Mälardalen University
SCHOOL OF INNOVATION DESIGN AND ENGINEERING
VÄSTERÅS, SWEDEN

DVA333 - Thesis for the Degree of Bachelor of Science in Engineering - Computer Network Engineering, 15 credits

HONEYPOT

To bee or not to bee: A study of attacks on ICS/SCADA systems.

Authors: Felix Albinsson
fen18014@student.mdh.se
Jesper Riedl
jrl18001@student.mdh.se

Examiner: Tiberiu Seceleanu
tiberiu.seceleanu@mdh.se
Mälardalen University, Västerås, Sweden

Supervisor: Svetlana Girs
svetlana.girs@mdh.se
Mälardalen University, Västerås, Sweden

Date: **2021-05-26**

Abstract

In the past, industrial control systems (ICS) and supervisory control and data acquisition (SCADA) systems were planned to run as isolated networks, and not interconnect with other networks e.g., the internet or other parts of a corporate's network. Because of the isolation, no cybersecurity mechanism was required. In the modern society, ICS/SCADA systems has evolved to communicate over public IP networks and has been incorporated in a company's intranet or directly to the internet. This integration opens up for threats that were not envisioned at the time when the system was created. When ICS/SCADA systems get exposed to the internet, there is a risk that vulnerabilities in the systems get exploited by a malicious force. This can lead to data loss, destruction of data and devices, damage to infrastructure, financial losses for the company, and even loss of human life could occur. To mitigate and prevent attacks it is crucial to understand the attacks and the behaviour of the attacker. One way to achieve this is setting up a system that mimics the real system. This fake system is separated from the production network and closely monitored. The data collected can be analysed and used to gain knowledge about the attacks.

This thesis will present a possible way to study attacks on an ICS/SCADA system using a honeypot designed for this purpose. To do this, a suitable honeypot had to be found that could collect relevant data regarding what kind of attacks that may be used against an ICS/SCADA system. This was achieved by experimenting with different set ups, and the collected data was analysed. This led to the use of T-pot as the chosen honeypot and the collected data showed that a lot of the traffic were directed towards the ICS/SCADA communication protocols Modbus and s7comm. To secure an ICS/SCADA system, it is important to gain knowledge about attacks and attack vectors. A honeypot can be a useful tool that provide information regarding attacks and attackers and can be a help in setting up a defence-in-depth strategy to improve the security in an ICS/SCADA network.

Table of contents

List of Figures

List of Tables

1. Introduction

Cybersecurity is always a present matter in a modern world and this matter grows when more and more systems and devices connect to the internet. There are billions of devices connected to the internet and the explosive growth of connected devices has also increased the possibility of data breaches [1]. The digitalization also involves other systems, e.g., Industrial Control System (ICS), Supervisory Control and Data Acquisition System (SCADA) and Distributed Control Systems (DCS) [2]. ICS used to be proprietary systems, nowadays they are open and standard technologies interconnected with other networks such as the internet. Since the Stuxnet attack on ICS/SCADA systems, the security issues on these systems have become a high priority for owners of critical infrastructures [3] [4].

When all these different systems get exposed to the wild internet, the attack surface grows and this could eventually lead to more attacks and an unsafe internet, data loss for companies and private persons, destruction of data and devices, damage to infrastructure, and financial losses. To mitigate attacks, it is important to understand the attack and the behaviour of the attacker. One way to do this is to use honeypots. A honeypot is a system set up to be probed, attacked, and compromised [2]. During an attack, the system generates data that can be used to study the behaviour of the attacker, patterns of the attacker and different techniques used.

The idea with understanding the attacker, gaining the knowledge of the attack and system's vulnerabilities is to prevent systems from getting hacked and make the companies and the society aware of how vulnerable and exposed they are.

In this thesis, we focus on industrial systems and use a SCADA honeypot to identify possible attacks and understand malicious actions aiming at ICSs. The behaviour of the attacker and her interaction with the system will be analysed using the information gathered by the honeypot. We will also analyse if a cloud service like AWS will have an impact on the interaction level of the attacker.

1.1 Problem Formulation

There are many different types of honeypots as they vary in what kind of ports and services that are open for exploits and what kind of protocols they are running. Some of them are made to look like a specific system, such as an IoT device or ICS and some are more generic. There are also different types of platforms that the honeypots can run on. These platforms can be real hardware based (directly on a hardware), on a virtual machine (e.g., VMware) on local hardware, or they can run on a cloud service like AWS. This opens possibilities for different approaches on how to set up a honeypot and the result that can be achieved.

In this thesis we aim to evaluate an ICS honeypot in AWS cloud service to get a better picture of the attacks targeting ICS and how they behave. We also want to evaluate if AWS can be used as a platform for an ICS honeypot or if there can be a risk that the attacker gets suspicious and does not interact after the first contact. To do this we want to compare a honeypot on AWS and a honeypot not in the cloud and analyse the data generated by the honeypots, see the differences in attacks and attackers. To address these problems the focus will be on the following research questions.

RQ1: Which honeypot is suitable to use for studying attacks targeting ICS/SCADA?

RQ2: What kind of attacks are used against the ICS/SCADA honeypot?

RQ3: How do the attacks and attackers differ between a honeypot placed in a cloud and a honeypot located on a local server?

2. Background

In this section we present the background information that we find necessary for the reader to understand the thesis contributions. We provide information about Industrial Control Systems, the concept of honeypot, the SCADA honeypot, and ports and protocols that can be used to gain access and control of a system or inject malicious code. There will also be a description of different kind of attacks that can be used to target a system.

2.1 Industrial Control System

Industrial Control System is a system used to control, manage, or regulate devices or physical systems in industry and infrastructure [5]. An ICS contains several loops e.g., control loop using actuators, sensors, and controllers to operate some regulated process, human interfaces for engineers to monitor and control the controllers, and tools for remote diagnostic and maintenance. These tools use different network protocols on a layered network architecture [6]. The communication protocols used between devices are ICS-specific and often legacy point-to-point or broadcast [5]. These protocols were designed with the belief that devices in an ICS are connected through dedicated cabling. However, nowadays these protocols may be layered on top of other transport protocols like User Datagram Protocol (UPD) and Transmission Control Protocol (TCP) and run over Ethernet. They may also use the existing IP-based networks, as well as the internet, to communicate. An ICS has progressed from being proprietary systems to having open structural design and standard technologies, that now interconnect with other networks and the internet [2], Figure 1 shows an example overview of the architecture of an ICS/SCADA system and how different parts are interconnected. Since the security aspect in this sort of system evolution might be uncharted this could potentially lead to security vulnerabilities [2].

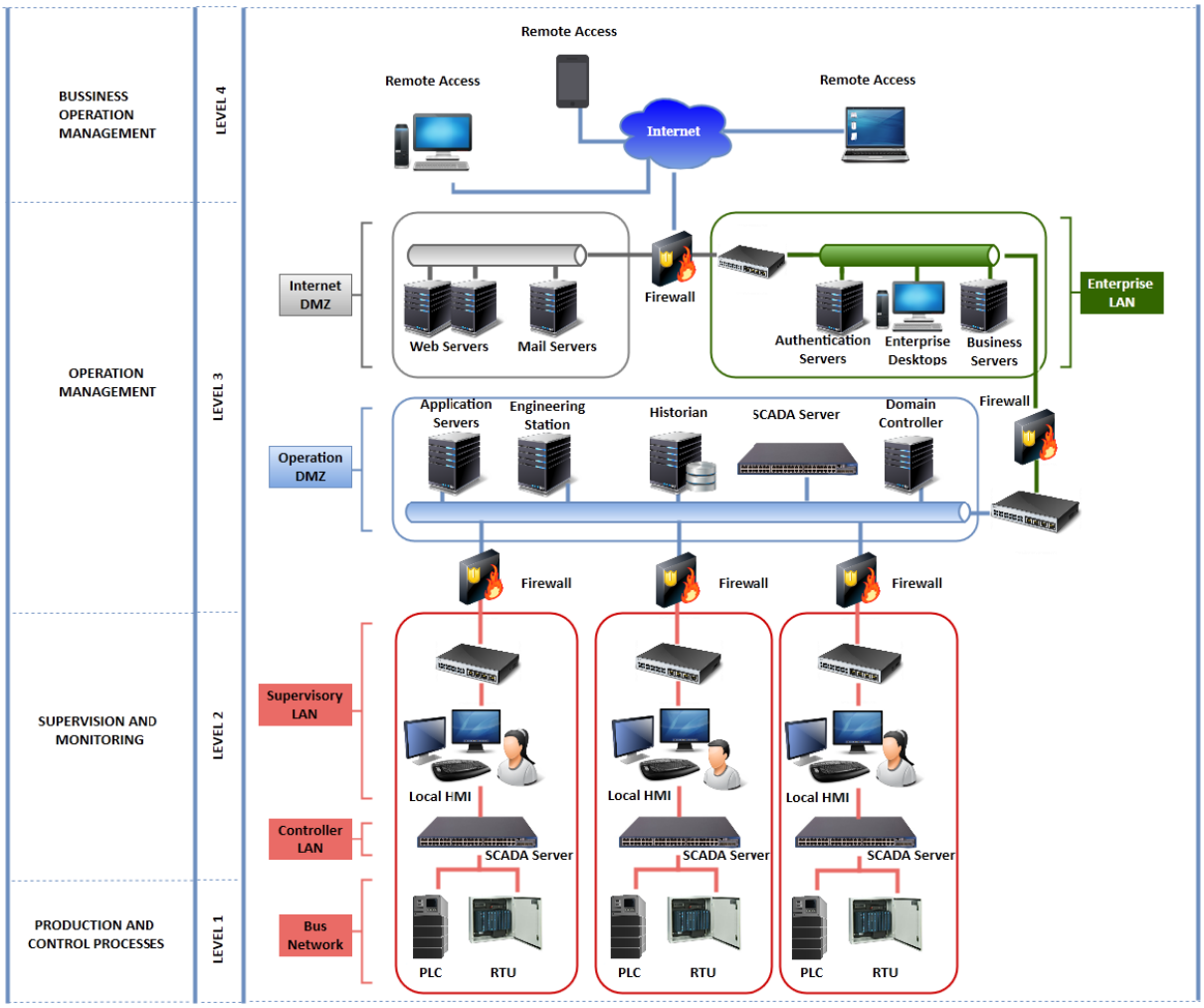

Figure 1: Overview ICS/SCADA architecture [7]

2.1.1 Supervisory Control and Data Acquisition System

One larger sub-group of the ICS is Supervisory Control and Data Acquisition System. A SCADA system is a system set of hardware and software that is used for controlling and monitoring geographically separated assets and process [3] where centralization of data acquisition and management is crucial to the system operation. The system infrastructure is typically composed of different parts [2] [8] such as a central computer, several field-based remote measurement and control units e.g., Remote Terminal Units (RTUs) and Programmable Logic Controllers (PLCs), a communication system to connect them e.g., a wide area telecommunications system, and an interface for the operator to access the system [2]. The field-based remote measurement and control units are used for measuring e.g., the environment, collect data, and send the data to the central computer that can control and supervise the field-based remote measurement and control units. The control center can run on standard Windows or UNIX operating system and for data storage traditional database software may be used.

2.1.2 ICS and SCADA communication protocols

Two very common communication protocols in ICS/SCADA systems are Modbus and Dynamic Network Protocol 3 (DNP3) [3]. Modbus is an application layer protocol and used for inter-controller or between controller communication and communication to field devices. Original Modbus Serial run over serial communication but have now been extended to run over TCP communication, called Modbus TCP. Modbus is designed for real time traffic in order to send and receive commands from e.g., a remote field unit. DNP3 is also an application layer protocol, it defines how control commands are communicated between SCADA devices in remote geographical areas. Like Modbus, DNP3 was created as a serial protocol but also got extended to work on an IP network.

Both these protocols are designed for real time purpose and to guarantee efficiency and reliability [3]. To achieve the needed efficiency, inefficient functions have been stripped away and this includes security functions, such as encryption and authentication. Nowadays, DNP3 supports authentication by a sheard key or by certificate. However, the secure mode of DNP3 is not generally used due to the lack of standardization of the key management in the SCADA system [3].

Figure 2 shows an overview of protocols used in the different levels of the infrastructure of an ICS/SCADA system.

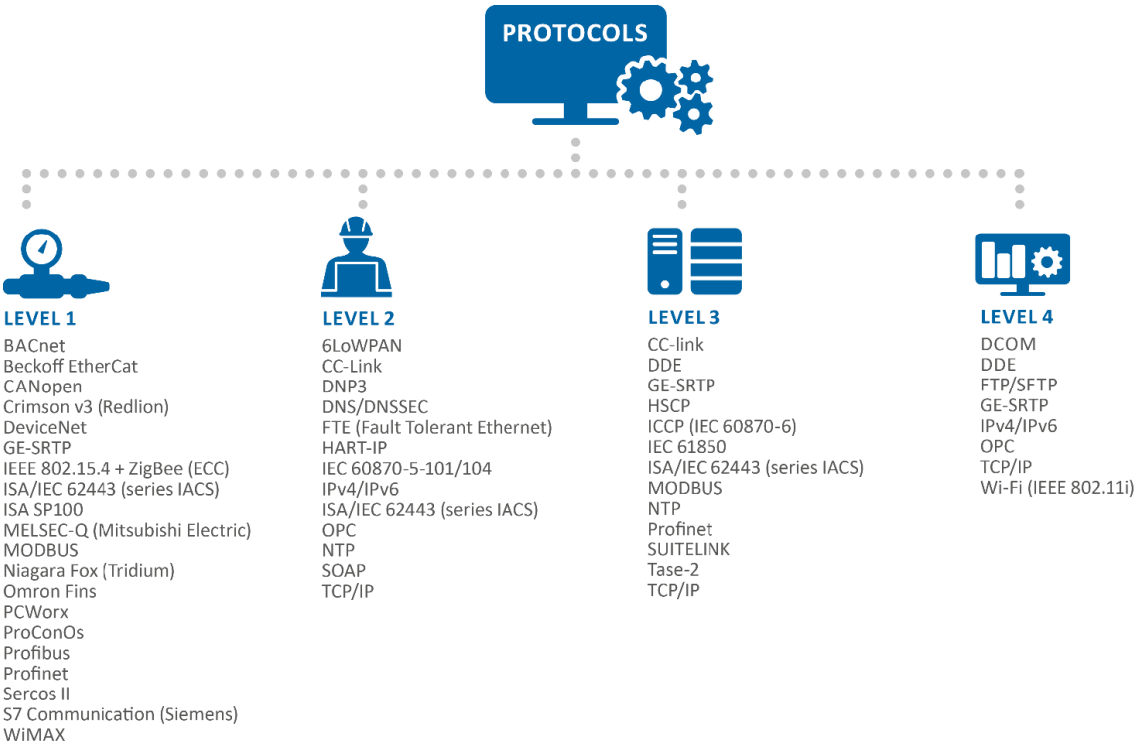

Figure 2: Commonly used protocols in ICS/SCADA systems [7]

The following protocols are some of the commonly used in the different levels of an ICS infrastructure and relevant for the thesis:

- IEC104 - This protocol is commonly used as communication protocol in SCADA systems for remote monitoring and control of large-scale systems [9]. IEC 104 encapsulate a message in a special Protocol Data Unit (PDU) over TCP/IP using port 2404. The payload of the TCP PDU contains a header, and data with the values and control messages between the RTU and the SCADA system.
- BACnet - Building Automation and Control Network is a communication protocol used for building automation and control systems [10]. The protocol is an international standard and is designed for building automation, control, and management. The protocol provides a way to interface BACnet over the internet for remote management and control. BACnet uses UDP port 47808.
- EtherNet/IP – Is a TCP/IP-based protocol used in an ICS [11]. The protocol is an adaptation of common industry protocol (CIP), which allows CIP communication to be transported over Ethernet. Ethernet/IP uses port 44818 to accept connection over TCP and UDP.
- File Transport Protocol (FTP) is a TCP/IP communication protocol used for file transfer over networks [12]. The protocol uses port 21 for commands and port 20 for the data exchange. FTP works in two modes, active and passive. The active mode strictly uses ports 21 and 20 for commands and data transfer. The passive mode uses port 21 for commands but dynamically assign a port in the range 1024 to 65535 for the data transfer.
- Hyper Text Transfer Protocol (HTTP) is an application layer protocol used in distributed collaborative, hypermedia information systems [13]. The protocol is used to transport data over the internet. The protocol uses port 80.
- Intelligent Platform Management Interface (IPMI) provides an interface to platform management services on a hardware [14]. These services provide a way to monitor, control, and

logging the hardware. This can be done over a LAN by encapsulating the IPMI message in remote management control protocol (RMCP) using UDP datagrams using port 623.

- Modbus is a protocol used for serial communication between industrial devices e.g., PLCs, RTUs and HMIs [15]. Modbus/TCP version can run over a network based on TCP/IP and encapsulate communication in an TCP segment using port 502.
- S7 communications protocol (s7comm) provides a way for communication between PLC devices and Siemens Step7 software [16]. The protocol can be used for obtaining data from a PLC in a SCADA system, program the PLC, data exchange between PLC, and for diagnostics. The protocol uses port 102 to be carried over TCP [17].
- Simple Network Management Protocol (SNMP) is a protocol which provide a way to make queries, monitor, and manage devices connected to IP networks [18]. The protocol uses UDP ports 161 and 162 to transport data.
- Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol used for downloading and uploading files over IP networks [19]. TFPT uses UDP for transport with port 69.

2.2 Types of Attacks

One of the purposes of using a honeypot is learning how to mitigate hacker attacks and to do so, one must know what sort of attacks are prevalent on the internet.

2.2.1 Denial-of-Service attack

Overloading a website with more traffic that it can handle can result in the website server crashing, not being able to deliver services to visitors of the website [20]. This is called a denial-of-service attack. Using many machines (e.g., using a botnet) in order to take down a website is called a distributed denial-of-service attack (DDoS).

2.2.2 Credential reuse

Sometimes websites or services get hacked, it is inevitable. And sometimes these hacks result in leaked credentials such as e-mail/username and password [20]. When information such as passwords gets passed around the hacker communities enough times, they create wordlists of probable passwords to use on their attacks on different websites/services. If a user reuses a password and/or if another user has that same password, chances are you will get hacked by not using a unique and secure password.

2.2.3 Malware

Malware is basically any code that could be harmful and was created with malicious intent. A few definitions of malware are worms, ransomwares, and remote access trojans (RATs) [20]. If there is a malware on a computer, a hacker might be able to control the machine and monitor all actions. They can also steal user's data, such as passwords. A great protection from malware is using an up-to-date antivirus program.

2.2.4 Port scan

A port scan is not harmful in itself, but rather is used as a probe to detect vulnerabilities in a system [20]. An attacker sends a request to a client with the aim of finding an active port that could be used to inject harmful code. A few terms in port scanning that are the majority of the scans are TCP Connect, SYN Scan, UDP scan, ACK scan and FIN scan.

2.2.5 Man-In-The-Middle

Man-In-The-Middle (MITM) is a type of attack where the hacker intercepts communication and puts him- or herself in the middle of two parties [20]. Normally a user has a private connection to a website, but during a MITM attack the entire communication between a user and what is thought of as a secure website is managed by the attacker.

2.3 Honeypots

A honeypot is a computer system that works as a decoy and its value lies in being probed, attacked and compromised [1] [21] [22]. The honeypot simulates a real system and could be used to either mislead an attacker away from a production system or to study the attacker and its pattern. In either way, any interaction with the honeypot can be deemed suspicious by definition, because the system is a decoy and there is no sense a benevolent user should interact with the honeypot system. A honeypot can be seen as a valuable tool to attract an attacker to penetrate and compromise the system and monitor these attacks and log the activities. The data logs can be studied to help understand the attacker, understand attack patterns and to prevent similar attacks in the future.

Honeypots can be classified based on what their exact purpose is and what level of interaction they offer. Looking at the purpose, honeypots can be categorised as research and production honeypots and from the level of interaction perspective, there are low, medium, and high interaction honeypots [1] [22].

Research honeypots are exclusively used in research and the purpose of them is to gain as much information as possible about the attacker and the attackers activities [1] [22]. By giving the attacker full access to penetrate the system and compromise it, valuable information can be gained about the attack. These honeypots tend to be complex as they are designed to collect broad information about the attacker [1]. The collected data can help the forensic scientists to better understand the attacker and its patterns.

Production honeypots are placed in the production network and can help the company to protect itself from an attacker and malicious activities [22]. With the assistance of the honeypot, an organisation can safeguard its internal network structure and help the network technicians to identify attacks [1]. Razali *et al.* [1] points out that these honeypots can have fewer functions than research honeypots and can be simpler in terms of implementation. This can be seen as a trade-off between ease of operation and the amount of collected data.

Low-Interaction honeypots emulate a small set of services [1] [2] [22]. These services can be secure shell (SSH), Telnet, and file transport protocol (FTP). These types of honeypots typically do not provide the attacker any access to an operation system to interact with. Because of the limitation of the system, the attacker's interaction with the system is probably short and limited to login attempts. These types of honeypots can be good for static evaluation due to the minimal response to an attacker.

Medium-interaction honeypots are running false susceptible services and can interact with the attacker by fake requests [2]. Like the low-interaction honeypots, these honeypots do not provide an operation system for the attacker to interact with [1], but they contain a higher level of services to attract the attacker. The result of these honeypots is that they generate sufficient quantity of replies that can cause a follow-up of the attack.

High-interaction honeypots are not created on simulation of services or operation systems [2]. Instead, they depend on authentic operation systems or genuine services that an attacker can interact with. These honeypots are complex systems, and the implementation and maintenance levels are higher for low- and medium-interaction honeypots [1]. A high-interaction honeypot offers an unrestricted operation system and environment with a vast set of services installed for the attacker to exploit. This will generate a lot of data about the attacker and the attack pattern that can be analysed and used to gain information about the intrusion.

2.3.1 ICS/SCADA honeypots

SCADA honeypots attempt to imitate an authentic SCADA system [8]. Conpot and T-pot are two honeypots that can be used to emulate an ICS/SCADA system.

Conpot is a low-interaction honeypot [8] and it can be used to collect information about the motives and techniques of attackers targeting ICS. Conpot logs events of HTTP, FTP, TFTP, SNMP, Modbus/TCP, S7comm, BACnet, IPMI, IEC 104, EtherNet/IP and, CIP [23].

T-pot is a multi-platform honeypot that contains several honeypots and services [24]. It can be run in different modes, each mode using a different set-up of honeypots and services [25]. The honeypot daemons and assistance modules are dockered, which allows T-pot to run several honeypot daemons on the same network interface. This also allows each honeypot to be contained within its own environment. The supporting tools help collecting, present and visualize the data. T-pot uses Conpot as one of the implemented honeypots. Cowrie is another of the implemented honeypots, which is a low-interaction honeypot made for logging SSH events [26].

System requirements are:

- 8 GB RAM.
- 128 GB SSD.
- Network via DHCP.
- A working, non-proxied, internet connection.

Figure 3 shows rough overview of the architecture of a T-pot, just to show that T-pot incorporates a lot of services.

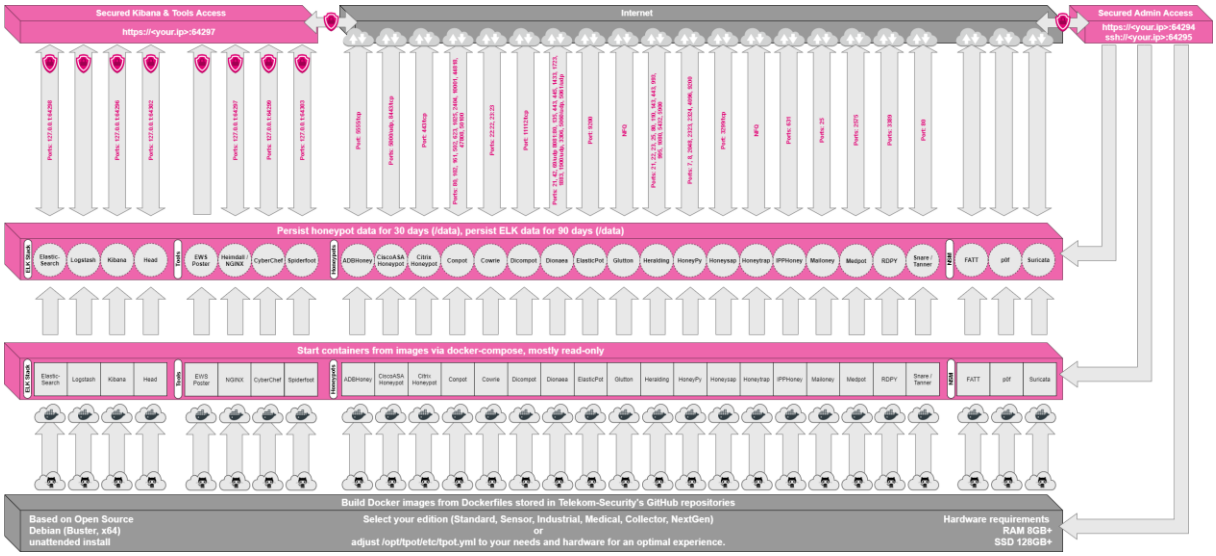

Figure 3: T-pot architecture [25]

T-pot includes the following tools [25]:

- Cockpit - a lightweight web-ui for docker, os, real-time performance monitoring and web terminal.
- Cyberchef - a web app for encryption, encoding, compression and data analysis.
- ELK stack - visualize all the events captured by T-pot.
- Elasticsearch Head - a web frontend for browsing and interacting with an Elastic Search cluster.
- Fatt - a pyshark based script for extracting network metadata and fingerprints from pcap files and live network traffic.

- Kibana – an analytics and search dashboard for Elasticsearch.
- Spiderfoot - an open-source intelligence automation tool.
- Suricata - a network security monitoring engine.

2.3.2 Protocols, ports, and services

Both Conpot and T-pot are built to expose some common protocols and services that are used in an ICS/SCADA system. Figure 2 shows and overview of commonly used protocols in an ICS/SCADA system. Some of these protocols are used by the honeypots to emulate an ICS/SCADA system. T-pot and Conpot emulates some of the commonly used protocols like IEC104, BACnet, EtherNet/IP, s7comm, IPMI, Modbus, SNMP, FTP, HTTP, TFTP. T-pot and Conpot also imitate two other services, one is Guardian AST which emulates a gas pump monitoring system [8]. The service uses port 10001. The other one is Kamstrup which is emulating a smart meter [8]. The service is using ports 1025 for the kamstrup_protocol and 50100 for the kamstrup_managment_protocol.

2.4 Cybersecurity tools and services

Different tools and services are used in the name of cybersecurity. Cybersecurity tools and services include portscanners, online mass-scanners, IDS:s and online vulnerability databases.

2.4.1 Nmap

Nmap (short for Network Mapper) is a vulnerability scanning tool used to identify what ports and services are running on a system [27]. The general idea of a port-scan tool like Nmap is that it sends a request to a host (i.e., an IP address) and sees what ports are open on that specific host. The ports are often well-known as to what service they are associated with. An administrator/hacker can therefore deduce what services that are run on a host just by seeing the ports that are open.

2.4.2 Shodan

Shodan is online search tool and a crawler explicitly designed for finding devices exposed across the internet e.g., ICS and SCADA systems and another network equipment [28]. Other scan services also exist, two examples are Censys and Binaryedge. Shodan is however the most prominent scan service used online.

2.4.3 Suricata

Suricata is a combination of an IDS, IPS and network security monitoring engine. It is run by the non-profit foundation, Open Information Security Foundation (OISF) and is open source based [29]. Using Suricata, data packets that passes through one's network can be scanned and determined if it is malware.

Suricata uses different solutions for its implementation, where two main ingredients are:

- Common Vulnerabilities and Exposures (CVE)
- Emerging Threats (ET)

T-pot implements Suricata when determining what sort of exposure or vulnerability the hackers try to use.

2.4.4 Common Vulnerabilities and Exposures

Common Vulnerabilities and Exposures (CVE) is a list of publicly disclosed computer security flaws [30]. CVEs are assigned ID numbers to help identify what sort of vulnerability or exposure that has been identified. They contain short technical information as to what the CVE is referring to. CVEs are used by many administrators to help securing networks.

2.4.5 Emerging Threats

Emerging Threats (ET) Intelligence provides threat intel for IPs and domains involved in both suspicious and malicious activity [31]. ET is a database that can be implemented in IDS and IPS systems, including Suricata. It notices trends and logs timestamps of a threat and the type of threat and exploit kit used.

3. Related work

There has been a wide variety of studies using honeypots over the years. Studies are ranging from using honeypots to detect ransomware [32] to capture IoT attacks [33] or to analyse possible attacks on industrial systems [2]. There is no doubt about the usefulness of analysing the information a honeypot could generate, the question is how and where it could be applied best, and above all what sort of information can be acquired.

Amit Tambe *et al.* [34] suggests getting a general honeypot architecture using simple IoT devices. The devices could be placed at the same physical location and using a VPN it is possible to make it look more geographically distributed than it is. An attacker would have a hard time distinguishing these fake devices from real vulnerable devices and thus information about attack methods.

Not all implementations need to have a concrete problem formulation beforehand. A study with a more general approach has also been made [35] where honeypots are implemented and information is afterwards analysed, resulting in lessons learned and what steps to take from there.

There are some different approaches on which type of platform and which type of honeypot that have been used to study attacks on an ICS/SCADA system. Some of these implementations and results are presented in [2] [5] [24] [8]. Amine Belqruch *et al.* [2] observed by using a medium-interaction SSH honeypot in a network, that the honeypot could be a helpful tool for monitoring and defence purposes. The honeypot could also help an ICS/SCADA administrator learn how someone is trying to attack the network. But these results may be different if they were conducted in a live environment and not in a closed and controlled laboratory environment which they used. Michael Dodson *et al.* [5] points out in their recommendation that the honeypot should be a high-interaction honeypot to avoid being fingerprinted by the attacker and to mislead attackers and get a better understanding of the hacker's intentions.

The authors of [24] have analysed attacks against ICS/SCADA by using T-pot implemented in AWS. T-pot is a multiplatform honeypot that can be run in different modes to emulate different systems, e.g., industrial mode (ICS/SCADA). The collected data stretches over a period of 13 days and they mention that the time frame may have been too short for collecting sufficient data regarding some of the ICS ports. Some interesting aspects here are the use of AWS and the short time conducting the experiment. By conducting the experiment for a longer period and not using a cloud service, it could possibly lead to different results. Michael Dodson *et al.* [5] have some recommendations based on their study regarding the use of AWS. They write that honeypots should use a realistic IP address and not a cloud service like AWS since ICS devices are unlikely to be connected via a cloud service provider.

Arthur Jicha *et al*. [8] analysed Conpot, which is a low-interactive ICS/SCADA honeypot. The focus of their paper is the evaluation of the effectiveness of Conpot as an ICS/SCADA honeypot. This is done by studying which ports Nmap and Shodan determines are open and compare them to the ports opened by Conpot. This knowledge can be useful when implementing an ICS/SCADA honeypot to make sure the open ports on the system correlates to the ports the ICS/SCADA honeypot have open.

Daniele Antonioli *et al*. [36] made an implementation with a virtual ICS honeypot using frameworks from different honeypots. Their main goal was to create an ICS honeypot that is high-interaction with a few specific requirements. These requirements are creating a realistic (multiple services supported), low cost, reconfigurable honeypot that targets ICS domains and is usable both for research and production. Their conclusion is mainly how difficult it is to first define what high-interaction honeypot actually means in an ICS/SCADA environment, they do however formulate a useful classification of how a virtual, server-based ICS/SCADA honeypot can be used to trick an attacker thinking they are inside a real ICS environment.

Ramachandruni and Poornachandran [37] made an extensive study on what sort of attack vectors an ICS/SCADA system might have to be hardened against. They set up a honeypot in a University network and reported that most of the attacks were against Modbus based devices. They concluded that they might have gotten more targeted if they were not in a University network range since this is a known range and potentially easily recognized as something else than an ICS/SCADA system IP address.

The survey shows that if we would like to compare AWS versus a local VM we might have to rethink the validity of the results if the local VM is placed within our own University's IP range. Best case scenario would be to place a honeypot within the DMZ network of a real industry. Regardless of the sort of the network, honeypots can help in logging unusual traffic and serve as the sacrificial system in the event of an attack. More attack vectors might exist, and SCADA vendors must identify them and share them with the industrial community. SCADA systems are very crucial and sensitive systems so more attention is required in securing them.

4. Method

Figure 4 shows the steps that are followed while working on this thesis project. The work starts with a literature study, which will be an ongoing task during the entire thesis when we need to find new information or validate the data we have already gathered. A rough review will be made in the beginning of the project by reading papers we find interesting. If they seem relevant and the need for discussion may arise, they will be cited in the report [38, p. 25]. When relevant, each paper's conclusion and data will be summarized and discussed in order to support or invalidate our theories and claims. During the study, if we find any data odd or any conclusions wrong, counterarguments will be made as to why we oppose the authors conclusions to their own data. The focus in the beginning will be collecting data and organize it, later in the thesis the collected data will be compared to other studies to form an understanding and conclusion on our own results.

Zobel gives guidelines [38, p. 25] on how to proceed as the study continues. Some factors are more important than others, a few of them are how closely related other authors' work is to ours and how influential they may have been in guiding us. He also suggests being very inclusive in adding papers for discussion. As our own thesis work goes on, less relevant papers may be removed, and the more relevant ones will take a bigger role in the literature study part.

The different steps in Figure 4 will provide us with a structured way to answer our research questions. During the literature study, information will be gathered about ICS/SCADA systems and different honeypots and how they work. When studying more papers about ICS/SCADA systems we will learn if there are other suitable honeypots we can use and if there indeed is a problem with using ICS/SCADA system honeypots on a cloud service. This will be combined with our own investigation and answers can be found to the different research questions. During the setup and evaluation, more knowledge about the chosen honeypot will be collected. This will help us answer the first research question. The experiment part allows us to test different setups of the honeypot and a brief evaluation of the collected data will guide our setup process and how to setup the honeypot so collected data is relevant and valid to answer research question 2 and 3. The collected data will be analysed and presented as results. The results will be discussed, and conclusions will be made.

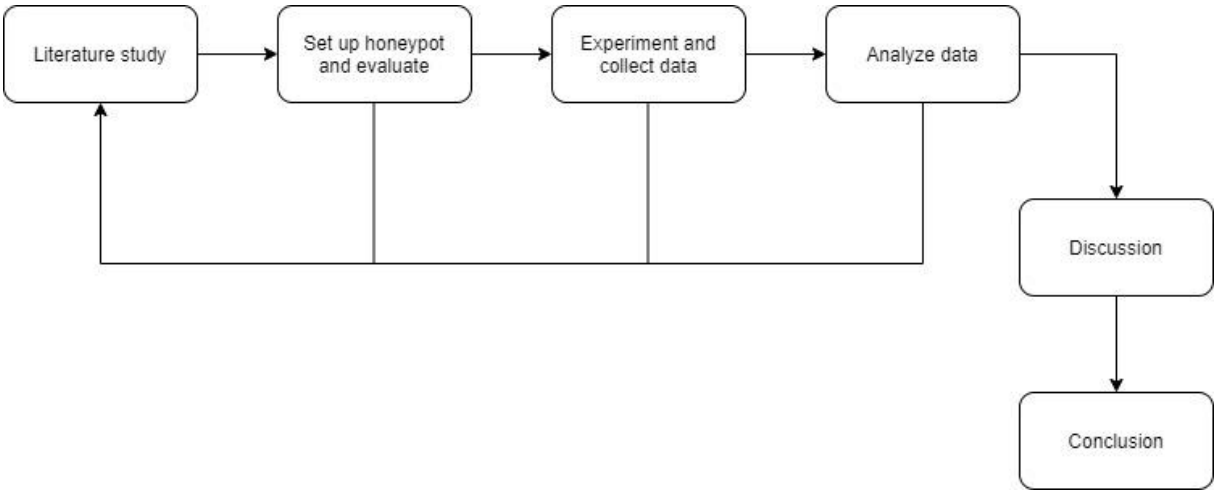

Figure 4: Chart of workflow

5. Ethical and Societal Considerations

Due to the nature of honeypots and the goal of this project it is important to consider the ethical and societal aspects. One purpose of the honeypot is to lure an attacker to penetrate and compromise the honeypot, this is done for a research purpose. Mokube and Adams [39] mention a few legal issues and challenges in implementing honeypots. These are entrapment, privacy, and liability.

With liability the authors [39] mean that an attacker might use the honeypot to harm others. This is more of an issue with a high-interaction honeypot, which we do not plan to implement. The use of low-interaction honeypots and how they work will prevent an attacker to implement irreversible effects on the system and will not be able to use it to harm others.

We classify our honeypots as research honeypots. Mokube and Adams [39] point out that this type of honeypot is used to gain information about an attacker and can provide information how an organization can protect themselves from threats. Every country has different laws regarding the collection of information, which makes this matter more complex. Our opinion is however that the engineers Code of Honour fully support our honeypot approach [40, pp. 20-21].

Entrapment can be a possible aspect of ethical consideration using a honeypot. According to Mokube and Adams [39] entrapment applies when a government acts in a way that causes a perpetrator to commit a crime. They mean that entrapment cannot be applied on private operated honeypots. As we see it and how we intend to use the honeypots, we believe that entrapment is not an issue for us. Our purpose is for research and we are not advertising our system and in that we are not inviting any one to attack and penetrate our systems, also we are acting independently from the government. If someone finds the honeypot machines and attacks them, we believe they cannot use entrapment in their defence.

Regarding privacy, due to the nature of our honeypots and how they collect and store data, there is some consideration to account for. The collected data can contain information that may reveal who is attacking, from where the attack is originated, data that may have been uploaded or tried to be uploaded, ASN of the network owner, and which domain is registered to an IP-address. If we gather information that could potentially link a single user to it, consideration will be taken.

One societal consideration is that hackers could take an advantage of honeypots and learn how they operate and work around them. In this way honeypots could in essence create a better hacker and potentially make it harder to secure a system. But we think the advantage of honeypots and the information that can be gained from the collected data outweighs the expected disadvantages. Because a hacker needs to interact with the honeypot to gain the knowledge and with this interaction the honeypot will collect data about the hacker, this information could be used to gain knowledge about the attacker and the techniques used.

6. Honeypot selection and setup

Conpot [2] is one option to study attacks on ICS/SCADA systems. The question might arise whether it is suitable on a cloud service, since the IP range of e.g., AWS is publicly known, this might deter attackers when they realize it is not a local ICS system. Initially we cannot say for sure if this is a problem, but a possible counteraction is relaying the data from a cloud server to a local machine with an IP not from a cloud service. This is done using a proxy or VPN service. T-pot is another option to study attacks on ICS/SCADA systems. But like Conpot the same question and problem might be relevant here. Another concern with both these honeypots is if a scan service e.g., Shodan and Nmap recognise them as honeypots and flag them. This could possibly lead to attackers backing off and not initiate an interaction with the honeypot.

Both T-pot and Conpot can be suitable to study attacks on an ICS/SCADA system. T-pot is a multi-platform honeypot which includes, among other things, Conpot. One crucial factor is the availability of hardware resources. If resources are not a problem, then T-pot can be a good choice. But if the resources are limited, then Conpot can be a better choice as it requires less resources than T-pot. T-pot´s advantage is the included tools Kibana, Cockpit, Cyberchef, ELK stack, Elasticsearch Head, Fatt, Spiderfoot and Suricata. These tools provide a graphic web user interface and an admin control panel for monitoring, extract data and visualize it. To extract data from Conpot's logs in a structured and readable way it would probably be best to build a parser in e.g., Python. Based on these aspects, T-pot was the choice to use. This would save us time from building a parse, that we instead could spend on analysing the collected data.

The setup contains one honeypot on a local machine and one honeypot in the cloud using AWS. Both honeypots are T-pot with the industrial flavour. Over time, modification had to be done to the honeypots to avoid getting fingerprinted as honeypots on Shodan. T-pot offers a wide variety of setups where the user can choose what sort of system he or she wants to mimic. To answer our research questions, a choice was made to use the industrial template. This template includes setting up honeypots to mimic different commercial and industrial systems, not all of which are connected to an ICS. Through several test setups on both the local and cloud honeypots, an implementation was found that gets the honeypot machines flagged as an Industrial Control System on Shodan.

As the intention was to be as authentic as possible using the instruments given by T-pot, new settings were tried if and when Shodan managed to flag the machines as honeypots, rather than as pure ICS:s. When the machines got flagged as honeypots, a new public IP was also set up so that in the eyes of an outsider it would look like a completely new system.

In the first test setup, the honeypots were setup with an un-filtered ingoing and outgoing connection. This means that every single port (range 0-65535) was opened and accessible for a hacker to get in through. The result of this was that just in a few days Shodan had no trouble flagging the machines as honeypots.

Before the second setup, changes were made in the "industrial.yml" file that T-pot uses to set up the docker honeypot containers. The not needed honeypot variations were removed from the configuration file, e.g., meaning honeypot variations such as Dicompot and Medpot were removed. These variations are more suited for the medical industry rather than industrial, hence the decision to not install them. New IP addresses were acquired on both machines. Instead of having all ports open, the ports open to the machines were narrowed down to only the ones necessary for the honeypots to work as intended. The open ports setup can be seen in Table 1.

PORT	SERVICE
21	FTP
22	SSH
23	Telnet
69	TFTP
80	HTTP
102	s7comm
161	SNMP

502	Modbus
623	IPMI
1025	Kamstrup
2404	IEC104
10001	Guardian AST
44818	EthernetIP
47808	BACnet
50100	Kamstrup Management
64294-64297	T-pot admin interface

Table 1: The necessary ports to run the honeypot machines with all ICS services used.

Shodan flagged both machines as honeypots in a few days. This is where the decision was made to start blocking specific ports as we were suspecting the static responses given when probing the respective ports gave away that the machines were honeypots.

Including showing that the machine is a honeypot, Shodan also shows what ports they noticed were open. This means it is relatively easy to determine what port is a giveaway if one manages to check Shodan just as they discover a specific port open while they flag the machines as honeypots.

In the next test setup, port 2404 was closed and the rest of the ports from Table 1 were kept open. Shodan flagged the machines as honeypots again. This process kept going where Shodan still managed to flag both machines as honeypots and more ports were closed before acquiring new IP addresses.

Right before the last test setup, two final ports were decided to be closed on the local machine and three on the cloud machine. Due to the limited time we chose to close more than one port. The ports closed at the same time in the last setup are port 21, 69, and 50100. Based on earlier setups we have concluded that port 21, port 69, and port 50100 are major triggers to get flagged as a honeypot. This is probably not due to the ports themselves, rather the answer the ports give when probing them. For example, port 21 gives a static response with a nonsense string (this string includes “Technodrome”, which is a term from a famous comic book series). Shodan could easily flag this specific string as being connected to a honeypot. Probing port 50100 gives a static MAC address as a string response. Using the same logic this string could also lead to Shodan flagging the machine as a honeypot.

Table 2 shows a short summary of the approach on closing ports in our different setups when Shodan flagged the honeypots as honeypots.

PORT	**SERVICE**	**PORTS CLOSED**
21	FTP	Fifth setup
22	SSH	
23	Telnet	
69	TFTP	Fifth setup
80	HTTP	
102	s7comm	
161	SNMP	
502	Modbus	
623	IPMI	
1025	Kamstrup	
2404	IEC104	Second setup (cloud). Never open on local machine
10001	Guardian AST	
44818	EthernetIP	
47808	BACnet	
50100	Kamstrup Management	Third setup, reopened fourth setup, fifth setup

Table 2: Ports closed during the different setups.

In the end, ports 21, 69, 2404, 10001 and 50100 were kept closed and the rest were open. All of the closed ports were decided based on our trial-and-error-method. Between every test setup, a few days

could go without almost any attack at all, see figure 5. A similar pattern was seen on both the cloud honeypot and the local honeypot. The attacks started again when the machines showed up on Shodan. Shodan does not flag the machines as anything at first, just that they are online and exist.

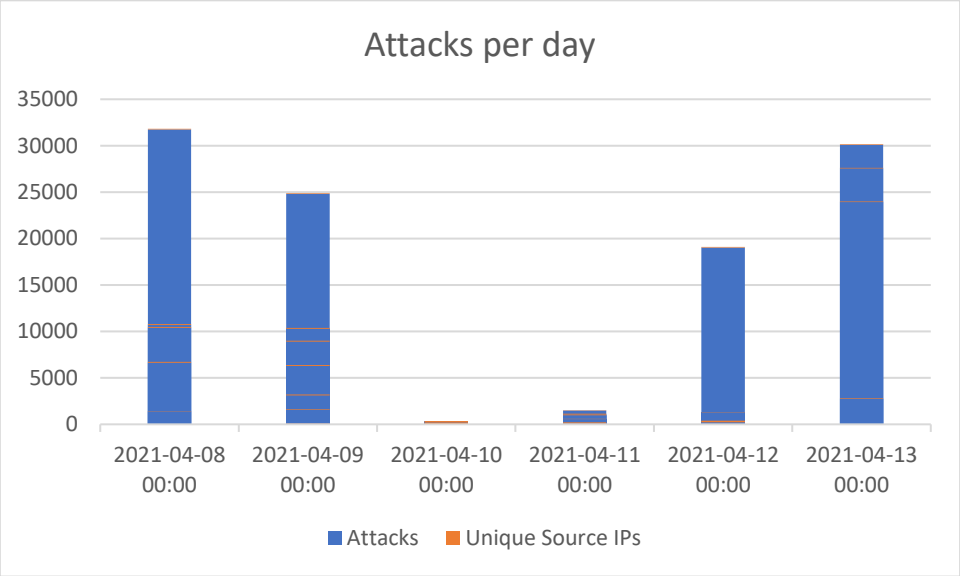

Figure 5. Old IP and new IP with a drop of attacks in the gap. The attacks began to grow again when the new IP showed up on Shodan.

Port 44818 triggered Shodan to flag the system as an ICS. Having the honeypot machines flagged as ICS systems is crucial for getting accurate results.

In conclusion, the working ports for having a functioning, industrial honeypot without it getting tagged as a honeypot are 22, 23, 80, 102, 161, 502, 623, 1025, 44818, and 47808. This is based on the time interval we ran our test. Other result may show up if the test we conducted would have been done during a longer time interval. We are currently avoiding ports 21, 69, 2404, 10001, and 50100 as we believe the response is what gives the machine away as a honeypot.

One can probably avoid detection on Shodan and still leave all the ports opened. This could be done by customizing the static responses given when probing each respective port. The perfect custom strings would need in-depth knowledge of ICS/SCADA systems; however, it would probably be adequate to just change small parts of the already existing strings to something similar to avoid fingerprinting.

7. Results

The results of the process of obtaining information and collecting data during the different setups will be presented and explained down below. The result is structured in a way related to the research questions.

First, Figure 6 and Figure 7 show the top 10 of most frequent types of attacks on the two machines that were used in the experiment. Most of the attacks launched on the cloud machine are brute-force attacks on SSH (port 22). A difference is seen on the local machine where the most frequent attack is labelled as CVE-2020-11899 [41], which in broad terms is an IPv6 vulnerability where the attacker gets read access on memory where sensitive data might exist.

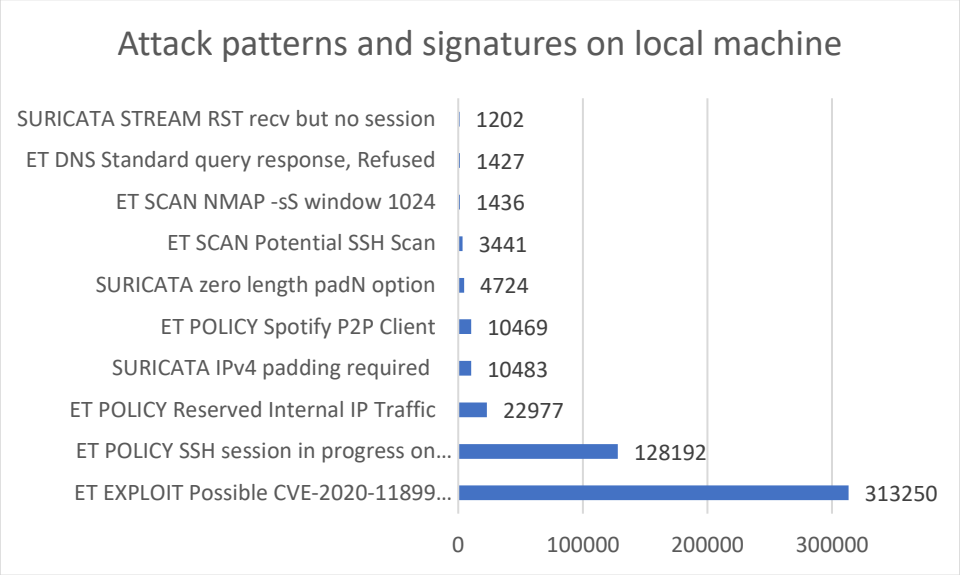

Figure 6. Attack patterns based on Suricata, which uses ET and CVE:s to determine type of attack used..

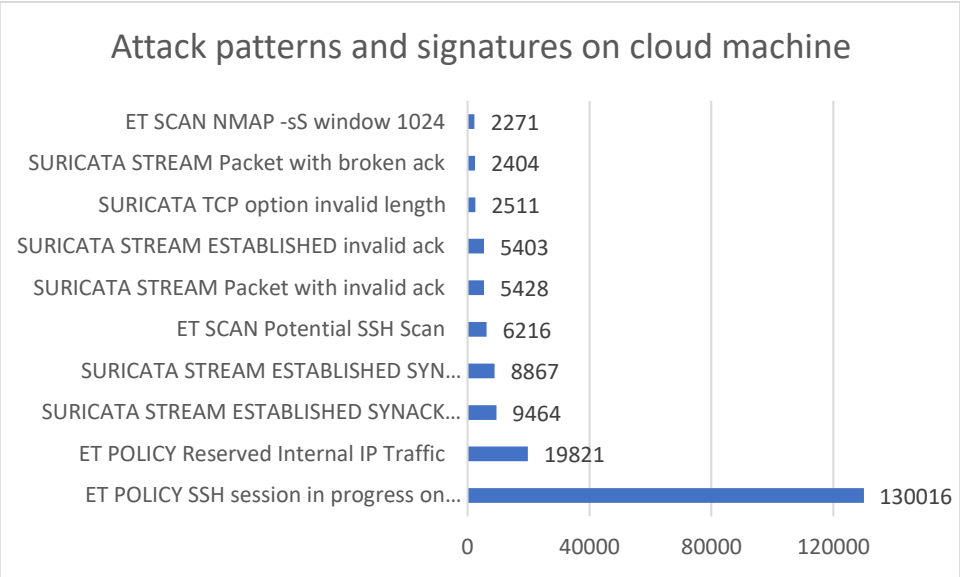

Figure 7. Attack patterns based on Suricata, which uses ET and CVE:s to determine type of attack used..

If factoring out the IPv6 out-of-bounds attack, a similar pattern is seen on both machines. The most frequent attack by far is SSH brute-force attack. Other attacks occurring, albeit less frequent, can also be seen in the figures.

Attacks against protocols associated with specific ICS/SCADA industry systems can be seen in Figure 8 and Figure 9 below. While http is the most targeted protocol, it is also a more generic protocol than many other. For example, kamstrup, s7comm and Modbus are specific ICS/SCADA protocols, and they are all subject for attacks, as can be seen in Figure 8 and Figure 9.

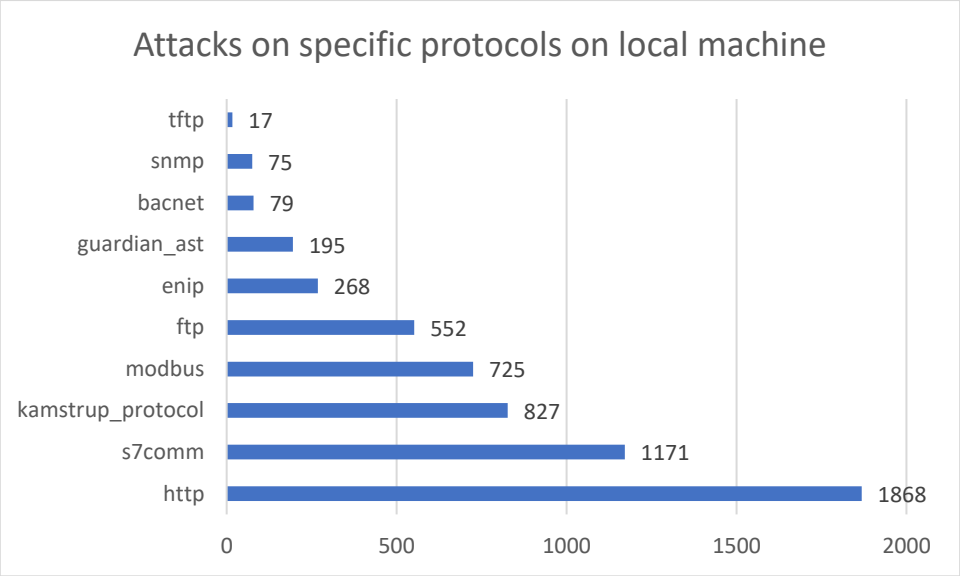

Figure 8. The protocols targeted on the local machine.

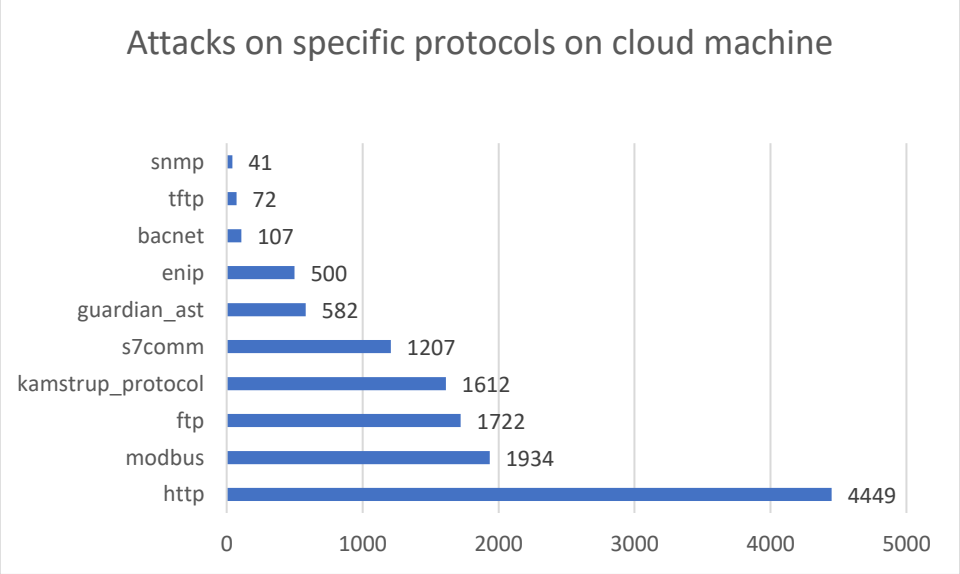

Figure 9. The protocols targeted on the cloud machine.

It is important to point out that while protocols such as http, ftp and tftp are used in a bigger picture in industrial systems, these are not protocols specifically designed for usage in ICS/SCADA systems, as can be read in the next section. They can however be used to acquire non-authorized access within an ICS/SCADA system if the security is not properly addressed.

7.1 Attacks on ICS/SCADA protocols results

During the period 2021-03-27 until 2021-05-04 there were 5937 attacks on the ICS/SCADA protocols in the cloud and 3194 attacks on the local machine. Figure 10 and Figure 11 show how the attacks were distributed over the protocols. Protocols IEC104, kamstrup_management_protocol and guardian_ast are

underrepresented or missing in Figure 10 and Figure 11, this is because of the close-down of ports over time during the different setups. The IPMI protocol on port 623 is missing completely from the graphs, although in the log files there are entries that confirm that there have been activities to port 623. But the logs do not reveal what the payload is. Log examples from the different protocols can be found in Appendix A.

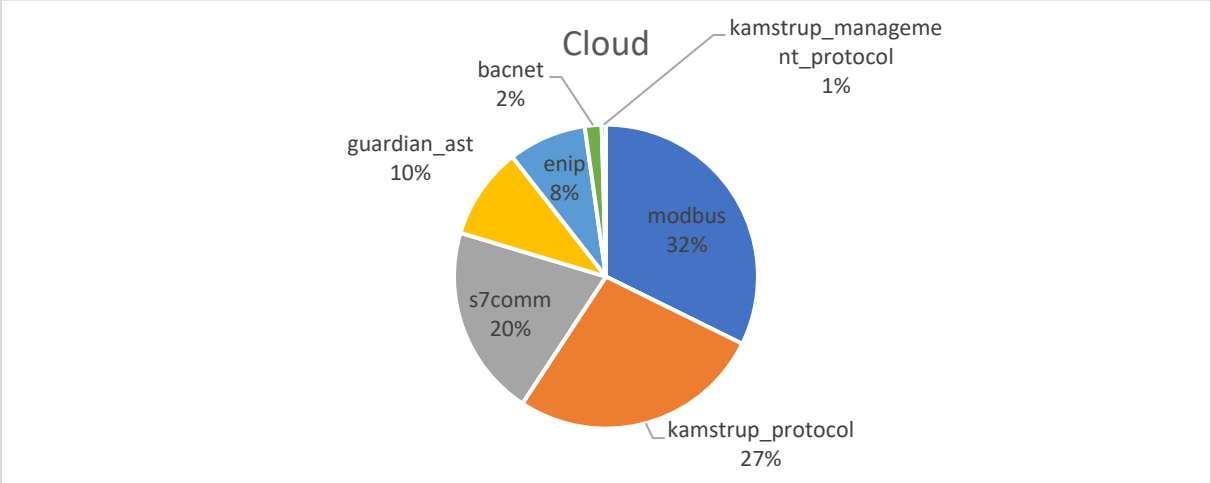

Figure 10: Distribution of attacks on ICS/SCADA protocols in the cloud

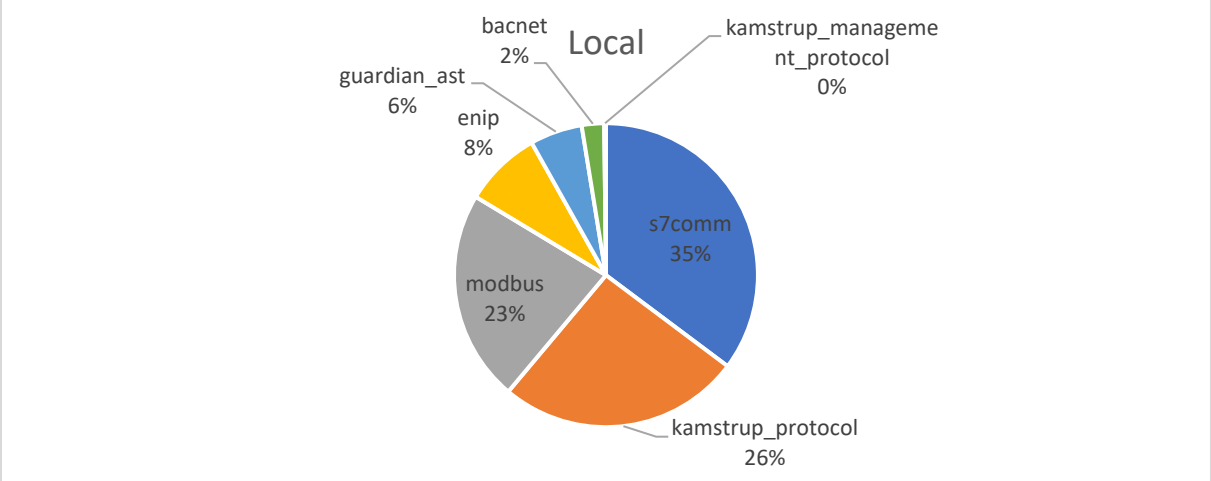

Figure 11: Distribution of attacks on ICS/SCADA protocols on the local machine

Figure 12 and Figure 13 present a brief overview of the top 30 source IP addresses from where attacks or probing were made. A predominant part of these addresses are associated with a scan service, e.g., Shodan, Censys, and BinaryEdge among others.

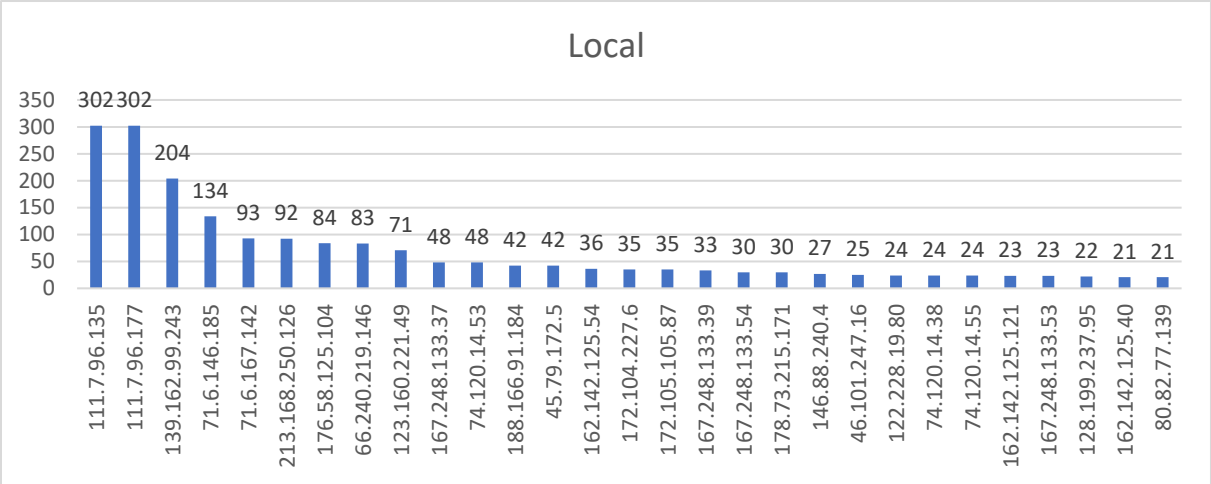

Figure 12: Top 30 attack source IP on local machine

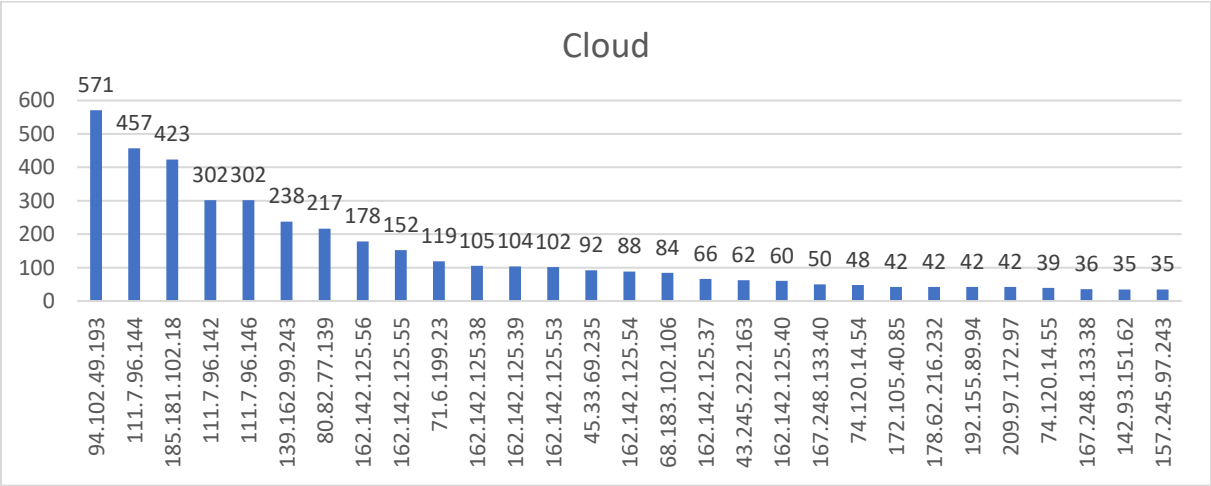

Figure 13: Top 30 attack source IP on cloud machine

Table 3 shows how many of these top 30 attack source IP-addresses on the cloud and the local machines that belong to scan services. Shodan, Ipip, Censys and BinaryEdge are all scan services and the "Other" are the ones that we only can see belong to a specific ISP or ASN.

SERVICE	CLOUD TOP 30	LOCAL TOP 30
Shodan	4	4
Ipip	1	1
Censys	13	11
Binaryedge	5	6
Other (not known to us as a scan service)	7	8

Table 3: Top 30 IP-addresses, how many belongs to a scan service.

The results of traffic and specific commands sent to the different ICS/SCADA protocols on the local machine and the cloud machine are represented in Table 4.

Protocol	Cloud Number of attacks	Local Number of attacks	Specific commands Request and response
S7comm	1210	1125	PDU type 1 and 7
Modbus	1923	719	Function code 17 and 43
kamstrup_protocol	1602	827	No valid requests
kamstrup_management_protocol	26	6	No valid commands
guardian_ast	581	178	I20100 command attempt, Non ^A command attempt
BACnet	494	263	No PDU sent
EtherNet/IP	76	106	Valid request
IEC104	-	-	

Table 4: Number of attacks, commands sent, and PDU type on the cloud and local machine.

The PDU types received on port 102 (s7comm) were of type 1 and 7. PDU type 1 is a "job request" PDU and is sent by the master and it could be, e.g., read/write memory, read/write blocks, start/stop device, setup communication [42] [43]. PDU type 7 is a "request user data" PDU and contains a request/response id.

Regarding Modbus, only two different function codes were used. Function code 17 is "report server ID" (serial line only) and is used to read the description of the type, the status, and other information specific to a remote device [44]. Function code 43 is "read device identification" and is used to read the identification of a remote device and get further information of the physical and functional description.

Regarding the other protocols, the logs showed traffic activities on those ports, but with less information compared to Modbus and s7comm. EtherNet/IP and guardian_ast showed traffic with valid commands and requests but not any information about these commands. Kamstrup_protocol, kamstrup_management_protocol, and BACnet did not show any valid request or valid commands, the logs only showed that connection had been made but not much more.

Number of attacks on the cloud and local machines are similar and does not show much of a difference. The traffic patterns during the studied time interval for the different ICS/SCADA protocols are shown in graphs below. One noticeable observation on the local machine is the absence of traffic to the ports between 16:th of April to 20:th of April. This is due to an internal discussion on which ports to open and close, during this time all ports were closed.

Figure 14 and Figure 15 show the traffic patterns for s7comm on both the local and the cloud honeypot. The patterns are similar to each other and not much different.

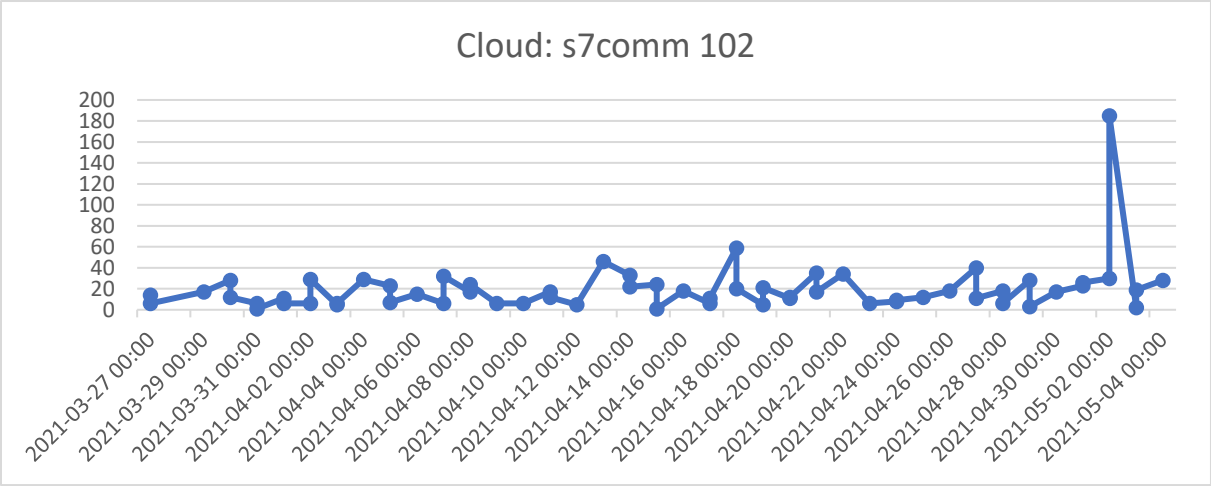

Figure 14: s7comm traffic over time on the cloud honeypot

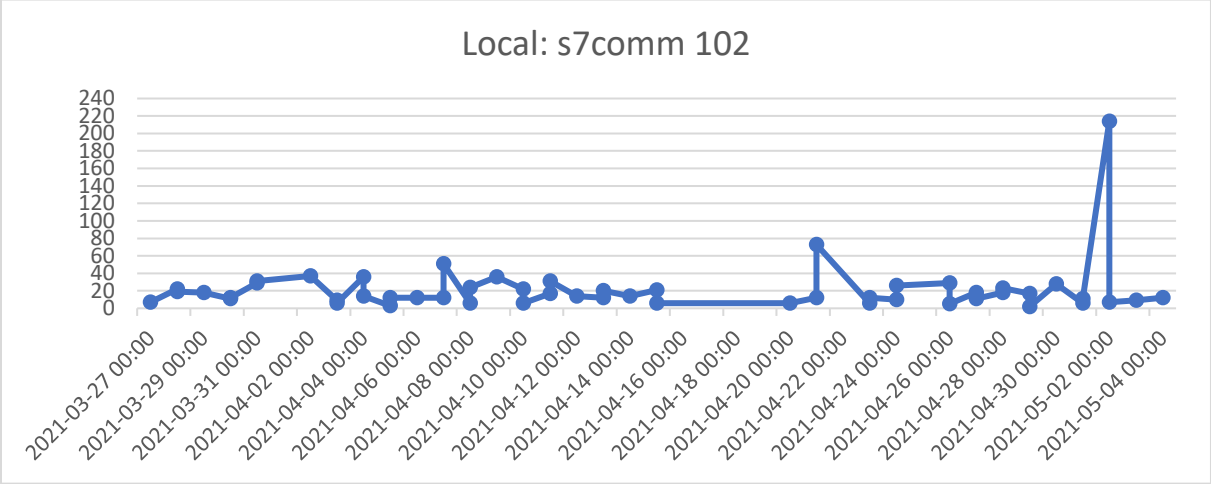

Figure 15: s7comm traffic over time on the local honeypot

Next, Figure 16, Figure 17, Figure 18, Figure 19, Figure 20, Figure 21, Figure 22, Figure 23, Figure 24 and Figure 25 show the traffic related to Modbus, kamstrup_protocol, guardian_ast, BACnet and EtherNet/IP. These figures show there is a difference in the total amount of attacks on the local honeypot compared to the cloud honeypot. There is approximately twice as many against the cloud honeypot. One reason for this may be that the IP range AWS uses could be better known and therefore scan-services scan the IP range more frequently.

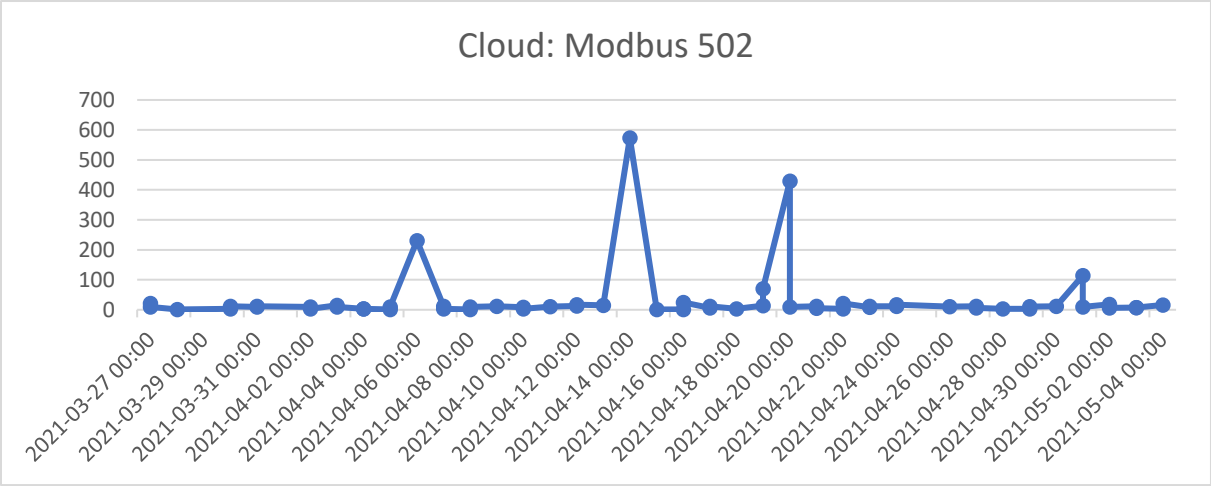

Figure 16: Modbus traffic over time on the cloud honeypot

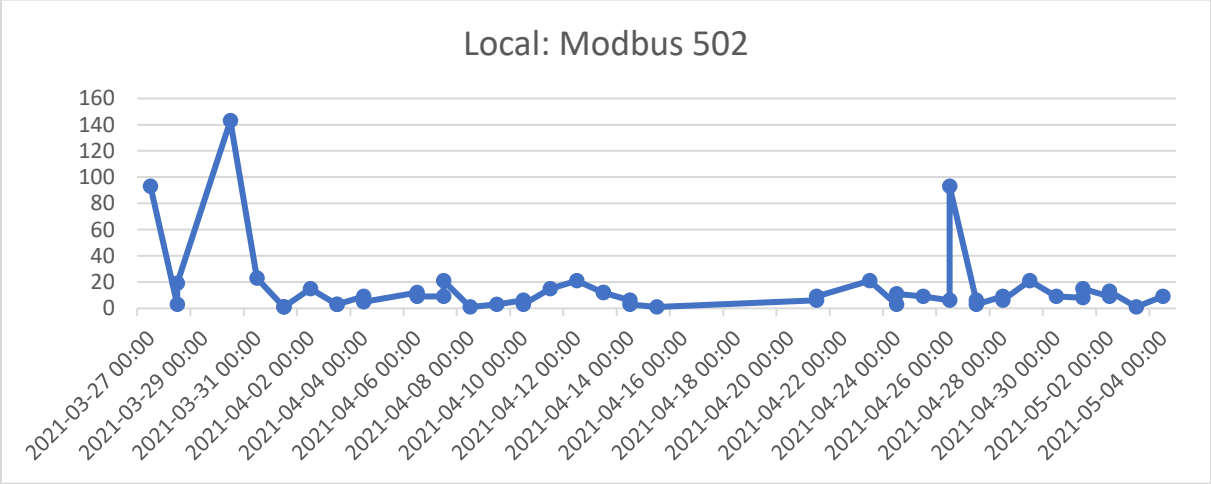

Figure 17: Modbus traffic over time on the local honeypot

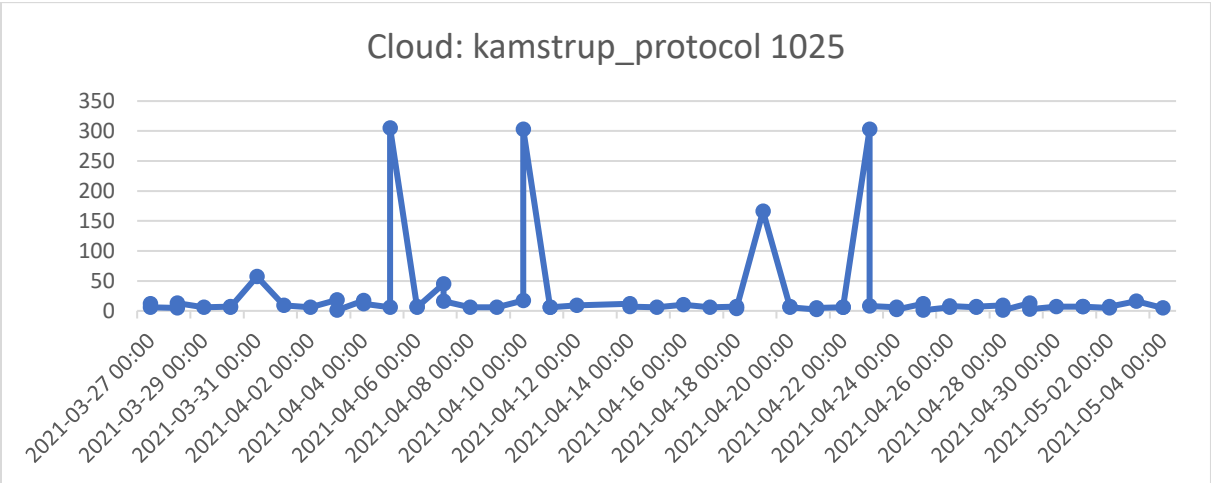

Figure 18: kamstrup_protocol traffic over time on the cloud honeypot

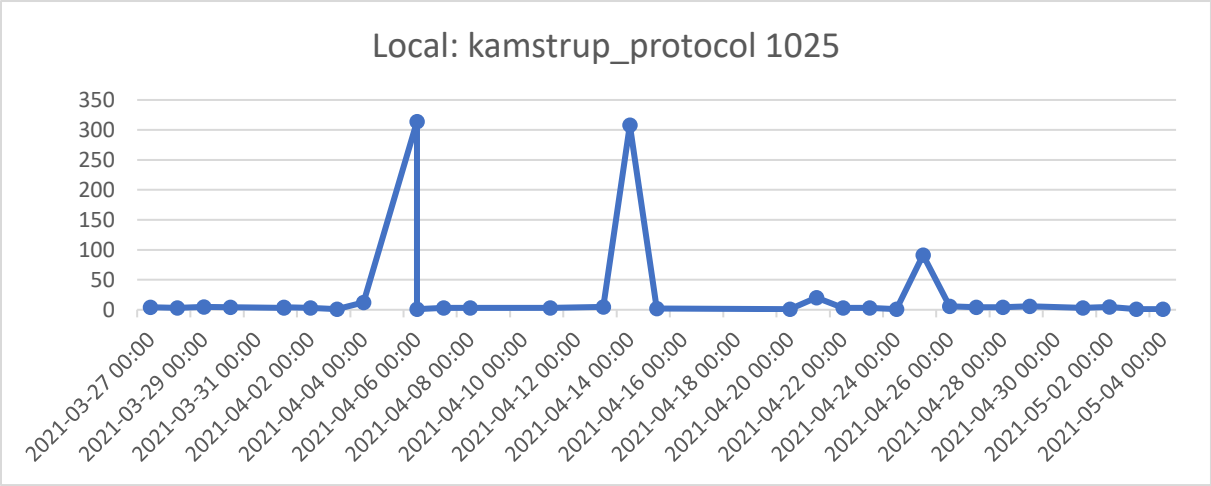

Figure 19: kamstrup_protocol traffic over time on the local honeypot

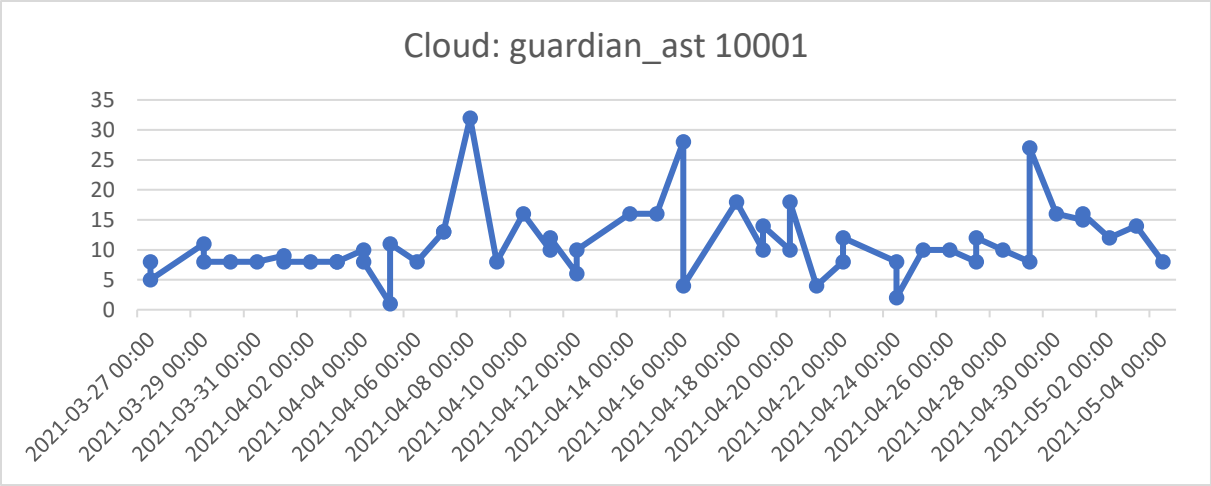

Figure 20: guardian_ast traffic over time on the cloud honeypot

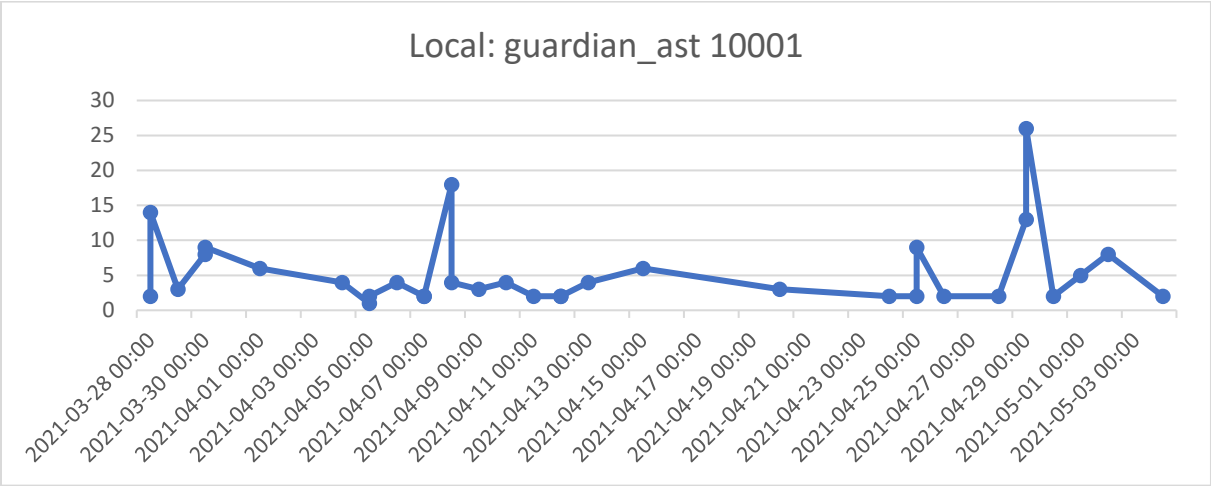

Figure 21: guardian_ast traffic over time on the local honeypot

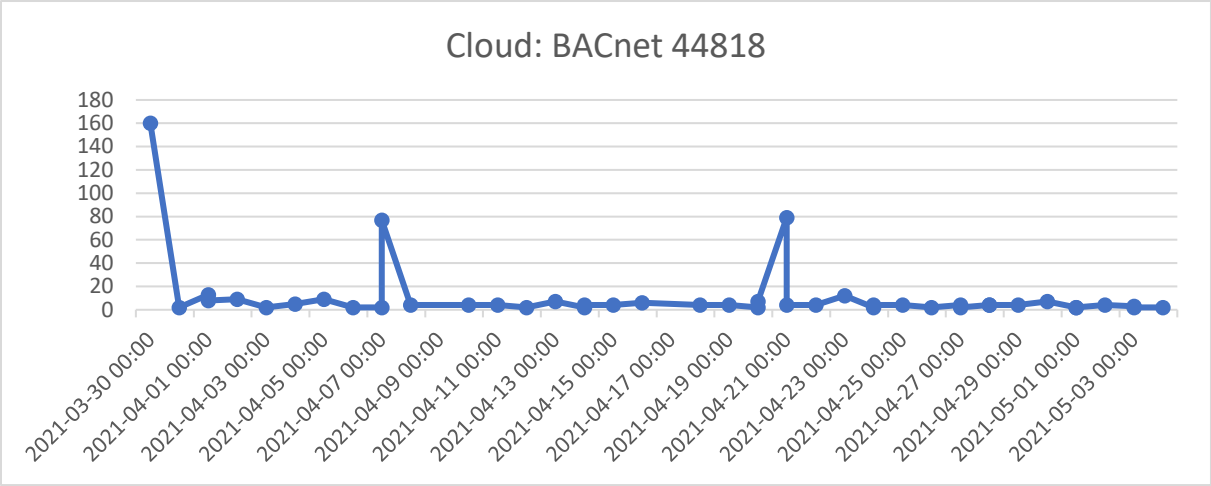

Figure 22: BACnet traffic over time on the cloud honeypot

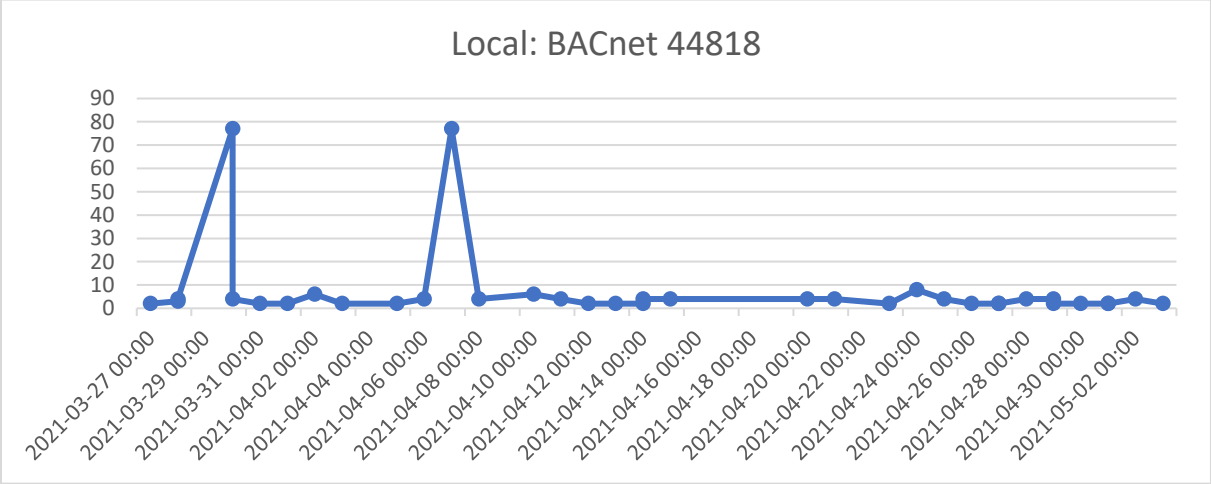

Figure 23: BACnet traffic over time on the local honeypot

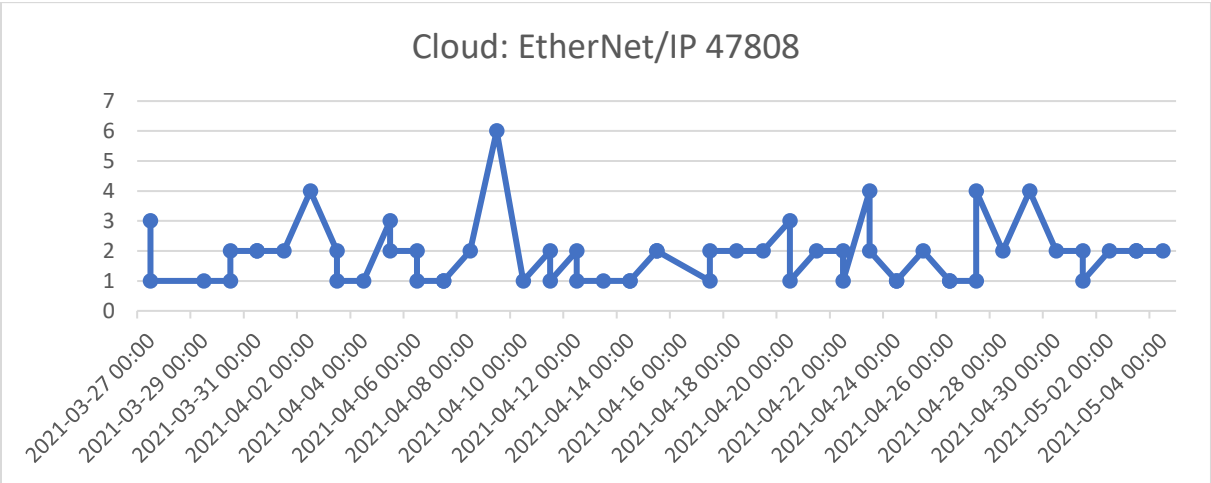

Figure 24: EtherNet/IP traffic over time on the cloud honeypot

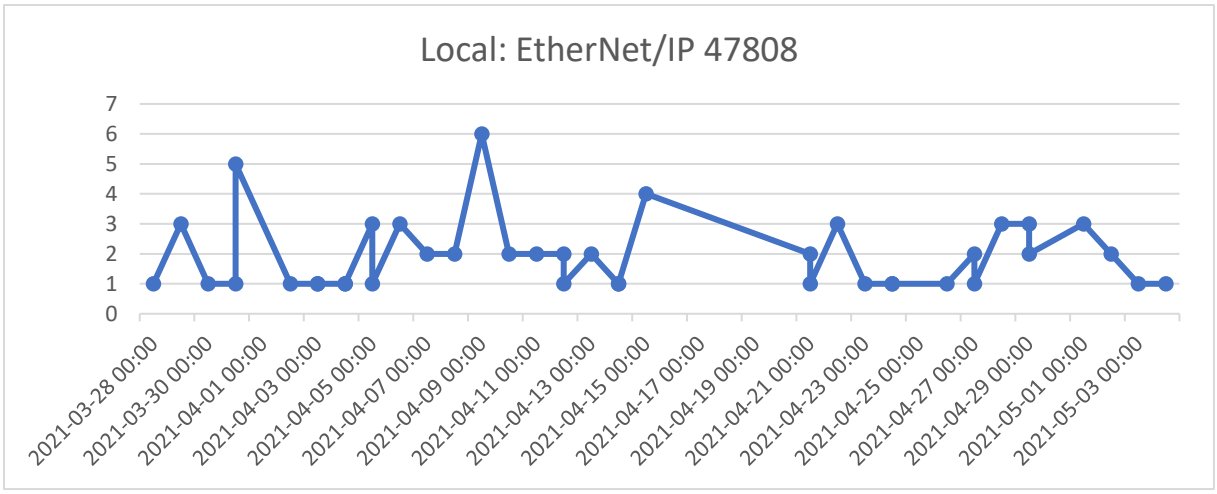

Figure 25: EtherNet/IP traffic over time on the local honeypot

Figure 26 and Figure 27 illustrates traffic to kamstrup_management_protocol. Due to the early closing of this port and the reopening and closing again there is not much activity here.

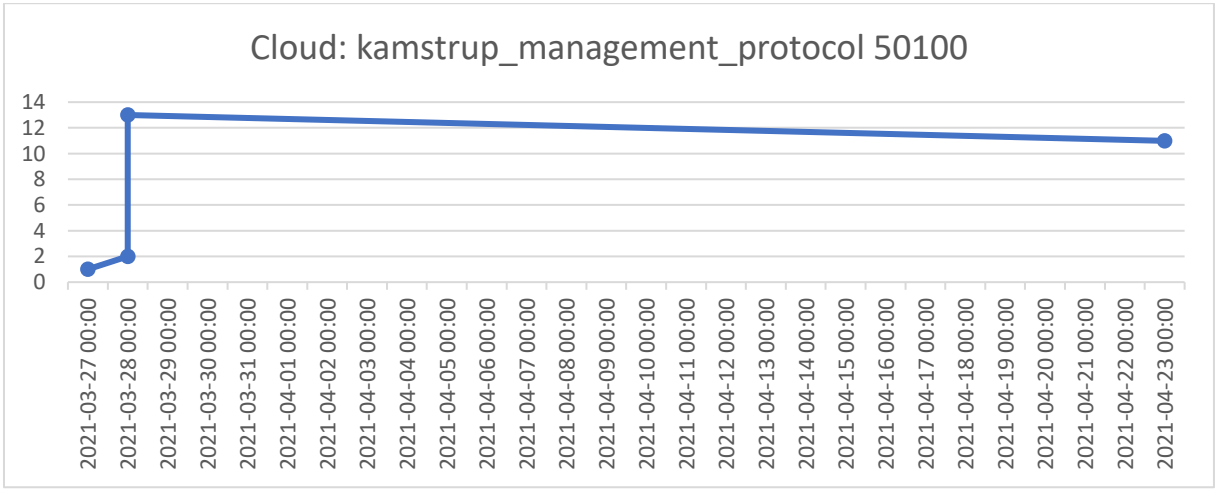

Figure 26: kamstrup_management_protocol traffic over time on the cloud honeypot

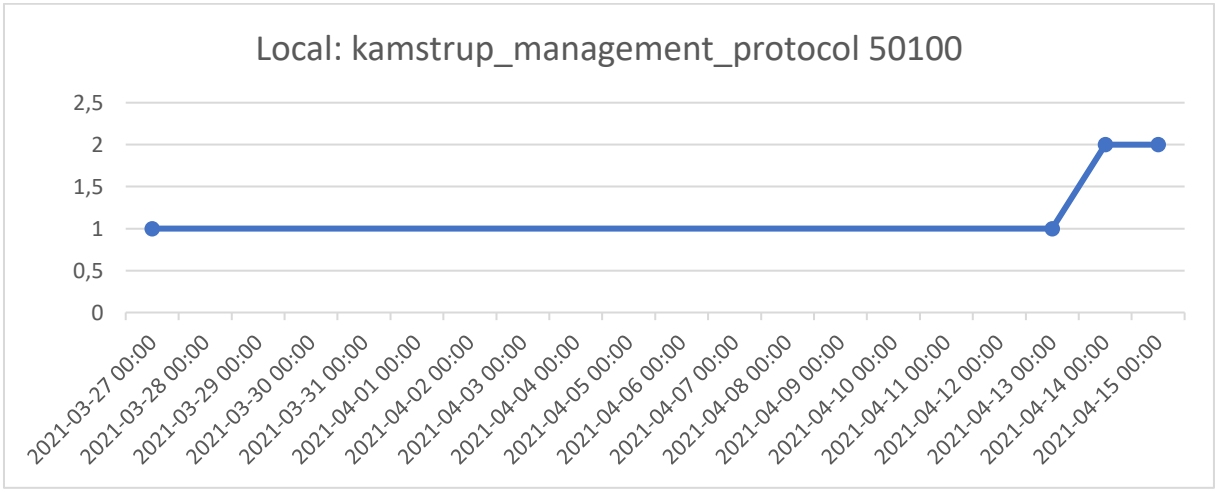

Figure 27: kamstrup_management_protocol traffic over time on the local honeypot

8. Discussion

Our thoughts regarding the results from selecting and setting up the honeypots and the collection and analysing of the data will be discussed in the following chapter.

8.1 RQ1

To study attacks on an ICS/SCADA system an evaluation of ICS/SCADA honeypots has been done. Two honeypots were looked at, T-pot and Conpot. Both have the capability to emulate an industrial control system. T-pot is a multi-honeypot service which, among other things, uses Conpot for the industrial mode. Advantage with T-pot is that it also contains data monitoring tools which makes it easy to get a good overview of the system and the attacks. However, it also runs other services and open ports for these services that are not related to an ICS/SCADA system. This could make an attacker suspicious of the system when the system deviates from a real ICS/SCADA system, and possibly lead to the attacker backing off. The lack of research articles regarding T-pot makes it hard to evaluate based on the information from others. Arthur Jicha *et al.* [8] analysed Conpot and compared discovered open ports using Shodan and Nmap. Their results and conclusion regarding ports can be useful when setting up a honeypot and using T-pot as a honeypot to analyse attacks on an ICS/SCADA system, since T-pot is using Conpot. However, we still must test it and make our own conclusion about its efficiency on collection of data concerning attacks on ICS/SCADA systems. Conpot does not come with the monitoring tools that T-pot has, which makes Conpot data not as easy to extract and monitor. But instead, it runs less services than T-pot and therefore does not open ports that are not relevant to an ICS/SCADA system. This could lead to a system that looks more like an ICS/SCADA system and attackers might not get as suspicious and might stay in the system longer once they breached it. Regarding resources, T-pot is more resource heavy than Conpot and due to our limits in resources this will be crucial for how long and how many honeypots we can run.

To answer the question which honeypot is suitable to study attacks on an ICS/SCADA system, both Conpot and T-pot have the potential to work. But additional aspects must be considered when implementing the honeypot to get closer to an answer. One aspect is the risk of the honeypot getting flagged as a honeypot on Shodan or the attacker recognise the hardcoded responses and parameters on the honeypot by collecting information of the system using Nmap. As mentioned earlier, this can have an effect on attackers and how and if they interacted with the system. If the honeypot system clearly deviates from a real ICS/SCADA system, then there is a risk that an attacker who specifically wants to target an ICS/SCADA system avoids the system, and no data can be collected from them. To avoid getting fingerprinted by an attacker and scan sites such as Shodan, modifications to the honeypot have to be done. R. N. Dahbul *et al.* [45] tries to address these problems and implementing enhancement on honeypots. This can be done by changing static responses such as banners from different services and closing ports that are not used in an ICS/SCADA system. The enhanced honeypots were tested by security experts to validate the result of the implementations. The result of the test showed that the modification of the honeypots had some effect. The enhancements did not make the honeypot undetectable, but it made them harder for the expert to identify them as honeypots.

Our own tests done by closing downs ports not relevant to ICS/SCADA system made it take longer time for Shodan to flag our honeypots as honeypots. By closing ports and moving the honeypot on the local machine behind a firewall we could avoid it getting flagged on Shodan. However, the static responses from the different services that show up on Shodan or collected with Nmap can still easily give the system away as a honeypot. The honeypot in the cloud still got flagged as a honeypot after closing down ports, the biggest difference implementation-wise that could be seen were after the first test setup when we closed all ports that do not have any connection to the honeypot.

By continuously closing down ports which we suspected led to our honeypots getting flagged as honeypots on Shodan, we could see a potential problem with the honeypot which we think can be related to the generic responses from the honeypot. If the honeypots easily get flagged and fingerprinted by the different scan services, there is a high risk that an attacker knows the machine is a honeypot.

Even if the low-interaction honeypot system deviates from a real system there is still potential in the collected data to contain valuable information. Before an attacker knows that the system is a honeypot and not a real system, they still must do recognition attacks (or use online scan services) and fingerprint the system. This collected data can tell us there is an interest in the system and that an attacker potentially wants to do something malicious. This can tell an administrator of a network or a researcher that someone is having an interest in the system and may be trying to breach it. Mokube and Adams [39] states that low-interaction honeypots can be used to analyse spammers and as an active countermeasure against worms. The collected data revealed activities from different scan services that collected information about the honeypots. An administrator could use this information when creating rules for a firewall, an IDS or IPS and block the scan services IP addresses. By doing this, less information regarding a system is less visible on the open internet.

A few actions can be taken to improve the usefulness of T-pot and Conpot and consequently avoid detection and fingerprinting, as we see it, to make the honeypot harder to detect. These actions are:

- Close ports not relevant to an ICS/SCADA system.
- Hide behind a firewall and have strict rules how to handle open ports and drop traffic.
- Reconfigure the hardcoded responses in the honeypot configuration files.
- Change hardcoded static data to dynamical changeable data.

We think these improvements can have a positive effect on concealing the machines from being detected as honeypots.

8.2 RQ2 and RQ3

Most countries consider ICS/SCADA systems such as electrical energy networks and systems as critical infrastructures, as pointed out by Erdal Irmak and Ismail Erkek [46]. Because of advances in IT, SCADA systems are also integrated more and more into the networks connected to conventional network systems. Precautions must therefore be taken when setting the systems up before going online. However, cybersecurity has not yet progressed for ICS and SCADA as far as for other network fields. Moreover, securing such systems is more complex than it is for, e.g., a typical conventional company network. In a worst-case scenario, loss of human life could occur if not being careful with the industrial machines existing within ICS/SCADA systems. It is therefore of utmost importance to understand the cyberattack vectors in these systems and how to harden them [37]. Ramachandruni and Poornachandran [37] mention that threats to ICSs are many. Some of them are hostile governments, single intruders, terrorist groups, botnets, employees with malicious intents, malwares, human mistakes, and equipment failure. We consider ICS/SCADA honeypots as an essential tool to help in the cybersecurity field. Based on analysing the results, an understanding of who, what and where regarding cyberattacks can be formed. By extension, an understanding can be gained in how to set the systems up so no outsider might gain non-authorized access.

There are three main aspects in the cybersecurity of ICS/SCADA systems [46]; hardware-sided, software-sided and communication-sided. Hardware-sided attacks are attacks done by being physically present at the system and injecting code or changing settings. Our honeypot setups do not generate results that can be analysed to improve the physical security on ICS/SCADA systems, but rather give an understanding in software-sided and communication-sided attacks. Software-sided attacks are categorized for example as source code change, buffer overflow attacks, SQL injection and cross-site scripting. Analysing our results we can see a lot of these, particularly buffer overflow attacks. The communication-sided attacks are by far the most frequently occurring attacks on our honeypots. The communication protocols in SCADA systems are many, and the main focus on the honeypot setups in this thesis work. Modbus is the most used communication protocol in ICS/SCADA systems [46], which by our results is also one of the most attacked ICS/SCADA protocols in the honeypots. S7 communication protocol (s7comm) was also one of the most targeted protocols. Eigner *et al.* [47] write that S7 Programmable Logic Controllers (PLCs), which use the S7 communication protocol, are estimated to constitute over 30% of the worldwide PLC market. This is probably why the s7comm protocol was one of the most targeted protocols in our own setups. The s7comm protocol is a proprietary

one [47], meaning very little information can be found about attacks against it. Our viewpoint is that industry leaders need to be more open about their proprietary protocols to increase the security of them.

The ICS/SCADA communication protocols are often tunnelled over conventional network systems like IPv4, which is where encrypted tunnelling protocols such as SSH come into play. SSH has the possibility to send information secure over an insecure IPv4 connection, which is possibly why SSH is so attractive for the hackers to breach. Our results show that the most targeted protocol overall was SSH. This can be explained partly through the fact that ICS/SCADA communication protocols can be sent over SSH. Another contributing factor would also be that SSH in itself is a very profitable protocol to target and try and get access through, even in conventional network systems outside of ICS/SCADA systems [2] [46].

As can be seen in the results, a IPv6 out-of-bounds attack was seen on the local machine but not on the cloud machine. The probable answer as to why is that the cloud service provider does not provide a local IPv6 address on a Linux machine whereas an IPv6 link-local address by default is created on a local machine when installing Linux.

The large number of attacks on upper layer protocols such as http and ftp can be explained by several factors. As a hacker, when trying to gain access to a system, a strategy that is often used is to first do reconnaissance on a system, then compromise the system and later use pivoting once inside the system. Pivoting is the practice of gaining access to a system or account inside a network, and then using that access to further the privilege escalation until access is gained to core systems inside an intranet. Giovanni Apruzzese *et al.* [48] mention more in-depth in their study several key factors of defending an ICS/SCADA system, including when trying to defend against pivoting (or lateral movement as it is also called). During the reconnaissance part, port-scans are often done to gather information about active systems and open hosts. When compromising the system, a known or zero-day vulnerability is used to get access. During the pivoting phase, the compromised systems are used to get further privilege on other systems inside the network. We can see the same behaviour in our own honeypot setups where attackers first often try to target SSH to get administrator access, and then try to execute malicious code that utilizes vulnerabilities in ICS/SCADA systems. When inspecting the signature of the payload, one can determine if it is malicious code. This is however only possible if it is a vulnerability known from before. If the attackers are using zero-day exploits there is no precedence for how the malicious code might look like and attackers would have a greater chance getting privileged access to our system. Since we are not in a real, major live ICS/SCADA system, the risk of this is probably slim to none since attackers would not want to want to risk the discovery of a zero-day exploit in the off-chance that our systems might be real. Attackers would probably only use zero-day exploits after doing thorough reconnaissance against a specific ICS/SCADA target.

Giovanni Apruzzese *et al.* [48] also mention the difficulty in detecting hackers practicing pivoting. The attacks might consist of signature-based, anomaly-based and/or protocol-specific. Our results show a variety of these different sorts of attacks and we can draw the conclusion that pivoting might be a main goal for a serious hacker when first probing the system. Ramachandruni and Poornachandran [37] mention that a great way to protect an ICS/SCADA network is to do a risk assessment and employ a defence-in-depth strategy (DiD). DiD is a layered strategy where several defence mechanisms are used to deflect an attack, where one defence mechanism takes over if a first one fails. Based on our own results, the idea of a layered strategy as a security solution is adequate.

During the time of the experiment, we saw very similar attack patterns for the honeypots installed on a local machine and in the cloud. However, this could partially be caused by the limited time that the honeypots were active; we believe that if we had the time to run our tests for a longer period, there might have been a bigger difference. With that stated, what conclusion can be made from the differences we did notice? Is it possible to answer why the cloud machine sometimes had twice the attacks on some protocols against it, when comparing to the local machine? As mentioned in the results, this could be attributed to the IP range AWS uses. It is a more well-known range and more frequently scanned by scan-services and similar entities. When looking at how many attacks per day and the peak of those attacks, most protocols had roughly the same peak when comparing the machines. For example, the kamstrup protocol had about twice the attacks in total on the cloud machine. However, on days when the protocol was most targeted, both machines had a peak of about 300-320 attacks. This could be

because when an attacker does think it is an actual ICS/SCADA environment, the same set of attacks are run to try to get access to the system. This could be an automated set of attacks, hence the similarities in the number of attacks on peak days. All in all, the results both from the cloud and the local honeypot are usable to understand the attackers, what malicious code the attackers use, and the several attack vectors targeted. If both the honeypots were to be online for a longer period of time, an educated guess is that the differences would even out, and even more similar results would be noticeable.

9. Conclusions

In this thesis our focus was on ICS and usage of an ICS/SCADA-honeypot to identify possible attacks and identify malicious activities targeting ICS/SCADA systems. The behaviour of the attacker and his or her interaction with the system was analysed using the data gathered by the honeypot. We also analysed if a cloud service like AWS would have an impact on the interaction level of the attacker.

The thesis work is great for getting a general idea in what protocols are targeted and exploited. It gives a pointer in how much thought that should put into a network's security implementation. T-pot is suitable for studying attacks against ICS/SCADA systems, it combines great honeypots such as Conpot and with online IDSs and presents the information collected in an understandable way using different graphical tools.

When analysing the thesis results, a few different possible directions could be taken if one were to do similar tests and get even more accurate results. One can configure the static responses to make the honeypots harder getting flagged on scan services such as Shodan. One can also use Shodan API to know exactly when a new service on the machine's IP address is discovered. Using the API would eliminate the need to manually check Shodan to see when and if something new is discovered.

If the time frame would have been longer to run the experiment another setup strategy could have been applied to possibly get a better result of how the honeypot got flagged. We could have started with only one port open for a specific protocol, if or when Shodan flagged it we could have taken an appropriate action, either by closing the port or modified the static response from the honeypot. By doing this for all the protocols we would have a better list of what triggered the flagging and, in the end, possibly a honeypot that would avoid detection better. The time frame between a machine going online and Shodan flagging it ranged from anywhere from a few days up to weeks. We did not have enough time to follow such a strict methodology where only one port at a time was opened. Shodan also does not mention explicitly their algorithm for flagging machines, so this would only be one factor out of possibly many to consider.

To protect an ICS/SCADA system network, one must be well-read on what sort of attack vectors that exist. After learning about the attack vectors, one can maximize the security in the network by using a defence-in-depth strategy when choosing security implementations. If a hacker does get through one security layer, pivoting is then avoided by another defence layer taking over.

To acquire serious data, one would need to place the honeypots in a real ICS/SCADA network. There is also a chance that the differences seen can be attributed to chance alone. Not enough data was collected to know how much of a difference there is between a cloud and local honeypot mimicking ICS/SCADA systems. Based on the results we did collect, a few differences in the number of attacks and a few discrepancies in what sort of malicious code injected into the cloud machine versus the local machine was observed. We believe the results lean towards a somewhat negligible difference when analysing what sort of attacks that are launched to the machines. One can use a honeypot in the cloud or choose a honeypot on a local machine, the results are still valid and can be used to get a better understanding of the ICS/SCADA system security.

References

[1] M. F. Razali, M. N. Razali, F. Z. Mansor, G. Muruti och N. Jamil, ”IoT Honeypot: A Review from Researcher’s Perspective,” i *2018 IEEE Conference on Applications, Information and Network Security (AINS)*, Langkawi, Kedah, Malaysia, 2018.

[2] A. Belqruch och A. Maach, ”SCADA security using SSH honeypot,” i *NISS19: Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, Rabat, Morocco, 2019.

[3] Z. Drias, A. Serhrouchni och O. Vogel, ”Taxonomy of attacks on Industrial Control protocols,” i *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, Paris, France, 2015.

[4] S. Al-Rabiaah, ”The "Stuxnet" Virus of 2010 As an Example of A “APT” and Its “Recent” Variances,” i *2018 21st Saudi Computer Society National Computer Conference (NCC)*, Riyadh, Saudi Arabia, 2018.

[5] M. Dodson, A. R. Beresford och M. Vingaard, ”Using Global Honeypot Networks to Detect Targeted ICS Attacks,” i *12th International Conference on Cyber Conflict*, Tallinn, Estonia, 2020.

[6] K. Stouffer, V. Pillitteri, S. Ligtman, M. Abrams och A. Hahn, ”Guide to Industrial Control (ICS) Security,” May 2015. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf. [Accessed 7 March 2021].

[7] European Union Agency for Cybersecurity, ”Communication network dependencies for ICS/SCADA Systems,” European Union Agency For Network And Information Security, EU, 2016.

[8] A. Jicha, M. Patton och H. Chen, ”SCADA Honeypots An In-depth Analysis of Conpot,” i *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, Tucson, AZ, USA, 2016.

[9] L. Salazar, N. Ortiz, X. Qin och A. A. Cardenas, ”Towards a High-Fidelity Network Emulation of IEC 104 SCADA Systems,” i *CPSIOTSEC'20: Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy*, Virtual Event USA, 2020.

[10] S. U. Cho och S. H. Hong, ”Fault Tolerant BBMD in the BACnet/IP Protocol,” 2006 IEEE International Conference on Industrial Technology, Mumbai, India, 2006.

[11] F. Tacliad, T. D. N. Nguyen och M. Gondree, ”DoS Exploitation of Allen-Bradley’s Legacy Protocol through Fuzz Testing,” ICSS 2017: Proceedings of the 3rd Annual Industrial Control System Security Workshop, San Juan PR USA, 2017.

[12] W.-s. Jung, S.-M. Kim, Y.-H. Goo och M.-S. Kim, ”Whitelist representation for FTP service in SCADA system by using structured ACL model,” i *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, Japan, 2016.

[13] A.-S. Brylinski och A. Bhattacharjya, ”Overview of HTTP/2,” i *ICC '17: Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, Cambridge, United Kingdom, 2017.

[14] W. Fischer, ”IPMI Basics,” 22 December 2014. [Online]. Available: https://www.thomas-krenn.com/en/wiki/IPMI_Basics. [Accessed 29 April 2021].

[15] B. Phillips, E. Gamess och S. Krishnaprasad, ”An Evaluation of Machine Learning-based Anomaly Detection in a SCADA System Using the Modbus Protocol,” i *ACM SE '20: Proceedings of the 2020 ACM Southeast Conference*, Tampa, FL, USA, 2020.

[16] E. D. L. Morales, ”HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems,” Arizona, 2020.

[17] ”S7comm,” 11 August 2020. [Online]. Available: https://gitlab.com/wireshark/wireshark/-/wikis/S7comm. [Accessed 29 April 2021].

[18] Wikipedia, ”Simple Network Management Protocol,” 18 April 2021. [Online]. Available: https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol. [Accessed 30 April 2021].

[19] Wikipedia, ”Trivial File Transfer Protocol,” 19 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol. [Accessed 30 April 2021].

[20] S. S. Gadde, R. K. S. Ganta, G. Gupta, R. Raghava och R. Mohan, ”Securing Internet of Things(IoT) Using HoneyPots,” *International Journal of Engineering & Technology*, vol. 7, nr 2.7, pp. 820-824, 2018.

[21] C. Kreibich och J. Crowcroft, ”Honeycomb – Creating Intrusion Detection,” *ACM SIGCOMM Computer Communication Review*, vol. 34, nr 1, pp. 51-56, 2004.

[22] A. Mairh, D. Barik, K. Verma och D. Jena, ”Honeypot in Network Security: A Survey,” i *ICCCS '11: Proceedings of the 2011 International Conference on Communication, Computing & Security*, Rourkela Odisha, India, 2011.

[23] GitHub, ”Conpot,” November 2020. [Online]. Available: https://github.com/mushorg/conpot/tree/master/conpot/protocols. [Accessed 7 March 2021].

[24] S. M. Z. U. Rashid, M. J. Uddin och A. Islam, ”Know Your Enemy: Analysing Cyber-threats Against Industrial Control Systems Using Honeypot,” i *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, Dhaka, Bangladesh, 2019.

[25] M. Ochse, S. Harderecker, A. Vorbach och B. Kober, ”GitHub/tpotce,” 4 September 2020. [Online]. Available: https://github.com/telekom-security/tpotce/blob/master/README.md. [Accessed 7 February 2021].

[26] M. Oosterhof, ”GitHub/Cowrie,” 2019. [Online]. Available: https://github.com/cowrie/cowrie. [Accessed 16 February 2021].

[27] M. Ferranti, ”Network World,” 17 August 2018. [Online]. Available: https://www.networkworld.com/article/3296740/what-is-nmap-why-you-need-this-network-mapper.html. [Accessed 7 March 2021].

[28] E. L. Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao och G.-J. Ahn, ”HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems,” i *CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event USA, 2020.

[29] Open Information Security Foundation, ”Suricata IDS,” 2021. [Online]. Available: https://suricata-ids.org/about/. [Accessed 6 April 2021].

[30] RedHat, Inc, ”RedHat, Inc,” 2020. [Online]. Available: https://www.redhat.com/en/topics/security/what-is-cve. [Accessed 11 April 2021].

[31] Proofpoint Inc., ”Proofpoint,” 2021. [Online]. Available: https://www.proofpoint.com/us/products/advanced-threat-protection/et-intelligence#. [Accessed 30 April 2021].

[32] C. Moore, ”Detecting Ransomware with Honeypot Techniques,” i *2016 Cybersecurity and Cyberforensics Conference (CCC)*, Amman, 2016.

[33] W. Zhang, B. Zhang, Y. Zhou, H. He och Z. Ding, ”An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks,” *IEEE Internet of Things Journal*, vol. 7, nr 5, pp. 3991-3999, May 2020.

[34] A. Tambe, Y. L. Aung, R. Sridharan, M. Ochoa, N. O. Tippenhauer, A. Shabtai och Y. Elovici, ”Detection of Threats to IoT Devices using Scalable VPN-forwarded Honeypots,” i *CODASPY '19: Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, New York, 2019.

[35] E. Vasilomanolakis, S. Karuppayah, P. Kikiras och M. Mühlhäuser, ”A honeypot-driven cyber incident monitor: lessons learned and steps ahead,” i *SIN '15: Proceedings of the 8th International Conference on Security of Information and Networks*, Sochi, 2015.

[36] D. Antonioli, A. Agrawal och N. O. TIppenhauer, ”Towards High-Interaction Virtual ICS Honeypots-in-a-Box,” i *CPS-SPC '16: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, Vienna, Austria, 2016.

[37] R. Ramachandruni och P. Poornachandran, ”Detecting the network attack vectors on SCADA systems,” i *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Kochi, India, 2015.

[38] J. Zobel, *Writing for Computer Science*, 3rd red., London: Springer Publishing Company, Incorporated, 2015.

[39] I. Mokube och M. Adams, ”Honeypots: Concepts, Approaches, and Challenges,” i *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*, Winston-Salem, North Carolina, USA, 2007.

[40] K. Säfsten och M. Gustavsson, *Forskningsmetodik – För ingenjörer och andra problemlösare*, Lund: Studentlitteratur AB, 2019.

[41] National Institute of Standards and Technology, ”National Institute of Standards and Technology,” 21 July 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-11899. [Accessed 20 May 2021].

[42] A. Kleinmann och A. Wool, ”Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensics,” *Journal of Digital Forensics, Security and Law*, vol. 9, nr 2, pp. 37-50, 2014.

[43] G. Miru, ”The Siemens S7 Communication - Part 1 General Structure,” 30 January 2016. [Online]. Available: http://gmiru.com/article/s7comm/. [Accessed 9 May 2021].

[44] Modbus Organization, ”MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3,” 26 April 2012. [Online]. Available:

https://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. [Accessed 9 May 2021].

[45] R. N. Dahdul, C. Lim och J. Purnama, ”Enhancing Honeypot Deception Capability Through Network Service Fingerprinting,” i *IOP Conf. Series: Journal of Physics: Conf. Series 801 (2017) 012057*, Indonesia, 2016.

[46] E. Irmak och İ. Erkek, ”An overview of cyber-attack vectors on SCADA systems,” i *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, Antalya, Turkey, 2018.

[47] O. Eigner, P. Kreimel och P. Tavolato, ”Identifying S7comm Protocol Data Injection Attacks in Cyber-Physical Systems,” i *5th International Symposium for ICS & SCADA Cyber Security Research 2018 (ICS-CSR 2018)*, Hamburg, Germany, 2018.

[48] G. Apruzzese, F. Pierazzi, M. Colajanni och M. Marchetti, ”Detection and Threat Prioritization of PivotingAttacks in Large Networks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 8, nr 2, pp. 404-415, 2020.

Appendix A – traffic examples to ICS/SCADA protocols

This appendix contains example of traffic target the open port belonging to the ICS/SCADA protocols. The text is a copy of raw log files created by Conpot.

Modbus – port 502

conpot_default.log.8.gz:2021-04-26 16:34:53,689 Modbus traffic from 71.6.167.142: {'request': b'0000000000020011', 'slave_id': 0, 'function_code': None, 'response': b''} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:54,313 Modbus traffic from 71.6.167.142: {'request': b'0000000000020111', 'slave_id': 1, 'function_code': 17, 'response': b'11110101ff'} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:54,655 Modbus traffic from 71.6.167.142: {'request': b'000000000005012b0e0100', 'slave_id': 1, 'function_code': 43, 'response': b'2b0e0101000003000075369656d656e73010753494d41544943020653372d323030'} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:55,003 Modbus traffic from 71.6.167.142: {'request': b'0000000000020211', 'slave_id': 2, 'function_code': 17, 'response': b'11110101ff'} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:55,349 Modbus traffic from 71.6.167.142: {'request': b'000000000005022b0e0100', 'slave_id': 2, 'function_code': 43, 'response': b'2b0e0101000003000075369656d656e73010753494d41544943020653372d323030'} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:55,696 Modbus traffic from 71.6.167.142: {'request': b'0000000000020311', 'slave_id': 3, 'function_code': None, 'response': b''} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

conpot_default.log.8.gz:2021-04-26 16:34:56,064 Modbus traffic from 71.6.167.142: {'request': b'000000000005032b0e0100', 'slave_id': 3, 'function_code': None, 'response': b''} (08aa4b0a-aff0-4a8c-8ac9-97a80cd527ce)

S7comm – port 102

conpot_default.log.3.gz:2021-05-02 08:45:45,524 New s7comm session from 176.58.125.104 (a5df9b85-d405-4c45-b513-364edc7489c4)

conpot_default.log.3.gz:2021-05-02 08:45:45,948 Received S7 packet: magic:50 pdu_type:1 reserved:0 req_id:0 param_len:8 data_len:0 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:45,949 Received S7 packet: magic:50 pdu_type:1 reserved:0 req_id:0 param_len:8 data_len:0 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:46,142 Received S7 packet: magic:50 pdu_type:7 reserved:0 req_id:0 param_len:8 data_len:8 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:46,144 Received S7 packet: magic:50 pdu_type:7 reserved:0 req_id:0 param_len:8 data_len:8 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:46,244 Received S7 packet: magic:50 pdu_type:7 reserved:0 req_id:0 param_len:8 data_len:8 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:46,245 Received S7 packet: magic:50 pdu_type:7 reserved:0 req_id:0 param_len:8 data_len:8 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

conpot_default.log.3.gz:2021-05-02 08:45:46,346 Received S7 packet: magic:50 pdu_type:7 reserved:0 req_id:0 param_len:8 data_len:8 result_inf:0 session_id:a5df9b85-d405-4c45-b513-364edc7489c4

kamstrup_protocol – port 1025

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,003 New Kamstrup connection from 192.241.216.62:43576. (2f1a9554-afbb-420d-b44a-7faefe8b1039)

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,008 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x45

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,009 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x48

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,009 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x4c

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,009 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x4f

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,009 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x20

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x7a

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x67

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x2d

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x30

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x34

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,010 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x31

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,011 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x36

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,011 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x61

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,011 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x2d

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,011 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x32

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,011 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x35

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,012 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0x31

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,012 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0xd

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:19,012 Kamstrup skipping byte, expected kamstrup_meter request magic but got: 0xa

conpot_kamstrup_382.log.8.gz:2021-04-28 00:41:28,838 Kamstrup client disconnected. (2f1a9554-afbb-420d-b44a-7faefe8b1039)

conpot_kamstrup_382.log.8.gz:2021-04-28 00:42:00,857 Session timed out: 2f1a9554-afbb-420d-b44a-7faefe8b1039

kamstrup_management_protocol – port 50100

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:48,256 New kamstrup_management_protocol session from 176.58.125.104 (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:49,835 Kamstrup management traffic from 176.58.125.104: {'request': '#ST\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:51,668 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x0bn\x00\x00\x00\x00\x01\x04beio', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:53,500 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x00\x07\x00\x08\x00\x03\x00\x04\x00\x05\x00\x06', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:55,346 Kamstrup management traffic from 176.58.125.104: {'request': '\r\n\r\n', 'response': ''} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:56,924 Kamstrup management traffic from 176.58.125.104: {'request': 'GET / HTTP/1.0\r\n\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:54:58,764 Kamstrup management traffic from 176.58.125.104: {'request': 'OPTIONS / HTTP/1.0\r\n\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:00,609 Kamstrup management traffic from 176.58.125.104: {'request': 'OPTIONS / RTSP/1.0\r\n\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:02,535 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x1e\x00\x06\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x07version\x04bind\x00\x00\x10\x00\x03', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:04,363 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x0c\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:06,192 Kamstrup management traffic from 176.58.125.104: {'request': 'HELP\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:08,439 Kamstrup management traffic from 176.58.125.104: {'request': 'l\x00\x0b\x00\x00\x00\x00\x00\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:10,266 Kamstrup management traffic from 176.58.125.104: {'request': 'GET /nice%20ports%2C/Tri%6Eity.txt%2ebak HTTP/1.0\r\n\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:12,117 Kamstrup management traffic from 176.58.125.104: {'request': '\x01default\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:14,100 Kamstrup management traffic from 176.58.125.104: {'request': 'OPTIONS sip:nm SIP/2.0\r\nVia: SIP/2.0/TCP nm;branch=foo\r\nFrom: <sip:nm@nm>;tag=root\r\nTo: <sip:nm2@nm2>\r\nCall-ID: 50000\r\nCSeq: 42 OPTIONS\r\nMax-Forwards: 70\r\nContent-Length: 0\r\nContact: <sip:nm@nm>\r\nAccept: application/sdp\r\n\r\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:15,943 Kamstrup management traffic from 176.58.125.104: {'request': 'TNMP\x04\x00\x00\x00TNME\x00\x00\x04\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:17,921 Kamstrup management traffic from 176.58.125.104: {'request': ':\x00\x00\x00/\x00\x00\x00\x02\x00\x00@\x02\x0f\x00\x01\x00=\x05\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00/\x00\x00\x00\x00\x00\x00\x00\x00\x00@\x1f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:19,750 Kamstrup management traffic from 176.58.125.104: {'request': 'JRMI\x00\x02K', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:21,818 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x03\x00\x01\x00\x00\x00\x00\x00\x00\x00\x02\x00\x00\x00\x00\x0f\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:23,650 Kamstrup management traffic from 176.58.125.104: {'request': 'GIOP\x01\x00\x01\x00$\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x06\x00\x00\x00abcdef\x00\x00\x04\x00\x00\x00get\x00\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:25,553 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x04\x00\x00\x00\x08\x00\x00\x00\x01\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:27,621 Kamstrup management traffic from 176.58.125.104: {'request': '\x00\x00\x00\x00\x00', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

conpot_kamstrup_382.log.44.gz:2021-03-23 00:55:29,448 Kamstrup management traffic from 176.58.125.104: {'request': '\x01I20100\n', 'response': "\r\n? Command not found.\r\nSend 'H' for help.\r\n"} (30b6f9bc-f011-4e6a-9751-735a080bacd0)

guardian_ast – port 10001

conpot_guardian_ast.log.9.gz:2021-04-27 11:26:53,695 New guardian_ast session from 67.205.174.13 (63dc4598-1818-4737-b131-e9541c1e27ee)

conpot_guardian_ast.log.9.gz:2021-04-27 11:26:53,696 New GuardianAST connection from 67.205.174.13:61953. (63dc4598-1818-4737-b131-e9541c1e27ee)

conpot_guardian_ast.log.9.gz:2021-04-27 11:26:55,686 Non ^A command attempt 67.205.174.13:61953. (63dc4598-1818-4737-b131-e9541c1e27ee)

conpot_guardian_ast.log.9.gz:2021-04-27 11:26:55,687 GuardianAST client disconnected 67.205.174.13:61953. (63dc4598-1818-4737-b131-e9541c1e27ee)

conpot_guardian_ast.log.9.gz:2021-04-27 11:27:25,699 Session timed out: 63dc4598-1818-4737-b131-e9541c1e27ee

conpot_guardian_ast.log.9.gz:2021-04-27 11:51:12,739 New guardian_ast session from 183.136.225.16 (c07eda89-005d-4ba1-974f-5b7d0786099b)

conpot_guardian_ast.log.9.gz:2021-04-27 11:51:12,741 New GuardianAST connection from 183.136.225.16:56063. (c07eda89-005d-4ba1-974f-5b7d0786099b)

conpot_guardian_ast.log.9.gz:2021-04-27 11:51:12,742 I20100 command attempt 183.136.225.16:56063. (c07eda89-005d-4ba1-974f-5b7d0786099b)

conpot_guardian_ast.log.9.gz:2021-04-27 11:51:32,547 GuardianAST client disconnected 183.136.225.16:56063. (c07eda89-005d-4ba1-974f-5b7d0786099b)

conpot_guardian_ast.log.9.gz:2021-04-27 11:52:02,564 Session timed out: c07eda89-005d-4ba1-974f-5b7d0786099b

BACnet – 47808

conpot_default.log.20.gz:2021-04-19 00:18:14,349 New bacnet session from 146.88.240.4 (9c580b73-2606-43c7-ac63-47598760ce5a)

conpot_default.log.20.gz:2021-04-19 00:18:14,351 DecodingError - PDU: <PDU None -> None : 0a.00.11.01.04.00.05.ff.0c.0c.02.3f.ff.ff.19.4b>

EtherNet/IP – port 44818

conpot_default.log.5.gz:2021-04-30 17:10:40,458 New enip session from 125.64.94.138 (0ae700e8-a805-429e-81c2-acf224c85a11)

conpot_default.log.5.gz:2021-04-30 17:10:40,465 (header.(command.((byte)) : => request.enip.command = 99 (format '<H' over array('B', [99, 0]))

conpot_default.log.5.gz:2021-04-30 17:10:40,466 (header.(command.((byte)) : => request.enip.length = 0 (format '<H' over array('B', [0, 0]))

conpot_default.log.5.gz:2021-04-30 17:10:40,467 (header.(command.((byte)) : => request.enip.session_handle = 0 (format '<I' over array('B', [0, 0, 0, 0]))

conpot_default.log.5.gz:2021-04-30 17:10:40,467 (header.(command.((byte)) : => request.enip.status = 0 (format '<I' over array('B', [0, 0, 0, 0]))

conpot_default.log.5.gz:2021-04-30 17:10:40,469 (header.(command.((byte)) : => request.enip.options = 0 (format '<I' over array('B', [0, 0, 0, 0]))

conpot_default.log.5.gz:2021-04-30 17:10:40,526 (list_identity.(CPF.(empty.((noop)) -- limit='request.enip.CIP.list_identity...length' == 0; ending at symbol 0 vs. None

conpot_default.log.5.gz:2021-04-30 17:10:40,527 EtherNet/IP CIP Request (Client ('125.64.94.138', 44590)): {

conpot_default.log.5.gz: 'enip.command': 99,

conpot_default.log.5.gz: 'enip.length': 0,

conpot_default.log.5.gz: 'enip.session_handle': 0,

conpot_default.log.5.gz: 'enip.status': 0,

conpot_default.log.5.gz: 'enip.sender_context.input': array('B', hexload(r'''

conpot_default.log.5.gz: 'enip.options': 0,

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF': {},

conpot_default.log.5.gz: 'enip.command': 99,

conpot_default.log.5.gz: 'enip.length': 0,

conpot_default.log.5.gz: 'enip.session_handle': 0,

conpot_default.log.5.gz: 'enip.status': 0,

conpot_default.log.5.gz: 'enip.sender_context.input': array('B', hexload(r'''

conpot_default.log.5.gz: 'enip.options': 0,

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF': {},

conpot_default.log.5.gz: 'enip.command': 99,

conpot_default.log.5.gz: 'enip.length': 0,
conpot_default.log.5.gz: 'enip.session_handle': 0,
conpot_default.log.5.gz: 'enip.status': 0,
conpot_default.log.5.gz: 'enip.sender_context.input': array('B', hexload(r'''
conpot_default.log.5.gz: 'enip.options': 0,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].type_id': 12,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.version': 1,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_addr': '0.0.0.0',
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_family': 2,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_port': 44818,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.vendor_id': 1,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.device_type': 14,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_code': 54,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_revision': 2836,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.status_word': 12640,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.serial_number': 7079450,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_name.length': 20,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_name.string': '1756-L61/B LOGIX5561',
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.state': 255,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].input': bytearray(hexload(r'''
conpot_default.log.5.gz: 'enip.input': bytearray(hexload(r'''
conpot_default.log.5.gz:2021-04-30 17:10:40,554 EtherNet/IP CIP Response (Client ('125.64.94.138', 44590)): {
conpot_default.log.5.gz: 'enip.command': 99,
conpot_default.log.5.gz: 'enip.length': 0,
conpot_default.log.5.gz: 'enip.session_handle': 0,
conpot_default.log.5.gz: 'enip.status': 0,
conpot_default.log.5.gz: 'enip.sender_context.input': array('B', hexload(r'''
conpot_default.log.5.gz: 'enip.options': 0,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].type_id': 12,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.version': 1,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_addr': '0.0.0.0',
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_family': 2,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.sin_port': 44818,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.vendor_id': 1,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.device_type': 14,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_code': 54,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_revision': 2836,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.status_word': 12640,
conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.serial_number': 7079450,

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_name.length': 20,

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.product_name.string': '1756-L61/B LOGIX5561',

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].identity_object.state': 255,

conpot_default.log.5.gz: 'enip.CIP.list_identity.CPF.item[0].input': bytearray(hexload(r'''

conpot_default.log.5.gz: 'enip.input': bytearray(hexload(r'''

conpot_default.log.5.gz:2021-04-30 17:10:40,819 EtherNet/IP CIP Request (Client ('125.64.94.138', 44590)): {

conpot_default.log.5.gz:2021-04-30 17:10:40,823 EtherNet/IP CIP Response (Client ('125.64.94.138', 44590)): {

IPMI – port 623

conpot_ipmi.log.9.gz:2021-04-26 02:54:07,960 New IPMI traffic from ('184.105.247.230', 57165)

conpot_ipmi.log.9.gz:2021-04-26 02:54:07,960 New IPMI session initialized for client (('184.105.247.230', 57165))

conpot_ipmi.log.9.gz:2021-04-26 02:54:07,960 Connection established with ('184.105.247.230', 57165)

conpot_ipmi.log.9.gz:2021-04-26 02:54:07,961 IPMI response sent to ('184.105.247.230', 57165)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,892 New IPMI traffic from ('108.197.74.123', 80)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,893 New IPMI session initialized for client (('108.197.74.123', 80))

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,895 Connection established with ('108.197.74.123', 80)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,895 IPMI response sent to ('108.197.74.123', 80)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,930 Incoming IPMI traffic from ('108.197.74.123', 80)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,938 Incoming IPMI traffic from ('108.197.74.123', 80)

conpot_ipmi.log.8.gz:2021-04-27 11:25:30,960 Incoming IPMI traffic from ('108.197.74.123', 80)