# Advanced Persistent Threat detection for Industrial Control Systems

Research Project 2
SNE Master
University of Amsterdam

Steffan Roobol: sroobol@os3.nl
Dominika Rusek: drusek@os3.nl

July 5th, 2020

**Abstract**—In recent years there has been a surge of Advanced Persistent Threat (APT) type of attacks on Industrial Control systems. Many organizations lack visibility into the ICS network events, preventing the defenders from taking an appropriate response. In this research, we design a Proof of Concept (PoC) application for manufacturing environments, based on the open source IDS Zeek, that analyses the network traffic and detects potential adversary techniques from the ICS MITRE ATT&CK framework. This PoC was evaluated using a data set containing 12 adversary techniques. The evaluation shows that it has a low ratio of False Negatives (with a True Positive Rate of 0.945), but a higher rate of False Positives (with a Positive Predictive Value of 0.750). The Matthews Correlation Coefficient (MCC) for the PoC application is in a range between 0.7 and 0.8, which indicates a strong positive relationship between the predicted values of our PoC and the actual values in the data set. The techniques from the ICS MITRE framework are mapped to the ICS Killchain to provide perspective where in the adversary campaign the technique can be placed. Our research shows that providing such a perspective can help interpreting an analysis of anomalies in ICS networks.

**Keywords - ICS, IDS, Zeek, ICS MITRE ATT&CK, anomaly detection, threat hunting, ICS Killchain**

## 1 Introduction

Defending Industrial Control Systems (ICS) against a wide range of adversaries is becoming a challenge due to increasing interconnectivity with IT systems, which expands possible attack paths. As Andy Greenberg described in his book "Sandworm", in recent years adversary groups like nation-state attackers are shifting their attention towards the industrial systems, as these shape the way we live in modern society [1]. From energy grids, bridges, airports, and other critical infrastructure to the manufacturing of goods we use every day, we are surrounded by ICSs.

ICS is a general term that encompasses several types of industrial systems such as Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), or Supervisory Control and Data Acquisition (SCADA), which all have strategic significance due to potentially serious consequences in case of malfunctions. As stated by Greenberg, we are in a new era of cyber war, where hackers can cripple other nations by attacking their industrial control infrastructure and hence impacting the lives of the civilians [1].

The highest priority of the ICSs is their availability and the safety of the people, which is a key difference from IT networks. Before deployment in production, hardware and software needs to be extensively tested to meet a variety of safety standards, which takes substantial amounts of time. As a consequence, industrial systems very often have no built-in or outdated security mechanisms such as encryption, authentication, or authorisation. Furthermore, the industrial systems are vastly different from the traditional IT infrastructure because of different communication patterns, the use of proprietary protocols and real-time processing of automation and control systems. Due to a lack of security controls, gaining visibility into what is happening on the network and being able to act upon it, is thus an important defense measure. However, ICS environments tend to have very limited visibility and log aggregation capabilities, which would allow the detection of and control over what is happening on the internal networks, as researched by Dragos [2]. If monitoring is in fact in place, very often it is purely an enterprise-based detection mechanism, which focuses on level three and above of the Purdue model (explained in detail in Section 2.2) and is thus not suitable for monitoring the industrial communications below those levels [3]. There are various efforts in the industry to improve the status quo, however, proposed monitoring solutions are either highly academic and difficult to implement in practice, or they are commercial solutions with high licence costs.

The lack of visibility leads to a large dwell time, which in the case of ICS networks can be above ninety days [4]. The dwell time is the time between an adversary gaining access to the network and when they are detected and the

access is severed. Usually, the detection happens too late, and by that time, the physical control systems are already damaged. Following the cyber security principle of "Assume you are compromised", this research is an attempt to improve visibility into ICS networks, by creating a Proof of Concept application (named PoC application in this paper), which is built on top of open source monitoring solutions [5]. The goal of the PoC is that it can be easily deployed in an industrial environment and is highly modular to accommodate for specific industrial environment use cases. In doing so, this research enables site engineers to pave the way towards monitoring and proactive hunting for sophisticated adversaries within their networks.

This paper is structured as follows: Section 2 presents research efforts in the area of monitoring, detecting anomalies and threat hunting in ICS, Section 3 focuses on describing our research methodology, while Section 4 and 5 present the design and implementation choices of the PoC. Next, Sections 6 and 7 present the results and discuss their practical application, followed by the conclusion in Section 8. Finally, Section 9 suggests future work that could expand upon our research.

The main research question for this project is defined as follows:

> How can network analysis be used to discover the potential presence of Advanced Persistent Threats (APT) in Industrial Control Systems (ICS)?

To support the main research question the following subquestions have been defined:

- What are the network-based attack techniques used by APT groups documented in the ICS MITRE ATT&CK framework?
- How can existing monitoring solutions be improved upon to automate the detection of APT techniques in ICS networks?
- How can the detected techniques be mapped to the ICS Killchain to recognise the stage of adversary campaign?

## 2 Related work and background

### 2.1 Intrusion Detection Systems

Due to the criticality and complex requirements of Industrial Control Systems (ICS), a lot of research is being done in the area of intrusion detection solutions specific to those environments. Mitchell and Chen held a survey on different intrusion detection systems (IDS) for ICS [6]. They categorised IDSs based on two design dimensions: their data sources and the implemented detection techniques. For the data sources, they differentiated between host-based data and network-based data [6]. For the detection techniques, they distinguished misuse based techniques, focusing on detecting signatures of malicious traffic, anomaly based techniques and comparing a system's behaviour to its normal behaviour.

Some widely known open source IDSs are Snort, Suricata and Zeek (formerly known as Bro). Shah et al. evaluated the performance of two of them: Snort and Suricata [7]. Both tools are signature-based and alert only on known malicious traffic. To overcome the limitation of a signature-based solution, they developed a machine learning plugin for Snort. Gustavsson used Zeek to extract features from network captures and analysed them with machine learning algorithms in order to detect malicious traffic on IT networks [8]. Drakos implemented a series of policies in Zeek based on several network intrusion scenarios to detect Advanced Persistent Threats (APTs) on university networks [9].

Research has also been done to expand upon existing IT tools to provide support for ICS protocols. For instance, Snort has been expanded with generic ICS static signatures [10]. In addition to this, Zeek has also been customised by researchers to tailor it to ICS networks. Lin et al. have created DNP3 protocol parsers and Zeek policies to detect packets validating the semantics [11]. Udd et al. have taken another approach to extend Zeek in order to tailor it for industrial environments [12]. Their approach was to create automatic whitelisting and anomaly detection for the IEC 60870-5-104 protocol.

### 2.2 Threat hunting

Threat hunting in ICS has not been researched as extensively as threat hunting within IT environments. For instance, the well known and widely used MITRE ATT&CK Enterprise framework documents tactics and techniques used by adversaries and it is under constant development by a range of threat intelligence and security specialists [13]. The ICS version of the framework has been published in January 2020, after going through a community review [14]. It adds techniques used by adversaries worldwide, based on the public incident reports detailing attacks against ICS [15]. It allows to understand and distribute knowledge about adversary behaviours in industrial systems, which in turn can help with keeping industrial environments secure and operational.

For the purpose of this research, we define threat hunting as the process of proactive and iterative searching through networks on the lower levels of the Purdue model (level 0 to 2) to detect potential malicious presence, evading existing (if any) security solutions [3]. It should be noted that threat hunting activities require that a human (e.g. an analyst) interprets the results provided by any software in use [16].

The Purdue model classifies level 0 as the physical process levels, which consist of devices such as sensors and actuators. It classifies level 1 as a basic control level with devices such as PLCs, which are responsible for process control in industrial control systems. Level 2 is the area supervisory control, with devices such as Human Machine Interfaces (HMIs) that provide an interface to interact and control the PLCs from level 1.

Attacks on ICSs can be roughly divided into two types: indirectly targeted (accidental) and deliberate [17]. An example of adversary groups carrying out deliberate attacks are APTs. These are stealthy threat actors, defined by their long-lasting presence in compromised environments, typically a nation state sponsored group or groups performing targeted intrusions for specific goals. APTs often gain initial access via IT systems, however the enterprise domain is out of scope for this research.

In order to help defenders with deliberate attackers, the ICS Killchain framework was created by Michael Assante and Robert M. Lee [18]. It is based on the original Cyber Killchain from Lockheed and it assists defenders in visualizing and understanding adversary steps [19]. It consists of two stages; the goal of Stage 1 is getting access to the network, collecting information, gaining persistence and developing capabilities, whereas the goal of Stage 2 is the execution of targeted attack [18].

# 3 METHODOLOGY

## 3.1 Network based adversary techniques

The ICS MITRE ATT&CK framework specifies 81 unique techniques used by adversaries on ICS networks (level 0-2 of the Purdue model), along with various data sources that allow for the detection of those techniques [3]. The techniques represent "how" an adversary achieves its objective. The framework retains an abstraction level, which is an attempt to encompass multiple products while providing a level of technical specificity, making the techniques applicable for various ICS environments [15].

The focal point of our research was the first eight tactics from the framework [14]: Initial Access, Execution, Persistence, Evasion, Discovery, Lateral Movement, Collection and Command & Control. The Inhibit Response Function, Impair Process Control and Impact were not taken into consideration, because they list adversary goals and the impact they make on the industrial systems. In our method, we wanted to detect an attacker's presence and enable the defenders to stop them before they are able to reach those goals. Furthermore, we focused on the techniques that can be detected with network-based data sources such as network captures. Any host-based data sources and/or attack techniques were therefore out of scope. Table 7 in Appendix 10 specifies the techniques in scope.

## 3.2 Manufacturing testbed

We used an ICS testbed created and maintained by the Industrial Control & Communication Competence Center (IC4) [20]. The testbed represents a manufacturing environment and consists of an office network segment and three industrial network segments with hydraulic presses, a furnace for baking tiles, and dosing equipment. The simulation of ICS operations is performed with scripts carrying out actions such as read/writes to the PLCs and setting up connections to equipment on regular intervals.

The testbed contains various industrial devices such as PLCs, Human Machine Interfaces (HMIs), Input/Output devices, industrial switches and routers. Four industrial protocols are in use: Modbus, ProfiNET, Siemens S7comm, and EtherCAT. A full overview of the devices located in the manufacturing testbed is located in Table 8 in Appendix 11.

### 3.2.1 Data collection

We obtained VPN access to the manufacturing testbed in order to capture the network traffic. To gain visibility in all network segments (traffic passing through physical and virtual switches), we simultaneously captured traffic on two machines and merged it afterwards.

First, network traffic was captured during ninety minutes of normal operation. We call this our "baseline", which we used to get acquainted with the industrial protocols, to understand the environment, and to develop and test our PoC. Then, we performed the second network capture during which attacks were carried out. In order to prevent research bias, we did not create our own scripts to attack the environment. Instead, we used readily available scripts created by the researchers from Howest University College, Ghent University, and the IC4 research group [21]. Table 10 in Appendix 12 provides an overview of the attacks performed.

## 3.3 Proof of Concept design

We adhered to the following design principles while creating and evaluating the PoC application to detect adversary techniques on the network. The PoC created for this research works on network captures, which are inserted to the application periodically (e.g. every 10 minutes). The tool has been designed to be modular, allowing for easy modifications and enhancements and therefore making it possible to deploy it in other industrial environments as well.

### 3.3.1 Data processing

Figure 1 shows the data flow of our PoC and the components needed to process the data. The input we used are network captures, which can be processed by the "Data Logging Module", consisting of four main components of the Zeek platform.

First in the "Data Logging Module" are the protocol analysers. Zeek supports a set of parsers for commonly used protocols [22]. These include both Modbus and DNP3, which are widely used in industrial environments. The common protocol parsers are loaded by default on a Zeek instance. In addition to these protocol parsers, we used Zeek Package Manager to install ICS specific protocol parsers [23, 24]. Among those were protocol parsers for Bacnet, CIP, COTP, ENIP, S7comm, and ProfiNET. New protocol parsers can also be created with the Spicy framework and added to Zeek [25]. However, this was out of scope for our research.

Secondly, we extended the existing parsing and logging functionality of Zeek by creating custom logging logic using the event-driven scripting language that Zeek provides. Out of the box, Zeek provides a large number of scripts, but they do not cover ICS specific use cases [26]. For that purpose, we wrote our own scripts, covering the scope of the techniques from the ICS MITRE ATT&CK framework.

Next is the Zeek Logging Framework, which allows for storing network metadata from packet captures so that it can be searched, indexed, queried, and reported. There are three abstractions of the Logging Framework: Streams, Filters, and Writers. The log stream corresponds to a single log, e.g. **http.log** shows HTTP activity. Filters allow for determining which information gets written out, while Writers define the output format. Finally, each script we wrote takes advantage of the Zeek Notice Framework, which allows for raising notices and/or logging notices when called in a Zeek event or hook. Those notices are written to a file called **notice.log**. We tested our event scripts on publicly available network captures to ensure that notices would be raised on the right events [27, 28, 29, 30].

Both the **notice.log** file and the other **.log** files were used as input for the "Anomaly Mapper" seen in 1, for further processing. For the implementation of the "Anomaly Mapper", we created a web application written in ASP.NET Core 2.1 using C# 7.1. This PoC application can take the log files compressed in a **.zip** archive, and parses each log file to a collection of data objects. The application then selects all data objects belonging to the **notice.log** file, and maps them to other data objects that occurred at the same timestamp for the same connection UID. Then, both these data objects are subjected to a set of pre-defined rules, which are explained in Section 5. The output is a mapping of the events that happened on the network to the adversary techniques from the ICS MITRE ATT&CK framework.

The dotted line on Figure 1 indicates how the application should be used when deployed in a production environment. It indicates that searching for potential presence of adversaries on the network is an iterative process and

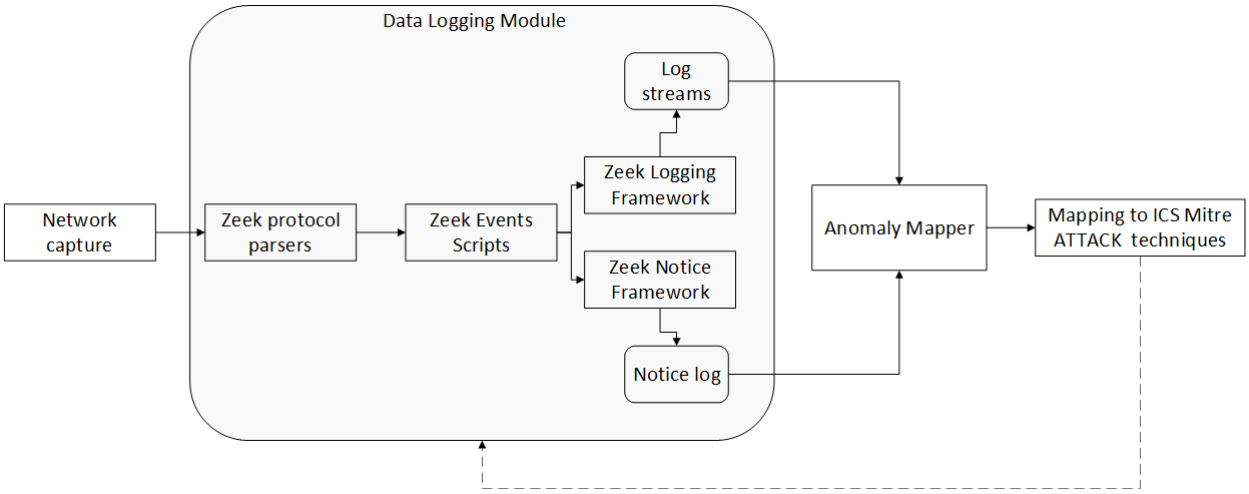


Fig. 1: PoC design used to search for potential adversary techniques on ICS networks.

that after obtaining and analysing the results, new scripts could be written and the application could be replayed or even expanded upon. This of course implies a human presence, which is absolutely necessary for any threat hunting activity, as the software is only meant as a support system for the analysts.

#### 3.3.2 Evaluation

Our PoC application was evaluated against the second data set from the manufacturing testbed containing various adversary techniques. We used a confusion matrix to perform the evaluation. Two classes - Regular vs Anomaly - were plotted against the Actual and the Predicted Condition. This leads to four categories of results: detected malicious traffic (True Positive), undetected malicious traffic (False Negative), legitimate traffic that the tool detects as malicious (False Positive), and legitimate traffic for which no detection is made (True Negative) [7]. As we investigated network traffic, the amount of regular, non-malicious events can rise very quickly. This means that the amount of True Negatives was much higher than the other three values of the confusion matrix, indicating an unbalanced data set. Predictive values that could be calculated based on the values from a confusion matrix would therefore not provide any meaningful insights. That is why we did not attempt to determine the amount of True Negatives, and why we did not provide a Negative Predictive Value, specificity or accuracy of our tool.

We did determine the amount of False Negatives of our experiment, to show the important network events that were missed by our PoC application. In addition to this, we calculated other relevant rates, which provide insights into the performance of our tool. The Positive Predictive Value (PPV) indicates the precision of the tool, showing how often an event is an anomaly, given that it is predicted as such. The True Positive Rate (TPR) or sensitivity shows the chance that an anomaly would be picked up by our tool and marked as an anomaly. Finally, since the actual value of the True Negatives was not calculated, we determined the Matthew's Correlation Coefficient (MCC) for two possible values of the True Negatives to determine a range of how well our PoC application performs as a binary classifier.

### 3.4 ICS Killchain mapping

Discovering potential adversary techniques present on the network is helpful, but further insight is needed to understand what the technique means, what could be its consequence on the production systems and what kind of other techniques can be expected on the network. Most of the time, techniques form a part of an adversary campaign. The network-based adversary techniques from the ICS MITRE framework have been mapped to various stages of the ICS Killchain to provide those insights [18].

## 4 Data Logging Module

The "Data Logging Module" is responsible for flagging potentially suspicious communication and logging it for purposes of further investigation. Before explaining the module, we provide a detailed description of ICS protocols supported by the PoC application.

### 4.1 Supported ICS protocols

Our PoC application supports the Modbus and S7 Communication (S7comm) industrial protocols. The "Data Logging Module" contains parsers for other ICS protocols as listed in Section 3.1, which allows for future expansion of the application, but no custom Zeek scripts were written to raise notices for these protocols.

#### 4.1.1 Siemens S7 protocol

S7comm is a Siemens proprietary protocol used by the SIMATIC S7 products, which includes PLC models from the S7-300/400 line and the newer generation from the S7-1200/1500 line. The protocol is used for PLC programming, exchanging data between PLCs, PLC data access by ICS systems, and diagnostic purposes [31]. There is no official specification available, however, there have been efforts to reverse engineer and model the Siemens S7 protocol [32, 33, 34].

There are two protocol versions: the standard one (protocol ID 0x32) and the new S7comm Plus (protocol ID 0x72) [34]. The new version is used in the S7-1200/1500 family of PLCs. Both versions of S7comm Plus are even less documented than the standard version of S7comm [34]. The devices in the manufacturing testbed are from the S7-1200 family and hence use both versions.

There are three types of roles a device can take in Siemens communication: the client, the server, or a peer [33]. In the client-server communication model, the HMI (client) initiates the communication by sending a query, and the PLC (server) responds by sending a response or by taking the action e.g. write operation specified in the query.

In peer-to-peer communication, the devices can exchange unsolicited data once the connection is established. S7 communication to and from a specific PLC is highly periodic [33]. In our network setup, the client-server communication model is present.

Figure 2 shows the structure of the protocol. S7comm runs on port 102/TCP, ISO-TPKT is used as a transport service (specified in RFC 1006 [35]) and ISO 8073 COTP (specified in RFC 905 [36]) is used as a connection-oriented transport protocol and contain the device type and addresses. S7comm is encapsulated in COTP.

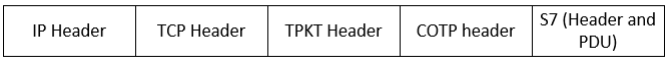


Fig. 2: S7comm protocol structure.

Each S7Comm packet consists of a header, a parameter part specifying which PLC variable should be accessed, and an optional data part (can be either Read Request, Read Response, Write Request or Write Response). The header includes the Protocol ID, ROSCTR (Remote Operating Service Control), the parameter length, the data length, the function code, and the item field.

A list of common S7comm function codes is shown in Table 1. S7comm also supports the use of user data subfunctions, which allow to read out PLC CPU values, for example `Read SZL (0x01)`. SZL stands for System Zone List, which is a memory area containing the diagnostic data and system state containing for instance the level of CPU protection [37]. The codes listed in Table 1 are searched for and used in the rule set of the PoC application.

| Value | Function code |
|---|---|
| 0x00 | CPU services |
| 0xF0 | Setup communication |
| 0x04 | Read Variable |
| 0x05 | Write Variable |
| 0x1A | Request download |
| 0x1B | Download block |
| 0x1C | Download ended |
| 0x1D | Start upload |
| 0x1E | Upload |
| 0x1F | End upload |
| 0x28 | PI-Service |
| 0x28 | PLC Control |
| 0x29 | PLC Stop |

Table 1: Job Request/Ack-Data function codes of S7 protocol [38].

#### *4.1.2 Modbus TCP*

Modbus is a data communication protocol, used for real-time distributed control. It is royalty-free and over time has become a de facto standard protocol for ICS environments [39]. There are several versions of the Modbus protocol, but the devices in the manufacturing testbed use Modbus TCP. It is an application layer protocol using port 502/TCP.

Modbus employs a master-slave communication model. The master device initiates the transaction (sends queries) and slaves respond by supplying the information or performing the requested action [39]. Only one device can be designated as a master (usually the HMI) and the remaining devices act as slaves (e.g. PLCs). It does not have long-term session semantics [39]. The protocol uses separate query-response sequences, however, the connection between the master and the slave is embedded in a single TCP connection. Depending on the PLC design, it can either accept a single TCP connection or allow for multiple concurrent connections on port 502.

The data model of Modbus is based on four primary tables:

- Discrete Input, single bit, read-only, data can be provided by the I/O system
- Coils, single bit, read-write, alterable by an application program
- Input Registers, 16-bit, read-only, data can be provided by the I/O system
- Holding Registers, 16-bit, read-write, alterable by an application program

A Modbus frame consists of the Application Data Unit (ADU), which encapsulates the Protocol Data Unit (PDU). The PDU has two fields: the payload and a function code. The payload field is limited to 252 bytes and it contains parameters that are specific to the function code used. The function code is a one-byte integer in the range of 1 to 127, but the Modbus standard specifies 19 of them [40]. There are three types of function codes: public, user-defined, and reserved. The public function codes are our focus because they are not implementation-specific. Table 2 shows the public function codes that can be used to access data.

| Function name | Function code |
|---|---|
| Read Discrete Inputs | 2 |
| Read Coils | 1 |
| Write Single Coils | 5 |
| Write Multiple Coils | 15 |
| Read Input Registers | 4 |
| Read Holding Register | 3 |
| Write Single Register | 6 |
| Write Multiple Holding Registers | 16 |
| Read/Write Multiple Registers | 23 |
| Mask Write Register | 22 |
| Read FIFO queue | 24 |
| Read File record | 20 |
| Write File record | 21 |

Table 2: Modbus public function codes used for data access [40].

### 4.2 Custom Zeek event scripts

As discussed in Section 3.1, our focus is on the 24 techniques from the ICS MITRE framework that can be detected on the network level with packet captures and/or network protocol analysis as data source. For each technique, the ICS MITRE framework provides a description and a procedure example as seen in the collected incident response reports. We took that data as our point of reference when creating a list of network activities we should monitor for in order to efficiently design our PoC and spot suspicious behaviour on the network. The full overview of tactics, techniques, and what to monitor in order to spot them is shown in Table 7 in Appendix 10.

We used that information while creating custom Zeek scripts that allowed us to flag potential techniques observed on the network. As the first step in our process, they produce a large number of False Positives and they are later inserted into the "Anomaly Mapper" for further processing. The mapping between the technique and the script used to detect it, is shown in Table 12 in Appendix 13. The code base is located in a Github repository [41]. The correct working of the scripts depends heavily on the Zeek protocol parser. All of them can be loaded simultaneously and run on a network capture to produce log files.

All the scripts we have written are specifically tailored to ICS environments and they can be roughly divided into two categories: ICS protocol related and other scripts. The ICS protocol related scripts are: `modbus_logging.zeek`, `s7comm_cotp_logging.zeek` and `http_user_agent.zeek`. The choice to log the HTTP user agent might not seem obvious, however, the PhoenixContact devices use the user agent MicroBrowser from SpiderControl, which is a web HMI solution. Phoenix HMI sends HTTP GET/POST requests to the PLC to read/write to the device using the MicroBrowser user agent and therefore it is important to register this communication. To understand the data exchange we also use `http_post_body.zeek`, which allows us to capture the data sent over HTTP POST requests.

We analyzed the baseline traffic of Modbus, S7comm, and HTTP requests from PhoenixContact devices to understand the communication patterns and we leveraged the fact that ICS is a static environment by embedding whitelist rules in the scripts to flag only unknown communication. As a result, when running the aforementioned scripts on a network capture, all communication using either Modbus, S7comm is sent to the `notice.log` file. Additionally, any HTTP requests using the MicroBrowser user agent, that do not originate or go to one of the listed IP addresses is logged in `notice.log` as well. Each entry includes the GMT timestamp, the UID, source and destination IP address, source and destination port, protocol used, notice message and action to be taken.

The other scripts do not have whitelists built in. The `arp_spoofing.zeek` script logs the ARP traffic and while doing so, builds an internal ARP cache, which is then used to determine when MAC/IP associations change. It intends to evaluate whether a MiTM attack is happening. The goal of `common_ports.zeek` is to register the use of common ports, as listed in ICS MITRE ATT&CK as the ones often used by adversaries. The `ftp_portable_executable.zeek` attempts to establish whether there are any executable files sent over the network. This is done by creating hooks for `FTP Request`, `FTP Reply` and for the type of file transferred. Furthermore, we log successful and failed attempts to establish RDP, SSH, VNC, and Telnet communication. All of these protocols are used actively in industrial environments and could be a means to get access to, as well as control industrial equipment such as a PLC. Moreover, for VNC we also log the client version, which allows determination of which version is used. The `smb_logging.zeek` script logs SMBv1 and SMBv2 commands sent over the network. As stated in the ICS MITRE framework, SMB is often used as a platform to spread malware or perform lateral movements. The purpose of the `tcp_scan_detection.zeek` script is to detect potential network scanning activities, which could indicate the presence of an adversary. Since ICS environments are very static and the equipment is fragile, a site engineer would never issue scanning of the industrial network.

## 5 Anomaly Mapper

The next step in our PoC is the "Anomaly Mapper", which applies a set of rules to the logs extracted with the "Data Logging Module" to further eliminate False Positives. The output is a list of techniques, which were spotted in the network capture. The code base with detailed comments explaining the thought process can be found on Github [41]. Figure 3 in Appendix 14 contains a flowchart of the steps taken by the application during its operation.

For the industrial protocols, we created a one-to-many mapping between the S7comm and Modbus function codes and the techniques from the ICS MITRE ATT&CK framework. They are shown in Table 13 and Table 14 of Appendix 15 respectively. Based on the function code seen in the logs, the program can determine which technique is used. S7comm functions correspond to eight various techniques from the framework, whereas the Modbus function codes map to two. The `CPU services` and `Setup Communication` functions from the S7comm protocol are not mapped to any of the techniques. This is due to the lack of publicly available information about the exact working of the `CPU services` command. Additionally, there are other subfunctions such as `Read SZL`, which are used to read out the CPU values. `Setup Communication` only indicates the start of the communication between two hosts and hence does not correspond to any of the techniques.

The HTTP POST parameters used by the PhoenixContact devices to communicate via MicroBrowser are not publicly disclosed. The request that we observed in the baseline traffic of the manufacturing environment is `uri:/cgi-bin/ILRReadValues.exe`, which retrieves the model and the function of the device, as well as the process parameters. We have thus mapped those requests to the "Role identification" and "Point & tag identification" techniques.

For filtering out False Positives in RDP, VNC, SSH, and Telnet communication, we have applied two rules: out-of-working hours and failure/success ratio of the login attempts. The first one specifies that all the traffic using the aforementioned protocols should be flagged if it happens outside of 7-19 business working hours. The failure/success ratio is set to flag if at least 3 failed attempts are seen before a successful login. The flagged events are mapped to the Command-Line Interface technique.

For FTP traffic, we log connections containing default credentials and/or executable files. These are mapped to the Default credentials technique and three others: System firmware, Module firmware, and Remote file copy. We also flag any other executable files shared on the network. In addition to this, the use of common ports is filtered by a rule triggering on a mismatch in protocols, e.g. if HTTP traffic is sent over a different port than 80 or 8080. Creating correct rules for the SMB protocol is a field of research on its own, but as specified in the ICS MITRE we only look at the Remote file copy [42]. The SMB connection is flagged if we see the following write options: File Write, Pipe Write, and Print Write [43].

The rule for the DNS tunneling is based on the SANS research conducted by Farnham and Atlasis [44]. If the length of the query is larger than 52, the Shannon entropy is higher than 4.0 and we spot at least 16 queries within 30 seconds, then we mark the communication as DNS tunneling,which is classified by the ICS MITRE framework under the "Connection Proxy" technique. Lastly, all ARP spoofing attempts flagged by the corresponding Zeek event script are marked by the application as a MiTM adversary technique.

The output from the "Anomaly Mapper" is a reduced list of adversary techniques on the network. They include the technique used, the tactic, the timestamp and the source and destination IP address (for network scanning only the source IP address). Some events correspond to more than

one technique, which means that for the same event we have several entries with the same timestamp. These were put in a group of techniques and considered as one entry.

## 6 Results

### 6.1 PoC detection performance

To verify the detection performance of our PoC, we evaluated it against the data sets from the manufacturing environment. When simulating the adversary techniques, the launch of WannaCry did not result in the generation of any network traffic.

When evaluating the PoC against the data set containing the attacks, we obtained 160 distinct alerts about techniques used. Those techniques covered 10 out of 12 adversary techniques that were used on the manufacturing testbed. The PoC did not detect the start/stop PLC attempt nor the WannaCry attack. To assess the detection performance of our PoC, we created a confusion matrix, as shown in Table 3.

| | | Actual | |
|---|---|---|---|
| | Total population | Anomaly | Regular |
| Predicted | Anomaly | 120 | 40 |
| | Regular | 7 | N/A |

Table 3: Confusion matrix detailing the anomalies predicted by our PoC and the actual anomalies.

The matrix shows that the PoC produced a total amount of 160 alerts (adversary techniques). From this number, 120 were correctly detected as an anomaly (adversary technique used on the network). These are our True Positives. There were 40 False Positives in our output and 7 packets were not detected by the tool (False Negative). Based on the values from the confusion matrix, we calculated the Positive Predictive Value and True Positive Rate shown in Table 4. The PPV has a value of 0.750, whereas the TPR is 0.945. Due to the fact that our dataset is unbalanced and due to the effort required to calculate the amount of True Negatives, we determined the range of the MCC value. The MCC value was first calculated by assuming the amount of True Negatives is equal to the amount of True Positives, which would result in an MCC of 0.695; secondly when assuming the amount of True Negatives is equal to the amount of packets in the network capture (2,841,495), which would result in an MCC of 0.842.

| Rates | Value |
|---|---|
| Positive Predictive Value (PPV) | 0.750 |
| True Positive Rate (TPR) | 0.945 |
| Matthews Correlation Coefficient (MCC) | $0.695 \leq MCC \leq 0.842$ |

Table 4: The values of the selected rates calculated from the confusion matrix.

### 6.2 Mapping techniques to ICS Killchain

This section provides an analysis on how the network-based techniques from ICS Mitre correspond to the ICS Killchain as explained in Section 2. Table 15 in Appendix 16 provides the full overview.

The Exploit of the public facing applications technique belongs to Stage 1: Cyber Intrusion Phase (Delivery), which has the goal of obtaining initial access to the internal network. Once on the network, an adversary attempts to establish command and control capabilities which belong to Stage 1: Management & Enablement (C2). The following three techniques can be mapped to this phase: Standard application layer protocol, Commonly used ports and Connection proxy.

Stage 1: Sustainment, entrenchment, development & execution (Act) encompasses a variety of techniques with the goal of collecting information, gaining persistence and performing lateral movements. These are: Control device identification, Network service scanning, Detecting operating mode, Detecting operating state, Point & tag identification, Role identification, Program upload, Default credentials, Remote file copy, Module & System firmware and MiTM.

In Stage 2 of the ICS Killchain, the attacker uses the knowledge obtained in Stage 1 to launch a targeted attack. Command line interface, Execution through API and Program download techniques are part of the Stage 2: ICS Attack (Deliver and Install/Modify) phase.

The techniques from ICS Mitre that correspond to Stage 2: Execute ICS Attack are Rogue Master Device, Spoof reporting messages, Utilize/change operating mode and Changing the program state. They all could result in actual damage to the industrial equipment.

## 7 Discussion

### 7.1 Analysis of the results

The number of techniques logged in our PoC application's output is 160 and is therefore higher than the amount of techniques we performed. This is because carrying out an adversary technique generates multiple packets on the network, which get logged as separate entries in the output.

Our results show that the TPR of our PoC application is 0.945, having only seven False Negatives out of 127 actual anomalies. This indicates that the application performs well at marking actual anomalies as anomalies, and misses only a small fraction of the adversary techniques we simulated on the network. Furthermore, the results show a PPV of 0.750, which indicates that the 25 percent of the events marked as anomalies by our PoC are not actually anomalies. This shows that the PoC application can still benefit with further tuning to reduce the amount of False Positives, as such an amount creates extra work for human analysts.

The two possible values calculated for the MCC were 0.695, assuming the amount of True Negatives was equal to the amount of True Positives, and 0.842 when assuming the amount of True Negatives was equal to the amount of packets in the network capture. Given that the amount of anomalous events in the network capture is very low compared to the amount of packets, it is unlikely that the amount of non-anomalous events is equal to the amount of anomalous events, as this would constitute a total of about 300 events in about 3 million packets. It is therefore more likely that the true value of the MCC is higher than the lower bound of the calculated range. This would mean that the MCC for our PoC application is at least higher than 0.7, which indicates a strong positive relationship between the predicted values of our PoC and the actual values in the data set. The calculated values of the TPR and MCC show that our PoC application is a good starting point as a binary classifier to detect anomalous network events in our manufacturing testbed, but the amount of False Positives should be lowered.

Out of the 160 detected techniques, 120 were marked as techniques we carried out and 40 were falsely categorized as attacker techniques. These False Positives can be explained as follows. First of all, on the network we used for capturing our packets, a script was running that created

an SSH connection every minute. For SSH connections, we created the rule that any connection set up outside of office hours (Monday - Friday, 7AM - 7PM) would be marked as a technique. As the script was running while we performed our attacks, the application detected those SSH connections as well. This resulted in 25 of the False Positives.

Furthermore, 14 False Positives were caused by the ARP spoofing detection script we used for Zeek. The script keeps track of ARP requests and replies that have been sent in the packet capture and marks any unsolicited ARP replies sent over the network as anomalous. The amount of truly anomalous ARP replies was six, so this approach warrants further tuning. Additionally, one False Positive was generated by the use of HTTPS over port 80. Our PoC marks any mismatch of protocols over a commonly used port as anomalous, and while the use of HTTPS over port 80 is unconventional, it does not match with any of the attacks we performed on the network.

The stop/start CPU PLC command was not detected by our PoC, even though the commands for altering the PLC outputs and memory were detected. These three attacks were issued using the same script created by the researchers from the IC4 research group against the Siemens S7-1200 PLC [21]. This family of Siemens PLCs supports both the standard and new S7comm protocol version. After in-depth analysis of the attack script, we noticed that in order to modify the program state, a function code `Write variable` is used. This function code is sent as part of the S7comm PDU, which in turn is encapsulated in a COTP packet. The stop/start CPU command, on the other hand, is issued using the new S7comm protocol version. Neither Wireshark nor the Zeek S7comm and COTP protocol parsers are able to parse the newer S7comm protocol. As a consequence, our application is also not able to extract the function codes and map it to any corresponding ICS MITRE ATT&CK techniques. This is an excellent example of how complex ICS protocols can be and that the protocol support of our PoC should be enhanced in the future.

Secondly, our PoC could not detect the WannaCry attack we carried out. When performing the attack, the malware did not generate any of the network packets that are expected to be seen with WannaCry. Therefore, both the Zeek parsers and the "Anomaly Mapper" could not pick up any events, because there were none to be seen.

When mapping the events that the PoC detected in the network captures to the ICS MITRE framework, we noticed that the Collection tactic appeared more frequently than the other ICS MITRE tactics. This can be explained by the scripts we used to simulate the attacks. Before making changes to a PLC, such as altering the program state, the scripts perform a read on the PLC to collect I/O values. We furthermore found that basing mapping rules on specific ICS protocol commands makes distinguishing between attacking techniques and adversary goals fairly easy, which provides extra insight based on the protocol command used. Also, as the ICS MITRE techniques are not sequential, the mapping to the ICS Killchain helps with the determination at what stage an attacker is in their campaign.

### 7.2 Limitations

Our research methodology is based on the techniques specified by the ICS MITRE ATT&CK framework. It is important to point out that those techniques are in turn based on documented public incident response reports detailing attacks against industrial systems. The framework does not contain data from incidents that either have not been discovered (due to lack of monitoring and/or security controls), or if discovered have never been publicly disclosed. For instance, the energy sector is more mature in terms of security controls, which also leads to incidents being detected faster than in other industrial sectors. This introduces a bias in the framework and thus in our approach because we only search for known techniques.

It is also important to reflect on the possibility that there could be two or more different attackers in the industrial environment at the same time. The approach we took allows to determine techniques being used on the environment and can attribute them to the tactics from the ICS MITRE framework, but we do not decide whether the techniques are part of one adversary campaign or more. This would require more advanced means of analyzing the network, e.g. the use of forensics, host-based event analysis, or even malware reverse engineering. One of the challenges of our research was obtaining the right data set. Because the data sets from actual ICS adversary campaigns are not publicly available, we had to simulate the techniques in a controlled environment. One of the drawbacks of this approach is the timing of the attacks performed. APT campaigns could span over days or even months, during which various techniques are performed. Since the attack simulation was performed in a shorter time span, the time between the techniques could not be taken into consideration while developing the detection rules. On the other hand, even though an APT could spend months preparing and gathering the information, certain actions like the PLC code download will be a one-time event.

Our PoC was built to discover 24 techniques from the ICS MITRE ATT&CK framework. In our simulation, however, we were able to cover only a subset of those. Our intention was to cover at least one technique from each tactic except the initial access, as this often happens via the IT segment. The attacks we performed are just examples of the type of attacks that could happen on the network and be classified in a certain category of techniques.

We should also reflect upon the vast landscape of ICS protocols and its variations. There are three steps required to support a new protocol in our PoC. First is writing the Zeek protocol parser, then creating a Zeek event script to flag certain events, and lastly inserting the logic into the "Anomaly Mapper" to filter out False Positives. For Modbus and S7comm we have accomplished all of those steps. For ProfiNET we have completed the first two steps. It should be noted that even if a protocol parser is available, that does not mean that it is able to parse the specific variation of the protocol present in the environment, as we could see with the S7comm Plus.

Despite the fact that we attempted to make the PoC as adaptive to various manufacturing environments as possible, there are certain things that need to be taken into consideration before deploying it in production. One of them is establishing a baseline of static, allowed communication between HMIs and PLCs. Another aspect is verifying which protocols are in use in the environment. The same applies to the configuration of the working hours, which could differ from the manufacturing testbed. This could require additional time spent on understanding the environment and adjusting the application.

## 8 Conclusion

In order to answer our main research question, we first need to elaborate on the sub-questions. The first one is

focused on the attack techniques used by the APT groups. In our research we took the ICS MITRE ATT&CK framework as the reference and selected the techniques that can be discovered by means of network captures or network protocol analysis. Based on the recommendations from the literature, we have also focused on discovering those techniques that allow detection of an attacker's presence before they are able to reach the stage where damage can be inflicted on the industrial systems.

The second sub-question was about finding and improving existing monitoring solutions to accommodate for the detection of the APT techniques. Based on the results from the literature review, we have decided to use the Zeek platform, which allows for customization due to its scripting language. We built upon Zeek by writing event scripts tailored to ICS networks, to trigger on suspicious events. We then created our own parser engine, the "Anomaly Mapper", in order to further analyze anomalies, to rule out False Positives and to map them to the adversary techniques from the ICS MITRE. Our results show that the PoC application performed well as a binary classifier with a Matthews Correlation Coefficient range between 0.7 and 0.85 and a True Positive Rate nearing 0.95. On the other hand, the False Positive Rate should be improved to lessen the amount of required manual verification of the output.

The last sub-question was about mapping the ICS MITRE techniques to various stages of the ICS Killchain. We have provided an in-depth analysis of the correlation between the network-based events from the ICS MITRE framework to the ICS Killchain adversary campaign stages, which puts the techniques used by adversaries in perspective.

To answer our main question "How can network analysis be used to discover the potential presence of Advanced Persistent Threats (APT) in Industrial Control Systems (ICS)?", we can conclude that it is possible to improve upon open source tools to detect anomalies in ICS networks and flag them as potential adversary techniques. Further development of the PoC application is required to improve its detection performance and support more ICS protocols.

Our approach allows defenders to deploy the tool in their industrial networks, start gaining more visibility into the network events and begin with threat hunting activities.

## 9 Future work

As part of future work, further enhancements of the PoC are necessary. New protocol parsers should be written to accommodate the wide variety of different ICS protocols in use. For instance, S7comm Plus, EtherCAT and ProfiNET should be supported. The rules of the PoC should also be expanded and improved upon to decrease the amount of False Positives, by fine tuning detection rules. Furthermore, the PoC should be evaluated against different industrial environments. Support for host-based events should be added in order to cover all techniques from the ICS MITRE ATT&CK framework.

## References


[1] Andy Greenberg. *The Story of Sandworm, the Kremlin's Most Dangerous Hackers*. URL: https://www.wired.com/story/sandworm-kremlin-most-dangerous-hackers/. (accessed: 02.06.2020).

[2] Dragos. *2019 Year in Review, Lessons learnt from the front line of ICS cybersecurity*. URL: https://www.dragos.com/wp-content/uploads/Lessons_Learned_from_the_Front_Lines_of_ICS_Cybersecurity.pdf?hsCtaTracking=ea40a828-084b-4ee9-a0fc-0908864d3f8e%5C%7C4eafb14d-2e38-44e0-9e6d-08c2aea4a480. (accessed: 05.06.2020).

[3] *Purdue Enterprise Reference Architecture*. URL: https://en.wikipedia.org/wiki/Purdue_Enterprise_Reference_Architecture. (accessed: 01.06.2020).

[4] Rob T. Lee Robert M. Lee. *SANS 2018 Threat Hunting Survey Results*. URL: https://www.sans.org/media/analyst-program/Multi-Sponsor-Survey-2018-Threat-Hunting-Survey.pdf. (accessed: 02.06.2020).

[5] Yuri Diogenes and Erdal Ozkaya. *Cybersecurity - Attack and Defense Strategies: Infrastructure security with Red Team and Blue Team tactics*. Packt Publishing Ltd, 2018.

[6] Robert Mitchell and Ing-Ray Chen. "A survey of intrusion detection techniques for cyber-physical systems." In: *ACM Comput. Surv* 46 (2014), pp. 2–30.

[7] Syed Ali Raza Shah and Biju Issac. "Performance comparison of intrusion detection systems and application of machine learning to Snort system". In: *Future Generation Computer Systems* 80 (2018), pp. 157–170.

[8] Vilhelm Gustavsson. *Machine Learning for a Network-based Intrusion Detection System: An application using Zeek and the CICIDS2017 dataset*. 2019.

[9] Panagiotis Drakos. "Implement a security policy and identify Advance persistent threats (APT) with ZEEK anomaly detection mechanism". In: (2020).

[10] Jeyasingam Nivethan and Mauricio Papa. "Dynamic rule generation for SCADA intrusion detection". In: *2016 IEEE Symposium on Technologies for Homeland Security (HST)*. IEEE. 2016, pp. 1–5.

[11] Hui Lin et al. "Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol". In: Jan. 2013.

[12] Robert Udd et al. "Exploiting bro for intrusion detection in a SCADA system". In: *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*. 2016, pp. 44–51.

[13] The MITRE Corporation. *ATT&CK Matrix for Enterprise*. URL: https://attack.mitre.org/. (accessed: 31.05.2020).

[14] The MITRE Corporation. *ATT&CK® for Industrial Control Systems*. URL: https://collaborate.mitre.org/attackics/index.php/Main_Page. (accessed: 31.05.2020).

[15] Jacob Steele Otis Alexander MIsha Belisle. *MITRE ATT&CK® for Industrial Control Systems: Design and Philosophy*. Tech. rep. 2020.

[16] David Szili. *SANS Building and Maturing YourThreat Hunting Program*. URL: https://www.sans.org/media/analyst-program/building-maturing-threat-hunting-program-39025.pdf. (accessed: 01.07.2020).

[17] Kevin E Hemsley, E Fisher et al. *History of industrial control system cyber incidents*. Tech. rep. Idaho National Lab.(INL), Idaho Falls, ID (United States), 2018.

[18] Michael J Assante and Robert M Lee. "The industrial control system cyber kill chain". In: *SANS Institute InfoSec Reading Room* 1 (2015).

[19] Eric M Hutchins, Michael J Cloppert and Rohan M Amin. "Intelligence-driven computer network de-

fense informed by analysis of adversary campaigns and intrusion kill chains". In: *Leading Issues in Information Warfare & Security Research* 1.1 (2011), p. 80.

[20] *Industrial Control & Communication Competence Center*. URL: https://www.ic4.be/?lang=en. (accessed: 06.06.2020).

[21] Tijl Deneut. *ICSSecurityScripts*. URL: https://github.com/tijldeneut/ICSSecurityScripts. (accessed: 06.06.2020).

[22] Zeek. *Protocol Analyzers*. URL: https://docs.zeek.org/en/current/script-reference/proto-analyzers.html. (accessed: 22.06.2020).

[23] Zeek. *Zeek Package Manager*. URL: https://docs.zeek.org/projects/package-manager/en/stable/index.html.

[24] Zeek. *Zeek packages*. URL: https://zeek.org/packages/. (accessed: 22.06.2020).

[25] Spicy. *Spicy — Generating Parsers for Protocols & Files*. URL: https://docs.zeek.org/projects/spicy/en/latest/.

[26] Zeek. *Zeek Script Index*. URL: https://docs.zeek.org/en/current/script-reference/scripts.html.

[27] Wireshark Wikipedia. *Sample Captures*. URL: https://wiki.wireshark.org/SampleCaptures.

[28] Netresec. *Publicly available PCAP files*. URL: https://www.netresec.com/?page=PcapFiles.

[29] malware-traffic-analysis.net. *PCAPS for tutorial on exporting objects*. URL: https://www.malware-traffic-analysis.net/training/exporting-objects.html.

[30] PCAPAnalysis.com. *Pcap analysis*. URL: https://www.pcapanalysis.com/.

[31] Gilbert Peterson and Sujeet Shenoi. *Advances in Digital Forensics XIV: 14th IFIP WG 11.9 International Conference, New Delhi, India, January 3-5, 2018, Revised Selected Papers*. Vol. 532. Springer, 2018.

[32] Thomas Wiens. *S7comm Wireshark dissector*. URL: https://sourceforge.net/projects/s7commwireshark/.

[33] Amit Kleinman and Avishai Wool. "Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics". In: *The Journal of Digital Forensics, Security and Law: JDFSL* 9.2 (2014), p. 37.

[34] Ma Liang Cheng Lei Li Donghong. *The spear to break the security wall of S7CommPlus*. URL: https://www.blackhat.com/docs/eu-17/materials/eu-17-Lei-The-Spear-To-Break%5C%20-The-Security-Wall-Of-S7CommPlus-wp.pdf. (accessed: 29.06.2020).

[35] D.Cass M.Rose. *ISO Transport Service on top of the TCP Version: 3*. URL: https://tools.ietf.org/html/rfc1006. (accessed: 22.06.2020).

[36] Network Working Group. *ISO Transport Protocol Specification ISO DP 8073*. URL: https://tools.ietf.org/html/rfc905. (accessed: 22.06.2020).

[37] N Ben Aloui. "Industrial control systems dynamic code injection". In: *Cybersecurity Labs, DCNS Toulon, Toulon, France (grehack. org/files/2015/Grehack% 202015% 20-% 20Paper% 20-% 20Industrial% 20Control% 20Systems% 20Dynamic% 20Code% 20Injection. pdf)* (2015).

[38] Gyorgy Miru. *S7comm protocol constants*. URL: http://gmiru.com/resources/s7proto/constants.txt. (accessed: 22.06.2020).

[39] Niv Goldenberg and Avishai Wool. "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems". In: *International Journal of Critical Infrastructure Protection* 6.2 (2013), pp. 63–75.

[40] Modbus Organisation. *Modbus Application Protocol Specification V1. 1b3*. 2012.

[41] Steffan Roobol. *ICSMitreAnomalyParser*. URL: https://github.com/StefRoo/ICSMitreAnomalyParser.

[42] Ikram Ullah. "Detecting Lateral Movement Attacks through SMB using BRO". MA thesis. University of Twente, 2016.

[43] Microsoft. *Copy File (Local to Remote)*. URL: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-smb/01bf5dbe-2d51-45e1-b88f-8d6afac1ab98. (accessed: 29.06.2020).

[44] Greg Farnham and A Atlasis. "Detecting DNS tunneling". In: *SANS Institute InfoSec Reading Room* 9 (2013), pp. 1–32.

## 10 APPENDIX: NETWORK-BASED TECHNIQUES IN SCOPE

| Tactic | Technique | Explanation | Monitoring mechanism |
|---|---|---|---|
| Initial access | Exploit public facing applications | Exploiting devices with direct communication to the Internet or IT | Monitor port 80, 8080, 443, 8443 |
| Execution | Change program state | Change the state of the program on control device | Monitor function codes of Modbus, S7comm |
| Execution | Command line interface | Used to interact with systems and execute commands | Monitor Telnet, RDP, VNC, SSH |
| Execution | Execution through API | API calls can be used to engage functions on a device | Monitor HTTP(S) |
| Execution | Man in the middle | Intercept the traffic with the purpose of blocking, logging, modifying or injecting traffic | Monitor for ARP, DNS and IP spoofing |
| Persistence | Module firmware | Install malicious or vulnerable firmware onto modular hardware devices | Monitor for binary files, portable executable, files transferred over FTP |
| Persistence | Program download | Load malicious program logic on a device | Monitor for uploading programs to PLC from an outside IP address |
| Persistence | System firmware | Install malicious or vulnerable firmware onto devices | Monitor for binary files, portable executable, files transferred over FTP |
| Evasion | Rogue master device | Impersonate master device to communicate with a slave | Monitor for uploading programs to PLC from inside IP address |
| Evasion | Spoof reporting messages | Modify reporting messages to not reflect the actual state of the operation | Monitor for ARP spoofing |
| Evasion | Utilize/change operating mode | Put controllers in alternate mode of operation | Monitor function codes of Modbus, COTP, S7comm |
| Discovery | Control device identification | Determine the make and model of the device | Monitor function codes of Modbus, COTP, S7comm |
| Discovery | Network service scanning | Host and service discovery on the network | Monitor for TCP scanning |
| Lateral movement | Default credentials | Usage of default credentials on control devices | Monitor authentication over FTP and VNC |
| Lateral movement | Remote file copy | File copy between the systems | Monitor for SMB file transfer |
| Collection | Detect operating mode | Determine current operating state of PLC | Monitor for gathering information about current PLC state e.g stop/prog/run/remote/invalid |
| Collection | Detect program state | Determine current state of a program on a PLC | Monitor for gathering information about current state of a program on a PLC e.g. running/halted/paused/exception |
| Collection | Point & tag identification | Determine inputs, memory locations, outputs | Monitor for scanning I/O of the PLC |
| Collection | Program upload | Download project file from a PLC to gather information | Monitor for file download from the PLC using outside IP address |

| | | | |
|---|---|---|---|
| Collection | Role identification | Determine role of a device | Monitor for scanning I/O of the PLC |
| Command and control | Commonly used port | Use of common ports to blend in with normal network activity | Monitor commonly used ports in ICS: TCP/22, TCP/80, TCP/443, TCP/UDP/53, TCP/UDP/5353, TCP/8080, TCP/23, UDP/161, TCP/502, TCP/102, TCP/2000, TCP/44818 |
| Command and control | Connection proxy | Use of proxy to direct network traffic between systems | Monitor for DNS tunneling |
| Command and control | Standard application layer protocol | Command and control capabilities over commonly used protocols | Monitor HTTP(S), RDP, Telnet, Modbus |

Table 7: Selected techniques from ICS MITRE ATT&CK framework and corresponding monitoring mechanisms.

## 11 APPENDIX: INDUSTRIAL EQUIPMENT INVENTORY

| Function | Model | IP address |
|---|---|---|
| Router | eWON Flexy 201 | 10.20.1.1 |
| Switch | Beckhoff CU2008 | N/A |
| PLC | Beckhoff CX9020 | 10.20.1.10 |
| HMI | Beckhoff CP6606 | 10.20.1.11 |
| I/O Island | Rockwell 5096-L306ER/A CompactLogix 5380 | 10.20.1.30 |
| PLC | Mitsubishi FX5U-32M | 10.20.1.112 |
| PC | Win7 OPC UA | 10.20.1.15 |
| PC | Win10 OPC UA | 10.20.1.25 |
| Router | Siemens S623 | 10.20.2.1 |
| Router | IXON IXRouter 3 | 172.20.2.2 |
| Switch | Siemens XB208 | 10.20.2.5 |
| PLC | Siemens S7-1200 | 10.20.2.10 |
| HMI | Siemens KTP400 | 10.20.2.11 |
| I/O Island | Siemens IM151-3 | 10.20.2.20 |
| PLC | Siemens S7-1500 | 123.145.120.102/29 |
| PLC | Phoenix ILC 390 PN | 10.20.3.10 |
| HMI | PhoenixContact WP 06T | 10.20.3.11 |
| I/O Island | PhoenixContact IL PN BK | 10.20.3.20 |
| PLC | Schneider TM241CE40R | 10.20.30.149 |
| PLC | PLCnext | 10.20.3.120 |

Table 8: Industrial equipment inventory from the IC4 manufacturing testbed.

## 12 APPENDIX: TECHNIQUES PERFORMED ON THE MANUFACTURING TESTBED

| Action | Technique | Tactic | Target | Execution date GMT +2 |
|---|---|---|---|---|
| ARP spoofing | MiTM | Execution | 10.20.2.5 | 2020-06-27 15:28 |
| Establishing SSH session | Command Line Interface | Execution | Siemens Scalance Industrial Switch, 10.20.2.5 | 2020-06-27 15:31 |
| Establishing SSH session | Command Line Interface | Execution | Kali Linux 10.20.20.22 | 2020-06-27 15:31 |
| Establishing RDP session | Command Line Interface | Execution | Beckhoff Win7, 10.20.1.15 | 2020-06-27 15:32 |
| Altering outputs (scans device beforehand) | Change Program State | Execution | S7-1200 PLC, 10.20.2.10 | 2020-06-27 15:34 |
| Altering memory (scans devices beforehand) | Change Program State | Execution | S7-1200 PLC, 10.20.2.10 | 2020-06-27 15:35 |
| Uploading executable file | Module Firmware, System Firmware, Remote file copy | Persistence, Lateral movement | Phoenix Contact PLC, 10.20.3.10 | 2020-06-27 15:36 |
| Stop/start CPU PLC (scans device beforehand) | Utilize/Change Program Mode | Evasion | S7-1200 PLC, 10.20.2.10 | 2020-06-27 15:38 |
| TCP scan | Network Scanning | Discovery | Industrial network segment, 10.20.1.0/24 | 2020-06-27 15:39 |
| WannaCry | Exploitation of Remote Services, External Remote Services, Remote File Copy | Lateral movement | Beckhoff Win7, 10.20.1.15 | 2020-06-27 15:51 |
| Device scanning | Detect Operating Mode, Control Device Identification | Collection | S7-1200 PLC, 10.20.2.10 | 2020-06-27 15:51 |
| DNS tunneling | Connection proxy | Command and control | From 10.20.20.22 to remote server | 2020-06-27 15:52 |

Table 10: Detailed overview of the techniques performed on the manufacturing testbed.

## 13 APPENDIX: TECHNIQUES WITH CORRESPONDING CUSTOM ZEEK SCRIPT(S) TO DETECT IT

| **Zeek script** | **Technique** |
|---|---|
| arp_spoofing.zeek | Man in the middle, Spoof reporting messages |
| common_ports.zeek | Exploit public facing applications, Execution through API, Commonly Used Port, Standard application layer protocol |
| dns_spoofing.zeek | Man in the middle |
| dns_tunneling.zeek | Connection proxy |
| ftp_portable_executable.zeek | Module firmware, System firmware, Default credentials |
| http_user_agent.zeek | Change program state |
| http_post_body.zeek | Change program state, Role identification, Point & tag identification |
| modbus_logging.zeek | Change program state, Program download, Rogue master device, Utilize/change operating mode, Control device identification, Detect operating mode, Detect program state, Point & tag identification, Program upload, Role identification |
| rdp_logging.zeek | Command line interface, Standard application layer protocol |
| s7com_cotp_logging.zeek | Change program state, Program download, Rogue master device, Utilize/change operating mode, Control device identification, Detect operating mode, Detect program state, Point & tag identification, Program upload, Role identification |
| smb_logging.zeek | Default credentials, Remote file copy |
| ssh_logging.zeek | Command line interface |
| tcp_scan_detection.zeek | Network service scanning |
| telnet_logging.zeek | Command line interface, Default credentials, Standard application layer protocol |
| vnc_logging.zeek | Default credentials |

Table 12: Custom Zeek scripts used to detect techniques used on the network.The code is uploaded to Github [41].

## 14 Appendix: Flowchart of Anomaly Mapper program flow.

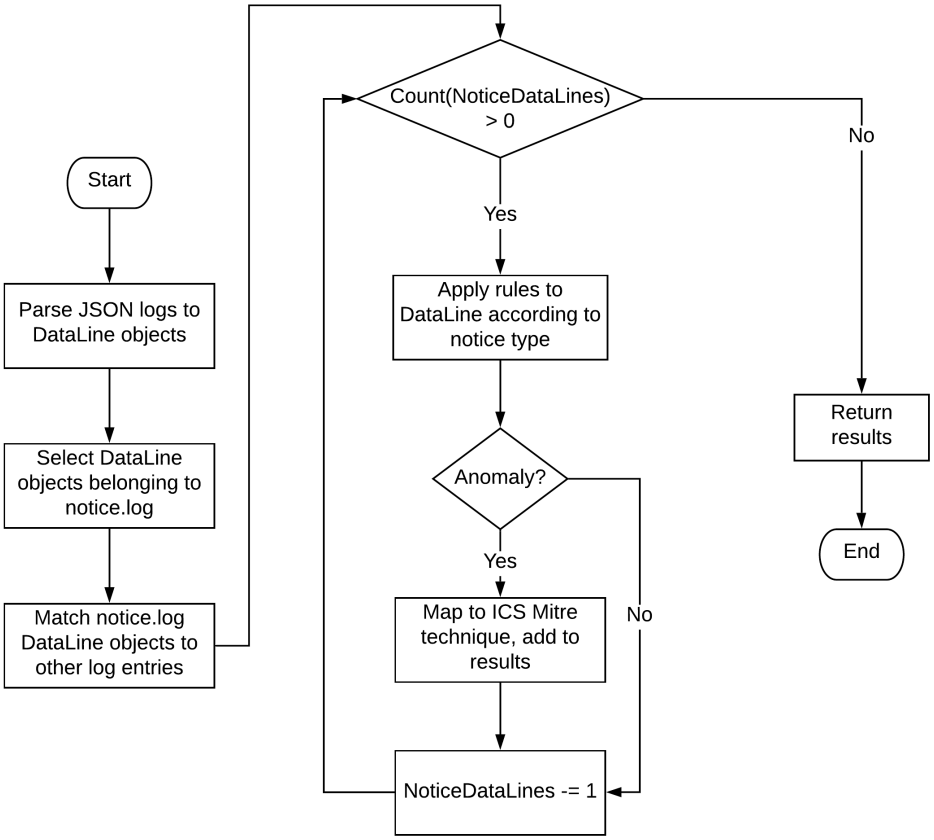


Fig. 3: Flowchart explaining the Anomaly Mapper operation.

## 15 APPENDIX: ANOMALY MAPPER - FUNCTION CODE MAPPING TO ICS MITRE FRAMEWORK

| Function name | Technique |
|---|---|
| Read Variable | Point & tag identification,<br>Detect program state,<br>Role identification |
| Write Variable | Change program state |
| Request download | Program download |
| Download block | |
| Download ended | |
| Start upload | Program upload |
| Upload | |
| End of upload | |
| PI Service | Utilize or change operating mode |
| PLC Control | |
| PLC Stop | |
| Read SZL | Detect operating mode |

Table 13: One-to-many mapping between the S7comm function codes and the techniques from ICS MITRE framework.

| Function name | Technique |
|---|---|
| Read discrete inputs | Point & tag identification |
| Read coils | |
| Read input registers | |
| Read holding register | |
| Read FIFO queue | |
| Read file record | |
| Read/Write multiple registers | Point & tag identification,<br>Change program state |
| Write single coil | Change program state |
| Write multiple coils | |
| Write single registers | |
| Mask write register | |
| Write file record | |

Table 14: One-to-many mapping between the Modbus public function codes and the techniques from ICS MITRE framework

## 16 APPENDIX: ICS MITRE FRAMEWORK MAPPING TO ICS KILLCHAIN

| **ICS MITRE technique** | **ICS Killchain phase** |
|---|---|
| Exploit public facing application | Cyber Intrusion (Delivery) |
| Standard application layer protocol | Management & Enablement (C2) |
| Commonly used ports | |
| Connection proxy | |
| Control device identification | Sustainment, entrenchment, development & execution (Act) |
| Network service scanning | |
| Detecting operating mode | |
| Detecting operating state | |
| Point & tag identification | |
| Role identification | |
| Program upload | |
| Default credentials | |
| Remote file copy | |
| Module firmware | |
| System firmware | |
| MiTM | |
| Command line interface | ICS Attack (Deliver and Install/Modify) |
| Execution through API | |
| Program download | |
| Rogue Master Device | Execute ICS Attack |
| Spoof reporting messages | |
| Utilize/change operating mode | |
| Changing program state | |

Table 15: One-to-many mapping between the ICS MITRE ATT&CK framework and the ICS Killchain.