On the Significance of Process Comprehension for Conducting Targeted ICS Attacks

Benjamin Green
Lancaster University
Lancaster, United Kingdom
b.green2@lancaster.ac.uk

Marina Krotofil
Hamburg University of Technology
Hamburg, Germany
marina.krotofil@tuhh.de

Ali Abbasi
University of Twente
Enschede, Netherlands
a.abbasi@utwente.nl

ABSTRACT

The exploitation of Industrial Control Systems (ICSs) has been described as both easy and impossible, where is the truth? Post-Stuxnet works have included a plethora of ICS focused cyber security research activities, with topics covering device maturity, network protocols, and overall cyber security culture. We often hear the notion of ICSs being highly vulnerable due to a lack of inbuilt security mechanisms, considered a low hanging fruit to a variety of low skilled threat actors. While there is substantial evidence to support such a notion, when considering targeted attacks on ICS, it is hard to believe an attacker with limited resources, such as a script kiddie or hacktivist, using publicly accessible tools and exploits alone, would have adequate knowledge and resources to achieve targeted operational process manipulation, while simultaneously evade detection. Through use of a testbed environment, this paper provides two practical examples based on a Man-In-The-Middle scenario, demonstrating the types of information an attacker would need obtain, collate, and comprehend, in order to begin targeted process manipulation and detection avoidance. This allows for a clearer view of associated challenges, and illustrate why targeted ICS exploitation might not be possible for every malicious actor.

KEYWORDS

ICS; SCADA; OT; Reconnaissance; MITM

1 INTRODUCTION

Industrial Control Systems (ICSs) play a crucial role in the monitoring, control, and automation of operational processes, some of which form part of a nations critical infrastructure [7]. Example industries include water, oil, gas, electricity, and manufacturing.

A variety of works have sought to identify security challenges in devices [4], network protocols [24], and the overall cyber security culture [14] of ICSs. Although these works provide a solid foundation for the identification of social and technical challenges within the context of ICSs, they have also unintentionally contributed to the notion that ICSs are an easy target for a variety of malicious actors. While there is truth to this notion, when seeking to develop a targeted attack in which one achieves unauthorised, undetected, physical process manipulation (as opposed to more opportunistic disruption caused through the use of denial of service (DoS) tools, for example), the assumed ease of attack can be challenged.

There exists little knowledge with regards to the collection of required information about a target ICS, its comprehension, and inclusion within a targeted attack. In this paper we demonstrate how and where an attacker might obtain the required information to achieve a desired level of "process comprehension", a term we coined to describe the understanding of system characteristics and components responsible for the safe delivery of service (e.g. treatment of water). This includes all relevant physical and computational attributes. In addition, we discuss how inability in obtaining essential information can become a significant hurdle for attackers. Through the use of two practical examples, we demonstrate how a targeted attack on operational processes can be achieved at the network layer, and host layer, further highlighting the aforementioned complexities.

The practical examples selected for inclusion here, are based on Man-In-The-Middle (MITM) attack scenario, due to their prevalence in existing works e.g. [6, 21, 25, 33, 35, 37], with execution typically achieved on unauthenticated and unencrypted protocols, widely considered to be an easy task. While engagement with readily available MITM tools [34] can quickly achieve some results, targeted manipulation would require broader contextual knowledge. What system/application is being attacked? How does the system fail? What components, functions, or variables need to be manipulated in order to achieve the desired effect? etc. all of which feed into a requirement for high levels of process comprehension. The discovery of information aiding this process is often referred to as reconnaissance, and presents the most essential step for successful attack execution.

Reconnaissance of operational processes and control infrastructure configuration/design, requires specialised knowledge. Moreover, due to a large number of vendors and the proprietary nature of individual components, open source tools for data gathering are not readily available. This knowledge is not easily obtainable from public sources, and is therefore often accumulated through hands-on work in real-world facilities. Therefore, this paper provides a step-by-step example of ICS reconnaissance, required for the successful establishment of process comprehension, and execution of network-based and host-based MITM attacks.

As we can not demonstrate reconnaissance and subsequent attacks on a real-world ICS, we have opted to derive our discussion and examples from a testbed environment. Therefore, the following section provides high-level details of the testbed in question.

CPS-SPC'17, November 3, 2017, Dallas, TX, USA

ACM ISBN 978-1-4503-5394-6/17/11…$15.00
https://doi.org/10.1145/3140241.3140254

1.1 Experimental testbed

ICSs are typically large complex environments. When analysing a full-scale ICS, it is easy to become overwhelmed with complex interdependencies and details. In order to provide a meaningful, yet easy-to-follow discussion around process comprehension, we have sought the use of a small scale, but realistic testbed environment [11, 13, 29]. Our testbed is designed to mirror typical architectures for local and remote monitoring of a physical process. Furthermore, all equipment and applications are selected to represent typical hardware and software found in real-world utilities. Figure 1 illustrates the architecture or our testbed. A Siemens TP1500 Human Machine Interface (HMI) is responsible for local process monitoring and control via a Siemens ET200S programmable Logic Controller (PLC), whereas a Schneider ClearSCADA Master Terminal/Telemetry Unit (MTU) is responsible for remote process monitoring via a Schneider SCADAPack32 Remote Terminal/Telemetry Unit (RTU). It addition, the MTU communicates with a KepWare Historian for collection of historical operational data (non-alarm based). All of which will play a crucial role in the attack design and execution.

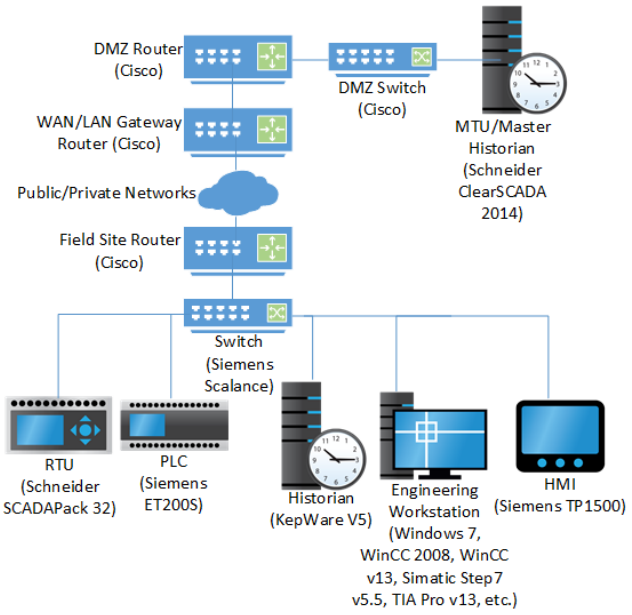

Figure 1: Network diagram of the testbed

Through use of the described testbed, we explore the process of information gathering required to conduct a targeted attack, including associated sources. In defining a selected damage scenario, we highlight the gaps and challenges of individual data sources, their unique context specific nuances, and propose alternate sources where possible. In this way we emulate the task of process comprehension, and illustrate how obtained knowledge is used during the attack phase.

Key differences existing in the targeting of ICSs vs. traditional IT infrastructures. Therefore, the following section is included to cover salient points, providing a baseline for further discussion throughout the remainder of this paper.

2 PRELIMINARIES

The work of Assante and Lee [3] provides a description of the process a malicious actor may step through when targeting an operational facility, referred to as "The Industrial Control System Cyber Kill Chain". This work discusses the difference between targeting conventional IT systems vs. ICSs, and acknowledges variations in attack difficulty dependent upon adversary goals.

While the ICS cyber kill chain offers a high level view, with case study examples, the large task of "Attack Development and Tuning" is presented as a single stage, with few details provided on how an attacker passes from reconnaissance to attack execution. Furthermore, details about specific information an attacker is required to obtain for attack development, and means of its collection, are also limited. The methods by which an attacker obtains information may influence his ability to proceed through the kill chain. As we demonstrate in subsequent sections, PLC control logic, for example, if retrieved directly from the PLC, may not contain sufficient details to allow for a complete understanding of its function, therefore additional information gathering and offline work would be required.

In traditional IT exploitation, a common goal is to remain undetected. In most ICS scenarios this is not possible. Attacks on ICSs, often referred to as cyber-physical attacks, alter process physics in the real-world, which cannot be hidden by simply erasing log files. If equipment is damaged, or a plant is misbehaving, it will be noticed and addressed. Unless the attackers intention is to create instantaneous impact, any targeted attack must induce a "hiding" element, directed at preventing immediate response (See Figure 2). Response prevention is achieved by "blinding" an operator and/or a control system responsible for operational process monitoring and response, this includes data historians responsible for long-term storage of process data. Launch of a hiding element must to be initiated prior to process manipulation. In the context of our testbed, to hide the effects of a targeted attack, the adversary would be required to blind the local HMI and remote MTU. With that, the scope for process comprehension and weaponisation sees a significant increase.

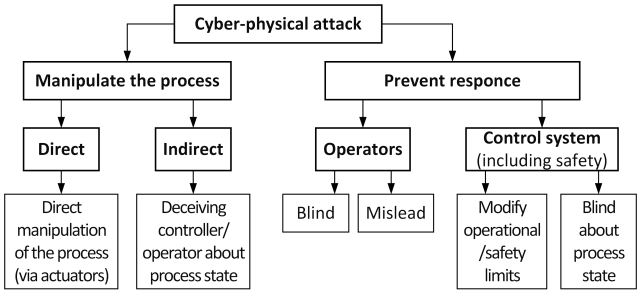

Figure 2: Constituents of cyber-physical attack

In differentiating attacker requirements between conventional IT systems and ICSs, questions on how sufficient process comprehension can be achieved as an unauthorised person are raised. At its core, when targeting ICSs, an attacker will ideally have a fundamental understanding of engineering concepts and practices

applied within the target domain. This will act as a starting point, with baseline knowledge of the information required to execute a successful targeted attack.

2.1 Points and Tags

One unique characteristic seen across ICSs is their use of points. A concept not applied in conventional IT systems, points provide a key differentiator between these two distinct domains. Points are responsible for all aspects of an ICS, essentially denoting a data source or controllable function. Each point is assigned a unique ID (e.g. I1.1, DB1.DBX1.1, etc.), these can be input channels, memory locations, etc. on top of which more logical "friendly" names can be assigned (e.g. "Valve 1"), these are referred to as tags. Points can be *soft* or *hard*. A hard point denotes a physical input or output to/from sensors and actuators. A soft point denotes results derived from mathematical calculations and logic actions. From a software perspective, a tag is considered much the same as a variable name, typically allocated as per industrial site naming conventions. In essence, points and their associated tags, link external devices and process parameters with ICS applications, and can have either Read, Write, or Read/Write permissions.

Discovering relationships between physical processes, points, tags, network protocols, monitoring and control functions, etc. is one of the most time consuming task in achieving adequate levels of process comprehension. The following section provides example data sources from our testbed, demonstrating the availability of information across multiple devices, with no single device providing a complete picture.

3 DATA SOURCES

In this section we take a selection of operational resources and discuss the information they can provide towards process comprehension. Not all ICS environments will contain certain devices, or device functionality will be integrated (e.g. RTUs may have the required computational resource and functionality to act in a dual capacity, i.e. as an RTU and data historian), this all adds to the challenge of achieving adequate process comprehension. The included examples provide a base for further discussion, and a reference point to Sections 4 and 5.

3.1 PLC Configuration

PLCs provide the computational resource by which physical operational processes can be automated, controlled, and monitored. Therefore, in the eyes of an attacker, PLCs are an attractive target for achieving targeted operational process manipulation. While an attackers end goal will be to understand which points/tags must be manipulated to achieve the desired outcome, a PLCs configuration can provide a wealth of additional information, all of which can be applied during various stages of process comprehension. To aid further discussion, information which can be obtained through PLC configuration includes:

- Core design of the operational process
- Operator ability to control (manual vs. automated process control)
- Use of sensors and actuators
- Addressing schemes and data point structures
- Controller-integrated safety and alert functions
- Network-related information (protocols, interfaces, addresses/IDs of neighbouring devices, etc.)

Depending upon the source from which an attacker obtains PLC configuration, and how the PLC has been configured, greatly impacts the detail of information provided. Taking two examples from one Siemens ET200S PLC, these differences can be observed.

Figure 3 depicts an example of PLC logic taken from a control engineer workstation/laptop or backup server. Acquisition of the control logic file (or project file) can be achieved through the use of conventional, non-ICS specific, attack tools and techniques. Once obtained, the use of official vendor software would be the easiest and most convenient way for an attacker to review such a resource. As can be seen in Figure 3, the example rung of logic has a title and accompanying descriptive text, which, among other things, provides an initial mapping between points and operational process components (Valve 2). With this metadata, an attacker can quickly begin to understand the rung's purpose. In addition, should the attacker view the contents of Data Block (DB) 1 (the area of memory in which the points shown here reside), accompanying properties of each point could also be provided. These include the point address (e.g. DB1.DBX10.1), its format (e.g. Boolean, REAL, etc.), and descriptive text (e.g. "Valve number 2"). In this example, logical friendly tag names have not been applied to replace point addressing, if applied, further descriptive text would be available here.

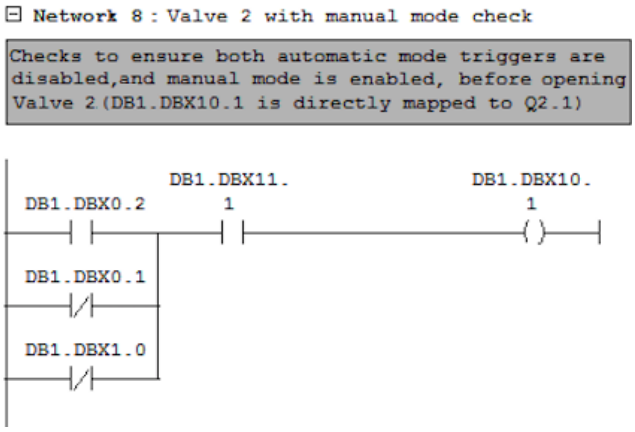

Figure 3: Commented PLC logic

Figure 4 provides an example of control logic taken directly from the PLC. Its acquisition could be achieved through the use of official vendor software, existing public exploits [5], or the development of custom tools, possibly making use of ICS specific programming modules/resources [31]. This example presents the worst case scenario, when data point addressing is applied directly within the logic. When no title, descriptive text, friendly tag names, etc. are provided, it becomes extremely difficult to analyse control logic without additional information. It is worth noting, that should friendly tag names have been applied, they can be extract directly from a Siemens PLC [4].

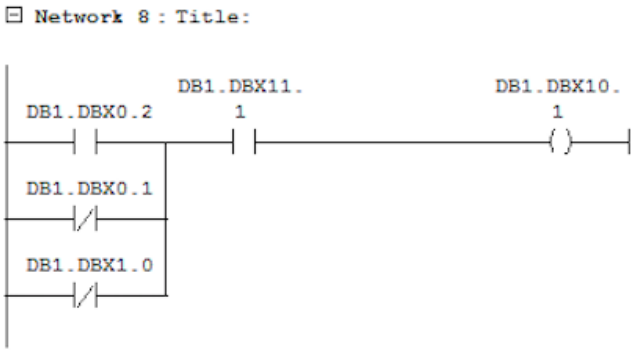

Figure 4: Un-commented PLC logic

In the analysis of PLC configuration, obtaining a greater level of process comprehension is made possible, further allowing for the identification of target points for successful process manipulation. Where information may be absent, point descriptions for example, access to alternate operational resources could prove highly valuable when examined in collaboration. Collation of data from historians, HMI/workstations, or wiring/P&ID Diagrams, would offer some if not all information required to fill gaps.

3.2 HMI/Workstation Configuration

Compact HMIs (HMI panels) and larger scale operator workstations, offer an interface by which users can monitor and control physical operational processes. Dependent upon the scale of processes operations, these can range from a single touch screen unit, used a handful of times a week, to multiple large-scale desktop workstations, manned 24/7. In the examples provided here, we are explicitly considering local HMIs/workstations, not those used within remote management centres as part of RTU/MTU systems.

In the eyes of an attacker, HMIs can be considered both a valuable target and added complication. Valuable in that their functionality can be harnessed by an attacker to manipulate operational processes as an authorised user would. Typically containing a graphical representation of the process architecture, serving as an additional source of information. However, HMIs also offer a window into attacker efforts (process state), an added complication for the attacker to address in the form of a hiding element within the attack process. Information which can be obtained through HMI configuration includes:

- Operator view into the core operational process
- Operator ability to control (manual vs. automated process control)
- Operator view of sensors and actuators (including added scaling)
- Addressing schemes and data point structures
- HMI/controller-integrated safety and alert functions
- Network-related information (including protocols, interfaces, neighbouring devices, etc.)

As with the PLC control logic, depending upon the source from which the attacker obtains HMI configuration, and how the HMI has been configured, greatly impacts the detail of information provided.

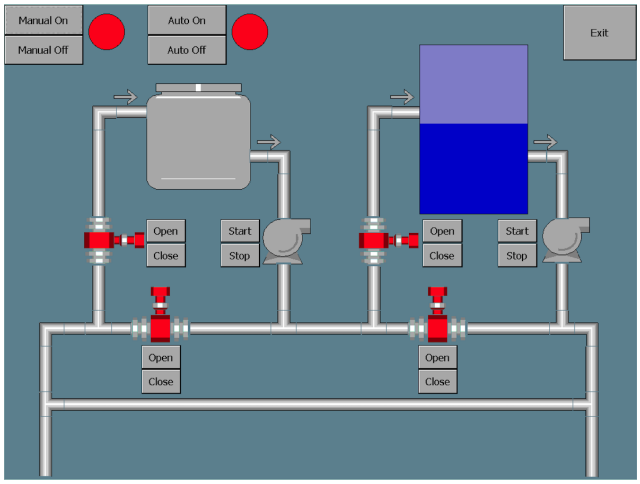

Figure 5: HMI GUI

Taking two examples from one Siemens HMI, these differences can be observed.

Figures 5 and 6 provide examples of HMI configuration taken from an engineer workstation/laptop or backup server. Acquisition of which can be achieved through the use of conventional, non-ICS specific, attack tools and techniques. Once obtained, the use of official vendor software would be the easiest and most convenient way for an attacker to review such a resource. This view offers a very quick and easy way for the attacker to understand not only how the interface looks like to the operator, but what functionality it offers. Additional nuances related to information that can be obtained from HMI configuration is rarely discussed (e.g. scaling of PLC data), yet can all be seen off-line through this resource. While the attacker can model process behaviours through a HMIs runtime or associated network traffic, having the ability to view additional factors off-line (tag names, applied scaling, etc.) provides a valuable secondary resource.

Name	Connection	Data type	Address
Auto Mode HMI On/Off	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 0.1
Manual Mode On/Off	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 0.2
Pump 1 Manual On/Off	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.2
Pump 1 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.2
Pump 2 Manual On/Off	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.5
Pump 2 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.5
Valve 1 Manual Open/Closed	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.0
Valve 1 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.0
Valve 2 Manual Open/Closed_0	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.1
Valve 2 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.1
Valve 3 Manual Open/Closed_1	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.3
Valve 3 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.3
Valve 4 Manual Open/Closed_2	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 13.4
Valve 4 State	Field Site 3 PLC (ET200S)	Bool	DB 1 DBX 10.4
Water Level	Field Site 3 PLC (ET200S)	Real	DB 1 DBD 2

Figure 6: View of the control logic tags on the HMI

HMI configuration taken directly from the HMI/workstation, is likely to be a compiled runtime file, and therefore not an easily readable/editable configuration file as previously discussed (however, we acknowledge that some organisations may choose to store backup configuration files directly on the HMI station). Therefore, while the information presented in Figure 5 would still be accessible (via execution of the compiled runtime files in official vendor software), extraction of additional information would require a more creative approach. For example, should the attacker load this runtime file while connected to the operational network, a view of live operational processes, and associated network traffic could all be captured and modelled. Alternatively, given adequate time, an attacker may choose to view the runtime file in a HEX editor, thus allowing for extraction of the same information found in the full configuration file. Figure 7 provides an example of this, with a PLCs IP address details highlighted (one of the PLCs this HMI communicates with). However decoding all relevant information would not be a simple task, consuming significant time and effort.

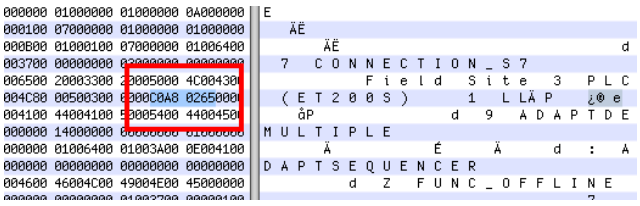

Figure 7: View of HMI configuration file in HEX editor

The application of information discussed here towards achieving a high level of process comprehension is significant, especially where an attacker is required to hide his actions from 24/7 monitored operational environments. Furthermore, the use of data as seen in Figure 6 can be collated with that discussed in Section 3.1 to close gaps opened through access to control logic acquired directly from the PLC (i.e. missing friendly tag names, descriptions, and engineer comments). Should an attacker only have access to runtime files, the techniques described here can be applied to obtain more comprehensive configuration details. However, collation of information from alternate sources, such as the PLC, could also be used if in a fully documented format (i.e. containing logical tag names, descriptions, and engineer comments).

3.3 Historian Configuration

Data historians are responsible for the collection of operational data across an industrial facility, usually via PLCs or RTUs, providing a centralised point, from which the data can be forwarded to top end systems (sometimes cloud-based) for data analytic purposes. While the use of this data is not related to raising operational alarms, users of the data could challenge data quality should they feel it exceeds normal limits, therefore acting as an indication of operational issues.

Information which can be obtained through historian configuration includes:

- Addressing scheme and data point structure
- Data points of importance to decision makers

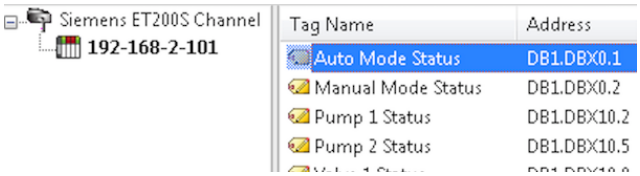

Figure 8: Historian GUI

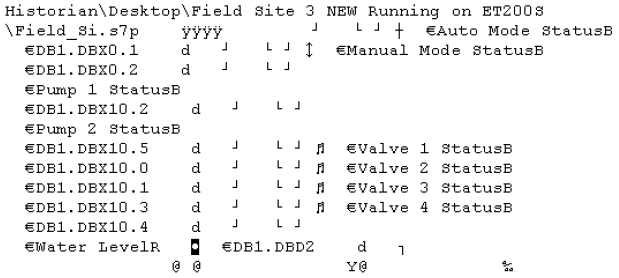

Figure 9: Historian configuration file open in text editor

- Network-related information (including protocols, interfaces, neighbouring devices, etc.)

As with the aforementioned PLC and HMI, historian configuration can be obtained from multiple locations. However unlike these devices, in the example provided here, there is little difference in the information available across each location. Our two examples are based on the Kepware v5 historian software package. As this is a Windows based application, the use of conventional attack tools and techniques can be applied in its retrieval. Figure 8 depicts a configuration example taken directly from the historian, and viewed using the vendors official software. As can be seen in this figure, the use of official vendor software provides a clean and easy way for attackers to begin reviewing this resource. Should official software be unavailable to an attacker, alternate approaches can be adopted. Figure 9 provides an example of configuration opened in a standard text editor (Notepad). Although this viewing option is less informative than that achieved through official vendor software, significant details on points are still presented. This includes friendly naming conventions mapped directly to point addressing.

Although less prevalent and detailed in configuration compared to a HMI, historians can also offer a valuable resource for process comprehension. While alarm generation is not a common function, any alarm states will be recorded, ultimately providing a window into an attackers actions, so must be considered as part of any attack strategy. Furthermore, the use of information as seen in Figures 8 and 9 can be collated with those discussed in previous sections to close gaps opened through data accessed directly from a PLC and HMI (i.e. missing friendly tag names, descriptions, and engineer comments).

3.4 Network Traffic

Network traffic offers insight into the relationships between components across an industrial facility. When seeking to achieve targeted process manipulation, understanding traffic flows around the network is of significant value, allowing for a better understanding of

```
▸ Frame 100: 181 bytes on wire (1448 bits), 181 bytes captured (1448 bits) on interface 0
▸ Ethernet II, Src: Siemens-_4b:c1:f0 (28:63:36:4b:c1:f0), Dst: SiemensA_f5:12:af (00:0e:
▸ Internet Protocol Version 4, Src: 192.168.2.100, Dst: 192.168.2.101
▸ Transmission Control Protocol, Src Port: 49158, Dst Port: 102, Seq: 2795, Ack: 1695, Le
▸ TPKT, Version: 3, Length: 127
▸ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
▾ S7 Communication
  ▸ Header: (Job)
  ▾ Parameter: (Read Var)
      Function: Read Var (0x04)
      Item count: 9
    ▸ Item [1]: (DB1.DBX 0.2 BIT 1)
    ▸ Item [2]: (DB1.DBX 10.1 BIT 1)
    ▸ Item [3]: (DB1.DBX 10.0 BIT 1)
    ▸ Item [4]: (DB1.DBX 10.3 BIT 1)
    ▸ Item [5]: (DB1.DBX 10.5 BIT 1)
    ▸ Item [6]: (DB1.DBX 10.2 BIT 1)
    ▸ Item [7]: (DB1.DBX 10.4 BIT 1)
    ▸ Item [8]: (DB1.DBX 0.1 BIT 1)
    ▸ Item [9]: (DB1.DBX 2.0 REAL 1)
```

Figure 10: Traffic between HMI and PLC

what/how devices are interacting with operational data, beyond the review of individual device configuration.

Information which can be obtained from the network traffic includes:

- High-level data flows (e.g. networked systems interactions)
- Granular system-system data requirements (e.g. functionality, protocols, data read/write parameters, etc.)
- Operational process behaviour (e.g. which process data passed between devices, at which frequency, and under which conditions)
- Numerical process data, status of equipment, alarms, etc.

Figure 10 provides an example of traffic captured between a HMI and PLC, viewed in the open source packet analyser Wireshark [36]. There are many ways in which data such as this can be captured. Approaches include compromising network equipment allowing for direct capture, ARP poisoning, wireless sniffing, etc. Most approaches can be considered non-ICS specific. However, where we see an increase in the use of wireless sensor technologies (operating over less conventional radio frequencies), for example, attackers may be required to adopt more tailored approaches. Depending on the protocol in use, viewing traffic captures with packet analysers as shown here can provide an easy method by which information extraction can be achieved. In the case of proprietary protocols, an attacker would be required to develop customised parsers.

Although breaking down a protocol from network traffic can be achieved through the use of readily-available tools, challenges arise in the interpretation of extracted information. Collating network traffic and information from any of the previously discussed sources, would yield a greater depth of detail on operational process behaviour, significantly contributing to an attackers level of process comprehension. In short, reviewing network traffic presents a detailed map of information flows and data parameters. Should an attacker be able to establish what each data point within the network traffic represents (e.g. a specific sensor/actuator), traffic capture becomes a useful source for comprehensive operational process behaviour modelling.

3.5 Piping and Instrumentation Diagram

Piping and Instrumentation Diagrams (P&ID) are a valuable source of information on process design. P&ID presents a schematic illustration of the functional relationship of piping, instrumentation, and system components used in the operational process. P&ID includes unique identifiers for core components. In collaboration with

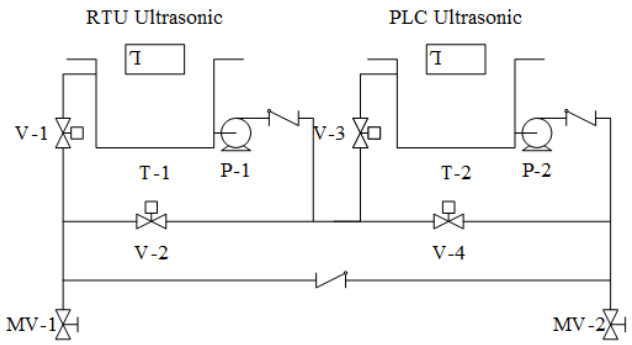

Figure 11: P&ID of the testbed

wiring diagrams, P&ID provide a base-line view of the controllers (PLC/RTU) monitor and control capability within the operational processes (i.e. what aspects of the operation process controllers have visibility and control over through associated sensor and actuators). Therefore, as an attacker resource the value of accurate and comprehensive P&IDs is significant.

Information which can be obtained from the P&ID includes:

- Complete design of the operational process
- Detailed list of the operational equipment, its type and sizes
- Mechanical equipment (which is often not included in control logic)

Figure 11 shows an example P&ID depicting our testbed environment. The style and notation of P&ID may differ between facilities, this is due to adherence against differing standards, as well as individual facility regulations and engineers personal preferences. As this data would be stored on a conventional IT based resource (engineer workstation/laptop or backup server), non-ICS specific tools and techniques would be used in its acquisition. Once obtained, basic graphics packages can be used to perform a review.

The other related documentation on operational process design may include One-Line Diagrams (often contain pertinent information on safety interlocks), Cause & Effect Diagrams, Cable Schedule Diagrams, Project Interconnection Diagrams, and Instrument I/O (input/output) Lists. While the use of this information in attack planning is clear, two points are of particular importance. Firstly, when collated with information derived from a PLC's configuration, gaps resulting from un-commented configuration can be closed, allowing an attacker to achieve significant process comprehension, including the PLC's role and function. Secondly, any non-controlled based (mechanical) safeguards can be observed and considered in the attacks design. Taking an example from Figure 11, we can see this process employs check valves, physically preventing the flow of water in certain directions. Therefore, attacker objectives may need to be adapted accounting for incurred limitations.

3.6 Other (Policies, Procedures, Reporting Functions, System/Component Constraints, etc.)

There exist many other sources of useful information. Examples include operational process policies, procedures, reporting, device/system constraints, social/organisational structures, etc. some

of which will be available and applicable to members of the public. Prominently written by technical authors, these documents include details on how physical processes should be operated and maintained. Other examples include reporting procedure (e.g. periodic physical (non-controller based) checks), guidelines for members of the public who observe suspicious/dangerous situations in service delivery (e.g. taste of water) or within operational facilities (e.g. visibility of smoke/fire inside an operational resource), and technical specifications of individual components (e.g. for how long a centrifugal pump can run dry before damage is incurred). Some of this data is available within the public domain, e.g. manufacture websites. Other (proprietary) data will be stored on conventional IT based resources (e.g. engineer workstations/laptops and backup servers), meaning non-ICS specific techniques can be applied to their acquisition.

Recent attacks in the Ukraine offer a useful example of how this type of information can be applied by an attacker. Once successful manipulation of operational processes had been achieved, the attacker identified reporting services offered to members of the public. These reporting services were provided in the form of a call centre able to act on any issues in service delivery. The attacker caused a DoS to phone lines use by this service, effectively preventing the public from providing any information which could have been used to collate control system data with actual impact across the estate [23].

3.7 Attacker Goal

Through the use of data presented across previous sections, subsequent sections will discuss how a targeted process manipulation attack can be achieved, while simultaneously addressing the challenge of detection avoidance. The attacker goal is to take water from Tank 1 (T-1), and move it into Tank 2 (T-2), causing Tank 2 (T-2) to flood (See Figure 11). Although simple in nature, the level of process comprehension required is still significant. Two example attacks are provided to aid this discussion, both with the same objective, however one will be executed on the network level, the other on the hardware level.

4 NETWORK BASED MITM ATTACK

This section describes the processes an attacker needs to go through in order to achieve targeted process manipulation and detection avoidance in our testbed environment at the network level. Discussions related to obtaining network access, and evading security controls (network segregation, firewalls, IDS/IPS, etc.) are out of scope.

4.1 Network/Device Enumeration

Network and device enumeration presents one of the initial challenges faced by an attacker. For this task, off the shelf tools such as NMap[28] offer both ICS and non-ICS specific enumeration capabilities. In addition, more focused tools can be applied at this stage, e.g. PLCScan [30]. In addition to the use of tools to actively perform device enumeration, the use of P&ID, network diagrams, wiring diagram, PLC configuration, HMI configuration, historian configuration, etc. could all offer insight into devices operating on the IP network, including their associated addresses.

```
if (ip.proto == TCP && tcp.src == 102 && DATA.data+20 == "\x09") {
    msg("Found 9");
    DATA.data+24="\x00";
    DATA.data+30="\x00";
    DATA.data+36="\x00";
    DATA.data+42="\x00";
    DATA.data+48="\x00";
    DATA.data+54="\x00";
    DATA.data+60="\x00";
    DATA.data+66="\x00";
    DATA.data+72="\x42\x96\x96\x96";
}
```

Figure 12: Example of the static EtterCap filter

4.2 MITM Attack on the HMI

Prior to the execution of process manipulation, it is critical to ensure any means by which attacker actions can be observed are considered and addressed. The first and most prevalent device in this testbed use-case is a local HMI. While the use of existing tools and attack techniques is somewhat trivial, applying them to specific operational processes is not. To execute an effective MITM attack on the HMI, there exist a set of prerequisites. Establishing the level of operational process visibility (process measurements, equipment status, alarms, etc.) in parallel to control functions (e.g. start/stop pumps) available to operators is critical. Detailed knowledge of process visibility on the HMI is key, as it provides a starting point on which one can design a MITM attack. Specifically, which monitoring data points offer a window into attacker actions, and therefore must be substituted with the spoofed values, depicting "normal" operational behaviour. This information can be captured through access to the HMI configuration. Once understood, obtaining access to live operational data feeding the HMI is required to build a picture of normal process values, and in the case of our testbed, understand the sequence by which data is requested by the HMI from the PLC (see Figure 10).

Access to the operational network will allow an attacker to adopt several approaches to this task. While we acknowledge there are examples of capturing network traffic over a period of time and simply replaying it to the HMI, such an approach lacks intelligence and if poorly applied could indicate an ongoing attack (e.g. by replaying responses to operator requests which have not occurred).

Through the use of an Ettercap [9] and an associated filter we are able to prevent all legitimate traffic from the PLC reaching the HMI, in its place static values will be sent. This filter will make the operational process appear static, with the water level in Tank 2 remaining unchanged at what can be considered a normal value (based on a review of previously acquired network traffic).

Figure 12 provides an example Ettercap filter constructed specifically for our testbed attack scenario. The highlighted code depicts identification of network traffic parameters specific to the target, prior to the manipulation of values.

The example provided here is relatively simple. A more sophisticated MITM attack should seek to offer interactive functionality (i.e. acknowledge user interactions and respond accordingly with spoofed data). However, as described here, even when creating the simplest of MITM attacks, significant effort is required to understand what values need be falsified, how they appear within the protocol, and ensure they do not exceed normal limits.

This same approach can be applied towards the prevention of historian visibility into attacker actions.

4.3 Replay Attack on the RTU

Through a review of Figure 11 we can see an additional sensor has been placed on Tank 1, feeding directly into the RTU. It is common practice to see a single sensor used to feed more than one device, or to apply duplicate sensors across operational processes. While this adds a level of resilience, we have seen actions such as these taken for more practical reasons (i.e. the RTU being unable to extract data from PLC, thus requiring additional hard wired signals). RTU configuration can be acquired and analysed in a similar way to that of the PLC. Hence its exclusion from Section 3. As can be inferred through a review of our P&ID, the sensor value associated with Tank 1 provides a direct indication of what is happening in Tank 2. It is for this reason the RTU must be considered within the attack scenario. Furthermore, as its core function is to generate alarms for remote operational process monitoring, data from this sensor could be used to immediately notify remote operators.

In the context of our testbed, we opted to build a replay attack script through experimentation with the RTU and official vendor software. A similar approach was adopted by [4]. This script (see Figure 13) changes the IP address parameters of the RTU, taking it offline. However, out of ethical considerations, the payload has been excluded from our figure, preventing re-use against similar RTUs.

This attack impacts the RTU's ability to communicate with the MTU, and therefore remote operators. This is not as risky as it may seem at first glance. RTUs may only communicate with central systems a handful of times a day (unless an alarm is generated requiring immediate attention). In addition to this, it is common to see communication failures. These could be due to low quality remote communication mediums, power outages, ageing equipment, etc. Reviewing resources as described in Section 3.6 can offer further details on service level agreements, communication windows, etc. An additional analysis of network traffic would allow the attacker to map these details to operational data, all aiding the selection of an optimal attack window for maximum disruption of process observability.

4.4 Control Request Injection

Once effective blinding of local and remote operators has been achieved, the attacker can proceed with execution of our pre-defined damage scenario (overfilling Tank 2). A comprehensive understanding of not only PLC configuration, but also electrical/mechanical safeguards, component limitations, etc. are all required at this stage. Access to the P&ID or HMI configuration would provide a good start point, offering a baseline understanding of how the process could be manipulated (i.e. which pumps and valves need to be run/opened). The P&ID can also provide insight into relevant safeguards. A review of PLC configuration is then required to identify controller based safeguards, and confirm appropriate data points to modify in order to achieve the desired impact.

The use of programming aids [31] provide a simple way in which attackers can send requests to the PLC. However, the construction of those requests must be carefully thought out. Figure 14 provides an example attack script, creating specially crafted requests using Snap7 [31]. From a review of the PLC configuration, the attacker would be able to identify one safety mechanism. Automatic mode

```
import socket

TARGET_IP = "192.168.2.200"
MODBUS_TCP_PORT = 502

PAYLOAD = '0000000000000000'\
          '0000000000000000'\
          '0000000000000000'\
          '00'.decode('hex')

try:
    print "Connecting."
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print "Connecting.."
    s.connect((TARGET_IP, MODBUS_TCP_PORT))
    print "Connected"
    print "Sending Payload"
    s.send(PAYLOAD)
    s.close()

except:
    print "Failed, Please Try Again."

else:
    print "Payload Sent"
```

Figure 13: Example of the replay script in Python

must remain disabled, with manual mode enabled, to be able to start Pump 1 and open Valve 3, resulting in Tank 2 becoming overfilled. As can be seen in Figure 14, there are three payloads within this script, each writing values to the different areas of PLC memory, accounting for these requirements. The highlighted code denotes information specific to the testbed environment i.e. PLC IP address, hardware profile, and target data block. Should an operator attempt to interfere with this attack via the HMI, their requests would be immediate overwritten by the attack script, as it continues writing to the PLC until interrupted by the attacker.

```
import snap7

TARGET_IP = "192.168.2.101"
TARGET_RACK = 0
TARGET_SLOT = 2
TARGET_DB_ONE = 1
TARGET_BYTE_ONE = 0
TARGET_BYTE_TWO = 13
TARGET_BYTE_THREE = 1
PAYLOAD_ONE = bytearray(b'\x04')
PAYLOAD_TWO = bytearray(b'\x0C')
PAYLOAD_THREE = bytearray(b'\x00')

try:
    print "Connecting."
        client = snap7.client.Client()
    print "Connecting.."
        client.connect(TARGET_IP, TARGET_RACK, TARGET_SLOT)
    print "Connected"
        while True:
            client.db_write(TARGET_DB_ONE, TARGET_BYTE_ONE, PAYLOAD_ONE)
            client.db_write(TARGET_DB_ONE, TARGET_BYTE_TWO, PAYLOAD_TWO)
            client.db_write(TARGET_DB_ONE, TARGET_BYTE_THREE, PAYLOAD_THREE)
            print "Sending Payload"
            print "(Enter ctrl+c to End)"
except KeyboardInterrupt:
    print "Stopped"
    print "Disconnecting"
    client.disconnect()
    print "Disconnected"
except:
    print "Failed, Please Try Again"
```

Figure 14: Example of the command injection script in Python

Although the example attack presented here is simple, it shows the level of information required in order to achieve successful process manipulation. It would be advisable for an attacker to extract data from the PLC where possible, to identify any issues arising from the attack. For example, where a centrifugal pump run dry time has been identified, ensuring the pump does not reach defined limits may be required as to not raise the attention of operators or members of the public who may hear or see signs of pump damage, resulting in further investigation/reporting.

5 HOST BASED MITM ATTACK

This section describes the processes an attacker would go through in order to achieve targeted process manipulation and detection avoidance at the host level. Compared to the network based attack, exploitation at the host level has some advantages. The first being that PLCs are usually responsible for controlling the underlying operational process. Therefore, attacking the most essential component becomes an obvious choice. However, additional motivations in relation to the lack of protection mechanism developed for such devices can also be a contributing factor [1]. Furthermore, we have seen a recent increase in code-reuse vulnerabilities within PLC Operating Systems [18–20], adding to the relevance of host based attacks. Here we present a Pin Control Attack, as a method to execute a targeted MITM on the PLC I/O, a technique we first presented in 2016 [1].

5.1 Pin Control Attack

Pin Multiplexing. System on Chips (SoCs) usually employ hundreds of pins connected to the electrical circuit. Some of these have a single defined purpose. Since different equipment vendors with diverse I/O requirements will use SoCs, the SoC manufacturer produces its SoCs to employ a defined physical pin for multiple mutually exclusive functionality, depending on the application [22]. The concept of redefining the functionality of a pin is called Pin Multiplexing, and is one of the necessary specifications of the SoCs design [10]. Figure 15 illustrates the concept of pin multiplexing. Part 1 shows a general view of the SoC. Part 2 depicts pins with different I/O peripheral options. As can be seen, the SoC has multiplex pins for JTAG/SPI, SPI/GPIO, MMC/GPIO and I2C/GPIO. Each multiplex pin gives a vendor options to choose between those two functions. Part 3 shows how SoCs peripherals are located inside a SoC, and how one multiplex pin is connected to two peripherals.

Pin Configuration. Embedded SoC I/Os (e.g. ARM, MIPS, or PowerPC) are controlled with a pin based approach, and must be configured accordingly, otherwise they can not function correctly. Once the system starts, I/O can be configured by applications or drivers, by first mapping them into virtual memory. PLCs must configure pins that are to be used for *reading* values into *input* mode, and pins that are to be used for *controlling/writing* values to *output* mode. The PLC usually configures its pins by first mapping the physical I/O memories to a virtually mapped I/O, as illustrated in Figure 16. Through a dynamic analysis of the Codesys PLC runtime, we can observe how it maps physical I/O memory at 0x2020, to virtually mapped I/O. This virtually mapped I/O can be detected by attackers for use in a Pin Control Attack.

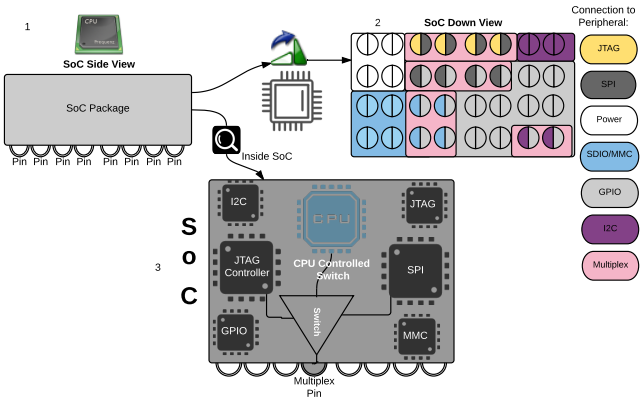

Figure 15: Pin multiplexing on SoC

```
[b6e47f54] open("/etc/3S.dat", O_RDONLY) = 8 <0.001979>
[b6df334c] close(8) = 0 <0.001878>
[b6e47f54] open("/proc/cpuinfo", O_RDONLY) = 8 <0.001354>
[b6df334c] close(8) = 0 <0.007677>
[b6f2c7e4] open("/dev/mem", O_RDWR) = 8 <0.001182>
[b6e53998] mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED, 8, 0x2020)
[b6f2bdd4] close(8) = 0 <0.001246>
[b6f2c7e4] open("/dev/[illegible]", O_RDWR) = 8 <0.001340>
[b6f2c7e4] open("/dev/spidev0.0", O_RDWR) = 9 <0.001886>
[b6e4fadc] ioctl(9, 0x80016b01, 0xb6a8d714) = 0 <0.001888>
```

Figure 16: Mapping of a physical I/O memory to a virtually mapped I/O

5.2 MITM using Pin Control

Through use of a Pin Control Attack, one can target the interaction between PLC firmware and PLC I/O. This attack causes the firmware to assume that it is interacting effectively with the I/O, while in reality connections between the I/O and PLC process are being manipulated. The greatest advantage of Pin Control attacks, is that they eliminate the necessity for hiding process manipulation effects across neighbouring devices (e.g. HMI and Historian). This is because the PLC will already "read" modified (good but spoofed) process values. However, there are three requirements an attacker must first satisfy.

PLC runtime privilege. In recent years multiple works have shown that PLCs from a variety of vendors including Siemens [4, 32], ABB [15], Schneider Electric [16, 17], Rockwell Automation [20], and WAGO [8], are vulnerable to system-level code execution via memory corruption or back-door password vulnerabilities.

Knowledge of the physical process. Based on our previously described reconnaissance/process comprehension phase, and defined attack objectives, we are required to hold the readings from our ultrasonic sensor (causing it to remain in a healthy but static state), while opening Valve 3 and starting Pump 1.

Knowledge of the PLC and mapping between I/O pins and the logic. The PLC type will allow an attacker to understand where hard-coded memory regions, such as multiplexing registers or pin configuration registers, reside. These hard-coded memory regions can be different across PLC models. Therefore, an attacker must be aware of the specific PLC in use. Additionally, an attacker must have adequate knowledge on the mapping between I/O pins and control logic. The attacker must know which inputs/outputs require

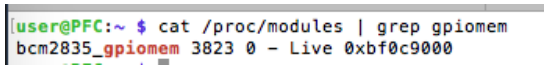

Figure 17: Request to provide virtual addresses of all digital I/O interfaces

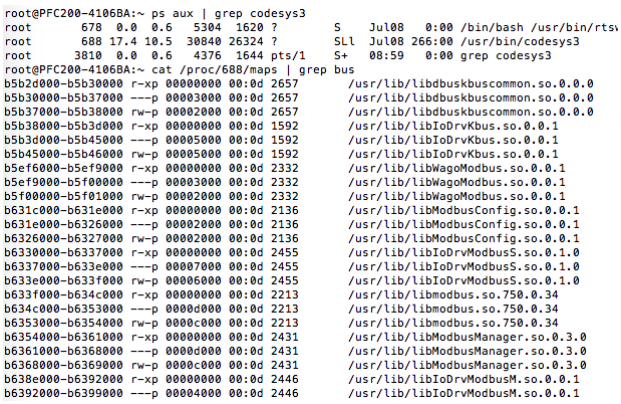

Figure 18: List of all virtually mapped I/O interfaces and their ranges

modification, to impact operational processes in the desired way. This information can be extracted from the HMI, PLC, Historian, etc., as described in previous sections. Furthermore, we note that the work presented by McLaughlin et al., [26, 27] can also be used to discover mapping between I/O variables and the physical world.

In some cases, the mapping between I/O pins and control logic is already available in the PLC OS. For example, should the attacker look to find I/O memory ranges for digital outputs (e.g. Pump 1 and Valve 3), it can simply be requested from the OS, providing the base address of the digital I/O memory (e.g. using /proc/$pid/maps or /proc/modules) and then effectively writing any desired value to it. Figure 17 illustrates legitimate request to the OS in order to find virtually mapped I/O address of the GPIO (digital input/output) interfaces. From this we now knows that all digital I/O interfaces of the PLC runtime are located at 0xbf0c9000. 0xbf0c9000 is a four-byte address, meaning that 32 digital I/O are being controlled by a four-byte memory address in the PLC SoC. Similarly, an attacker can find all analogue interfaces virtually mapped I/O addresses. Figure 18 illustrates how an attacker can find all analogue I/O interfaces virtually mapped I/O, stored in the maps file of a PLCs operating system.

In some cases, even the knowledge of mapping is not required. For example, in our testbed, since we only have one analogue interface actively available on the PLC for receiving ultrasonic readings, we can simply ask the PLC OS to provide the range of I/O memory that is being used for analogue inputs, then read the values stored in these pins. If a specific memory region changes (becomes active) it means that pin is currently reading a process value. With that, we are always able to immediately determine the I/O memory range of the ultrasonic sensor.

5.3 Process Manipulation using Pin Control Attack

One approach available to execute the attack, is through use of the multiplexing approach (since it is much simpler).

Multiplexing the ultrasonic pin. We first start by multiplexing the ultrasonic analogue pin to an alternative function by simply writing a zero value to the *analogue pin multiplex pin* (system wide available) which is connected to the ultrasonic sensor. By multiplexing the sensor pin, we prevent values being written to the analogue input pin connected to the PLC. As a result the PLC runtime can then only read previously written value from memory.

Sending command to open Valve 3 and Start Pump 1. After successfully executing the attacks hiding stage, we write value one (ON command) to the virtually mapped digital I/O memory (in our case 0xbf0c9000) for Pump One and Valve Three. This can be achieve via a single instruction. As a result, Pump 1 will start, and Valve 3 will open, overfilling Tank 2. However, the tank level value remains the same since we are in control of the analogue input pin, blocking all write operations from the ultrasonic sensor.

The RTU has its own dedicated ultrasonic sensor, so while both HMI and Historian will be blinded by this attack, the RTU will not. Therefore, one would look to apply this same approach to the RTU, made possible by similarities between the hardware used in PLCs and RTUs.

6 SUMMARY AND CONCLUSIONS

Due to the lack of basic in-built security mechanisms across the majority of ICS protocols and devices, they are often considered to be an easy target, even for a low skilled threat actors. While there is substantial evidence to support such a notion, when considering targeted attacks on operational processes, exploiting cyber vulnerabilities with conventional IT focused skills becomes insufficient. On the example of widely applied Man-in-the-Middle attack scenarios, we illustrated the effort an attacker must invest prior to final weaponisation. The mechanics of a Man-in-the-Middle attack are relatively simple and well understood, however the preliminary reconnaissance require to achieve a successful target attack is not. In cyber-physical systems security we are concerned specifically with attacks that result in physical impact. To achieve this, attackers must obtain a high level of process comprehension, identifying ways in which operational process manipulation can be achieved. This refers to discovering information about process design from documentation and configuration files. Without detailed knowledge, it is unlikely an attacker can achieve more than obvious, noticeable, immediately detectable disruption.

There is no a single source of information an attacker could consult. On the example of a small-scale test-bed, we have illustrated the types of information an attacker would need to gather, where and how it can be obtained, processed, interpreted, collated, and fully comprehended, in order to begin targeted process manipulation and detection avoidance. Understanding what information is critical for the attacker in order to achieve their desired completeness of process comprehension, is critical for planning defence activities. By restricting attackers access to critical information, e.g. point/tag-physical device mapping, effectively interrupts the kill chain. It also allows for more targeted monitoring of malicious

activities (the reconnaissance process can be noisy). Understanding what and where information exists within both enterprise and control networks, is critical for limiting an attackers ability to achieve desired levels of process comprehension. Classification of information, minimisation of information repositories, as well as strict access control and auditability, have to be prioritised, and become part of security hardening strategies.

This brings us towards an important security issue of the "path of least resistance". The attacker is likely to follow the easiest attack execution path. This is why protection of services utilising Open Platform Communications (OPC), for example, becomes critical. OPC services offer a standardised, vendor-independent, legitimate connection to ICS devices and systems, without the need to reconstruct complex point-to-tag label mapping. Discovering such likely attack avenues allows for the identification of monitoring capabilities, ideal when strategising on attack detection avoidance. In addition, achieving a comprehensive view of an ICSs social and technical make-up is of great importance. This has been discussed in our previous work [12], and continues to be of focus in future works.

Scaling of attacks when considering simple vectors as applied in Section 4.3, may be possible if the target organisation has deployed this same device across their estate. However, when considering other attacks described here, even where the operational facility is similar in nature (e.g. a water treatment works), the scale of production will vary, impacting the design of operational processes. Undoubtedly this will alter how targeted operational process manipulation can be achieved. This is before we begin to consider any diversity in equipment, the age of components, nuances in operational processes, device configurations, etc. Such diversity would allow an attacker to apply the same techniques as described in Section 4 or 5, however process comprehension (see Section 3) would be required on a facility-to-facility basis.

Where technologies are developing towards comprehensive monitoring and management of ICS networks and protocols, there too exists developments in low-level attack techniques, such as executing attack code directly at the micro-controller level. In the hacking community this is sometimes referred to as a "Race to the Bottom"; as defences are introduced at a certain level of computational resource, the attackers shift their exploitation down one level.

REFERENCES

[1] Ali Abbasi and Majid Hashemi. 2016. Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. *Black Hat Europe* (2016).

[2] Ali Abbasi, Jos Wetzels, Wouter Bokslag, Emmanuele Zambon, and Sandro Etalle. 2014. On Emulation-Based Network Intrusion Detection Systems. In *Research in Attacks, Intrusions and Defenses*. Springer, 384–404.

[3] Michael J Assante and Robert Lee. 2015. *The Industrial Control System Cyber Kill Chain.* Technical Report.

[4] Dillon Beresford. 2011. Exploiting Siemens Simatic S7 PLCs. In *Black Hat USA*.

[5] Dillon Beresford. 2017. Siemens Simatic S7-300 - PLC Remote Memory Viewer (Metasploit). (2017). https://www.exploit-db.com/exploits/19832/

[6] Jonathan Butts and Sujeet Shenoi. 2013. *Critical Infrastructure Protection VII*. Springer, Washington DC.

[7] CPNI. 2017. Critical National Infrastructure. (2017). https://www.cpni.gov.uk/critical-national-infrastructure-0

[8] DigitalBond. 2012. 3S CoDeSys, Project Basecamp. (2012). http://www.digitalbond.com/tools/basecamp/3s-codesys/

[9] Ettercap Project. 2017. Ettercap - About. (2017). https://ettercap.github.io/ettercap/about.html

[10] Prosenjit Ghosh, Pritpal Singh Hira, and Shelly Garg. 2013. A method to make SoC verification independent of pin multiplexing change. In *Computer Communication and Informatics (ICCCI), International Conference on*. 1–6.

[11] Benjamin Green, David Hutchison, Sylvain Andre Francis Frey, and Awais Rashid. 2016. Testbed diversity as a fundamental principle for effective ICS security research. In *SERECIN*.

[12] Benjamin Green, Marina Krotofil, and David Hutchison. 2016. Achieving ics resilience and security through granular data flow management. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 93–101.

[13] Benjamin Green, Anhtuan Lee, Rob Antrobus, Utz Roedig, David Hutchison, and Awais Rashid. 2017. Pains, Gains and PLCs: Ten Lessons from Building an Industrial Control Systems Testbed for Security Research. In *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17)*. USENIX Association.

[14] Benjamin Green, Daniel Prince, Jerry Busby, and David Hutchison. 2015. The impact of social engineering on Industrial Control System security. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. ACM, 23–29.

[15] ICS-CERT. 2012. ABB AC500 PLC Web Server Buffer Overflow Vulnerability. (2012). https://ics-cert.us-cert.gov/advisories/ICSA-12-320-01

[16] ICS-CERT. 2012. Schneider Electric Modicon Quantum Vulnerabilities (Update B). (2012). https://ics-cert.us-cert.gov/alerts/ICS-ALERT-12-020-03B

[17] ICS-CERT. 2014. Schneider Electric Modicon Quantum Vulnerabilities (Update B). (2014). https://ics-cert.us-cert.gov/alerts/ICS-ALERT-12-020-03B

[18] ICS-CERT. 2015. Schneider Electric Modicon M340 Buffer Overflow. (2015). https://ics-cert.us-cert.gov/advisories/ICSA-15-351-01

[19] ICS-CERT. 2015. Yokogawa Multiple Products Buffer Overflow Vulnerabilities. (2015). https://ics-cert.us-cert.gov/advisories/ICSA-15-253-01

[20] ICS-CERT. 2016. Rockwell Automation MicroLogix 1100 PLC Overflow Vulnerability. (2016). https://ics-cert.us-cert.gov/advisories/ICSA-16-026-02

[21] BooJoong Kang, Peter Maynard, Kieran McLaughlin, Sakir Sezer, Filip Andrén, Christian Seitl, Friederich Kupzog, and Thomas Strasser. 2015. Investigating cyber-physical attacks against iec 61850 photovoltaic inverter installations. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 1–8.

[22] Kernel.org. [n. d.]. Pin Control Subsystem in Linux. ([n. d.]). https://www.kernel.org/doc/Documentation/pinctrl.txt

[23] Robert M Lee, Michael J Assante, and Tim Conway. 2016. *Analysis of the cyber attack on the Ukrainian power grid.* Technical Report. http://www.nerc.com/pa/CI/ESISAC/Documents/E-ISAC

[24] Munir Majdalawieh, Francesco Parisi-Presicce, and Duminda Wijesekera. 2007. DNPSec: Distributed network protocol version 3 (DNP3) security framework. In *Advances in Computer, Information, and Systems Sciences, and Engineering*. Springer, 227–234.

[25] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. 2014. Towards understanding man-in-the-middle attacks on iec 60870-5-104 scada networks. In *Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014*. BCS, 30–42.

[26] Stephen McLaughlin and Patrick McDaniel. 2012. SABOT: Specification-based Payload Generation for Programmable Logic Controllers. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS '12)*. ACM, New York, NY, USA, 439–449. https://doi.org/10.1145/2382196.2382244

[27] Stephen E McLaughlin. 2011. On Dynamic Malware Payloads Aimed at Programmable Logic Controllers. In *HotSec*.

[28] Nmap.org. 2013. Nmap - Free Security Scanner For Network Exploration & Security Audits. (2013). http://nmap.org/

[29] Ben Paske, Ben Green, Dan Prince, and David Hutchison. 2014. Design and Construction of an Industrial Control System Testbed. In *PGNET*. 151–156.

[30] PLCScan. 2013. PLCScan - PLC Devices Scanner. (2013). https://code.google.com/p/plcscan/

[31] Snap7. 2016. Snap7 - Overview. (2016). http://snap7.sourceforge.net/

[32] Ralf Spenneberg, Maik Brüggemann, and Hendrik Schwartke. 2016. PLC-Blaster: A Worm Living Solely in the PLC. (2016).

[33] Anurag Srivastava, Thomas Morris, Timothy Ernster, Ceeman Vellaithurai, Shengyi Pan, and Uttam Adhikari. 2013. Modeling cyber-physical vulnerability of the smart grid with incomplete information. *IEEE Transactions on Smart Grid* 4, 1 (2013), 235–244.

[34] Aaron Turner. 2017. Tcpreplay. (2017). https://github.com/appneta/tcpreplay

[35] David Urbina, Jairo Giraldo, Nils Ole Tippenhauer, and Alvaro Cardenas. 2016. *Attacking fieldbus communications in ICS: Applications to the SWaT testbed.* Cryptology and Information Security Series, Vol. 14. 75–89.

[36] Wireshark.org. 2016. About Wireshark. (2016). https://www.wireshark.org/

[37] Yi Yang, Kieran McLaughlin, Timothy Littler, Sakir Sezer, Eul Gyu Im, Z Q Yao, B Pranggono, and H F Wang. 2012. Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid SCADA systems. (2012).