ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

TC 11 Briefing Papers

PCaaD: Towards automated determination and exploitation of industrial systems

Benjamin Green[a], Richard Derbyshire[b,*], Marina Krotofil[c], William Knowles[d], Daniel Prince[e], Neeraj Suri[f]

[a] Research Fellow, School of Computing and Communications Lancaster University, UK
[b] PhD Student, School of Computing and Communications Lancaster University, UK
[c] Visiting Researcher, Hamburg University of Technology, Hamburg, Germany
[d] Visiting Researcher, School of Computing and Communications Lancaster University, UK
[e] Senior Lecturer, School of Computing and Communications Lancaster University, UK
[f] Distinguished Professor, School of Computing and Communications Lancaster University, UK

ARTICLE INFO

Article history:
Received 21 February 2021
Revised 27 July 2021
Accepted 29 July 2021
Available online 8 August 2021

Keywords:
PLC Programming Practices
Reconnaissance
Process Comprehension
C2
ICS
SCADA
OT

ABSTRACT

Over the last decade, Programmable Logic Controllers (PLCs) have been increasingly targeted by attackers to obtain control over industrial processes that support critical services. Such targeted attacks typically require detailed knowledge of system-specific attributes, including hardware configurations, adopted protocols, and PLC control-logic, i.e., process comprehension. The consensus from both academics and practitioners suggests stealthy process comprehension obtained from a PLC alone, to execute targeted attacks, is impractical. In contrast, we assert that current PLC programming practices open the door to a new vulnerability class, affording attackers an increased level of process comprehension. To support this, we propose the concept of Process Comprehension at a Distance (PCaaD), as a novel methodological and automatable approach towards the system-agnostic identification of PLC library functions. This leads to the targeted exfiltration of operational data, manipulation of control-logic behavior, and establishment of covert command and control channels through unused memory. We validate PCaaD on widely used PLCs through its practical application.

1. Introduction

Acting as the bridge between physical industrial processes and enterprise systems, Industrial Control Systems (ICSs) deliver wide-spread monitoring, control, and automation capabilities to a broad spectrum of end-users. The Purdue Enterprise Reference Architecture model (PERA) (Williams, 1994) provides an approach to compartmentalize the complexity of ICSs into hierarchical layers. Each layer affords system users with access to industrial processes and the data they generate. The lower the layer, the closer associated devices are to the processes they oversee, with Programmable Logic Controllers (PLCs) providing a primary interface to op-

* Corresponding author.
E-mail addresses: b.green2@lancaster.ac.uk (B. Green), r.derbyshire1@lancaster.ac.uk (R. Derbyshire), marina.krotofil@tuhh.de (M. Krotofil), w.knowles@lancaster.ac.uk (W. Knowles), d.prince@lancaster.ac.uk (D. Prince), neeraj.suri@lancaster.ac.uk (N. Suri).

https://doi.org/10.1016/j.cose.2021.102424
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

erational components (pumps and valves) via sensors and actuators.

A number of historical attacks have demonstrated the willingness of attackers to target ICSs (Derbyshire et al., 2018; Miller et al., 2021), with initial access obtained via malicious USBs, project files, software updates, the supply chain, public facing systems, etc. (Falliere et al., 2011; ICS-CERT, 2014; Mirian et al., 2016; Slay and Miller, 2007). However, there still exists a primary challenge, once access is obtained, how can a cyber-physical attack be undertaken, i.e., an attack in which industrial operational process manipulation is achieved. A comprehensive body of existing work details the challenge attackers face in the development of cyber-physical attacks, this is largely focused on obtaining an adequate level of process comprehension. Process comprehension is defined as “the understanding of system characteristics and components responsible for the safe delivery of service” (Green et al., 2017). The described challenges align to a lack of a single resource by which attackers can obtain sufficient process comprehension to conduct a cyber-physical attack. We see this not only in the identification/understanding of physical operational process characteristics (drive controllers, safety doors, proportional-integral-derivative controllers, etc.), but also the interconnectivity and broader configuration parameters (communications interfaces, alerting functions, engineer access, etc.) of devices an attacker may choose to target.

While there exist a number of tools and techniques one can use to develop a level of process comprehension through the targeting of PLCs alone, they are limited by functionality, scope, and detectability (Beresford, 2011; dark lbp, 2020; Nmap, 2020). The holy grail would be to stealthily (avoiding detection) obtain complete Process Comprehension over a network/at a Distance (PCaaD) targeting only PLCs, while simultaneously preventing any disruption to their operation. We assert that current PLC programming practices provide a segue into capability of this kind, and provide validation through the exploration of widely used control-logic (PLC code) library functions, developed by device vendors for use by programmers. This leads to the following five exploitation capabilities: (1) the remote enumeration of control-logic library functions, (2) the exfiltration of operational process data and configuration parameters, (3) the targeted manipulation of control-logic behavior, impacting operational processes and configuration parameters, (4) the establishment of covert command and control (C2) channels through unused memory, and (5) the end-to-end environment-agnostic automation of 1–4.

This paper serves as an significant step in developing PCaaD capability, forming a greater understanding of the role PLC programming practices play in process comprehension techniques, using library functions as an explorative base. Through this, we begin to develop capability aligned to automated environment-agnostic cyber-physical attacks, and build upon an emerging vulnerability class based on control-logic constructs. Therefore, the novel contributions of this work are:

- A stealthy method to enumerate library functions based on memory allocation.
- A targeted approach to data exfiltration and operational process/device configuration manipulation.
- A method allowing for the establishment of a covert C2 channel via unused memory.
- An automated process to enact remote enumeration, exfiltration, exploitation, and covert C2 channel creation.

The remainder of this paper is structured as follows. Section 2 covers related work. Section 3 details a threat model/set of attack vectors. Section 4 provides a background on PLC program structures. Section 5 develops our main contribution of PCaaD, which is subsequently validated in Section 6. Section 7 provides a set of lessons learnt, including a process flow for automated PCaaD and attack execution. Section 8 concludes the paper and offers areas for future work.

2. Related work

Over the last decade, there has been an increasing volume of research targeting the exploitability of embedded systems used in industrial settings. This reflects both the large number of “low-hanging fruit" vulnerabilities, and an increased interest from attackers towards the disruption of industrial processes. To date, research efforts have predominately focused on real-time operating systems, firmware vulnerabilities, industrial protocols, and bypassing traditional security controls (Abbasi et al., 2016; Biham et al., 2019; Drias et al., 2015; Nochvay, 2019; Wardak et al., 2016).

Only a small subset of existing work focuses on controller programming security implications. Kottler et al. (2017) explore the formal verification of ladder logic (a control-logic programming language). Eckhart et al. (2019a) consider security implications within the wider system development lifecycle. While Serhane et al. (2018, 2019) examine coding practices that could cause unsafe conditions, in the majority of discussed practices, an attacker is required to push new control-logic to target systems. In a similar vein, the development of malicious control-logic to cause denial-of-service conditions has also been explored (Govil et al., 2017). More recently, Fluchs (2020) describes an initiative backed by the International Society of Automation to define “The Top 20 Secure PLC Programming Practices”, with a community driven approach to identify additional practices moving forwards. Finally, the work of Ljungkrantz and Akesson (2007) provides an empirical exploration of PLC programming practices using library components. This work showcases the wide spread adoption of libraries and the potential impact of homogeneity in control-logic design. However, it does not consider the cyber security implications of such practices.

Few works have addressed the need for process comprehension from an attacker’s perspective; a critical precondition when seeking to achieve operational impact beyond simple denial-of-service (Gollmann et al., 2015; Green et al., 2017). Research here has focused on the exploitation of configurational practices (Wardak et al., 2016), or wider attack scenarios and taxonomies (Drias et al., 2015).

Research exploring physics-aware attack payloads for industrial processes are also limited (Garcia et al., 2017; McLaughlin and McDaniel, 2012). While some elements of control-logic analysis in these works is done autonomously, payload design still relies on a “human-in-the-loop”. In the

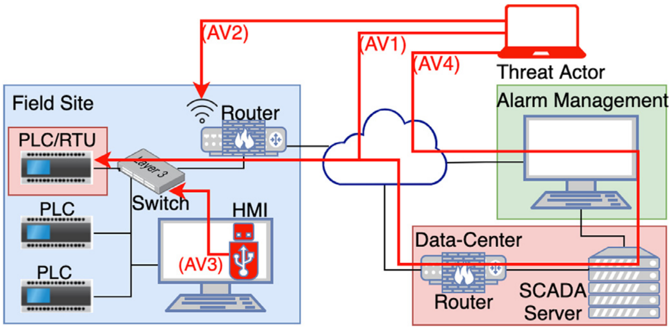


Fig. 1 – Threat Model.

closest work to ours (Dobrushin and Flint, 2019), the authors created an approach to automate the analysis of control-logic and Human Machine Interface (HMI) project files, before building an attack payload. However, the prerequisite in obtaining such files presents an obstacle.

Currently, Stuxnet and Industroyer form the most sophisticated ICS-focused attacks to date (Cherepanov, 2017; Falliere et al., 2011). Stuxnet applied a novel approach to target identification using known characteristics within system data blocks (SDBs), a component of Siemens PLC control-logic. However, this was highly targeted as SDBs are unique to each implementation only providing PLC hardware parameters. While Stuxnet embodied precision, Industroyer manipulated every identified variable on the Remote Terminal/Telemetry Unit (RTU) (set all states to 0), without understanding the targets associated operational processes.

To summarize, the security implications of PLC programming has received limited attention, a critical gap noted by others (Eckhart et al., 2019). Where it exists, there has been no examination of how deployed control-logic could be stealthily enumerated to support process comprehension. We assert current programming practices play a key role in a PLCs exploitability, providing PCaaD without a priori target system knowledge.

3. Threat model

To support discussions throughout the remainder of this paper, the following system under consideration and set of example attack vectors are presented. This offers insight into how PLCs can be targeted by multiple threat actor categories, with varying capabilities and resources (Derbyshire et al., 2020).

3.1. *System under consideration*

Fig. 1 provides an overview of infrastructure architecture frequently found in distributed ICS applications, such as water and energy (Stouffer et al., 2015). Within the Field Site (e.g., a water pumping station) there is a Windows-based HMI (Siemens, 2020b), used by trusted operators to monitor and control physical operational processes via the PLCs (Siemens, 2020a). There are two PLCs used to monitor, control, and automate operational processes. There is also an additional PLC, the PLC/RTU, which performs a similar process automation role but also communicates with a remote Top End System (TES), the Supervisory Control and Data Acquisition (SCADA) Server (COPADATA, 2020). Historically, PLCs only communicated with devices inside the Field Site, with dedicated RTUs forwarding operational data to TESs. However, due to the increased computational resource and connectivity available in modern PLCs, they now act in a dual-purpose capacity, providing RTU capability/interconnectivity with TESs (Gouglidis et al., 2018). There is a network switch, and WiFi router, providing the Field Site with local and remote communications. The remote SCADA Server communicates with the PLC/RTU via its boundary router (in a real-world setting there would be multiple Field Sites communicating with a TES for infrastructure-wide visibility). Finally, the Windows-based Alarm Management Workstation accesses operational data/systems via the SCADA Server.

3.2. *Attack vectors*

Overlaid onto Fig. 1 we have four example attack vectors (AV1-4), each with a set of threat actor and defence profiles.

3.2.1. *Attack vector 1*

Despite growing awareness of cyber security threats to ICSs, ICS devices are being exposed to the Internet without suitable security measures (Mirian et al., 2016; Shodan, 2020).

For this attack vector, we assume the PLC/RTU has been configured on a public IP address for remote SCADA Server access. This allows the threat actor to directly access the PLC/RTU and execute malicious commands with no defensive controls to circumvent. This could be enacted by a low-skilled threat actors.

3.2.2. *Attack vector 2*

For added convenience, the use of wireless technologies is becoming more prevalent in ICSs. Conventional WiFi (802.11) for example, can be used to established connectivity between engineering laptops, mobile HMIs, PLCs, etc (Siemens, 2021a). However if incorrectly configured (i.e. security and transmission range), they can induce additional risk outside of a Field Site's physical perimeter (Slay and Miller, 2007).

For this attack vector, we assume the WiFi access provided by the router is open and insecure. This allows the threat actor to directly access the internal Field Site network and execute malicious commands with no defensive controls to circumvent. This could be enacted by a low-skilled threat actor.

3.2.3. *Attack vector 3*

While Field Sites can be isolated using network-based controls, access can still be obtained through physical means (Falliere et al., 2011). For example, trusted system operators, engineers, via the supply chain (3rd party service providers and pre-infected device introduction), etc. (F-Secure, 2014; ICS-CERT, 2014; Slay and Miller, 2007).

For this attack vector, we assume the connection between the Field Site and Data-Centre is secured based on the use of a VPN and an associated IP address/port rule-set. The rule-set permits the remote SCADA Server to communicate with the PLC/RTU, all other traffic is blocked. In addition, the WiFi access has also been secured with a strong WPA2 key. Here a trusted HMI operator inserts a USB stick containing malicious code into the HMI, which then executes malicious commands autonomously against all devices within the Field Site network via the switch. This could be enacted through the use of a malicious insider, or alternatively a high-skilled threat actor who is able to infect a trusted users USB stick (e.g. via the supply chain).

3.2.4. *Attack vector 4*

Direct access to Field Site devices via networked communications may only be possible through existing trusted systems. Through the initial compromise of trusted systems, and subsequent lateral movement, the desired level of access can be achieved. Furthermore, social engineering is often viewed as a primary initial access technique, and impacts ICS environments in the same way as conventional IT systems (Lee et al., 2014; Liang et al., 2017).

For this attack vector, we assume the connection between the Field Site and Data-Centre is secured based on the use of a VPN and an associated IP address/port rule-set. The rule-set permits the remote SCADA Server to communicate with the PLC/RTU, all other traffic is blocked. In addition, the WiFi access has also been secured with a strong WPA2 key. Here the threat actor compromises the Internet connected Alarm Management Workstation via a malicious email. From this initial access, the threat actor then compromises the SCADA Server, which is used to execute malicious commands against the PLC/RTU.

4. PLC program structures

PLCs are available from a range of vendors, with varying deployments in multiple industrial settings dependent upon operational requirements (Stouffer et al., 2015). They sequentially execute a series of instructions, referred to as a “Program”. However, this paper uses the term *control-logic* to provide a clear distinction during discussion. At a fundamental level, control-logic interfaces with Input/Output (I/O) channels, and based on input states, adjusts output states. Control-logic can provide additional, more complex functionality. This includes establishing configuration parameters for specific network protocols, emailing system users in the event of an operational incident, and connect to remote engineer workstations in the event of a system failure.

The BSI/IEC standard 61131-3:2013 (British Standards Institute, 2013a) outlines five PLC programming languages. These are split into two categories, *Graphical* (Ladder Diagram, Function Block Diagram, and Sequential Function Chart), and *Text Based* (Instruction List, Structured Text). These languages are vendor and application agnostic, although vendor specific language subsets are often provided.

BSI/IEC 61131-3:2013 (British Standards Institute, 2013a) also defines the concept of Program Organization Units (POUs). The following definitions use, and expand, the terminology of BSI/IEC 61131-3 to provide a generalized model, abstracting away from vendor specific terminology, to support subsequent discussions:

- **Programs**: Are the highest level of organizational unit. They control program execution enabling responses to cyclic, time-based, or interrupt-driven events during program execution. They are composed of specific instructions but also Function Blocks and Functions.
- **Function Blocks (FB)**: Contain code that store their values permanently in memory, remaining available post Function Block execution.
- **Functions**: Provide discrete common functionality, for example, ADD or SQRT. Function POUs can use global variables to permanently store data, but do not have their own dedicated memory (local variables).
- **Variable Blocks (VB)**: Store program data and can be global (gVB), or local (fVB). The latter of which are associated with Function Blocks to provide long term data storage. The VB is an addition to the POU model as defined by IEC 61131-3. This standard describes the use of variables in a general sense, with limited ties to their storage.

Control-logic can be written in Program, FB, and Function POUs. Typically, there are additional specialist elements for accessing and addressing other system’s components, including peripherals and timers. However, comprehension of these is not necessary for the current discussion, and are therefore omitted.

A further noteworthy control-logic attribute is the use of pointers in, and between, VBs. This is useful for common information, such as configuration data, or for central recording of operational parameters. For example, consider two FBs. The first processes a water level, translating an I/O channels raw analogue reading into a total water volume. The second is required to access the total water volume for an additional calculation. Instead of writing the total water volume value to the second FBs fVB (copying/duplicating the value), the fVB would contain a pointer to a memory location where the value has been stored by the first FB.

The use of Functions and FBs support code reuse patterns within an organization across multiple deployments (Jacinto, 2017). However, vendors often supply a library of Functions and FBs spanning commonly required functionality, aiding the development of control-logic. These libraries are also referred to as “instruction sets” (Rockwell Automation, 2008). The results of a study into two automotive assembly facilities, identified the repetitive use of libraries across their controller base (Ljungkrantz and Akesson, 2007). This process allows for the managed development/deployment of control-logic, ensuring suitability in one operational zone prior to widespread replication across the remaining estate.

It is worth noting a key difference compared to library Functions in conventional IT software. On the surface, these POUs appear similar to their IT equivalents, providing code reuse. However, they execute sequentially in the control-logic, rather than undertaking complicated execution stack management and sub-routine calls. The implication here is that when a Function or FB is used multiple times, it must be copied into the control-logic multiple times. With FBs a repeated fVB is allocated each time. In this way library Functions and FBs are more like tested and verified code *snippets*, cut and pasted into the control-logic to save time. While their purpose in a given infrastructure will differ, their deployment/construct is identical. For example, a Count-Up FB could be used to count the number of products coming off an assembly line, or the number of times a pump is turned on. The code and context surrounding these FBs can be bespoke and highly varied, but the FBs remain the same. If a PLC programmer in the water industry uses a Siemens provided Count-Up library FB in a PLC, it will be identical in every way to a PLC programmer in a factory also using the same Siemens provided Count-Up library FB. This represents a key concept we exploit with our PCaaD approach.

5. The PCaaD approach

Given the previously described generic operating and programming model applied to PLCs, there exists the possibility of enumerating control-logic by observing VB memory. Where a PLC permits remote access to its memory across the network (e.g. HMI and TES interactions to monitor and control operational processes are done in this way), an avenue is provided for remote extraction of its content. Furthermore, PLCs often include additional network functions allowing for remote interrogation, these have been used in Nmap fingerprinting scripts (Nmap, 2020).

The memory layout of fVBs is consistent across implementations. Through the identification of patterns in the memory layout of a fVB, it becomes possible to identify (enumerate) them and their associated FBs. Once a FB is known, use of the data contained within its fVB can then be interpreted and exploited. Our advocated PCaaD process consists of a two phased approach to enumeration, *Data Retrieval* and *VB Determination*, the completion of which allows for targeted *Exploitation*.

5.1. *Enumeration phase 1: Data retrieval*

Data Retrieval is the first phase of PCaaD, and focuses on retrieving only the necessary information required for subsequent *VB Determination*. This can be applied to each of the four attack vectors described in Section 3, where we state “execute malicious commands".

The data retrieval phase is common in many attack approaches and is often found in wider reconnaissance activities (Assante and Lee, 2015). Typical reconnaissance techniques focus on identifying services running on a given system, and any additional freely available information which may be useful to the attacker. Current PLC reconnaissance tooling is limited, identifying basic parameters such as manufacturer, model, and firmware version (Efanov, 2017; Nmap, 2020).

Through the exploration of a range of PLCs, this work identifies the following three common data retrieval methods which may be used during this stage. Each has implementation specific pros and cons, which are highly dependent on the attacker’s objectives and modus operandi (See Section 6).

- **Metadata:** The majority of vendors provide network functions to query control-logic meta information. These functions do not provide information regarding the current operation of control-logic, but rather information about how it (and the wider PLC) is configured. Within a traditional IT context, this is comparable to querying the manifest of a shared code object (e.g. DLL on Windows or Shared Objects in Linux).
- **Bulk Transfer:** A PLC operating system will often provide a bulk transfer operation, allowing engineers to extract the current state of POUs (gVBs, fVBs, Program, FB, and Functions), supporting diagnostic fault finding, scheduled backups, etc. Within a traditional IT context, this is comparable to a web-server with direct file store access.
- **Memory Address Interrogation:** PLCs provide the ability to remotely interrogate internal memory locations for their current state. This functionality is used to provide operational monitoring and control capability, supporting the retrieval of one or more data items through the specification of memory locations. Within a traditional IT context, this is comparable to SNMP Object requests using a known Management Information Base.

5.2. *Enumeration phase 2: VB determination*

VB Determination, is the process of identifying which fVBs and associated FBs, have been included within a PLCs control-logic, through the analysis of retrieved data from Phase 1. Note that Functions do not have associated VBs and so cannot be identified in this way.

A simple first order approach can be derived from the use of *Metadata* retrieval approaches. As with interrogating shared code object manifests, a fVBs *Metadata* contains attributes one can use in the determination of its associated FB.

Through the use of a *Bulk Transfer*, or a byte wise download, we can obtain fVBs in their entirety. Once obtained, a search for unique attributes (similar to those in *Metadata*), can be conducted to determine its associated FB.

As previously discussed, each fVB has a static memory layout. This contains variables used by the associated FB, and is consistent across all operational and deployment contexts (e.g. water, energy, a testbed). A consistent static memory layout allows for the identification of distinct characteristics forming signatures, to identify fVBs and their corresponding FB using *Memory Address Interrogation*. This concept can be considered similar to rainbow tables, providing a set of pre-computed signatures ready for use during an attack. The following characteristics have been initially selected for fVB signatures:

- **fVB Size:** The fVB size (quantity of allocated bytes) is fixed for all instantiations of the FB across the control-logic base, and is dependant on the number and type of variables used.
- **Known Values:** Some FBs are pre-set with default (and thus known) values for variables in the associated fVB. Here it would be possible to map these known defaults and use them as an indicator of potential fVB match.
- **Variable Usage:** Examination through memory interrogation, to reveal potential data types. For example, if a memory location was only ever set to 0x00 or 0x01, it could indicate a potential boolean data type. Once the data type and memory offset is determined for enough data types, it is possible to map this to a known fVB layout.
- **Data Type Features:** In some PLC hardware architectures it is known that variable allocation is based on defined bit boundaries. Consider an architecture adopting a 16 bit memory boundary. A boolean data type would occupy more than a single bit. The use of numerous boolean variables in the fVB would create a signature of unused memory, which can be used to identify the fVB.

It is worth noting, that for in-house developed FBs (not publicly available library FBs), an attacker would not have an established signature for their associated fVB ahead of an attack - except with the use of other intelligence (i.e. precursory attacks to obtain PLC source code from either the target facility, or a subcontractor where PLC programming is outsourced). However, given the identification of an unmatched VB, it is then possible to generate a signature. When applying PCaaD to other devices within the target infrastructure, this new signature can then be used to identify where the unknown FB is reused.

5.3. *Exploitation based on PCaaD*

Using the aforementioned techniques, an attacker is able to identify which library FBs are included within the broader control-logic base. With this newly acquired information, an attacker is now presented with a range of options to launch an attack. Here we present three attacks, one of which is a storage-based covert C2 channel.

5.3.1. *Attack 1: Exfiltrate FB variables*

This attack extracts data tied to operational process state, and/or PLC configuration, dependant upon the FBs in use. It can be applied to each of the four attack vectors described in Section 3, where we state "execute malicious commands".

It has been asserted that library FBs require well defined fVBs, containing characteristics one can use to develop signatures. Once a FB is known, the associated fVB can be targeted to extract variable states, using a small number of *Memory Address Interrogation* requests.

As discussed in Section 4, VBs may contain pointers to alternate memory locations. Pointers typically have well defined static structures, which can be decoded. Therefore, if a FB is using pointers in its associated fVB, pointing to variables in a gVB, their state can also be retrieved, albeit with an additional *Memory Address Interrogation*. Furthermore, the use of pointers supports process comprehension, as it allows an attacker to identify gVBs that are used by FBs.

5.3.2. *Attack 2: Targeted manipulation of FB operation*

This attack gains fine grained control of FBs, to subvert PLC or operational process behaviors. It can be applied to each of the four attack vectors described in Section 3, where we state "execute malicious commands".

Previously demonstrated attacks either required a priori information on the target PLC (Falliere et al., 2011), adopt brute-force techniques (Robles-Durazno et al., 2019), or focus on denial of service (DoS) impact (Beresford, 2011). For example, in the case of Stuxnet (Falliere et al., 2011) it was widely reported the only way in which this attack was achievable, was through a complete attacker implementation of the target infrastructure based on significant intelligence.

The approach discussed thus far enables PCaaD against an unknown system (no requirement for a priori intelligence or replication of the target infrastructure). An attacker knows how the FB variables are being used, therefore has a greater level of understanding on how they can be manipulated. For example, consider a FB responsible for counting how many litres of water have been treated. To set this value back to 0, the attacker has two options, overwrite the integer representing the total value, or toggle the count reset bit to 1 and then 0.

5.3.3. *Attack 3: A Novel storage based covert channel*

This attack utilizes unused PLC memory, to create a covert channel. It can be applied as a combination of attack vectors 3 and 4 described in Section 3, where we state "execute malicious commands", to create a covert channel between the SCADA Server and the HMI

As previously discussed, the presence of unused memory acts as a key characteristic in the identification of fVBs and their corresponding FBs. However, it can also be used to establish a covert communications channel. Unused memory is present due to alignment with bit boundaries, and is often not considered when processing data in the memory location. For example, an 8 bit value allocated in a 16 bit boundary system, would result in 8 bits of unused memory. This is a concept similar to the use of file slack space for hiding data, and is used in tools such as bmap (Mulazzani et al., 2013).

This approach to covert channel creation, exploits the observation that a PLC is in a trusted position within an operational network. As shown in our scenario (Fig. 1), the PLC/RTU is required to communicate within its local Field Site network (i.e., with the HMI and other PLCs), as well as the data-center network (i.e., the SCADA Server). This means an external party could pivot via the PLC/RTU from the Data-Centre network to the Field Site network. A covert channel of this nature could be used for two primary purposes: C2 and *Data Exfiltration*.

For the C2 channel, an attacker uses the PLC/RTUs unused memory to issue commands from the C2 server (SCADA Server) to the C2 client (HMI). The C2 client periodically checks the unused memory for these commands and executes them accordingly. A channel built on this approach needs to consider the following:

- How the C2 Sever (SCADA Server) and Client (HMI) synchronize on a subset of the unused memory.

- What periodicity for checking and writing to unused memory should be used, as this may be dynamic dependant on network conditions (e.g., round trip times between systems).
- How reliable does the channel scheme need to be, versus the communications overhead of introducing increased robustness. An increased communication overhead could lead to increased detectability.
- The possible commands, how they are encoded, and how results are relayed.

The final point listed here leads onto our second covert channel use: a data exfiltration channel. Once a C2 channel is established it also becomes possible to transfer more than simplistic commands and responses. In much the same way as FTP applications have a control and data channel, a separate channel could be used to bulk transfer larger volumes of data. Adopting the same approach as the C2 channel, this secondary channel can be used to send and receive data. As with the C2 channel there are similar challenges on reliability, resiliency, and speed of communications, versus usability and detectability.

The concept of utilizing unused memory and a PLC as a covert channel for attack, gives rise to PLCs being used as a means for attack, not just the target of an attack. This opens up a new class of security challenge which must be considered when deploying PLCs. Section 6 demonstrates how this approach, and the two prior attacks, can be practically achieved.

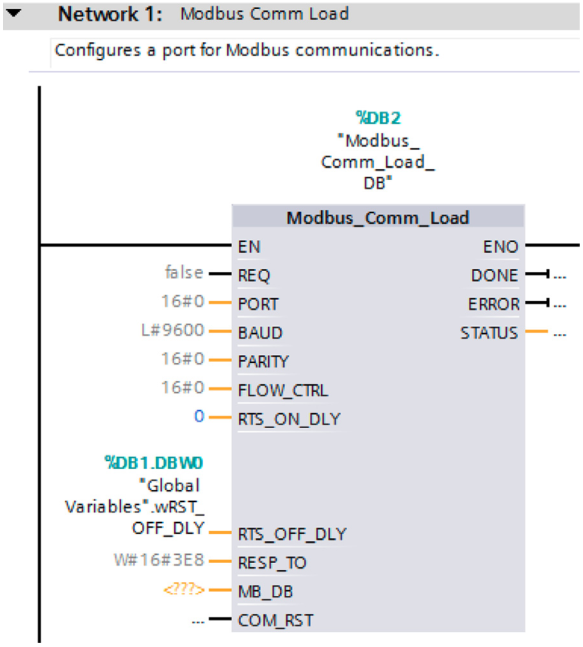
5.4. *Summarizing PCaaD*

This section has introduced the key PLC concepts required to understand the general security problem class of PCaaD. It has also demonstrated the feasibility of PCaaD by exploiting the code reuse patterns of common PLC software libraries. These software libraries provide a commonality across PLC implementations, regardless of operational or deployment context (e.g. water, energy, a testbed).

It is argued that this commonality provides a mechanism to identify FB signatures, which gives rise to a higher level of process comprehension. Memory features are shown to provide an approach by which FB signatures can be identified in a stable and repeatable way.

It is anticipated that machine learning approaches could be used for more advanced fVB identification, with mapping based on the features identified here. In addition, it would be expected that other features of fVB will be identified and used to provide robust signatures. However, using the techniques outlined here, it is possible to perform enumeration of all FBs available from vendor libraries.

Given the level of process comprehension which can be obtained through PCaaD, more sophisticated attacks can be performed. This includes configuration and operational data exfiltration, as well as fine grained variable manipulation. This section also introduced the concept of a storage based covert channel, via a PLCs unused memory. This covert channel opens a new class of security challenge for PLCs, such that they are now not only the target of attack, but also the means by which an attack occurs.


Network 1: Modbus Comm Load

Configures a port for Modbus communications.

Fig. 2 – Modbus Library Function.

6. PCaaD proof of concept

To facilitate practical proof-of-concept exploration, we used a Siemens 300 series PLC [1], and the Siemens TIA v13 platform as a programming agent (Siemens, 2020c). The library functions discussed herein are inbuilt into TIA v13 (Professional). We summarize this section with a note to the described techniques applicability across a broader PLC base..

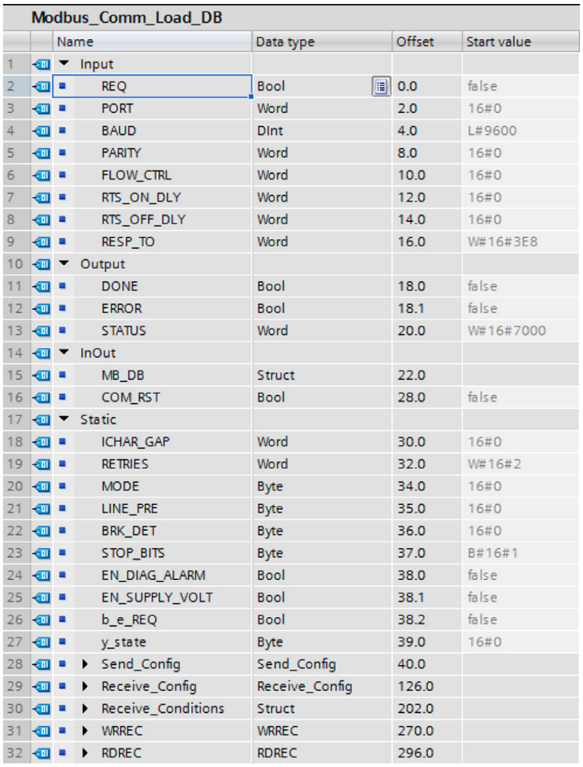
6.1. *Siemens PLC ecosystem*

Siemens 300 series PLCs support Ladder Diagrams, Statement List (Instruction List), Function Block Diagram, Graph, and Structured Text programming languages. When programming these devices four primary blocks are used to build control-logic: Organization Blocks (OB) (i.e. Program POUs), Function Blocks (FB), Functions (FC), and Data Blocks (DB) (i.e. Variable Block POUs). These are aligned to the previously described BSI/IEC 61131-3:2013 (British Standards Institute, 2013a).

Within OBs, FBs, and FCs, one is able to write control-logic. DBs are used to store data, more specifically, variables called by OBs, FCs, and FBs. There exist a number of additional symbol types where data can be generated, outputted, and stored. These can be summarized as I/O Signals (I, Q, etc.), Marker Memory (M, MB, etc.), Peripheral I/O (PIW, PQB, etc.), and Timers and Counters (T & C) (PLCDev, 2020).

Fig. 2 depicts the library FB ***Modbus_Comm_Load*** residing on a Ladder Diagram rung. This FB is provided by Siemens as part of their TIA Communications library, and is responsible for establishing the configuration of a port, from which the PLC can communicate over serial using the Modbus protocol. To the

[1] Selected based on global adoption, making it a representative use-case (Siemens, 2020a).



Modbus_Comm_Load_DB

	Name	Data type	Offset	Start value
1	Input			
2	REQ	Bool	0.0	false
3	PORT	Word	2.0	16#0
4	BAUD	DInt	4.0	L#9600
5	PARITY	Word	8.0	16#0
6	FLOW_CTRL	Word	10.0	16#0
7	RTS_ON_DLY	Word	12.0	16#0
8	RTS_OFF_DLY	Word	14.0	16#0
9	RESP_TO	Word	16.0	W#16#3E8
10	Output			
11	DONE	Bool	18.0	false
12	ERROR	Bool	18.1	false
13	STATUS	Word	20.0	W#16#7000
14	InOut			
15	MB_DB	Struct	22.0	
16	COM_RST	Bool	28.0	false
17	Static			
18	ICHAR_GAP	Word	30.0	16#0
19	RETRIES	Word	32.0	W#16#2
20	MODE	Byte	34.0	16#0
21	LINE_PRE	Byte	35.0	16#0
22	BRK_DET	Byte	36.0	16#0
23	STOP_BITS	Byte	37.0	B#16#1
24	EN_DIAG_ALARM	Bool	38.0	false
25	EN_SUPPLY_VOLT	Bool	38.1	false
26	b_e_REQ	Bool	38.2	false
27	y_state	Byte	39.0	16#0
28	Send_Config	Send_Config	40.0	
29	Receive_Config	Receive_Config	126.0	
30	Receive_Conditions	Struct	202.0	
31	WRREC	WRREC	270.0	
32	RDREC	RDREC	296.0	

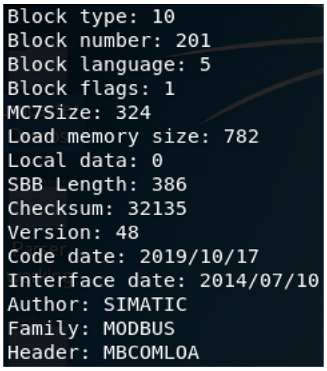
Fig. 3 – Modbus Library Function Data Block.

left of the ***Modbus_Comm_Load*** FB are a set of inputs, these are configuration parameters (port, baud, parity, etc.). To the right of the ***Modbus_Comm_Load*** FB are a set of outputs generated by the function (done, error, and status). By default, some of the inputs are pre-issued and can be seen in grey. These can be left unchanged if their states match the required configuration. Alternatively, inputs can be replaced directly within the rung, as can be seen with the blue ***RTS_ON_DLY*** input (0), or with variables stored in global DBs (gVBs), as can be seen with ***DB1.DBW0*** (the address for global variable ***wRST_OFF_DLY***) applied to the ***RST_OFF_DLY*** input.

Fig. 3 depicts the local DB (fVB) of library FB ***Modbus_Comm_Load***. This DB stores all local variables (including inputs) used by the FB. Where global variables are defined as inputs, FBs have two options. The first is to copy the current state of a global variable into the local DB counterpart during every control-logic cycle. The second is to configure a pointer targeting the global variable's location (DB address), in this instance the pointer will be stored as a local variable within the FBs DB. The latter option is typically used where larger data inputs are required (for storage and performance efficiencies).

6.2. *The application of PCaad*

The following subsections describe how control-logic can be leveraged by attackers to achieve PCaaD via library DB/FB enumeration, leading to our three attack cases: (1) *Exfiltrate Function Block Variables*, (2) *Targeted Manipulation of Function Block Operations*, and (3) *A Storage Based Covert Channel*. To establish communications with our PLC, the Python SNAP7 library was used (Molenaar and Preeker, 2013). This library allows for the crafting of Siemens S7 packets, the primary network protocol used by the PLC, and affords us with the ability to issue requests (e.g. Read, Write, and Upload) as per vendor specifications (Kleinman and Wool, 2014).



```
Block type: 10
Block number: 201
Block language: 5
Block flags: 1
MC7Size: 324
Load memory size: 782
Local data: 0
SBB Length: 386
Checksum: 32135
Version: 48
Code date: 2019/10/17
Interface date: 2014/07/10
Author: SIMATIC
Family: MODBUS
Header: MBCOMLOA
```

Fig. 4 – Get Block Info Example.

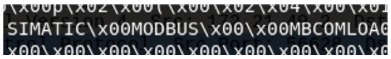
6.2.1. *Enumeration phases 1 and 2*

As described in Section 4, PLCs provide network access to VBs for use by HMIs, TESs, etc. For our PLC this means direct access to DBs (local and global). For example, in Fig. 2, we provided the input ***wRST_OFF_DLY*** at address ***DB1.DBW0***, this address would be used during the configuration of HMIs, allowing operators to read and depict its current state on a graphical display.

The following three data retrieval techniques discussed in Section 4 (*Metadata, Bulk Transfer,* and *Memory Address Interrogation*) exploit access granted to FB DBs over the network in order to enumerate their associated FBs, this can be considered an information leakage vulnerability. To recap, these techniques can be applied to each of the four attack vectors described in Section 3, where we state "execute malicious commands".

Metadata (Get Block Info) - The first technique one can apply towards the enumeration of a FBs DB, makes use of the inbuilt PLC feature ***Get Block Info***, allowing for the extraction of metadata parameters, as seen in Fig. 4. The family and header fields are of greatest importance, allowing us to ascertain which FB is using the DB to store its local variables. In the example provided here, this DB is used by a ***MODBUS*** family FB, so has an affiliation with Modbus communications. The header ***MBCOMLOA*** is a shortened tile for the related FB ***Modbus Comms Load***, previously described in Fig. 2. Therefore, it can be established that this DB is being used by the ***Modbus Comms Load*** FB.

While this technique allows for the enumeration of a DBs associated FB, its reliance on the built in network function ***Get Block Info*** impacts its detectability. In monitoring a ***Get Block Info*** request as it traverses the network, Wireshark's (Wireshark, 2020) in-built protocol recognition is able to clearly identify its purpose. This is also true of nextgen security products (Checkpoint, 2021; Claroty, 2021). As this request is not commonly used within live industrial networks, it would raise a red flag, and could be blocked as part of an environments default security configuration profile.



```
\x00p\x02\x00(\x00\x02\x04\x00\x01
SIMATIC\x00MODBUS\x00\x00MBCOMLOA
x00\x00\x00\x00\x00\x00\x00\x00\x0
```

Fig. 5 – Block Upload Example.

Bulk Transfer (Block Upload) - The second technique we have identified makes use of the inbuilt feature ***Upload***. This is a network function constructed to extract POUs in their entirety from the PLC. With PLCs, it is important to note that in certain situations we talk from the device's perspective. This is industry derived terminology consistent between vendors. Therefore, when using the term *upload*, we are referring to the PLC uploading data to the user, not the user uploading data to the PLC.

In sending a DB ***Upload*** request to the PLC, the entire byte-code of that DB will be returned. We examined this byte-code, and found the previously discussed family and header parameters stored in clear text (see Fig. 5). Running a parser over the byte-code allows us to clearly identify the DBs related FB.

As with ***Get Block Info***, the ***Upload*** function has challenges aligned to detectability. While this request is more common, it only occurs when an engineer requires a copy of PLC control-logic. Therefore, where an engineer is not present, its identification on the network would raise a red flag. It too could form part of an environment's default security configuration profile, requiring an engineer to connect directly with each PLC, rather than from a remote location.

Memory Address Interrogation (Read Requests) - To avoid detection challenges identified with the two former enumeration techniques, we have developed an approach based on ***Read*** requests. This is a specific network function applied to the extraction of variable data from the PLC. HMIs, for example, will execute ***Read*** requests to PLCs in order to extract data for use by operators. This makes it extremely common, and thus if observed on the network would be considered operational behavior (network traffic).

Siemens allocate static memory structures for our PLCs FBs in a minimum of 16bit blocks, even where only a single bit is required. This form of memory allocation can be observed in Fig. 3, where the ***REQ*** variable (boolean) resides at address 0.0 (byte 0, bit 0), and the next variable, ***PORT*** (word), starts at address 2.0, i.e. byte 0 bit 1 to byte 1 bit 7 are all unused and populated with zero states by default.

The size of some library FB DBs raises challenges in mapping all possible variable data combinations, without first conducting a tedious manual review of all possible state combinations (i.e. the more variables, the more valid variable state combinations). While machine learning approaches, on initial inspection, appear to be a feasible solution, they would rely on the ability to capture all variable state combinations. Dependent upon the FB, some variables are set once and do not change (e.g. BAUD in Fig. 3). Therefore, static variables such as these would require manual updating to allow for a complete picture of all possible variable state combinations to be captured. As such, mapping all unused memory offers a viable alternative, with an increased level of performance due to the focused comparison of unused memory alone. This would be achieved through a review of library FB DBs within the Siemens TIA programming agent. For example, Figs. 3, 8, and 9 are screenshots of three library FB DBs, within these screenshots you can observe the DBs construct, i.e., you can see each variable, its type, and memory offset. As this information is openly available within the TIA programming agent, one can build a clear and accurate picture of DB size, and unused memory.

Through the development of a comprehensive signature set (rainbow table) based on an offline analysis of all library FB DBs, focusing on their overall size and the location of any unused memory, we are now able to enumerate any Siemens library FB aligned to a DB using only ***Read*** requests. The ability to achieve DB and associated FB enumeration through the use of ***Read*** requests alone, offers a stealthy and effective technique when compared to the aforementioned approaches.

Our approach begins by ***Reading*** every DB byte into an array. This allows us to ascertain DB size, from which we have a view of possible associated FBs based on our signature set (i.e. we have narrowed the scope of possible FBs due to their static DB size). We then check values at defined offsets (indexes within the array), where we expect to see unused zero value filled memory (again, based on our signature set), e.g. unused offset/byte 1 as previously noted in Fig. 3. Dependent upon the requirements and construct of a FB and its DB, there can be in excess of 10 complete bytes of unused memory, in addition to multiple instances of partially used bytes (up to 7 bits of unused memory within a byte).

Given current programming practices, we have demonstrated how remote control-logic enumeration can be achieved using only ***Read*** requests, thus achieving stealthy PCaaD. This is significant, with Siemens providing library FBs spanning a number of critical areas (see Table 2 for a small selection of example FBs), including Communications, PID (Proportional, Integral and Derivative) Control, Safety, Remote Administration, and Alerting. The following subsections explore ways in which this information can be built upon by an attacker, to further develop their level of process comprehension, and execute targeted attacks.

6.2.2. *Attack 1: Exfiltrate function block variables*

Having successfully enumerated a local DB based on its association to a given library FB, it becomes possible to exfiltrate the data it contains. There are two distinct techniques (Direct and Pointer Decoding) that can be applied to the exfiltration of a FBs data. These techniques are based on how the FB obtains/creates data stored within its local DB. To recap, these techniques can be applied to each of the four attack vectors described in Section 3, where we state "execute malicious commands".

Direct Read Requests - Taking the Modbus Comm Load function from Fig. 2, and its associated DB in Fig. 3, there are a number of local variables that may be of interest to an attacker. For example, at offset 4 there exists a double integer storing the baud rate setting. Knowing that this DB is aligned to the ***Modbus Comm Load*** function, and having analyzed the structure of the DB offline in TIA portal to understand its contents, we can construct a ***Read*** request specifically targeted at this offset to obtain the current baud rate.

If the variable in question is being stored in its entirety within the FBs local DB, a ***Read*** request targeted directly at the variable location can be used to extract its current state. For

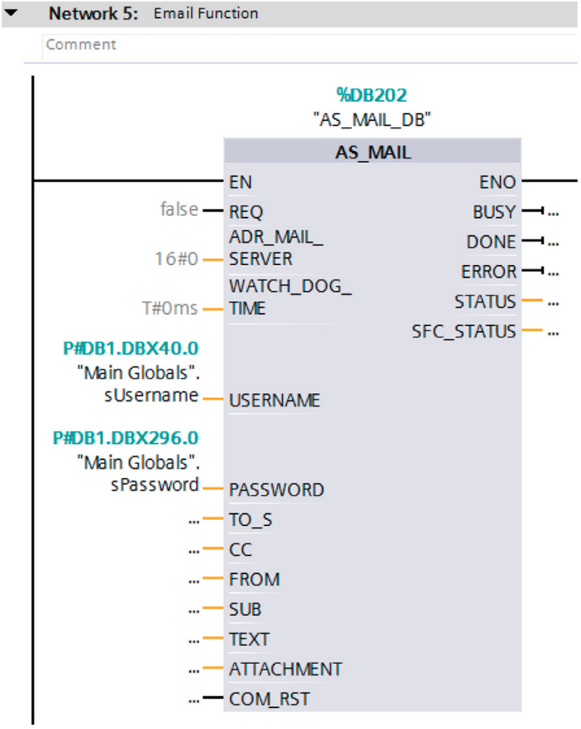
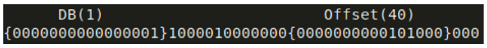


Fig. 6 – Email Library Function.



```
      DB(1)                        Offset(40)
{0000000000000001}1000010000000{0000000000101000}000
```

Fig. 7 – Pointer Structure.

example, to extract the baud rate variable from the Modbus Comm Load DB in Fig. 3, we would construct our request to *Read* 32 bits (double integer) from DB 1, starting at a byte offset of 4. This information develops the attacker's level of process comprehension and builds a picture of device-to-device data flows (Green et al., 2016), of value when constructing a targeted attack.

Pointer Decoding - As previously described, should a function's input variable require a large block of memory, a pointer will be applied. Therefore, should we wish to exfiltrate data from a pointer input, we must first obtain the pointer address. This is achieved by constructing an initial *Read* request targeting the pointers location within the FBs local DB. We must then decode the pointer address, prior to the formulation of an additional *Read* request targeting the newly decoded address.

The construct of pointers within the Siemens ecosystem is unique, more specifically, they contain an address based on the Siemens addressing scheme. Taking the variable ***sUsername*** from Fig. 6 as an example, this has an address of ***DB1.DBX40.0***, it is this address (the variables starting bit) that would be placed inside a pointer. Note the ***P#*** before the address, this denote the use of a pointer.

Fig. 7 provides an example pointer structure, highlighting the address location of ***DB 1*** with an offset of **40**. With this information we can begin constructing a new *Read* request. To complete our request, we must also ascertain the target variable size. An offline analysis of the FB input will provide this information. For example, Fig. 6 is a library FB taking email username and password details, these must be strings. Strings have a standardized size of 256 bytes, as can be see in Fig. 8 (the location of this pointer). Therefore, ***Reading*** 256 bytes from ***DB 1*** at a byte offset of **40**, will provide us with the username ***test@test.com***. Likewise, ***Reading*** 256 bytes from ***DB 1*** at a byte offset of ***296***, will provide us with the password ***mypassword***. Both the username and password are stored in clear text. The use of this information to an attacker could prove highly valuable, especially where the exfiltrated credential set is applicable across multiple systems within the target environment.

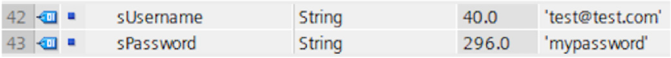
The two exfiltration techniques described here cover all variables stored within a FBs local DB. While the exfiltration of baud rate and credentials demonstrate a clear benefit to attackers, they represent just two examples from a much larger set (i.e. thousands) of FB variables across the Siemens library. With Siemens library FBs spanning an array of capabilities as noted in Section 6.2.1, and examples provided in Table 2, the wider ramifications of data exfiltration, particularly with regards to the development of process comprehension, is significant.

6.2.3. *Attack 2: Targeted manipulation of FB operations*

The use of ***Write*** requests are required to manipulate FB behavior through the targeting of local DB variables. These requests are typically seen where operators modify operational process behavior via a HMI (e.g., starting/stopping a pump). They are, therefore, common permissable commands on an industrial network. To recap, this technique can be applied to each of the four attack vectors described in Section 3, where we state "execute malicious commands".

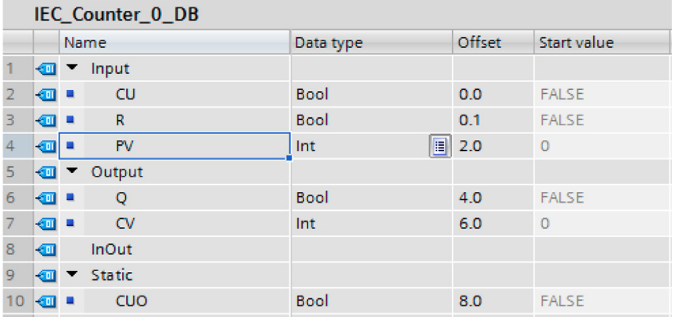
While ***Read*** requests have no limitations in their ability to execute as expected, the ability to successfully manipulate FB behavior has one: cycle time. Where variable states are updated during every control-logic cycle, overwriting them with a 100% success rate becomes a challenge (Robles-Durazno et al., 2019). This is the case for Siemens FBs inputs, with input states moved into a FBs local DB during every cycle. Using Fig. 2 as an example, Section 6.1 discussed the following three techniques to provide FB inputs: directly, from a global DB, and through the use of default values. Should a PLC programmer apply the first or second technique, input states will be written to the FBs local DB during every cycle of control-logic. Where default values are used, this limitation does not exist, and a singe ***Write*** request will overwrite their state.

The cycle time limitation can be circumvented under certain conditions. For example, the ***IEC_CU*** FB (see Fig. 9 for this FBs local DB structure) takes a boolean input (byte 0, bit 0) as a trigger, and provides an integer count output (byte offset 6). Upon a state change of this boolean trigger, the integer count output increases by one. There is a second input responsible for resetting the current count value back to 0 (byte 0, bit 1). Should a PLC programmer allocate a global variable to this input, an attacker would not be confidently able to target the local DB address with a single ***Write*** request to reset the current count. However, variables used within the FB and the FBs outputs can provide an alternative target. Writing a single **0** to the current count integer (byte offset 6), would reset the count without any limitations. This is a vulnerability induced



42	sUsername	String	40.0	'test@test.com'
43	sPassword	String	296.0	'mypassword'

Fig. 8 – Username and Password Global DB.



IEC_Counter_0_DB

	Name	Data type	Offset	Start value
1	Input			
2	CU	Bool	0.0	FALSE
3	R	Bool	0.1	FALSE
4	PV	Int	2.0	0
5	Output			
6	Q	Bool	4.0	FALSE
7	CV	Int	6.0	0
8	InOut			
9	Static			
10	CUO	Bool	8.0	FALSE

Fig. 9 – IEC Count Up DB.

through the way in which the FB code has been written, something only the vendor can address.

The ability to execute ***Write*** requests to a FBs local DB, opens the door to wide-ranging impact. Using the ***Modbus_Comm_Load*** function discussed in Section 6.2.2 as an additional reference point, an attacker could target the BAUD variable, placing the Modbus communication channel into a defective state. Dependent upon what the channel is being used for, this could impact the PLCs ability to communicate with other PLCs for critical operational data exchanges, or even prevent operators from receiving alarms. As with data exfiltration, the quantity and functionality offered within the Siemens FB library brings with it a significant operational risk due to the described manipulation (e.g., the manipulation of Safety Functions, of paramount importance in protecting human life).

6.2.4. *Attack 3: A Storage based covert channel*

In developing our enumeration technique based on ***Read*** requests, we focused on the identification of local FB DBs through unused memory. In the previous section on manipulating FB operations, we have demonstrated an attacker's ability to execute ***Write*** requests targeting a FBs local DB. The remainder of this section takes attack vectors 3 and 4 from Section 3, where security zoning has been established, and presents a method by which it can be violated through the combination of these two concepts.

Scenario - Unlike Attacks 1 and 2, that require the attacker to obtain access to a PLC alone, a broader set of pre-requisites are required for this attack. As described in attack vectors 3 and 4 (See Section 3), the Field Site PLC/RTU is permitted to communicate with all other Field Site devices and also the SCADA Server. All other Field Site devices are not permitted to communicate outside of the Field Site network (managed by perimeter firewalls, a baseline recognized practice to defend zone boundaries (British Standards Institute, 2013b; Stouffer et al., 2015).

Should an attacker compromise the SCADA Server (attack vector 4), allowing for direct interaction with it, and the devices it connects to, only one Field Site device would be accessible, the PLC/RTU. In contrast, should an attacker compromise the HMI (attack vector 3), all Field Site devices would be accessible. However, as the HMI is isolated within the Field Site network zone, remote interaction would not be possible. The ideal attack vector, would involve a method by which the functionality of both the SCADA Server (remote connectivity into the Field Site Network network) and HMI (access to all Field Site devices) could be leveraged. This is where their common resource, the RTU/PLC comes in, providing an ideal pivot point between the two devices. For additional clarity, the compromised SCADA Server acts as the ***CS-Server***, and the HMI acts as the ***C2-Client***.

Channel Operations - In order to establish a covert C2 channel between the C2-Client and the C2-Server via a FBs unused memory, each must first enumerate all library FB DBs using the *Memory Address Interrogation* technique defined in Section 6.2.1. This technique should be adopted as it is stealthy in nature, and thus avoids raising an alert through alternate approaches (*Metadata* or *Bulk Transfer*). Once enumerated, both parties must select the same DB and unused memory offsets to begin communicating. Our approach to the selection of a DB is defined by the quantity of available unused bytes of memory, i.e., both parties will use the DB that contains the most unused bytes of memory. Where two or more DBs meet this requirement with an equal quantity of unused bytes, the first DB will be selected (e.g., should DB3, DB5, and DB6 all contain 10 unused bytes, DB3 would be selected). Once selected, the first two unused bytes will be aligned to synchronization and data exchange, respectively.

Table 1 provides an overview of the binary synchronization states written to, and read from, the first unused byte of memory. Each party must adhere to this scheme in the establishment and continuation of communication exchanges.

A “what goes in comes out" approach has been applied in our current covert C2 channel. The C2-Server's operator (the attacker) will construct a terminal request it wishes the C2-Client to execute (e.g. ping 192.168.0.1). This request will be cut into individual characters, each of which are sent over the data exchange byte. To achieve this, each character is converted into its decimal counterpart (ASCII character encoding). Each character is reconstructed by the C2-Client, and once the entire request has been received, it is executed in the C2-Client's terminal. The terminal's response is then cut into individual characters, sent back to the CS-Server, reconstructed, and displayed.

While the design of this covert C2 channel data exchange appears simplistic, it is effective, and acts as a demonstrable tool in the use of unused FB memory as a pivot point to violate security zoning. Its application across Siemens library FB DBs is widely applicable, and in order to ensure it does not impact a FBs operation, a series of tests were undertaken. Through this testing, we identified that in some instances where only one bit within a two byte block is allocated, the FB would look for state changes across all sixteen bits. Therefore, the use of

Table 1 – Synchronization Byte.

Function	C2-Server	C2-Client
Hello		00000001
Hello Ack	00000011	00000000
Write	01000000	11100000
Reading	11110000	01100000
Read	00000000	00000000
Final Write	11111111	11111110
On Hold	00011000	00011000

Table 2 – Example Library Functions (Siemens TIA v13) Siemens (2020c).

FB Category	Example FB	Description
Basic Instructions	TON	Delays the setting of the Q parameter for the programmed duration PT
	CTU	Increments the value at the CV parameter
	SMC	Compares the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step
	ACK_GL (Safety)	Creates an acknowledgment for the simultaneous reintegration of all F-I/O or channels of the F-I/O of an F-runtime group after communication errors, F-I/O errors, or channel faults
	SFDOOR (Safety)	Safety door monitoring
Extended Instructions	SET_SW	Supports the switch from daylight-saving time to standard time in CPUs that are not equipped with time-of-day status
	TIMESTMP	Transmits messages with a time stamp of an IM153-2 to its instance DB
	RDREC	Reads the data record with the number INDEX from the component addressed using the ID
	SETIO	Consistently transfers the data from the source area spanned by OUTPUTS to the addressed DP standard PROFINET IO device, and, if necessary, to the process image
	PACK	Transfers data located between any addresses and a table
Technology	CONT_C (PID)	Controls technical processes with continuous input and output variables
	PULSEGEN (PID)	Implements a fixed setpoint controller with a switching output for proportional actuators
	MC_MoveAbsolute	Starts an axis positioning motion to move it to an absolute position
	EncoderSINAMICS	Integrates a SINAMICS drive in Easy Motion Control
	OVERRIDE (PID)	Implements an override control
Communication	PUT	Writes data to a remote CPU if the connection does not take place via a CP
	PG_DIAL (Remote Administation)	Transfers a telephone number and an event ID to a TS Adapter. Using the specified telephone number, the TS Adapter establishes a remote connection to a programming device/PC
	MODBUSPN	Enables communication between a CPU with integrated PN interface and a partner which supports the Modbus/TCP protocol
	AS_MAIL (Alerting)	Uses the Simple Mail Transfer Protocol (SMTP) to transfer an e-mail from a CPU to a mail server
	SMS_SEND (Alerting)	Transfers a telephone number, a service center number and an SMS message to a TS Adapter

unused memory would have an undesirable impact on FB operation, creating issues with the stable and expected execution of control-logic, and increase the chance of detectability. However, we found few instances that impact ones ability to establish a covert C2 channel through the use of this technique. Furthermore, in monitoring the status of the primary bit in use, if it is set to 1, the use of unused memory will have no impact on FB operation.

6.3. *Cross vendor generalization*

The PoC running example described throughout this section has been aligned to Siemens 300 series PLCs. Considering alternative Siemens PLC series, 400 series PLCs act as a mirror to the 300 series, making the techniques described here holistically applicable. ET200 series PLCs act as a bridge between the 300/400 series and the newer 1200/1500 series. Testing with an ET200S resulted in the direct applicability of our techniques. Finally, for 1200 and 1500 series PLCs, where library functions (e.g. ***TMAIL_C***) apply direct addressing by default, the described techniques are also applicable.

In the consideration of PLCs from other vendors, we conducted some initial experimentation, and although the ecosystem of the devices tested differs somewhat to Siemens, our fundamental concept (see Section 4) holds true to ascertain similar attacks. For example, Rockwell Automation’s Allen-Bradley PLC VBs are referred to as “Data Files" or “Control Blocks”, storing variables in a similar way to FB DBs in the Siemens ecosystem. These are read/write accessible over the network using direct addressing (tested with the pycomm3 SLCDriver (Ottoway, 2021)), and have a standardized construct. This construct is well documented by Allen-Bradley in their SLC 500 Instruction Set Reference Manual (Rockwell Automation, 2008). In reviewing this manual, one can identify certain functions contain unused bytes of memory in their associated Control Blocks (e.g., EtherNet/IP Explicit Message). Through our experimentation, we found this unused memory to be filled with zero values, thus our attack techniques could be applied in much the same way. In addition, we identified vendors providing library functions for one another’s devices. ABB, for example, provide library functions supporting drive integration with Siemens PLCs. These library functions allow a Siemens PLC to control an ABB drive (ABB, 2020). The FBs and associated DBs provided by ABB, harbour the same deficiencies as discussed in Section 6.2. Therefore, ABB devices become exploitable through this integration.

7. Lessons learnt

The following subsections provide an overview of salient points tied to a PLCs exploitability via the PCaaD process, highlight issues in PLC programming practices, offering potential mitigations, a response from Siemens, and detailing a process for automation PCaaD and attack execution.

7.1. *The impact of PCaad*

This work demonstrates the theoretical and practical application of PCaaD, targeting only the PLC. Existing approaches to develop a high level of process comprehension can be lengthy and involve data aggregated from multiple sources (Green et al., 2017). This aggregation of data is largely applied toward an attacker’s understanding of PLC control-logic. With a high level of process comprehension, targeted operational process manipulation is made possible. Without it, attackers are limited to primitive DoS attacks. The capabilities described in this paper provide increased process comprehension, and therefore the ability to strategically attack a PLC. While this does not provide full process comprehension, it demonstrates an approach which supports enhanced attack complexity (e.g. *data exfiltration, targeted manipulation of FB variables,* and *covert channel creation*). The described techniques can be adopted in parallel to existing approaches (Green et al., 2017), enhancing an attackers understanding of PLC functionality.

This can be exemplified when comparing targeted manipulation of FB operation attacks, to attacks demonstrated in previous work (Robles-Durazno et al., 2019), reducing the requirement for priori knowledge of a system to identify target variables, and also the need for rapid remote overwriting of PLC memory locations to maintain the desired effect (i.e. the cycle time limitation).

In addition to the increased level of enumeration/attack sophistication available through the use of PCaaD, this paper practically demonstrates the use of a PLC in a covert channel. It was through the identification of unused memory signatures during the PCaaD process, that gave rise to the possibility of a storage based covert channel. As such, the PLC can be used as a pivot point between protected Field Site networks and external networks. The exploitation of a PLC in facilitating an attack, rather than being the target of an attack, along with the mechanics of this storage based covert channel, are both novel. Importantly, this adds a previously seldom considered class of security challenge for PLC implementation, suggesting a reconsideration in the understanding of ICS security zoning, a primary defensive measure (British Standards Institute, 2013b). Applying the described approach highlights challenges in conventional segregation techniques to adequately prevent attackers from accessing “isolated" zones. This work practically demonstrates channel feasibility, and as such forces the ICS community to think differently about the role of PLCs in cyber attacks.

7.2. *Mitigations*

This work continues to open the door of a new vulnerability class, one derived through the creation of insecure control-logic. The exploitability of FB libraries is, in part, tied to their static and accessible memory structures, but also in how they have been written by vendors, and implemented by PLC programmers. Therefore, during the development of our PCaaD approach, and each exploitation technique, mitigations were actively sought. Here a number of approaches are presented that would aid in the reduction of PCaaD and associated exploitation susceptibility. These are presented across two levels, device-level (changes that can be applied to the PLC), and network-level (controlling/monitoring the flow of network traffic to and from the PLC). Some examples are aligned to the Siemens eco system to support discussion, but provide a valid viewpoint for other vendors.

7.2.1. *Device-Level mitigaiton*

Memory State Monitoring could be implemented to identify unexpected data within a defined memory structure. For example, in monitoring the BAUD variable in Figs. 2 and 3, should its state remain static at 9600, a single rung of validation logic would be required. This rung would check the current value during every cycle, if the value did not equal 9600, it would set a bit, thus raising an alert on local HMIs/SCADA systems. This concept could also be applied to all unused zero state filled memory, raising an alert if it becomes populated with data. This could be applied in a similar way to stack canaries in conventional applications.

Setting All FB Inputs even where default values are applicable, will induce the cycle time limitation discussed in Section 6.2.3. While this will not mitigate all attacks, it presents an additional challenge for attackers to overcome.

Filling Unused Memory with random data would make enumeration via read requests more challenging.

Additional Checks for Neighboring PLCs could be applied to validate received data. For example, considering the count function attack in Section 6.2.3, an additional check to identify state changes on the reset bit, before accepting a reset of the count value.

Read/Write Protection features within 300, 400, and ET200 series devices would prevent enumeration using the ***Upload*** technique. However, the applicability of alternate options discussed in Section 6.2.1 would remain unaffected.

Vendor Centric changes could be applied to the development and inclusion of library FBs. For example, FBs could be written in such a way as to allow for all data generated to be made available as defined outputs, fed into gVBs, meaning no local or remote access to a FBs fVB would be required. In addition, Siemens implement know-how protection on FB code, preventing programmers viewing pre-compiled logic. This principle could be applied to a FBs associated fVB, making it harder to map memory usage and generate signature sets.

Memory allocation within fVBs could be constructed in a more efficient way, reducing unused capacity, and thus advancing the technical requirements for successful enumeration using read requests alone, and limiting attackers ability to create covert channels.

A blanket rejection of all network-based requests targeting fVBs would offer holistic mitigation. This feature can be manually enabled (disabled by default) on gVBs used by 1200 and 1500 series devices, but is not yet available for fVBs.

Enable Block Optimization on Siemens 1200 and 1500 series PLC where possible (not currently available across all library FBs). This feature removes static addressing schemes and introduces memory randomization, with variables referenced based on name as opposed to address. It may still be possible to enumerate and attack these functions based on the use of standardized variable names, however this is outside of our current scope. The only drawback of this feature is that third-party devices (e.g., HMIs) requiring access to PLC data must support S7 Comm Plus. Where support is not available, a blanket disabling of block optimization may be adopted by PLC programmers, thus re-introducing static addressing schemes.

7.2.2. *Network-Level mitigation*

Network access restrictions form a well explored and recommended starting point towards appropriate mitigation (British Standards Institute, 2013b; Stouffer et al., 2015). If network-level access to fVBs can not be natively restricted, ensuring PLCs are isolated from widely accessible networks forms a baseline requirement. To explore this option in more depth, we setup the scenario from Fig. 1 in a testbed environment (Green et al., 2020), with the following four industrial security solutions residing on the Field Site perimeter (between the Field Site Router and Switch in Fig. 1). These were configured to control traffic flow between the Field Site network and external networks:

- Westermo Redfox (Westermo, 2021) - This devices is similar to traditional IT network security products. Here we configured a rule-set permitting only the SCADA Server access to the PLC/RTU based on IP addresses and port 102 (used by the Siemens S7 communications protocol).
- Siemens S623 (Siemens, 2021b) - This device has a set of basic templated rules. Here we permitted the use of the Siemens S7 protocol only. There are no additional options to specify IP addresses, ports, etc.
- Tofino Xenon (Tofino Security, 2021b) - This device goes one step further than the S623. Here we configured a rule-set permitting only the SCADA Server access to the PLC/RTU based on IP addresses and the Siemens S7 communications protocol.
- Checkpoint 1570R (Checkpoint, 2021) - Providing the most advanced defensive solution, here we configured a rule-set permitting only the SCADA Server accesss to the PLC/RTU based on IP addresses, and Siemens S7 protocol via read/write requests.

In addition to these, we also deployed the following passive intrusion detection solution, taking a mirror/SPAN feed of all Field Site communications via the Field Site switch.

- Claroty CTD (Claroty, 2021) - This solution first monitors the environment and builds a baseline understanding of normal behavior in “learning mode”. Once we had captured all normal behavior, we switched to “operational mode”, where alerts are then generated based on suspicious traffic falling outside of the initial trusted baseline.

The following set of tests (T) were undertaken. These were applied initially from an untrusted device (See attack vector 1 in Section 3), then again from a trusted device (See attack vector 4 in Section 3).

- T1: Enumeration via *Memory Address Interrogation (read requests)*
- T2: Exploitation via *Targeted Manipulation of FB Operations (write requests)*
- T3: C2 Channel - Client Setup
- T4: C2 Channel - Server Setup
- T5: C2 Channel - Server Instructs Client to Execute an Nmap Top 20 Port Scan (Nmap, 2021)
- T5s: C2 Channel - Client Executes an Nmap Top 20 Port Scan (Nmap, 2021) Inside the Field Site Network

Table 3 – Prevention Results.

Prevention										
Vendor/Device	Trusted/Untrusted	T1	T2	T3	T4	T5	T5s	T6	T7	T8
Siemens S623	Untrusted	N	N	N/A	N	N	N/A	N	N	Y
	Trusted	N	N	N/A	N	N	N/A	N	N	Y
Tofino Xenon	Untrusted	Y	Y	N/A	Y	Y	N/A	Y	Y	Y
	Trusted	N	N	N/A	N	N	N/A	N	N	Y
Westermo Redfox	Untrusted	Y	Y	N/A	Y	Y	N/A	Y	Y	Y
	Trusted	N	N	N/A	N	N	N/A	N	N	N
Checkpoint 1570R	Untrusted	Y	Y	N/A	Y	Y	N/A	Y	Y	Y
	Trusted	N	N	N/A	N	N	N/A	Y	Y	Y

Table 4 – Detection Results.

Detection										
Vendor/Device	Trusted/Untrusted	T1	T2	T3	T4	T5	T5s	T6	T7	T8
Claroty CTD	Untrusted	Y (A)	Y (A)	N/A	Y (A)	Y (A)	N/A	Y (A)	Y (A)	Y (A)
	Trusted	Y (E)	Y (E)	Y (E)	Y (E)	Y (E)	Y (A)	Y (A)	Y (A)	Y (A)

- T6: Enumeration via *Metadata (get block info)*
- T7: Enumeration via *Bulk Transfer (upload)*
- T8: TCP C2 Channel via Malicious.exe, executed on the HMI, operating over port 102 (i.e., a traditional staged Meterpreter session (Offensive Security, 2021a) used as a comparison against our novel C2 Channel).

Our results from these tests are summarized in Tables 3 (prevention) and 4 (detection). In Table 3 we focus on each device's ability to prevent the attacks across our set of tests. In Table 4 we not only sought to establish the detectability of each attack, but how it was classified i.e., an alarm (A) or an event (E). T3 and T5s are only applicable to our detection-based solution, as they generate traffic within the Field Site network from trusted devices, and do not traverse its perimeter.

Across the four prevention-based devices we observed a unique set of defensive features. Starting with the Westermo Redfox, this device operates in a similar way to traditional IT products, and therefore lacks the ability to filter traffic based on ICS protocol recognition. However, it provides a good level of defensive coverage against untrusted devices (e.g. attack vectors 1 and 2). Where trusted devices are compromised by an attacker (e.g. attack vectors 3 and 4) it became ineffective. The Siemens S623 displayed the weakest performance across our tests, and was unable to prevent any of our attacks from either untruster or trusted devices. It was only able to block the traditional C2 attack established with Metasploit. This was due to its limited configurability, focusing solely on permitting or denying specific protocols. The Tofino Xenon combines the functionality of the Westermo and Siemens devices, providing traditional IP and port based filtering, alongside ICS protocol recognition. This provided extra coverage in our tests to successfully block the traditional Metasploit C2 channel. However, it still failed to prevent the attacks detailed in this paper when executed from a trusted device. Finally, the Checkpoint 1570 represents the most advanced device tested. This device goes one step further than the Tofino Xenon, allowing more granular control over functions within a protocol. In our testing we only permitted read/write requests from the trusted SCADA Server (the minimum requirement for this server to perform its operational function), this meant two of our three data retrieval approaches (*Metadata* and *Bulk Transfer*) were blocked, further confirming the stealthiness of our *Memory Address Interrogation* technique.

The Claroty CTD solution was able to identify every attack. This is unsurprising given its visibility of all traffic (internal and external) flowing through the Field Site switch. However, where a security operations centre will typically only take action based on an alarm, malicious traffic classification is critical. From our results we can see that all traffic involving untrusted devices is raised as an alarm. However, the attacks detailed in this paper, excluding *Metadata* and *Bulk Transfer* data retrieval techniques, are all noted as events (E) when executed from trusted devices. These events are often tagged with the comment "Baseline deviation change, not risky change". Although the trusted devices are reading/writing to/from different memory addresses, the read/write functions themselves are not considered risky, and have been undertaken by these trusted devices as part of their normal operation. Therefore, while T5s is raised as an alarm due to previously unseen behavior, should the C2 Client be instructed to execute read/write requests towards any of the Field Site PLCs, the traffic would be logged as an event. Using the techniques outlined within this paper, the attacker can enumerate, exfiltrate, and attack the remaining Field Site PLCs via the C2 channel while avoiding alarm generation.

From this testing we can see that the PCaaD techniques and attacks described within this paper, particularly those only requiring read/write requests, are stealthy, and are especially effective when executed from trusted devices (threat vectors 3 and 4). To fully prevent our attacks, the Checkpoint 1570R would be required to provide an extra layer of granularity in their controls, defining the specific PLC addresses trusted devices can read/write to/from. This has been applied to other protocols, but is not yet available for the Siemens S7

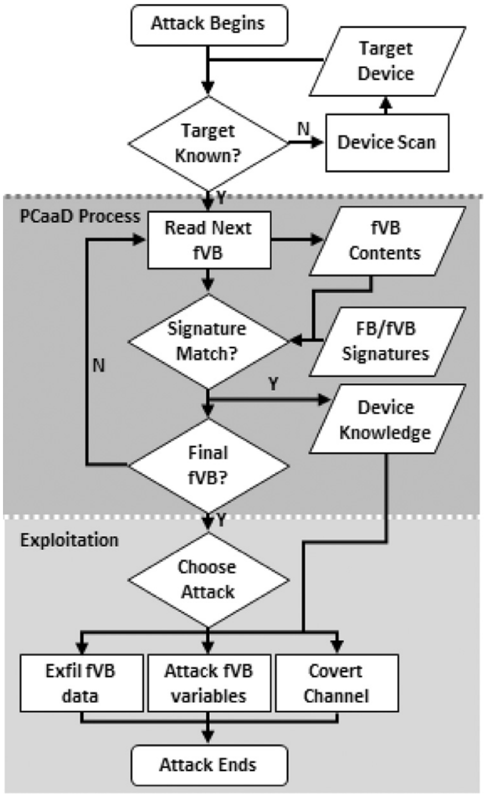


Fig. 10 – Automated PCaaD & Attack Execution Process.

protocol tested here. With detection based approaches, such as Claroty CTD, tweaks could be made to the classification of certain traffic patterns. For example, in our *Memory Address Interrogation* technique applied during T1, we read every byte within a DB sequentially, until we finally attempted to read an address that does not exist (denoting the end of the DB). The PLC response to this final request could be viewed as an indication of malicious activity, bolstered by the rapid sequential requests for each byte of data. Furthermore, where a write request targeting a new memory address is identified, this could be classified as an alarm, even if the device has historically made write requests in other addresses.

7.3. Vendor response

Following responsible disclosure practices, we contacted Siemens, Rockwell, and ABB to make them aware of the issues identified in this paper. Siemens response was as follows:

"The flat addressing in PLCs like S7-300 and ET200S CPU is a design decision from the 90s and cannot be easily changed without breaking existing installations. Siemens recommends customers to restrict network access to the affected devices, to apply Defense-in-Depth measures that can be found in the Operational Guidelines for Industrial Security, and to follow the recommendations in the product manuals. Siemens improved this behavior in the new PLC generation (S7-1200 and S7-1500) by creating the optimized Data Blocks and additional levels of protection to these PLCs."

It is worth noting, Siemens have committed to offering 300 series devices until 2023, with an additional ten years of support beyond this point (Siemens, 2020a).

7.4. Automated evaluation of control-Logic

The PCaaD capabilities presented here represent a first step towards the ability to automate targeted exploitation of operational processes and PLC configuration parameters. The endpoint of this research is to provide an integrated exploitation platform. Such a platform would enumerate using the identified, and expanded upon PCaaD approaches, and further integrate exploitation components extending beyond those already identified. Embedding this functionality into a single platform forms a linear attack offering, heightening each component/techniques collective value. This has been seen in IT penetration testing (e.g. Metaspolit). Fig. 10 provides a high-level process flow, depicting the functionality offered through the integration of each component described within in this paper. A PoC tool operating as outlined in Fig. 10 has been developed in Python by the research team, and was applied during the network-level mitigation testing.

The process incorporates manual and automated target selection (the latter is achieved through use of PLC Scan (Efanov, 2017)). Once a suitable target PLC is identified, the PCaaD process is initiated to enumerate control-logic and identify possible target memory locations for our three exploit categories. Additional flexibility is included to add signatures beyond the known vendor provided library FBs. Custom, in-house signatures can be added to the repository, supporting the enumeration and exploitation of in-house developed FBs/fVBs. As a whole, this platform will aid an organizations efforts to better understand the scale of exploitable control-logic within their estate, and to evaluate security zoning.

8. Conclusion and future work

This work has demonstrated the feasibility of stealthy, sophisticated, targeted attacks against industrial systems with no prior knowledge of the target PLC configuration or control-logic. An attack of this nature was previously considered to be impractical when targeting the PLC alone. However, through the exploitation of current PLC programming practices, code reuse patterns, and predictable memory allocation, such attacks are possible. Using library FBs as a primary use case, their identification and subsequent exploitation presents several security challenges. These challenges, aligned to successful PCaaD, give rise to sophisticated targeted attacks against previously unseen industrial systems, and the use of PLCs in the facilitation of attacks via storage-based covert channels.

A further benefit of the PCaaD approach described in this paper, is the ability to fingerprint custom FBs. These FBs, written in-house by PLC engineers for deployment across an organizations operational estate, can now be identified using our signature techniques. Given the wide-spread use of identical custom FB libraries within an organization (Ljungkrantz and Akesson, 2007), identifying custom control-logic offers added value, increasing the breadth of FB detection beyond publicly accessible/vendor provided libraries.

While our practical proof of concept focused on demonstrating the identified security challenges on Siemens PLCs, an initial exploration of two other prominent vendors highlights key parallels. The functional similarities between vendors, suggests other vendors' devices to be equally exploitable.

A selection of host and network-based mitigation techniques have been discussed. These included points raised by Siemens, aligned to capability embedded within their latest product range. This enhanced capability demonstrates a level of commitment by vendors, a positive step, as addressing issues at the device level should always take precedence over secondary wrap-around techniques, such as network-based detection/prevention. However, we believe features within these products could be circumvented, offering an additional direction for future work. Furthermore, due to the lifespan and operational context of industrial systems, combined with a reluctancy from vendors to provide suitable updates for existing products, end-users are often forced to adopt network-based defensive strategies. Through the testing of five commercial network-based security solutions, we have demonstrated the stealthiness of our approach, solidifying the importance of approaching detection across multiple dimensions, both network and host-based. Suggestions have been posited towards the improvement of network-based detection/prevention, alongside novel host-based techniques. These suggestions would mitigate the risk of attacks tested within the paper, presenting a starting point for continued development in the face of increasingly sophisticated attacks.

The research agenda for future work in this space will focus on two primary themes: Improvements in PCaaD techniques, and exploring further possibilities to utilise PLCs as an attack platform.

For PCaaD techniques, our first phase of work will be to widen the empirical exploration to a broader range of vendors. The aim of which is to develop a comprehensive tool, able to enumerate fVBs across a range of PLCs. To achieve this, further development in identifying memory features to provide more sophisticated signatures may be required. This includes the exploration of machine learning approaches. Our second phase of work will explore how PCaaD can be developed to yield a greater level of process comprehension. As a concept, process comprehension is extremely broad, including, but not limited to, hardware configurations, adopted protocols, and PLC control-logic. While the findings of this paper, in combination with existing techniques, bring us closer to understandings a PLCs purpose within an operational context, there is still room for improvement. For example, once a fVB has been identified, how can we better understand its place within the broader control-logic base, and furthermore, how can we tie it back to specific physical elements within the overall operational process. Developing PCaaD capabilities using stealthy techniques, offers a long-term research theme that can be subsequently used to drive and evaluate enhancements in host and network-based defensive strategies.

The use of PLCs as a tool during an attack will be further developed. This will focus on establishing a range of mechanisms to enhance our existing covert channel. More specifically, we will explore the tradeoff between channel features, detectability, robustness, and control overhead. While also exploring options for integration with existing toolkits, e.g., Metasploit (Rapid 7, 2021) and CobaltStrike (HelpSystems, 2021).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Benjamin Green: Conceptualization, Methodology, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Richard Derbyshire:** Conceptualization, Software, Writing – original draft, Writing – review & editing. **Marina Krotofil:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **William Knowles:** Conceptualization, Methodology, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Daniel Prince:** Conceptualization, Writing – original draft, Visualization, Supervision. **Neeraj Suri:** Conceptualization, Supervision.

Acknowledgements

This work was supported in part by EC H2020 CONCORDIA GA#830927.

REFERENCES

ABB, 2020. Drive Manager for SIMATIC. https://new.abb.com/drives/software-tools/drive-manager-for-simatic.

Abbasi A, Hashemi M, Zambon E, Etalle S. Stealth low-level manipulation of programmable logic controllers i/o by pin control exploitation. In: International Conference on Critical Information Infrastructures Security. Springer; 2016. p. 1–12.

Assante, M. J., Lee, R. M., 2015. The Industrial Control System Cyber Kill Chain. https://www.sans.org/reading-room/whitepapers/ICS/paper/36297.

Beresford D. Exploiting siemens simatic s7 plcs. Black Hat USA 2011;16(2):723–33.

Biham E, Bitan S, Carmel A, Dankner A, Malin U, Woo A. Rogue7: Rogue Engineering-Station Attacks on S7Simatic PLCs; 2019. https://i.blackhat.com/USA-19/Thursday/us-19-Bitan-Rogue7-Rogue-Engineering-Station-Attacks-On-S7-Simatic-PLCs-wp.pdf.

British Standards Institute. 61132-3:2013 - Programmable Controllers - Part 3: Programming Languages; 2013.

British Standards Institute, 2013b. Industrial Communication Networks Network and System Security Part 3-3: System Security Requirements and Secruity Levels (IEC 62443-3-3).

Checkpoint, 2021. Industrial Control Systems Security Gateway. https://www.checkpoint.com/products/industrial-control-systems-appliances/.

Cherepanov A. WIN32/INDUSTROYER: A new threat for industrial control systems; 2017. https://www.welivesecurity.com/wp-content/uploads/2017/06/Win32_Industroyer.pdf.

Claroty, 2021. Continuous Threat Detection. https://www.claroty.com/continuous-threat-detection/.

COPADATA, 2020. What is SCADA? https://www.copadata.com/en/product/zenon-software-platform-for-industrial-automation-energy-automation/visualization-control/what-is-scada/.

dark lbp, 2020. ICSSploit: Industrial Control System Exploitation Framework. https://github.com/dark-lbp/isf.

Derbyshire R, Green B, Hutchison D. "Talking a different language": anticipating adversary attack cost for cyber risk assessment. Computers & Security 2020.

Derbyshire R, Green B, Prince D, Mauthe A, Hutchison D. In: Security and Privacy Workshops (EuroS&PW), 2018 IEEE European Symposium on. An Analysis of Cyber Security Attack Taxonomies. IEEE; 2018.

Dobrushin, M., Flint, Y., 2019. Project RunAway. https://cs3sthlm.se/program/presentations/matan-yoav/.

Drias Z, Serhrouchni A, Vogel O. Taxonomy of attacks on industrial control protocols. In: 2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS). IEEE; 2015. p. 1–6.

Eckhart M, Ekelhart A, Lüder A, Biffl S, Weippl E. Security development lifecycle for cyber-physical production systems, 1. IEEE; 2019. p. 3004–11.

Efanov, D., 2017. PLCScan the Internet. http://scadastrangelove.blogspot.com/2012/11/plcscan.html.

F-Secure., Havex Hunts For ICS/SCADA Systems, 2014. https://www.f-secure.com/weblog/archives/00002718.html.

Falliere N, Murchu LO, Chien E. W32.Stuxnet Dossier; 2011. https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en.

Fluchs, S., 2020. The Top 20 Secure PLC Coding Practices Project. https://bit.ly/34DqoHi.

Garcia L, Brasser F, Cintuglu M, Sadeghi A-R, Mohammed O, Zonouz S. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In: Network and Distributed System Security (NDSS) Symposium; 2017. p. 1–15.

Gollmann D, Gurikov P, Isakov A, Krotofil M, Larsen J, Winnicki A. Cyber-physical systems security: Experimental analysis of a vinyl acetate monomer plant. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security; 2015. p. 1–12.

Gouglidis A, Konig S, Green B, Rossegger K, Hutchison D. In: Game Theory for Security Management - From Theory to Practice. Protecting Water Utility Networks from Advanced Persistent Threats: A Case Study. Boston: Springer Birkhäuser; 2018.

Govil N, Agrawal A, Tippenhauer NO. On Ladder Logic Bombs in Industrial Control Systems. In: Computer Security. Springer; 2017. p. 110–26.

Green B, Derbyshire R, Knowles W, Boorman J, Ciholas P, Prince D, Hutchison D. ICS Testbed Tetris: Practical Building Blocks Towards a Cyber Security Resource. 13th USENIX Workshop on Cyber Security Experimentation and Test 2020.

Green B, Krotofil M, Abbasi A. On the Significance of Process Comprehension for Conducting Targeted ICS Attacks. Proceedings of the 3nd ACM Workshop on Cyber-Physical Systems Security and Privacy. ACM, 2017.

Green B, Krotofil M, Hutchison D. Achieving ics resilience and security through granular data flow management. In: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy; 2016. p. 93–101.

HelpSystems. Software for Adversary Simulations and Red Team Operations. https://www.cobaltstrike.com/ 2021.

ICS-CERT, 2014. ICS Focused Malware. https://us-cert.cisa.gov/ics/alerts/ICS-ALERT-14-176-02A

Jacinto, J., 2017. Siemens Open Library Can Speed Project Development. https://bit.ly/2QUy96a.

Kleinman A, Wool A. Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics. The Journal of Digital Forensics, Security and Law: JDFSL 2014;9(2):37.

Kottler S, Khayamy M, Hasan SR, Elkeelany O. Formal verification of ladder logic programs using nusmv. In: SoutheastCon 2017. IEEE; 2017. p. 1–5.

Lee RM, Assante MJ, Conway T. In: Technical Report. German Steel Mill Cyber Attack. SANS; 2014. https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf.

Liang G, Weller SR, Zhao J, Luo F, Dong ZY. The 2015 ukraine blackout: implications for false data injection attacks. IEEE Trans. Power Syst. 2017;32(4):3317–18.

Ljungkrantz O, Akesson K. A study of industrial logic control programming using library components. In: 2007 IEEE International Conference on Automation Science and Engineering. IEEE; 2007. p. 117–22.

McLaughlin S, McDaniel P. Sabot: Specification-based payload generation for programmable logic controllers. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security; 2012. p. 439–49.

Miller T, Staves A, Maesschalck S, Sturdee M, Green B. Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems. International Journal of Critical Infrastructure Protection 2021.

Mirian A, Ma Z, Adrian D, Tischer M, Chuenchujit T, Yardley TM, Berthier R, Mason J, Durumeric Z, Halderman JA, Bailey M. An Internet-wide view of ICS devices. In: 2016 14th Annual Conference on Privacy, Security and Trust (PST); 2016. p. 96–103.

Molenaar, G., Preeker, S., 2013. Welcome to python-snap7's documentation!https://python-snap7.readthedocs.io/en/latest/.

Mulazzani M, Neuner S, Kieseberg P, Huber M, Schrittwieser S, Weippl E. Quantifying windows file slack size and stability. In: IFIP International Conference on Digital Forensics. Springer; 2013. p. 183–93.

Nmap, 2020. s7-info. https://nmap.org/nsedoc/scripts/s7-info.html.

Nmap, 2021. Port Specification and Scan Order. https://nmap.org/book/man-port-specification.html.

Nochvay A. CODESYS Runtime, a PLC control framework; 2019. https://ics-cert.kaspersky.com/media/KICS-CERT-Codesys-En.pdf.

Offensive Security, 2021a. Writing Meterpreter Scripts. https://www.offensive-security.com/metasploit-unleashed/writing-meterpreter-scripts/.

Ottoway, I., 2021. A Python Ethernet/IP library for communicating with Allen-Bradley PLCs. https://pypi.org/project/pycomm3/.

PLCDev, 2020. Symbol Table Allowed Addresses and Data Types. http://www.plcdev.com/symbol_table_allowed_addresses_and_data_types.

Rapid 7. Metasploit: The world's most used penetration testing framework.. https://www.metasploit.com/ 2021.

Robles-Durazno A, Moradpoor N, McWhinnie J, Russell G, Maneru-Marin I. PLC Memory attack detection and response in a clean water supply system. Int. J. Crit. Infrastruct. Prot. 2019;26:100300.

Rockwell Automation https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1747-rm001_-en-p.pdf.

Serhane A, Raad M, Raad R, Susilo W. Plc code-level vulnerabilities. In: 2018 International Conference on Computer and Applications (ICCA). IEEE; 2018. p. 348–52.

Serhane A, Raad M, Raad R, Susilo W. Programmable logic controllers based systems (plc-bs): vulnerabilities and threats. SN Applied Sciences 2019;1(8):924.

Shodan, 2020. Industrial Control Systems. https://www.shodan.io/explore/category/industrial-control-systems.

Siemens, 2020a. SIMATIC S7-300 - Proven Multiple Times! https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-300.html.

Siemens, 2020b. SIMATIC WinCC V7. https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada/simatic-wincc-v7.html.

Siemens, 2020c. Totally Integrated Automation Portal. https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html.

Siemens, 2021a. IWLAN the WLAN for challenging industrial applications. https://new.siemens.com/global/en/products/automation/industrial-communication/industrial-wireless-lan.html.

Siemens, 2021b. scalance S623. https://support.industry.siemens.com/cs/pd/17207?pdti=pi&dl=en&lc=en-CR.

Slay J, Miller M. Lessons learned from the maroochy water breach. In: International Conference on Critical Infrastructure Protection. Springer; 2007. p. 73–82.

Stouffer K, Pillitteri V, Lightman S, Abrams M, Hahn A. Guide to Industrial Control Systems (ICS) Security; 2015. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf.

Tofino Security https://www.tofinosecurity.com/products/tofino-xenon-security-appliance.

Wardak H, Zhioua S, Almulhem A. Plc access control: a security analysis. In: 2016 World Congress on Industrial Control Systems Security (WCICSS). IEEE; 2016. p. 1–6.

Westermo, 2021. Industrial Routing Switch. https://www.westermo.com/products/ethernet-switches/layer-3/rfi-219-t3g.

Williams TJ. The purdue enterprise reference architecture. Comput. Ind. 1994;24(2–3):141–58.

Wireshark, 2020. S7 Communication (S7comm). https://wiki.wireshark.org/S7comm.

Benjamin Green is an academic fellow in the School of Computing and Communications at Lancaster University, UK. His research involves both offensive and defensive elements of Industrial Control System security. He is involved in several related research projects, including Operational Technology Management after Cyber Incident (OT-MCI).

Richard Derbyshire is a PhD student in the School of Computing and Communications at Lancaster University, UK. His research involves both offensive and defensive elements cyber security, with a focus on penetration testing and risk assessment.

Marina Krotofil is a visiting researcher at Hamburg University of Technology, Germany. She is cyber security professional with over a decade of hands-on experiences in advanced methods for securing Industrial Control Systems (ICS) and other cyber-physical systems. Marina frequently collaborates with international organizations on the topics of critical infrastructure security and is a regular speaker at the leading conference stages worldwide.

William Knowles is a visiting researcher in the School of Computing and Communications at Lancaster University, UK. Within his industry work he specialises in goal-oriented security testing, and works to help organisations improve both their prevention and detection capabilities. His research interests primarily revolve around this area, with a particular focus on industrial environments.

Daniel Prince is a security and protection science lecturer at Lancaster University. He specialises in Cyber Risk Management and Network Security in complex socio-technical systems, particularly cyber physical systems and the financial services sector. He also works closely with organizations to help them understand the economic growth potential of cyber security.

Neeraj Suri is a Distinguised professor and Chair in Cyberseurity at Lancaster University, UK. His professional details are available at https://ssg.lancs.ac.uk/people/neeraj