means to translate policies written in the abstract configuration language into the actual configuration settings.

The SCM allows the specification of security policies independently from the target security control languages and at the same time guarantees the possibility to generate ready-to-deploy configurations. Therefore, it has been leveraged in the remediation framework proposed in this paper.

III. Remediation procedures

The workflow in Fig. 1 summarizes the approach presented in this paper. It starts from an abstract representation of a remediation strategy, the *Recipe*, which is independent of the operational environment where it will be applied. Recipes are easy to read and understand for easy validation and sharing. To be enforced, a Recipe must be transformed into a representation usable for remediation enforcement. This task is performed in two steps. First, an *Interpreter* refines a Recipe into a *Security playbook*, a workflow of machine-readable remediation instructions written according to a playbook model extending CACAO. Then, a *Deployment Engine (DE)* refines the Security playbook into a set of *Enforcement instructions*. These instructions include the configurations for the security controls available in the operational environment. When the operational environment does not own all the controls for the proper enforcement of the remediation, the instructions include directives to change the network topology, e.g., by adding

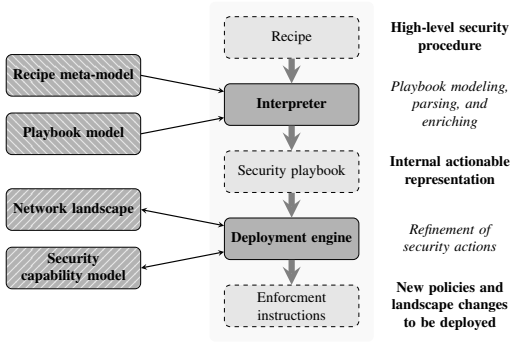

Fig. 1. Remediation workflow

```
list_paths from impacted_host to 'attacker'
iterate_on path_list
find_node of type 'firewall' in iteration_element with
     'level_4_filtering'
  if not found
    add_firewall behind impacted_host in iteration_element
         with 'level_4_filtering'
    add_filtering_rules rules_level_4 to new_node
    allow_traffic between impacted_host and
         'investigation_host' at new_node
  else
    add_filtering_rules rules_level_4 to found_node
    ...
  end_if
end_iteration
```

Fig. 2. A recipe for enforcing packet filtering policies

a new security control NSF or moving nodes in different positions. To this end, the *DE* leverages the description of the operational environment from a network perspective, the *Network Landscape (NL)*, which in our use cases is a graph-based representation of an SDN+NFV software network. Moreover, the *DE* accesses information about the Security capabilities available in *NL*, that is, the security features enforceable with the security controls available, formally specified according to the *SCM* (see Section II).

The persistence of the data is ensured by a *Knowledge Base (KB)* that stores the available high-level remediations in the form of Recipes, which security practitioners can prepare beforehand, directly actionable security playbooks, and enforcement instructions. The *KB* is structured to allow inferring behaviours and appropriate response strategies for a given threat. For this reason it also stores information about threats, attacks, and other CTI, which can originate from internal threat-hunting activities or, more importantly, from threat intelligence feeds.

A. Recipes and their meta-model

Remediation procedures are strategic information that can conceptually be placed at the highest levels of the cyber threat intelligence pyramid of pain since they increase the operability of shared CTI by enabling automated exploitation of it. Recipes, used to represent remediation strategies, are described using a domain-specific language (DSL), according to the *Recipe meta-model* grammar.

```
Recipe:
   statements*=Statement;
Statement:
   (SecurityAction | ConditionFlowConstruct |
   IterationFlowConstruct | ... );
IterationFlowConstruct:
   'iterate_on' iterationExpression=VarReference
      statements+=Statement
   'end_iteration';
SecurityAction:
   (ListPaths | FindNode | AddFirewall | AddFilteringRules |
   AllowTraffic | AddDnsPolicy | AddHoneypot | MoveNode |
   AddNetworkMonitor | Execute | Shutdown | Isolate | ... );
AllowTraffic:
   'allow_traffic'
   (('between' firstNodeExpression=VarReferenceOrString)
   ('and' secondNodeExpression=VarReferenceOrString)
   ('at' firewallNodeExpression=VarReferenceOrString));
```

Fig. 3. Recipe meta-model grammar excerpt

The Recipe meta-model adopts a semantic scheme composed of flow constructs and *Statements* expressing a remediation action (see Fig. 3). Each Statement is declared using an Action identifier, its unique name, and a set of arguments. Arguments enrich actions with context instructions like static values (e.g. `'level_4_capability'`), or variables whose value will be concretized by the Interpreter.

For example, the `firewallNodeExpression` variable indicates the identifier of the firewall, among the ones in the network, that will enforce the required policy. Furthermore, an action to determine if a specific network device is present in a network path will yield a Boolean variable named `found`. Variables do not have a strict type, but they can be inferred from the name and possibly from the context to which they belong. More structured values can also be handled but should be declared outside of the Recipe, directly into the memory store, which we chose to avoid jamming the lightweight syntax of Recipes with heavy declaration blocks.

Each action is associated with a function invoked to execute a job by interfacing with certain ReM engine capabilities. In the previous example, the function gathers information regarding the network topology by interfacing with the *NL*.

Some of the action arguments are mandatory for executing the action function, they are named *enabling constraints*. For instance, it is impossible to protect a victim without knowing its IP address or any reference. Together with the enabling constraints, other arguments are used to make decisions on the most appropriate Recipe for a specific threat.

The DSL has been developed leveraging the *textx* meta-language and the Arpeggio PEG parser with 'no grammar ambiguities' and 'unlimited lookahead' features [10]. The minimal syntax allows for expressing remediation workflows as close as possible to natural language. This is a valuable feature as an easily readable mechanism to document incident response measures may shorten the time frames when escalation between security levels is needed.

The meta-model aligns with existing security playbook description formats, as in the end, Recipes will be translated into security playbooks that are both machine-readable and

actionable. For instance, providing a way to define sequences of steps in a security procedure through flow constructs, implementing branching logic aligned with the CACAO Security Playbook specification. Moreover, it also provides a new declarative pattern for iteration, a "for each" construct that applies a given list of functions or security operations to each item of the collection passed to it.

The Recipe meta-model has been completed with predefined security actions, which emerged in our network and host-based threats use cases. These actions have been designed to be abstract enough to be deployed in multiple operational environments as they are irrespective of the vendor-specific security solutions adopted. They range from those typical of SDN environments, such as network segmentation, blocking links, monitoring, quarantining, or isolating hosts [11] [6], to features and orchestration capabilities of NFV infrastructures, like instantiating new NSF security controls to dynamically scale the available security features based on the current security needs. New security actions can be easily declared into the Recipe meta-model. The only requirement is the definition of corresponding functions that implement the needed behaviour when the deriving playbook is deployed.

Fig. 2 presents a Recipe that mitigates the risks from an infected host, recognised by its IP `impacted_host_IP`, as reported by a Threat Intelligence report. The actions forbid the infected host to communicate with the attacker host (i.e., the C2 of a botnet) and specify that if there are no packet filters to block this communication, a new one must be added before the victim. At the same time, it states that the defenders' host must reach the infected machine for further investigation (`investigation_host`), as the current policy or the new packet filter may prevent them from reaching it.

B. Security playbooks

While Recipes are better abstract, as they need to be written and understood by humans, our security playbooks are a machine-readable way to share remediation strategies and deployment constraints. This approach aligns with the Threat Intelligence sharing best practices, emphasising the importance of sharing higher-level information alongside raw artifacts.

The playbooks used in our work are represented according to a playbook model defined in compliance with the CACAO v1.0 for interoperability. Our playbook model extends the CACAO "Action" functionality to confer complete actionability. Indeed, as Shaked et al. report in their analysis [7], as of now, the CACAO standard falls short of providing consistent structural atomicity for actions comprising the workflow steps, thus making it difficult for different engines to interpret the workflow and deploy the playbook. Hence, to ensure actionability, each workflow step is associated with two new attributes that enrich the CACAO open vocabularies.

The first attribute is used for the Target type (target-type-ov), called `recipe-deployment-engine`, which indicates the actuator to be used for a given command or action of the workflow, the second one to the Command type (command-type-ov), called `recipe-security-action`, which identifies the Recipe specification for the individual commands. To comply with the CACAO standard normative, an Extension Definition accompanies the playbook, with a correct specification of our schema. Finally each element of the playbook employing it will simply reference the extension by its ID.

C. Translation and refinement

The *Interpreter* is tasked with generating a playbook which complies with the Recipe coming as input according to the Recipe meta-model and playbook model (see Section III-B). The interpreter gathers data to contextualize Recipes from the internal *KB* (e.g., victim and attacker data), then use them to generate the playbook. To this end, the collected data are organized semantically by means of an object-oriented model that enriches the CACAO model specification where each security action and flow construct inherits from generic Action and Flow classes. For instance in the Recipe shown before, Fig. 2, the action (`list_paths from impacted_host to 'attacker'`) has only one enabling constraint, the impacted host (`impacted_host`).

The task of generating ready-to-enforce configurations is performed by the *DE*, which is capable of interpreting Playbooks and complementing them with the information from the target operational environment, gathered from its formal representation, to make remediations practical. The *DE* completes the information of each workflow step as per the concepts defined in the Recipe meta-model and then it executes the functions associated with each action.

Indeed, how the *DE* carries out a given action in the Playbook depends on the action itself and on its function. Some actions may simply query the abstract representation of the operational environment, looking for edges and nodes (e.g., identifying attackers and victims in the network layout and the paths between them), checking the security features owned by the security controls, and also instantiating other state variables used by the next instructions. In other cases, it may run a branching flow, explore the network graph connectivity, or enforce a given high-level policy.

The *DE* exploits the action arguments, which are dynamically typed. Thanks to this feature, playbook behaviour can be directly specified working on the action class and the *DE* can understand argument semantics, e.g., it can match the correct host regardless of whether the specifier is an IP address, a DNS hostname, or some other accepted univocal identifier. We opted for maximum flexibility at the playbook level, as the CACAO standard is currently lacking a comprehensive normative coverage for artifact handling and specification as highlighted in past survey [1].

When playbooks provide instructions to configure security controls, these are not specified using their actual configuration languages. Configuration instructions are represented using abstract configuration languages specified according to the *Security Capability Model*. Therefore, the *DE* exploits the translation abilities of the *SCM* for refining abstract configurations into actual configurations for the target security controls.

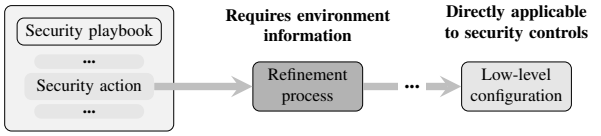

Fig. 4. Security action refinement

```
"security-action": {
    "description": "The security action used in the
        playbook conforming to the Recipe format",
    "disable_correlation": false,
    "misp-attribute": "text",
    "multiple": true,
    ...             }
```

Fig. 5. Excerpt of an attribute of the MISP Remediation object template

Moreover, the *DE* accesses the catalogue of VNFs and NSFs. This catalogue is helpful to identify security controls that can be deployed on demand on the *NL* when the interpretation of the playbooks highlights that some features needed for the remediations are missing. Hence, the result of some of these instructions may produce commands for the virtualized network environment (add nodes, move nodes) or abstract representations of the rules that need to be enforced by the selected security controls, as depicted in Fig. 4.

IV. Remediation sharing as CTI

Remediation recipes and deriving security playbooks open up the possibility of sharing threat remediation procedures and CTI data, conferring actionability alongside them. However, sharing is only possible if there are standard exchange modalities and formats, and this format is expressive enough to allow actionability. The next sections present an analysis of the state of the art and the solution adopted in our work.

A. Sharing actionable remediations through MISP

To ensure a strong security posture is imperative to integrate CTI into the security management pipelines, including threat response. However, the sharing of CTI is plagued by inconsistencies and a lack of quality data, and is often limited to low-level artifacts, such as IoCs. Actionability of shared CTI is one fundamental aspect that has been missing, and remediation sharing in the form of playbooks is one of the efforts in this direction. In this sense, the MISP platform can be leveraged to implement and integrate missing features.

The MISP data models currently provide multiple ways to enable the sharing of such information. Hence, we analysed them to verify the possibility of properly using them for sharing remediations in the form of security playbooks.

Two MISP objects are the most suitable for sharing security procedures. The first, named *Course of action*, comes with a limited set of parameters reported in Table I, which can be used to attach some high-level metrics and labels to a text-based description of the security procedure. From a thorough assessment of the two objects, the *Course of Action* appeared inappropriate for our needs as it is too generic, is not intended to specify structured procedures, and is unsuitable for carrying specific formats, e.g., JSON for the CACAO standard.

The second, named *Security playbook* [8], has been only recently added and accurately mirrors the CACAO standard metadata specification. It comes with several fields that specify temporal information related to the playbook (see Table I), such as the validity, creation, and modification time. Moreover, it also provides two fields to embed the actual playbook in its original format as a Base64 encoded string or as a MISP attachment attribute (i.e., `playbook-base64` and `playboook-file`).

B. Adding the MISP Remediation object

The MISP *Security playbook* object perfectly matches the schema of the playbooks we intend to use. However, our playbooks conforming to (and extending) the CACAO standard come with additional requirements to provide complete actionability. For instance, the enforcement of security policies, configurations, and landscape changes is bound to a particular *Security action*. This behaviour is currently missing from the CACAO standard, which simply permits the declaration of command types. Consequently, such parameters also miss in the MISP Security Playbook. Hence, we proposed a new object template for the MISP data model named *Remediation*, extending the *Security Playbook* object, to include a set of attributes that allow the sharing of more precise information and correlation of different types of security procedures.

The attributes in this object can be used to enrich the MISP object envelope with information from the inner playbook, enabling native event correlations at the MISP level. For instance, the `security-action` attribute, in Fig. 5, reports the playbook actions, while the `enabling-constraints` attribute specifies the minimal required parameters for playbook deployment. Table I presents the format of this object, reporting only some relevant additional fields against the *Security Playbook* object. Another relevant attribute is `security-capability` which reports the security features required by an actuator, also security control, aimed at enforcing the security actions in the playbook, according to *SCM* [9]. Moreover, we allow inserting auxiliary information regarding the playbook, such as the number of steps composing it, expected enforcement time, impacted services, etc. Hence, this new data model facilitates the integration and ingestion of actionable playbook-based remediations into existing security orchestration and sharing workflows.

V. Evaluation

The proposed approach has been evaluated in the context of the Palantir and Fishy EC-funded projects. These projects aim to develop unified cyber resilience frameworks, ensuring security features across heterogeneous and often insecure ICT infrastructures. Combining incident detection, recovery, and knowledge sharing provides Security-as-a-Service (SECaaS) for large enterprises and SMEs/MEs.

Their use cases cover a broad range of realistic scenarios, including a cloud-based infrastructure for autonomous

TABLE I
MISP CANDIDATE OBJECTS' TEMPLATES

Security playbook
description, labels, organization-type, playbook-impact, playbook-base64, playbook-creation-time, playbook-creator, playbook-type, playbook-id, playbook-abstraction, playbook-modification-time, playbook-priority, playbook-severity, playbook-standard, playbook-file, playbook-valid-from, playbook-valid-until, revoke
Course of Action
description, cost, efficacy, impact, name, stage, type
Remediation (some relevant extension attributes)
rem-version, security-actions, enabling-constraints, deployment-parameters, security-capabilities, security-controls

vehicles, an agriculture-based supply chain, a hospital, and industrial control systems. Moreover, to test the integration and new features, both projects have developed synthetic reference networks that are larger and more sophisticated than the implemented test beds.

The projects have selected a set of threats detected by internally developed threat intelligence tools. We have discussed, also with the test bed owners, potential remediation strategies and transformed them into a set of Recipes added to the ReM knowledge base. The Recipes have been presented to and approved by the project experts.

Recipes were shared between the two projects for analogous threats. Despite not being a complete validation of remediation sharing, it proves that Recipes, being abstract and landscape-independent, can be applied in an automated way to different test beds from different projects where threats are discovered with different means.

The validation tested the correctness of the entire ReM workflow. Starting from a knowledge base containing several recipes, the ReM received TIRs broadcast through brokers (RabbitMQ and Kafka) and then orchestrated the entire incident handling life cycle in an automated way.

All the remediations were evaluated according to the data in the knowledge base; the ones with satisfied enabling constraints were scored according to an estimated mitigation impact. Once the remediation is selected (in Fishy, this decision is made manually by an operator evaluating a set with best remediations, in Palantir, the decision is automatic), the Recipe is first converted in a Security Playbook that is later refined into configurations and landscape changes. The final CACAO-compliant security playbook is encapsulated in a MISP event as a MISP Remediation object, to be shared as CTI.

Experts have manually validated the effectiveness scores. Moreover, after applying the remediation actions, the test bed owners (manually) checked that the threats were actually mitigated, hence proving the correctness of the transformation process and the effectiveness of the proposed Recipe.

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented a model for describing, enforcing, and sharing remediation procedures. Remediations abstract from the actual infrastructure and available security controls. They can hence be applied to generic operational environments like SDN-enabled software networks. Remediations can also be shared as part of CTI feeds, enhancing the actionability of shared information and promoting the adoption of threat intelligence and response automation in security-sensitive environments. Future research will focus on integrating frequently handled CTI artifacts into the Recipe and playbook model using an ontology, transitioning our KB to a STIX-compliant graph-based taxonomy. This will require extending the ReM's custom formats with tailored custom STIX objects. Moreover, a reasoner that accesses the MITRE ATT&CK framework data can help select (and build) more tailored remediation strategies on a per-alert basis, and also adapt pre-existing solutions using Large Language Models. This work presents difficult challenges, including modelling remediation objectives and when they have been met as well as potentially unwanted behaviours that deviate from the desired course of action.

ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme, under projects PALANTIR (Grant Agreement No. 883335) and FISHY (Grant Agreement No. 952644).

REFERENCES

[1] D. Schlette, M. Caselli, and G. Pernul, "A comparative study on cyber threat intelligence: The security incident response perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2525–2556, 2021.

[2] B. J. Blažič, "The cybersecurity labour shortage in europe: Moving to a new concept for education and training," *Technology in Society*, vol. 67, p. 101769, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0160791X2100244X

[3] (ISC)2, "Cybersecurity workforce study," 2022. [Online]. Available: https://www.isc2.org//-/media/ISC2/Research/2022-WorkForce-Study/ISC2-Cybersecurity-Workforce-Study.ashx

[4] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 2016.

[5] A. Applebaum, S. Johnson, M. Limiero, and M. Smith, "Playbook oriented cyber response," in *2018 National Cyber Summit (NCS)*, 2018, pp. 8–15.

[6] T. Koulouris and M. Casassa-Mont, "Sdn 4 s : Software defined networking for security," 2017.

[7] A. Shaked, Y. Cherdantseva, and P. Burnap, "Model-based incident response playbooks," in *Proceedings of the 17th Int. Conf. on Availability, Reliability and Security*, ser. ARES '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3538969.3538976

[8] V. Mavroeidis, P. Eis, M. Zadnik, M. Caselli, and B. Jordan, "On the integration of course of action playbooks into shareable cyber threat intelligence," in *2021 IEEE Int. Conf. on Big Data (Big Data)*, 2021, pp. 2104–2108.

[9] C. Basile, D. Canavese, L. Regano, I. Pedone, and A. Lioy, "A model of capabilities of network security functions," in *2022 IEEE 8th Int. Conf. on Network Softwarization (NetSoft)*, 2022, pp. 474–479.

[10] I. Dejanovć, R. Vaderna, G. Milosavljević, and Ž. Vuković, "Textx: A python tool for domain-specific languages implementation," *Knowledge-Based Systems*, vol. 115, pp. 1–4, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705116304178

[11] F. Patzer, A. P. Meshram, and M. Hess, "Automated incident response for industrial control systems leveraging software-defined networking," in *Int. Conf. on Information Systems Security and Privacy*, 2019.