Implementation of a Security Information Event Management System in an Industrial Control System

Detecting attacks and correlating events

Magnus Korneliussen

Oppgave for graden
Master i Informatikk: informasjonssikkerhet
60 studiepoeng

Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

UNIVERSITETET I OSLO

Våren 2021

Implementation of a Security Information Event Management System in an Industrial Control System

Detecting attacks and correlating events

Magnus Korneliussen

Implementation of a Security Information Event Management System in an Industrial Control System

http://www.duo.uio.no/

Trykk: Reprosentralen, Universitetet i Oslo

Sammendrag

Utfordringene ved å fortsatt holde dagens systemer sikre under den konstante utviklingen de har bringer med seg utfordringer som ikke har blitt taklet før. Bruken av små enheter med lav prosesseringskraft har vært populært for å holde kostnadene nede i større systemer. Nyskapningen av disse som tillater at de kobler seg opp mot mer komplekse enheter åpner opp muligheter som aldri har blitt sett før både for produsentene og angripere. For å holde tritt med trusselbildet trenger vi bedre sikkerhetstiltak og verktøy som passer til oppgavene de skal utføre. Når enheter er essensielle for å forsyne en hel by med elektrisitet kan dukke under av et angrep som kan utføres gjennom enkle midler, burde samfunnet i helhet være ute etter gode og sikre løsninger. En vei å gå for å finne disse løsningene kan være ved å implementere disse enhetene virtuelt som gir mulighet til testing og utprøvning av forskjellige implementasjoner og variabler. Hensikten med dette prosjektet er derfor å undersøke hvordan slike enheter kan bli satt sammen i systemer som kan produsere tilnærmet lik trafikk til et levende miljø, og finne ut hvilke løsninger som egner seg best for å sikre de mot potensielle angrep.

Testene i dette prosjektet ble utført ved implementasjonen og installasjonen av en Programmable Logic Controller satt sammen med en vanntank, en human machine interface og to forskjellige sikkerhetsløsninger som skal tilsvare et industrielt kontrollsystem. Dette systemet ble da undersøkt for hvor komplett systemet er i forhold til hvor mye det ligner på et reelt system og trafikken det produserer. Med systemet testet ut ble det installert et Intrusion Detection System og en Security Information Event Management løsning som sikkerhetstiltak. Disse ble sammenlignet i forhold til bruk av tid på installasjon, dokumentasjon tilgjengelig, helhetlighet av løsningen, mulig informasjon som kunne hentes ut og hvor godt de detekterte angrep.

Resultatene fra testene viser at det virtuelle industrielle kontrollsystemet produserer tilnærmet lik trafikk og at det oppfører seg likt som et fullverdig industrielt kontrollsystem. Gjennom installasjonen og implementasjonen av en IDS og SIEM vises det at kompleksiteten til sistnevnte er av en mye større grad. Likevel er mulighetene og egenskapene til en godt oppsatt SIEM så instrumentelle til god sikkerhet at det sees på som en bedre løsning enn en IDS i et industrielt kontrollsystem. Begge løsningene klarer å detektere angrep, men når det blir testet et komplekst Man-In-The-Middle angrep er det bare SIEMen som har nok informasjon til å gjøre en riktig vurdering. Disse resultatene begrunnes med resultater av simuleringene, resultater av forsøk på detektering av angrep og kriterier satt for å måle forskjellige aspekter ved implementasjon og installasjon.

Abstract

The challenges of keeping today's systems secure under the constant evolution brings forth new challenges that need new solutions. The use of small devices with low processing power has been popular for a long time to keep costs down in larger systems. The development of these devices that allow them to connect to more complex devices opens up possibilities and risks that have never been seen before. To keep up with the threat image we need better security and tools that fit the tasks they should perform. When devices that are essential to supplying a whole town with electricity can kneel under because of an attack executed with not much effort, the society as a whole should be looking for well thought out solutions. A way to go to finding these can be by implementing these devices virtually which gives the opportunity to test different implementations and variables. The purpose of this project is to examine how these devices can be put together in systems to produce resembling traffic to live environments, and figure out which solutions are best suited to deal with potential threats.

The tests in this project were performed through implementation and installation of a Programmable Logic Controller put together with a water tank, a human machine interface and two different security solutions that correspond to an industrial control system. This system was then examined as to how complete the system is compared to real one and the traffic it produces. With this being done an Intrusion Detection System and a Security Information Event Management system was then installed as security solutions. These were compared with metrics of time used on installation, documentation available, completeness of the solution, possible log data available and how well they detected attacks.

The results from the tests show that the virtual industrial control system produces traffic and behavior that resembles a live environment. Through the installation and implementation of a IDS and a SIEM, it is shown that the complexity of the latter is of a large degree. With the opportunities and traits a well tuned SIEM possesses, it will still be seen as a better security solution in an industrial control system. Both solutions were capable of detecting attacks, but when a complex Man-In-The-Middle attack was tested only the SIEM had enough available information to determine if an event is a threat or not. These results are justified by the results of simulations, results of attempts on detection of attacks and the criterias set for measuring different aspects of implementation and installation.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Science at Universitetet i Oslo(UiO).

The project work was performed during the spring-semester of 2021, with the purpose of contributing with experimental and theoretical knowledge on securing industrial control systems with Security Information Event Management systems.

I want to send a big thank you to my external supervisor Storm Jon-Martin Pettersen, for excellent guidance, helpful feedback and being a positive influence throughout the work of this project. I would also like to thank Nils Gruschka for his feedback on my thesis. A special thank you goes out to my girlfriend and friends for helping to keep me sane through a difficult pandemic while completing my degree.

Oslo, May 2021

[illegible]

Magnus Korneliussen

Contents

List of Figures

Chapter 1

Introduction

This chapter contains the topic covered by the thesis, motivation behind it, contribution to the field and the research questions which will be answered.

1.1 Topic covered by the thesis

The threat landscape of today's cybercrimes is ever-evolving. Our systems keep getting more insecure each day they are not updated with protection and safeguards against the next attack. Especially systems handling critical infrastructure will be more prone to attacks in the future, with the major consequences it produces. A modern industrial environment for critical infrastructure has many physical devices and a lot of endpoints within the system that are affected by various factors. The field of Security Information and Event Management (SIEM) is based on collecting log data from all these devices that are connected to the network, which reflects the activity on the devices themself and the communication between them. The log data is then aggregated and normalized and analyzed with different aspects of security in mind. The key point of this is to be able to correlate occurring events and give more context to how or why these are taking place.

The context of a singular event can be crucial in how it is handled. In industrial environments, uptime is the ultimate goal, with availability in the information security triangle being the most important one. If an event is acted upon unnecessarily causing lower production or a halt in the service, the consequences can be catastrophic. This also insinuates that even though a SIEM system can be good in an environment with many different variables and devices, it also needs to go through the steps it takes fast and precisely enough to ensure uptime. This work will discuss the limitations and effectiveness of implementing and using a SIEM system compared to installing a less complex IDS in industrial environments.

A SIEM usually consists of multiple ways of generating the log data, with an example being an implementation of an IDS. This furthers the complexity of implementing it compared to only having one source to detect threats. With the vast importance of correct implementation being a crux in SIEM systems this will also be part of the topic and problematization of it.

1.2 Motivation

With the constant evolution of the Internet of Things and Industrial Internet of Things(ref til bakgrunn), and their applications in cyber-physical systems such as Industrial Control Systems, modern vehicles, and critical infrastructure. With the possibilities, these next-generation combinations of embedded devices provide, the benefit of negatively impacting them will increase. With the evolving effort to attack these systems, the security and general testing of solutions will be of great importance in the future [1]. I will try in this project to clarify some of these problems and find out if it is possible to make a simulation of an actual industrial control system together with a SIEM. If this is possible I would also like to explore how the SIEM picks up on certain attacks compared to an IDS and if it can detect and alert that they are happening. The contribution of this to the industrial information security community will hopefully be to showcase that it is possible to fully simulate an ICS environment. This can have the opportunity to help with simulating and testing different devices, security solutions, and network topologies. The end goal of this is an easily deployable solution to compare performance and security metrics.

1.3 Contribution of thesis

A SIEM system compared to an IDS gives more opportunity to correlate occurring events in a system, which makes it possible to contextualize what is going on and if taking action is necessary. The IDS can only pick up internet traffic that is directed towards or through it, it can not know if the action it alerts has any effect on the target. With a SIEM on the other hand, you can use the logs from the IDS and compare them to logs from the actual device the attack was executed on to find out if it had any impact. The problem with comparing and correlating these events in an Industrial Automation Control System is the delicate nature of the physical devices, and how they are affected by delay and time sensitivity. The vast complexity and size of a system where a SIEM could be applicable also raise concerns about how difficult it is to achieve a proficient solution in regards to efficiency, cost, and time. The size and complexity also speak to issues with the need for personnel who entirely grasp all components and how they're supposed to interact with each other. The SIEM generates more noise the more devices it has connected to it, so the previous issues mentioned also point to more alerts and possibly false alerts.

Chapter 2

Background

The following chapter reveals the information derived from research relevant to the thesis, which includes IDS, SIEM and some solutions, ModBus traffic and PyModbus, PLC, event correlation, effective attacks against industrial control systems, longevity of the current solutions and how they are impacted.

2.1 ICS

Industrial Control Systems is a general term used to describe several types of control systems often used in industrial sectors and critical infrastructures. These include supervisory control and data acquisition systems, distributed control systems, and Programmable Logic Controllers. It consists of a combination of control components that in conjunction achieve an industrial objective. There are two main components of how they work, and that is a system for producing output, and a part for controlling the system which produces the output. The part of the system primarily concerned with maintaining conformance with specifications is referred to as the controller. Because of the possible applications of an ICS, they are an important part of numerous countries' critical infrastructure. [2]

2.1.1 PLC

Programmable Logic Controllers are industrial small digital computers that are adapted to be used in the control of manufacturing processes or other tasks that need control. They are considered the smallest version of an ICS, and are used because of their flexibility, ruggedness, and easy programmability. With many of their applications being safety-critical, such as traffic control systems or chemical plants, their ability to be adapted to the situation are of vast importance. The hardware of a PLC consists of a CPU that is microprocessor-based, a memory, input- and output-points where signals can be received, and sent to actuators. Usually, a PLC is equipped with an operating system that allows it to load and run programs and that performs self-checks. The programs are developed and compiled on external devices, because of the lack of processing power it contains. The main difference in a PLC to a conventional system is the operation mode: the program on it is executed in a permanent loop: input is read, the already compiled program computes a new internal state and output, and then the output is updated.[3]

For programming the PLC the previous primary programming language was Ladder Diagram.[4] It brings many benefits but also some well-understood problems. With the evolution of the use-cases of PLCs, there has also been an evolution in how the programs for them are written. This has also led to standards being written with recommendations as to what programmable logic to implement. Some of the available solutions are the Signal Interpreted Petri Net(SIPN), Structured Text(ST), Function Block Diagram(FBD) and Sequential Function Chart(SFC).[5]

2.1.2 DCS

Distributed Control Systems are systems of sensors, controllers, and associated computers that are distributed throughout an industrial control system. The DCS is the same as intertwining a lot of PLCs together, because it has the ability to make adjustments to the input of numerous units at a time.[6]

2.1.3 SCADA

SCADA stands for Supervisory Control And Data Acquisition. It is not a full control system as such, but rather has the focus on a supervisory level. It is a purely software package that is positioned on top of hardware to which it is interfaced, generally via a PLC. [7]

2.1.4 Human Machine Interface

The human-machine interface is the platform for cognition and communication between human and machine. It's an approach to transmit information back and forth between the parties, and in regards to automation systems they can have a critical role in keeping a stable and safe system.[8]

2.1.5 Modbus and MODBUS/TCP

Modbus has become a de facto standard for industrial control systems. Many of the Modbus systems implement the communications layer using TCP as described in the Modbus over TCP/IP specification. The specification defines an embedding of Modbus packets in TCP segments where the TCP port number 502 is assigned to the Modbus protocol.[9] For the compatibility to stay intact with Modbus over serial lines, the payloads have to be limited to at most 253 bytes. For communication, the Modbus protocol uses a simple master-slave relationship between devices. This begins with the master device initiating a transaction where the slave(s) respond by supplying the requested data to the master or executing the command requested. It is only possible for one device to be designated as the master, while the remaining devices are slaves which usually are PLCs that control devices with simple input and output options.

The Modbus PDU is what the PLC provides as an interface based on the Modbus data model. It consists of "coil"(single-bit) and "register"(16-bit) tables, which each contain elements numbered 1 to n. For each table, the data model allows up to 65536 data items. The read and write operations that are associated with these items can access multiple consecutive data items that either are function code or payload. The function code is a single-byte integer with a range of 1 to 127. The Modbus standard has defined 19 of these 127 codes, where the most usual ones are codes for reading(1, 2 and 3) writing functions(5 and 6). When a successful request execution is made this is indicated by the slave returning a response packet that echoes the function code of the request, followed by relevant data according to the command that was executed.[10]

An easy solution for implementing Modbus/TCP is applying it through the use of Pymodbus. It is a full Modbus protocol implementation that uses twisted/tornado/asyncio for the asynchronous communications core.

2.1.6 IT and IoT

In ICS it is important to differentiate between operational technology(OT) and information technology. In previous years the heart of ICSs were only the OT, but in the modern day businesses incorporate IT based on the system functions desired in the overall system. For further referencing, the definitions are this[11]:

- OT - hardware and software that detects or causes a change through the direct monitoring and/or control of physical devices, processes and events in the enterprise

- IT - the technology involving the development, maintenance, and use of computer systems, software and networks for the processing and distribution of data

2.2 IDS

NIST describes an intrusion as an attempt to compromise CIA(Confidentiality, Integrity and Availability), or to bypass the security mechanisms of a computer or network.[12] Intrusion detection is the process of monitoring the events occurring in a computer system or network, and analyzing them for signs of intrusions.[13] Intrusion detection systems are a practical example on fuzzy evaluation of different criteria, and taking decisions by evaluating multi-dimension problems. It consists typically of a set of entities distributed through the system, whether it is in the network or on the hosts themselves. The different types of IDSs are all based on finding different ways to alarm about your network or computer system being the victim of an intrusion. The ways of implementing a system like this is either through a software or hardware system that automates this process. The detected threats are passed on as alarms to a security manager(s), to be handled and what actions must be taken and by whom.

2.2.1 Simplifying approaches

For the different detection technologies, they all come from two different ways of solving the problem; detection based on anomaly and misuse. A table created by H.J. Liao et. al.[13] divides it into another five different sub-classes; statistics, pattern, rule, state and heuristics.

Statistics-based uses anomaly and signature detection, and gets its information from audit data, user profiles and usage of disk and memory. It's not as complicated as the other types, but this in turn gives less accuracy.

Pattern-based only uses signatures, and gets information from audit records and signatures of known attacks. Also considered a simple solution, and provides less flexibility.

Rule-based uses mostly anomaly and signature for detection, and gets its information from audit records, rule patterns and from user profiles and policy. Is a lot more complex to get up and running, because the rules are not easily created and updated. If the rules created are extremely thorough this can still be an effective approach to intrusion detection.

State-based with state-transition analysis has the ability to use signature, anomaly and stateful protocol detection with information from audit records and state-transition diagrams of known attacks. It provides a lot of flexibility, and can detect across user sessions.

The heuristics-based approach has the ability to be used both as an IDS with anomaly and/or signature detection. It gets information from audit data, sequence of commands and should in time be possible to predict events. It can be self-learning, and should be fault tolerant. Another positive is that it can be easily configurable, which makes it scalable and flexible.

2.2.2 Accuracy in IDS

One of the biggest concerns when setting up intrusion detection for your system is using the most effective and accurate detection method suited to your specifics. From a security analysts point of view accuracy means the rate of alarms that are correct in comparison to how many you actually receive. False-positives means that an event is considered as an intrusion that has to be dealt with when it is actually not. This leads to more work for security analysts and with too many fake alerts the real ones have a possibility to be missed. Another great concern with accuracy is on the opposite side of the spectrum, with false-negatives. This means that an intrusion is not detected when it should be, opening up your system for malicious activity.

2.2.3 Detecting the threats to your system through three different angles

With an intrusion detection system three main methodologies exist; using Signature-based detection, Anomaly-based detection or Stateful Protocol Analysis. They all have their own drawbacks and positives attached to them, so there will never be a perfect way of handling detection for every system possible.

Signature-based detection

This methodology is based on storing signature profiles identifying patterns associated with network intrusions in a signature database, from which these signature profiles classification rules are generated. The data packets which are transmitted on the network that have corresponding classification rules are classified according to the previously generated classification rules. The now classified packets are forwarded to the signature engine, which compares them with the signature profiles. This methodology has some similarities with blacklisting, with the focus on threats and not what traffic and events are allowed in the system. This is a positive when it comes to protecting against well-known attacks. Mainly it will struggle with dealing with new threats that are yet to be discovered, which is a major problem in large systems that have a lot of possible devices and communication that can be exploited in different ways. With this method false-negative rates will be more of a problem than false-positives. Signature-based detection also requires a large library of signature profiles that continuously has to be worked on to stay up-to-date with current emerging threats.[14]

Anomaly-based detection

Similar to a signature-based method, the anomaly-based compares the current events and traffic to another previously saved behaviour. The difference lies in what the previously saved behaviour consists of, instead of comparing to signatures that are known ways of "attacking" the system it will rather compare to what is known as the "normal" behaviour. This normal behaviour and flow of traffic and events has to previously be established, and when the IDS finds an "anomaly" which can be defined with "an event that is suspicious from the perspective of security" it raises an alarm. This benefits the detection with that previously unknown attempts at intrusion is detected, because this will deviate from the established baseline behaviour of the system. Downfalls of using this approach is that it will generate more false-positives than the signature-based one if the "normal" behaviour isn't broad or established in a way that it reflects regular use properly. This way is better suited for an industrial environment because the baseline for "normal" behavior is less deviant with the strict confines of allowed operations.[15]

Stateful Protocol Analysis

A methodology that operates by comparing predetermined profiles of acceptable protocol behaviours for protocol states against observed activities to detect deviations and misbehaviors. In this sense "stateful" means that the IDS has the capabilities to identify and track the states of network, transport or application protocols that have a concept of state. This method is opposed to the "blacklisting" done with signature-based IDS, but instead "whitelisting" accepted protocol state behaviours to identify any abnormal packets that go outside this range. Shares a lot of similarities to anomaly-based, but the stateful protocol analysis relies on vendor-developed universal profiles that specify how particular protocols should and should not be used. This makes the IDS able to detect unknown attacks, which is more difficult when using an IDS based on "blacklisting". Again, using whitelisting requires a lot of work to capture every possible state transition that the system should be allowed to execute.

From an industrial point of view, stateful protocol analysis can be merged with tracking of cyber-physical states. Being able to do this the IDS can be used as a safeguard for the safety of human resources and physical assets in addition to IT assets. Another reason SPA can be a good method for industrial applications is that production and operations are normally streamlined, and has a well-known allowed set of state changes it goes through in the course of a production cycle. An enterprise IT system won't have such little deviation in how it is used, because the confines aren't as strict for what the devices and humans involved are allowed to do.[16]

Intrusion Prevention System

Another system that is enabled by having an IDS is an intrusion prevention system(IPS). When the IDS detects behavior that is deemed malicious, an IPS brings the opportunity of an automated response to specific behaviors. This is not as applicable in the industrial automation setting, because if you give full control over your system to conduct cyber-physical state changes if an alarm goes off, it can have devastating consequences. When actions for the system are automatically executed when a specific intrusion is detected, this can be manipulated by attackers to exploit the known behavior that the automatic action detection can lead to. With the physical dimension and the focus on always being operational these types of responses can either cause harm to employees on-site or cause the system to shut down. IPS can also become a serious bottleneck for an environment that is reliant on having no delays and a minimum of jitter to confer quality of service requirements. The reason for this is because IPSs must be placed in-line in order to actively stop attacks, but IDSs can be placed on mirrored ports, preventing a potential bottleneck.

NIDS

A specific way of implementing an IDS that targets the network that the hosts use to communicate with as well as the hosts themselves, the operating system and the applications. It captures network traffic at a specified network segment, and inspect for malicious activity.

2.2.4 IDS in an industrial environment - an intersection between safety and security

Challenges of an industrial environment is a different aspect than the regular enterprise IT systems. This includes quality of service, time-sensitive applications, cyber-physical states and a lot more traffic which requires specific monitoring. Attacks on e.g. a SCADA system is usually not based on the exploitation of a single packet[17], but instead a collection of packets attacking several vulnerabilities either over time or at once. The specific limitations of a SCADA system with the strict rules for traffic and state-changes opens up for a “whitelisting” practice, because the baseline traffic of a control system usually has low amounts of deviation. This methodology in a normal enterprise IT environment will cause a lot of false-positives, but in an industrial environment a practice like this is made possible because of the focus on cost-effective operations and the usually streamlined processes.

The approach of considering the entire control system as different devices and subsystems having states or cyber-physical states[18] is a way of viewing the security and safety of a control system intertwined. With well-known attacks such as the Stuxnet worm[19] and the sewage spill at the Maroochy Water Station in Australia[20] being possible primarily because of the control systems not having the correct information of equipment operation monitoring values, or the systems completely ignoring the physical systems operating tolerances. A proposed way of solving this is by tracking the cyber-physical states of the system, ensuring that when a state is evolving in the direction of failure, it can be stopped. This can be done with a NIDS solution that parses all packets going through the network with addition of sensors tracking physical information about the devices. Physical constraint algorithms can be applied to device command and data streams, with a stateful layer to represent and track the physical systems operational modes. A solution like this requires a lot of pre-existing knowledge of every device in the system, contextual information on how the devices interact with each other and it offers problems with scaling in terms of when a device is added or replaced the entire library of physical constraints and algorithms has to be revamped.

Security Operations Center(SOC)

Performing the tasks necessary to run a SIEM efficiently is having a SOC whose goal is to monitor security-related events from the companies assets which includes networking, perimeter defense systems like firewalls and IPS devices, application servers, databases, user accounts, sensors, and various devices that comprise the operations of the specific organization/company.[21] Each of these devices or assets can be monitored using a variety of sensors, and maintain their log files of activity. The SOC receives event information filtered out from the log files and sensor activity and triggers alerts that indicate if there is a possibility of malicious behavior from the inside or an intrusion from the outside. When an alert is triggered, the personnel in the SOC decides if the triggered alert is a false positive regarding e.g. updates or maintenance and it can be considered as harmless, or if the alert is indicating that malicious activity is happening. If the SOC gets an alert and it is considered to be of malicious intent, a message is forwarded to a team that coordinates a response to the incident and informs necessary parties that an attack is happening. This can be to the owner or operator of the involved assets or equipment, which can also be forwarded to law enforcement if a larger attack is discovered.

For a solution like this to be functioning the personnel of the SOC has to have good analytic and forensic capabilities and has to be aware of the threat-image to the devices and sensors they receive possible security-related events from. They also need to be aware of who needs the information they receive, to know the best practice when an attack happens to a specific part of the system, and who to inform to counteract and find the best solution on how to handle a security-related event or incident. The size of a SOC and the necessity of having one come down to the question of the size of your operation and the security budget that is appropriate for your company's assets. A SOC can consist of a single person on-site or a dedicated facility with hundreds of employees. If your system is large enough to consider implementing a SIEM it certainly is possible that also means that having a SOC is a possibility.

2.3 SIEM and SOC

SIEMS revolve around the same concept as an IDS. They are used as an important tool in e.g. security operations centers, which monitor security events related to an organization's IT assets. In an industrial environment the OT assets also should be monitored, to be collected, normalized and analysed from various sources in the organization. The difference between IT and OT assets lie in that IT deals with information assets and OT deals with the physical components. This means that IT assets consist of digital information flow and its data, and OT assets manage the operation of physical processes and the machinery used to execute the operations. In a power plant this can be a transformer which transforms voltage to the wanted specifications. The baseline of how a Security Information and Event Management System works is by collecting data, processing it to be human-readable and analyzed by either an automated system that correlates the events and gives alerts, or by personnel that has knowledge of what data can be considered malicious.

2.3.1 Security Operations Center(SOC)

Performing the tasks necessary to run a SIEM efficiently is having a SOC whose goal is to monitor security-related events from the companies assets which includes networking, perimeter defense systems like firewalls and IPS devices, application servers, databases, user accounts, sensors, and various devices that comprise the operations of the specific organization/company.[21]. Each of these devices or assets can be monitored using a variety of sensors, and maintain their log files of activity. The SOC receives event information filtered out from the log files and sensor activity and triggers alerts that indicate if there is a possibility of malicious behavior from the inside or an intrusion from the outside. When an alert is triggered, the personnel in the SOC decides if the triggered alert is a false positive regarding e.g. updates or maintenance and it can be considered as harmless, or if the alert is indicating that malicious activity is happening. If the SOC gets an alert and it is considered to be of malicious intent, a message is forwarded to a team that coordinates a response to the incident and informs necessary parties that an attack is happening. This can be to the owner or operator of the involved assets or equipment, which can also be forwarded to law enforcement if a larger attack is discovered.

For a solution like this to be functioning the personnel of the SOC has to have good analytic and forensic capabilities and has to be aware of the threat-image to the devices and sensors they receive possible security-related events from. They also need to be aware of who needs the information they receive, to know the best practice when an attack happens to a specific part of the system, and who to inform to counteract and find the best solution on how to handle a security-related event or incident. The size of a SOC and the necessity of having one come down to the question of the size of your operation and the security budget that is appropriate for your company's assets. A SOC can consist of a single person on-site or a dedicated facility with hundreds of employees. If your system is large enough to consider implementing a SIEM it certainly is possible that also means that having a SOC is a possibility.

2.3.2 Different tools for monitoring and collecting event information

SIEMS used security tools like IDSs and AVSs for detection previously. Every system used for gathering information from the system from different devices and sensors previously used their own interface specified by the vendor. Problems come from this because expertise was needed in how to operate each interface, and there existed no software to correlate the events identified across these tools. The smaller individual tools operated with little or no awareness of how the entire architecture was set up, so it meant more false positives to handle which in return meant a larger workload for the SOCs. SIEM systems are designed to face these obstacles, and they are solved by collecting the events from all the different tools, normalizing the data into a single readable format, sending all the events to a single interface so it can be analyzed by the personnel in the SOC.

2.3.3 Using SIEM systems to track auxiliary contextual information

SIEM systems can also be used to track and keep up-to-date information about your devices, because it can collect and normalize all data input coming from the various devices and sensors you have. This makes it so it can use something which is similar to a library function, with all new threats and vulnerabilities to your specific devices being up-to-date and giving your SOC the opportunity to act accordingly. In conjunction with the massive amounts of logs collected, post hoc forensic analysis and investigation is also a possibility to better combat attacks from APTs(Advanced Persistent Threats).

2.3.4 SIEM systems components

The devices mentioned below will send events through specialized connectors to either a database for logging or directly to a terminal managed by Security Analysts or to the SOC if the event that occurs is deemed as a security risk or has malicious intentions towards the system.

- Authentication device - a device that should authenticate users that are using the system.
- Firewall - a perimeter defense system that detects in- and egress traffic to the systems network.
- Network device - device logging network traffic inside the systems own network, can be set up with a network intrusion detection system(NIDS) to more efficiently deal with network traffic that should set off an alarm before it reaches the SOC or SAs.
- Web application firewall - a firewall that's a perimeter defence for detecting unallowed traffic to the web applications
- Application server - a device that hosts the application
- Host - computer or workstation
- Specialized connectors - The backbone of making the SIEM system work. These are devices that receive the events. To make these able to parse the input events from all the other devices, they have to be customized for each version, device type and vendor to manage to convert them into a common format that is readable from either the logdatabase or directly from the event-dashboard by the SOC or SAs.

2.3.5 Operational challenges for the SOC when using a SIEM system

Most problems with using a SIEM system for the SOC, are problems related to scalability and complexity of keeping track of all events being received when the systems become too large to be able to have a full overview of all its security devices and sensors. When a system needs expanding in regards to security devices and sensors, adding them to the event logs is not the hard part. Having security analysts in the SOC being able to understand when an event is a false positive because of bad rules, lack of context information for correlating it to another event is what makes it difficult. The rules have to be set up in a way that they catch all real-positives, and keep false-negatives to a minimum. With this in mind, when the rules are created for flagging an event, you will not try to make them as efficient as possible to minimize the workload of the SOC, but instead accept a large amount of false-positives to try to be as certain as possible u get no false-negatives. With the problem of the main-goal of most companies is that they want to be as cost-efficient as possible, security often gets a lower priority than it should. Being able to go through all the possible malicious events you work in a SOC, gets harder the bigger the system is. They will get more events in general, which also means more false-positives that still have to be checked out.

Another problem already mentioned is the contextual awareness of events happening. This comes from the fact that the SOC usually isn't connected to the regular operations of the company. They are only concerned with the security events the SIEM provides them with. Here communication is key, if one of the devices is supposed to be updated, it can trigger alerts that are a false-alarm. If this update has not been communicated to the SOC, they will treat it as a regular alarm, and waste even more time with excess false-positives that could've been avoided.

2.3.6 Challenges with storing, collecting, correlating, and analyzing events

As the world evolves, so does the scale of the systems that SIEM can be applied to. Progress in the technical development of devices like IoT devices has the computational capabilities to give even more log data than before, so having hardware capable of collecting all these events will be more costly than before. With these capabilities already being constrained by cost, it can lead to less secure practices which include less filtering of the ingress data, and accepting malicious data to enter the system. Another problem with getting in even more events is the possibility to store them. Even though an event might not be of importance when it doesn't give off an alarm, it can still be used later on for forensic purposes. Events can not be stored forever to be used like this, so all companies will need to find a balance that gives the right tradeoff concerning the storage costs and requirements for analysis. More data into the SIEM systems also provide problems with all the facets it should help improve, with more events from devices and humans, which needs more specific pattern recognition to find the real positives instead of flooding the event-stream with false positives.

2.3.7 SIEM in an industrial environment

The ability to process large amounts of data and being able to aggregate and normalize it and also being able to correlate it sounds like a perfect solution for an industrial environment. Industrial automation systems have a lot of traffic from different sources which usually has different vendors, protocols, physical limitations, and a low amount of processing power. Centralizing and aggregating the massive logs of data after collecting them, makes it possible to view events that can pose a security risk to your system. So the question is why isn't the use of SIEM systems more widespread in today's IACS environment landscape. The question probably comes down to cost-efficiency. The level of protection and amount of money willing to be spent to protect an asset should be aligned with the amount of money that will be lost if that asset is compromised in regards to confidentiality or integrity or lost for any reason with thought around availability. The massive cost of implementing a SIEM system with all its available capabilities may not be more cost-effective than dealing with the attacks that slip by a cheaper way of securing the system.

2.3.8 SIEM in a nutshell

To conclude the information gathered around SIEM systems is that they need to be applied by exquisite technical personnel to function properly. It gives a lot of responsibility to the people working in the companies SOC, and on its own, it is useless as it only gives the alert that something is happening that also needs to be analyzed to figure out further action. To have the opportunity to analyze and correlate events the logging needs to be done in a way that covers every device that picks up useful information for the security of the system. The SIEM system needs to be adaptable in a way that allows continuous reconfiguration of rules, amount of data collected from each device, being able to customize the event flow so that when the SOC finds correlating events the system can learn to show the specific events contextual information. Good communication between the SOC and the other moving parts in the company is essential. Without this, a regular update or patching to something novel can be viewed as a security risk that takes focus away from events that need it.

2.4 Elastic Stack as a SIEM

When considering the combined effort of different software and modules created by the company Elastic with help from others, it is usually referred to as the Elastic Stack, or ELK(Elastic, Logstash, Kibana)Stack for short. It is open-source software that provides a partially free and has managed to prove its efficiency in a large number of applications worldwide. This is some of the reasons it is a good fit for testing security in an experimental implementation. The main software components are:

- Elasticsearch used as a search and analytical system
- Logstash used as a software pipeline for data processing
- Kibana, which is a tool for visualization and navigation through the system
- Beats, which is a set of programs that are used for collecting and transporting logs, files, and packets

2.4.1 Elasticsearch

A distributed search and analytical system core which supports the architectural style REST API and sending data via JSON. By centralizing incoming data and supporting the clustered architecture, it allows the system to scale out. With innate abilities like transparency and reliability with a failure detection system, ensures that it maintains a high uptime rate. The kernel performs a real-time search over large volumes of heterogeneous data structures(documents). Document is the basic unit of information that can be indexed and is specified in the JSON format. The system has a well-developed API, and the list of supported languages for interoperability includes Java, Python, C++, and others.[22]

2.4.2 Kibana

Kibana is a software component that implements the visualization and navigation(user interface) in the Elastic Stack and other applications where it is applicable. The data is presented as a customizable and interactive dashboard in real-time with a lot of ready-for-use widgets.

2.4.3 Beats

The set of programs or collectors which are installed on client devices. The way the programs interact with elasticsearch and what data, logs, or files they collect and forward depends on what type of information you want to let Elasticsearch analyze. Examples are:

- Auditbeat which audits the activities of users and processes on the system it is installed on. It can also be used to detect changes to critical files, like binaries and configuration files, and identify potential security policy violations.[23]
- Packetbeat is a real-time network packet analyzer that provides visibility between the servers in your network. It captures the network traffic between the application servers, decoding the application layer protocols, and correlates the requests with the responses and it records interesting fields for each transaction.[24]
- Metricbeat periodically collects metrics from the operating system and services running on the server. It takes the metrics and statistics which it collects and sends them to the specified output.[25]
- Heartbeat daemon which is installed on a remote server to periodically check the status of your services, to determine whether they are available and reachable.[26]

2.4.4 Elastic Security and Elastalert

Elastic security is a package solution provided by Elastic that combines SIEM, endpoint security, threat hunting, cloud monitoring, and alerting for your Elastic Stack applications. Where it previously was free and readily available for use with smaller applications of the ELK Stack, it is now being placed behind a paywall. To have a way of monitoring the security of your applications placed in the elastic stack and not having to pay for the door to this wall there are solutions out there. Elastalert created by Yelp is a free and open-source solution and a simple framework for alerting anomalies, spikes, or other patterns of interest from data in Elasticsearch.[27] It is provided based on the ideology that if it can be seen in Kibana, ElastAlert can alert on it. It works by combining Elasticsearch with two types of components, rule types and alerts. Elasticsearch is periodically queried and the data is passed on the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match.

2.4.5 Logstash

The software pipeline is commonly used for data processing in the ELK stack before passing the data on to Elasticsearch. It simultaneously collects data from many different sources, primarily processes them, and sends it to a storage subsystem. It has a built-in parser that allows it to normalize heterogeneous data, determine the geographical coordinates by IP, to process information from various sources regardless of format and structure.[22]

2.5 Related work

This section of the literature study offers an overview of the literature that relates to my thesis and its research points. Some papers talk about general information security and how a SIEM solution can be the salvation to many problems. With my brief tenure on researching this topic, I have still been unable to find a complete and easy-to-navigate system for testing how a SIEM works without engineering one myself. It was observed that some of the papers talked about the pitfalls and difficulties of using a SIEM solution, but the consensus in the papers seems to be that for specific problems a specific solution does the job. The harder part is being able to implement a general out-of-the-box solution that is easy to set up and test.

Montesino, FEnz and Baluja mentions the need of achieving greater efficiency in the world of information security management. This involves reducing the complexity and points to automation as a possible solution.[28] This is not only difficult to achieve with regular office infrastructure but automating a SIEM system for a larger scale industrial setting can prove to be even more difficult. “The applicability of a SIEM solution: Requirements and Evaluation” states that an organization not only should consider factors than the technical side when evaluating what SIEM solution fits them best. They should instead also dive into the organizational and technical requirements that should be addressed and examine the applicability of a SIEM solution. This includes looking at how the company is structured in size, location and verticality, doing a general risk and asset run-down, how it can be possible to manage and hire a Security Operation Centre, and the compliance to regulations combined with the forensics capability of their data.[29]

I also had the joy of talking to someone from KraftCERT regarding SIEM solutions and their work within this field. The price of installing and the hardware of sensors needed for a SIEM to run properly are expensive as well as the usability of the sensor completely relies on the fact that the designer of the solution knows where to place it for maximum efficiency. This leads to the point that implementing a SIEM as your security solution not only demands financial resources, it also requires personnel who can dedicate time and effort. This is a necessity for being able to have even the possibility to put together hardware devices, sensors, technicians, security personnel, response teams, and the communication flow that a well-tuned SIEM requires.

On actually implementing a simulation of an IACS with a SIEM, I was not able to find any literature which was indicative of the complexity of even an entry-level solution. This thesis hopefully can help with broadening the specter of the difficulties and hardships of testing similar topologies.

Chapter 3

Method

This chapter will contain the different methods pursued to answer the questions I have previously described in this thesis.

3.1 Experiments

Goes under the classification of scientific method, where a factor or a few chosen variables variate and the others are kept at a constant so that the influence of the examined factor can be decided. The data from the experiment should be collected with accepted methods that can give reproducible results. In addition I will need to choose correct points of reference. The experiments should make the foundation of the scientific method, with hypotheses and theories. The sets of data and the methods should be commonly available, so the results from the experiment can be proven. Before these experiments can be conducted there has to be a hypothesis in place about an observed phenomenon or case study. All experiments also need to contain control groups:

- Negative controls that aren't expected to give any impact during the case study
- Positive controls that are expected to give an impact during the case study

Before the experiment the different variables that are in question have to be mapped, and of what types they are. They can either be continuous or categorical. Categorical variables have limitations in the number of possible values. Another thing that also has to be identified is the independent variables that variate in the background of the experiment that can result in confusion. It also has to be investigated if the variables covariate.

A baseline has to be established regarding how data is collected, and this is done by designing the study. All aspects and parts of the experiment are described as precisely as possible so it is reproducible for further research and testing.

The reason an experiment fits well into answering the research questions in this thesis is that they are best answered with practical scenarios rather than theoretical solutions. The main reason is that the possibility to manipulate an independent variable, control the others, and then being able to observe the results has great merit. This is because when you find an outcome, it gives the opportunity to find out the reason behind it.[30]

3.2 Literature Study

To conduct a literature study can be divided into two different types, where the focus is different. The method I chose is most similar to an approach that focuses on getting the answers with a more practical viewpoint aligned with the project I am working on. All the literature is in regards to either the theory behind or about the experiment I am writing about in my thesis. The sources the literature is derived from have great variance, with some sources being Google Scholar and UIOs research library and for more specialized content even videos from technology conferences. This is because the resources needed to explain and implement the solutions at the level I am interested in required a lot of different material.[31]

Chapter 4

Experiment

This chapter explains how the experiment is set up, what devices the topology contains, how they were configured and implemented, how the attacks work and an explanation of the two security solutions.

4.1 Experimental set up

To answer some of the questions that arose during the research period for my thesis and the writing of the preliminary essay, I wanted to conduct an experiment. The purpose of the experiment is to simulate how a live implementation of an ICS operates and behaves to enable conducting tests on performance and security. The basic topology needs devices that produce, store, manufacture, or transport and a way of controlling how they do it. The components required for an exact simulation in terms of size and complexity are not feasible to implement, because the number of devices and resources needed would exceed what was available. To execute the experiment the smallest version of an ICS was chosen, an application of a PLC regulating an individual process. For this kind of system to function properly a PLC is needed to work together with an input device and an output device. The state of the input device is continuously monitored and decisions are made based upon the custom program used to control the state of the output devices. The PLC and output devices are usually segregated into their own network, to defend the output device from tampering. The PLC itself should be connected to a version of an HMI to be able to control what input devices and programs it is monitoring. To also segregate this entire topology it should be separated from other network zones with a firewall. The first part of this chapter contains an explanation of how the experiment is designed, and how the devices have to be configured to be used in an intended way for the experiment. The implementation of software on the devices and what their purpose is will also be explained, and what modifications have been made to out-of-the-box solutions. The last part contains the execution of the attacks and the results of this.

The process which the input and output devices should monitor in the experiment is the levels and in-/outflow of a water tank. This tank is connected to and controlled by the “OpenPLC” PLC and the program it contains, which is connected to the HMI Node-RED to be able to do human interaction with the system. The network is segregated into a separate zone with the pfSense firewall.

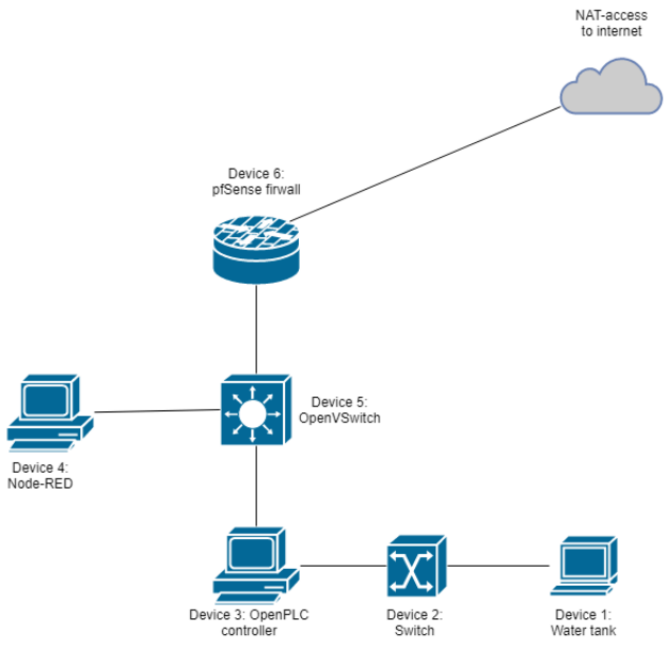


Figure 4.1: Depiction of basic topology

The water tank is only connected to the “OpenPLC” controller through a switch creating a separate subnet because it does not need to be directly connected to the internet. The “OpenPLC” controller and the “Node-RED” devices are connected to the OpenVSwitch, which again connects through the internet through the “pfSense”-firewall.

Enabling the devices to communicate with each other is set up through GNS3. It is a tool used for network simulation and emulation and supports both directly imported devices from the GNS library or virtual machines imported through one of the available virtualization platforms. The two main components that I’ve utilized that make GNS3 work are the GNS3-all-in-one software combined with the GNS3 virtual machine. The GNS3 VM works as a local server that creates the topologies which are designed through the software solution. When using virtualized machines imported through VirtualBox a UDPTunnel simulating network connections are set up on each of the network adapters available.

Devices that are not imported through the GNS3 library are virtual machines created through the software VirtualBox. Their documentation says it is a cross-platform virtualization application that enables the capability to run multiple OSes on an existing computer.[32] I have chosen this virtualization platform because it is integrated to work with GNS3 and because I had previous knowledge of how it operates before initiating my experiment.

The first device implemented is "Device 1: Water Tank", which is a Ubuntu-based live server. To make this work in the intended way of my system the water tank needs one network port to connect to the switch that is in turn connected to the OpenPLC controller. This is because the water tank does not need any other connections, to lower its attack surface. It has to simulate a server and gateway that generates ModBus traffic that is aimed to resemble a live industrial one. This is done through installing software called "Simulation-server-modbus-gateway".[33] The gateway holds three registers that are in use, where input register 1 holds the value of the tank, 2 holds the value of the inflow and 0 controls the value of the outflow.

The "Device 2: Switch" is imported through GNS3 and used as an out-of-the-box solution with no modification performed to have the needed functionality.

The "Device 3: OpenPLC" is also a Ubuntu-based live server. The setup necessary for it to function together with the water tank and the node-red machine is software that can work as a functional standardized Programmable Logic Controller.[34] It goes through three steps: check inputs, execute the program and update the outputs. Programming it to do the wanted task is done by feeding it with a Structured Text program, which sets the registers in the water tank to wanted levels. The program also automates that when the water tank reaches maximum or minimum level, it increases either the in- or outflow so that the level does not exceed either limit. All these tasks are made possible in my environment with the work of Thiago Alves. The OpenPLC soft-PLC for Linux is used to feed the slave device which is the water tank with input and deliver the output to the Node-RED machine for it to be visible. It is the only available controller that provides the entire source code.

"Device 4: OpenVSwitch" is imported through GNS3 and is a multilayer virtual switch produced by OpenVSwitch. It is designed to enable network automation through programmatic extension, it supports virtualization with VirtualBox and has kernel datapath distributed with Linux. It also supports creating SPAN ports, which in this case means a port that mirrors all traffic entering and exiting the switch, hence the position it has in the proposed topology. This means a security solution can plug into this port to attain the traffic of all devices connected to the switch.

For visualizing the output from the OpenPLC, “Device 5: Node-RED” is used as a human-machine interface. The device is implemented using an Ubuntu-based desktop computer. It is connected with the OpenPLC which fetches the output produced from the water tanks data. Node-RED is a programming tool used for wiring together hardware devices, APIs, and online services with a browser-based editor. Within the editor-tool the data from the different registers are used to show the different metrics of the water tank:

- Current level
- Minimum level
- Maximum level
- Outflow in liters
- Inflow in liters

This device is also used to control the web interface for the OpenPLC controller, which needs a graphical interface to operate. This means that the Structured Text script, slave device specifications, and program settings are managed through this device as well.

The final device of the basic topology is “Device 6: pfSense firewall”. It is used to set up the routing tables of the main subnet and provides a connection to the internet as well. With the choice of implementing static IPs in my topology DHCP is not needed, so it is used as an out-of-the-box solution with DHCP not enabled. Due to it being open-source with a web interface to customize it can be used as a simple security solution as well as basic routing and being a security perimeter it made a good fit in my topology.

4.2 Implementation of devices

4.2.1 Water tank

To implement the water tank, the software “Simulation-server-modbus-gateway” was used to simulate the required functionality. It was imported directly through

Github, and docker was also set up for it to be run via docker-compose. For simplicity, it also received a static IP address because the configurations for the software are not set up for non-static IPs. With both of these set up correctly, the PLC should be able to connect with the proper IP address and port number.

4.2.2 Switch

The basic ethernet switch was directly imported through the GNS3 library with no modifications.

4.2.3 OpenPLC

The OpenPLC device was implemented by following the guide on "OpenPLCproject.com/runtin
After installing git and downloading the OpenPLC software and then installing it, it is necessary to set up the IP for the OpenPLC device to be able to connect to the web interface from the HMI device. With the water tank being designated as a slave I/O from the web-interface, it is necessary to add this device in the configuration to establish a connection when running a program. With the water tank having a static IP and designated port, except for this, all settings should be the default. With all this in mind, the OpenPLC needs a PLC program which is fetched from Github[34]. The Structured Text program enables the water tank to function in the way intended.

4.2.4 OpenVSwitch

OpenVSwitch was directly imported through the GNS3 library, with modifications to ethernet port 7. This was turned into a mirrored SPAN port for future use in conjunction with a security solution.

4.2.5 Node-RED

For implementing Node-RED into the device node.js was installed and used to run npm to install the software. The IP address was set to be static for the sake of simplicity in the topology. For having access to a web-interface a flow has to be set up to be able to determine the input to the OpenPLC device and set the allowed levels of the water tank.

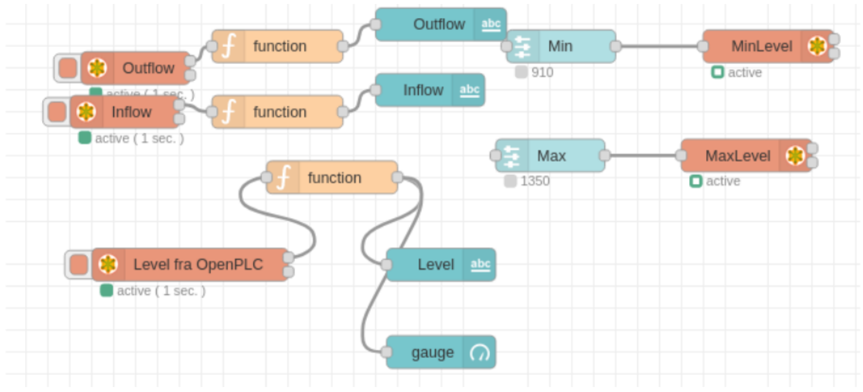


Figure 4.2: Shows how the flows in Node-RED is set up to enable the GUI in appendix 4

This is the HMI device of the topology, and it is later on supposed to also be the device for attacking the tested system. Pymodbus was installed for executing the attackers' python script, hping3 for executing a DoS-attack, and NMAP for the port scanner attack.

4.2.6 pfSense

The pfSense device was implemented as an out-of-the-box solution from pfsense.org as an ISO image to install on a FreeBSD virtual machine.

4.3 Security Solutions and their implementations

With the basic topology previously explained the next part of the experiment is implementing a security solution and testing their capabilities of detection with different attacks. The security solutions which are decided on earlier are an implementation of a IDS and a SIEM solution. This is the part where the experiment requires a different setup and implementation on the other devices in regards to what solution is chosen, so the chapter is split in two; one for the IDS and one for the SIEM.

4.3.1 IDS

When choosing what IDS solution to implement there are a lot of different ones out there to choose from like Snort, Suricata, Zeek, and Security Onion only to mention a few. With my limited time and hardware resources the four most important factors are performance, the difficulty of implementation and use, resource intensiveness, and adaptability of the rulesets. With Snort being known for easiness of installing and deployment with a lot of good resources available, it fits the needs of my experiment. Snort is an open-source Intrusion Prevention System, with the option to enforce rules that can be decided, created, or modified by the user. In my case, it will be set up as a packet sniffer and packet logger, and not as an IPS because this can be problematic in an industrial environment(REF TIL ESSAY HER). The implementation of Snort is done through downloading the software and creating directories for rules and logs.

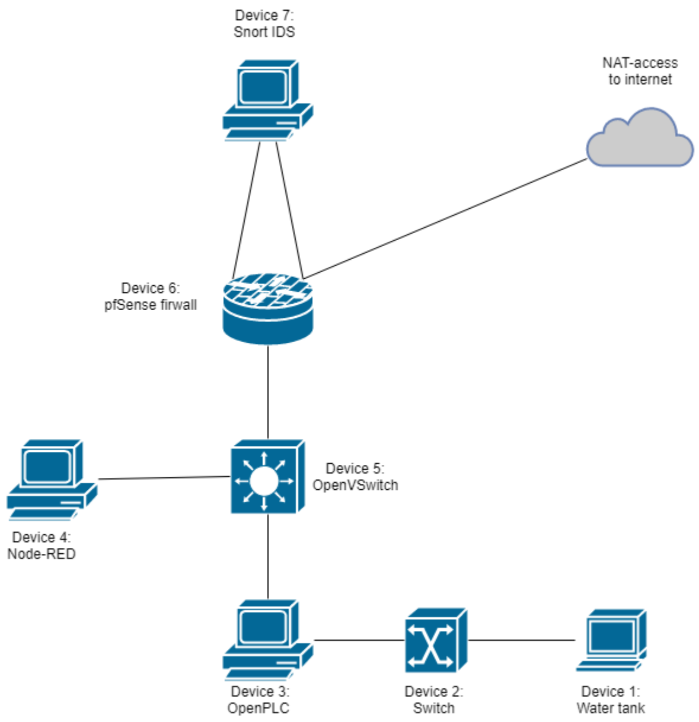


Figure 4.3: Depiction of basic topology

“Device 7: Snort IDS” is an Ubuntu-based desktop computer connected to the OpenVSwitch with two different ports because it utilizes the previously set up

SPAN port to attain the network traffic. The network card enp0s8 listens to the port that receives information from the span port, and this is the chosen interface that Snort also listens to. With how the topology is set up, for Snort to do its job no modification to the other devices is needed to get the detection up and running. Because it only operates with network traffic, which is mirrored through the OpenVSwitch, the other devices can stay as they are meaning it requires very little time and effort to implement.

4.4 SIEM

The difficulty of choosing a SIEM solution was an even harder choice because most solutions are extremely resource-intensive as well as being strenuous to implement. As well as not being something that is regularly implemented on systems it is not strictly necessary or extremely beneficial, the variety of available solutions which are both free and easy to set up does not match the counterpart in this experiment, the IDS. Because implementation of the SIEM is a part of this thesis two solutions were initially tested in production to determine the best direction to work with. Research on the topic led to some interesting options with older papers stating a lot of open source and free tools ready to be implemented. But with the industry moving forward and seeing the potential profits of this new and essential field within security, most solutions have moved on to locking away functionality behind paywalls. My original solution was supposed to be a modified version of the ELK stack previously mentioned in the related work chapter, but with their free security add-on being secluded behind a paywall I was interested in moving on to something else. Another solution that still was entirely free with decent documentation was Malcolm.

The first solution tested was Malcolm; it is a powerful and easily deployable traffic analysis tool suite for full packet capture artifacts (PCAP files) and Zeek logs. It is an open-source project created at the Idaho National Laboratory for Homeland Security and could be a good fit for what the experiment needed with some modification. The easiest way of deploying it was through downloading an ISO file and following the steps from their documentation[35]. After setting up and trying to capture the network traffic similar to how the Snort IDS receives it I was unable to get this working, as no traffic was showing up. After troubleshooting and finding no solutions to my problem, the only possibility left was installing and setting up another device for forwarding the networking logs with Hedgehog Linux. With the complications mentioned and lack of hardware resources, my lab environment would not be able to handle the already taxing device that is Malcolm with the addition of the Hedgehog log forwarder.

With other free solutions being sparse I found an alerting solution made by Yelp[27] that is set up to work with the Elastic Stack, so for the second implementation, the SIEM consists of Elasticsearch, Kibana, Beats, and elastalert. Elasticsearch is used for receiving logs and data from the Beats placed on necessary devices and then processed and visualized through Kibana. Elastalert is a framework used for alerting anomalies, spikes, and other patterns of interest from data in ElasticSearch.

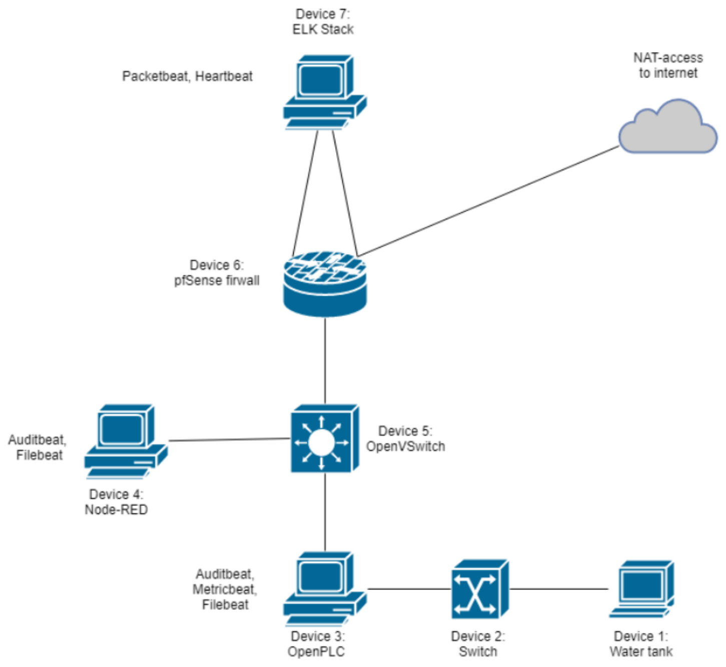


Figure 4.4: Depiction of SIEM and beats installed on various devices

“Device 7: ELK Stack” is an Ubuntu-based device that runs Elasticsearch, Heartbeat, Packetbeat, Kibana, and Elastalert. Because Kibana and Elasticsearch are set up on the same local device, all logs and output are forwarded here. The different beats installed on the main device of the ELK stack are Heartbeat and Packebeat. Heartbeat is a simple daemon that periodically checks the status of the services in my topology to determine whether they are available or not. For picking up network traffic from the SPAN port in a similar manner as the Snort IDS does, Packebeat listens to the network card that is connected to the mirrored port on the OpenVSwitch. In addition for Elasticsearch to have the necessary logs and data, the Beats has to be installed on the remote devices I wanted to monitor.

With “Device 3: OpenPLC” being where the different metrics for the water tank is decided, it is a possible point of weakness for attackers to exploit. The beats chosen for this device are Auditbeat, Metricbeat, and Filebeat. Auditbeat is used for monitoring and auditing the activities of users and processes on it, as well as detecting changes to critical files to its operation. Filebeat on the other hand forwards and centralizes log data from the specified output, in this case, the syslogs for the device and the possibility of sending the information about the PLC. Metricbeat is used to collect metrics from the operating system and from the services that run on the server. It takes the metrics and statistics and ships them to the specified output. “Device 4: Node-RED” can also be used to forward the different levels in the water tank, so here Auditbeat and Filebeat are also installed.

4.5 Attacks

This subchapter explains the attacks used to test the limitations and detection capabilities of the different security solutions chosen. For deciding what attacks to execute the metrics considered were severability to the uptime of the water tank, ease of execution, and type of attack.

4.5.1 Port scan

Port scanning is used to probe a server or host for open ports, which in an industrial setting can be proven devastating with how many devices and hosts usually are available. Ports that are unnecessary and dangerous to keep open are harder to keep track of the larger our system is and can be used to exploit vulnerabilities. [36]

NMAP Port scan

The port scan attack is done with the tool NMAP, which is specified to attack the subnet that the devices you want to target are a part of. To detect an attack like this the security solution needs to be able to find out if there is a sudden spike, or increase in frequency for querying many ports on different devices at the same time. This sort of attack can usually be detected by a firewall that limits between zones, but due to simplicity the attack will be executed from within the topology instead of from the outside.[37]

4.5.2 DoS

Denial of Service attacks is defined as denying the use of a service. This is usually done by flooding the specific service with requests and traffic to the point where the working load is beyond the capabilities of the device running the service. With their low processing power, PLCs and RTUs are explicitly vulnerable to an attack like this, but the effects will be minimized in this simulation because of how the devices are set up. The device controlling the water tank is not run on an actual

PLC, but on a Ubuntu-based server with a magnitude more processing power. So the actual strength of the attack needed to take it out of service is more significant than in an actual Industrial Automation Control System. To detect this attack the security solution has to alert when a specific device is having a sudden spike in queries or if it receives too many queries over a certain timeframe. A DoS attack from the outside can easily be blocked out by the correct firewall implementation because of how zone control should be in an ICS, but again for simplicity the attack is executed from within the network itself.[38]

Hping3

For executing this attack the Node-RED device was used to leverage a tool called hping3, which is a command-line oriented TCP/IP packet assembler/analyzer. The interface is inspired by the ping(8) Unix command, but hping3 is not only able to send ICMP echo requests. It also supports TCP, UDP, ICMP, and RAW-IP protocols, has a traceroute mode, the ability to send files between a covered channel, and many other features. In this case, it is used for sending a large number of packets as fast as possible in an attempt to deny the use of a service.[39]

4.5.3 MITM Setpoint attack

A man-in-the-middle attack is defined by the attacker being in between services or devices either impersonating the device or having actual control of it. In this attack, we assume that an adversary has gained control over the Node-RED device with no other devices being aware of it. The Node-RED device will then instead of going through the Node-RED password-protected interface try to directly change the level of the water tank without no one noticing. This can have a large impact on the water tank because with wrong information about the metrics the OpenPLC device will keep filling up the water tank even though it has reached its maximum levels. To detect a man-in-the-middle attack the devices in your system need rules of what operations and actions they are allowed to permit. Integrity and authenticity checks to make sure that no devices are compromised can also be used to mitigate consequences.[40]

Setpointattack.py

To execute this attack the script in the file “setpointattack.py” is used. It connects directly to the PLC and sets the maximum level of the tank to a value out of the legal scope. The idea is that the script directly changes the input to the “OpenPLC”-device, so that it is possible to damage the water tank by setting illegal values.

4.5.4 Modifications done to IDS and SIEM to detect attacks

For Snort to be able to detect these specific attacks extra rules have to be created in addition to the general rules created by the Snort community. This is done through editing the Snort.rules and writing custom rules based on the behavior it is supposed to report on. The ELK stack receives all necessary information for it to raise alerts when these attacks happen, but they have to be set up through Elastalert to do so. This is done through configuring one of the rule types and their desired metrics as of when something is wrong. The modifications to the rules used in both Snort and Elastalert are a simplified version of what would be made in a real environment to be a proof of concept that the security solutions can detect the attacks executed.

Snort rules

A rule file was written in "/etc/snort/rules" which contains an alert that goes off when other TCP traffic than to the port "502" is executed to protect from NMAP TCP scan. And for alerting the ping scan, it detects if any devices ping the "192.168.1.60" device which is the "OpenPLC" because its only functionality in this topology should be to control the levels of the water tank and sending information to the "Node-RED" device. To detect the DoS attack a rule is written that tracks if any port is connected to with a larger count than normal(in this case 70) every 10 seconds.

Elastalert rules

The rules in elastalert are placed in the example rules folder where two rules are used. The modified version of the example rule for spikes fetches the logs from the elasticsearch index "packetbeat-*" which is network traffic, to alert when an event spikes. The timeframe is 30 seconds, with the threshold being set at 30. If the value of connections to port 502 exceeds 30 by 20, it will forward a message to the service of your choice. I chose Telegram because it provides easy setup with the telegram bot to send the alerts into a Telegram channel of your choice.

The second rule for detecting a port scan can be used in two different ways, depending on the settings chosen. It fetches the logs from elasticsearch index "packetbeat-*", to alert when the frequency of an event exceeds the specified amount in the time frame selected. In my case, it detects the number of events on either the TCP- or ICMP-traffic to alert to telegram as mentioned in the last rule.

The third rule for detecting whether the specified MITM attack is still a work in progress and will be referenced in the future work section.

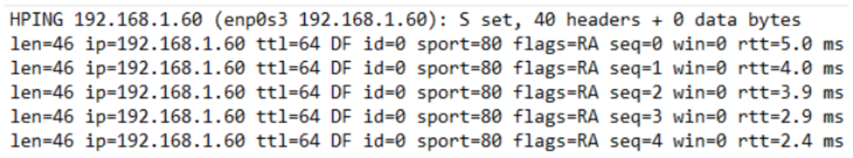
4.6 Attack execution

The observations in this experiment focus on tracing the commands used to execute the attacks to find out if they were successful, and if the security solution managed to send an alert that fires off based on the parameters set in its respective ruleset.

4.6.1 Dos Attack

During the DoS attack, the hping3 tool sent 5000 packets with the command “hping3 -i u20 -S -p 80 -c 5000 192.168.1.60”.

- -i u20 specifies interval of 20 microseconds to wait between each packet
- -S sets the SYN tcp flag
- -p 80 determines to send packets to port 80
- -c 5000 specifies to send 5000 packets

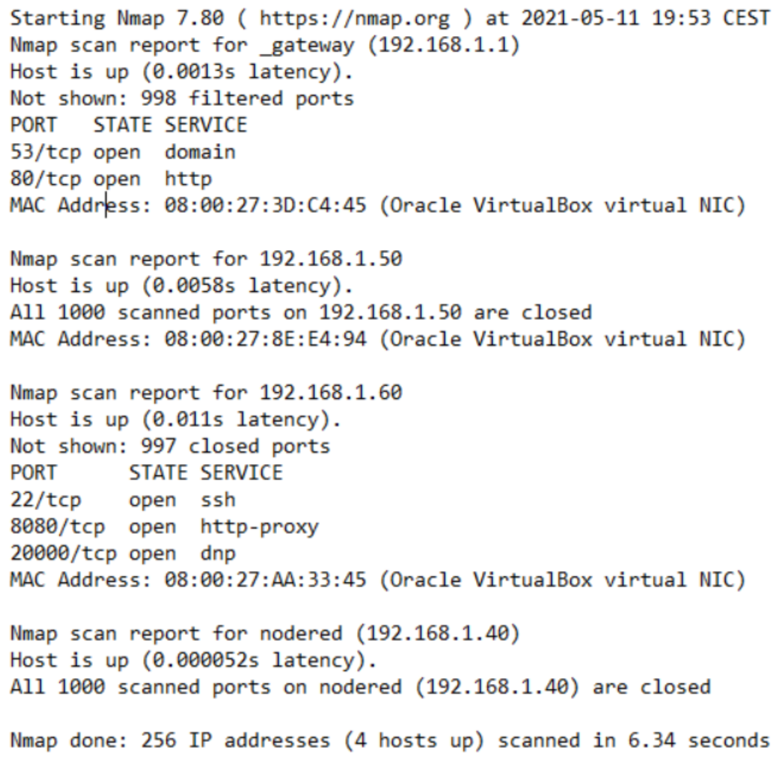


```
HPING 192.168.1.60 (enp0s3 192.168.1.60): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.60 ttl=64 DF id=0 sport=80 flags=RA seq=0 win=0 rtt=5.0 ms
len=46 ip=192.168.1.60 ttl=64 DF id=0 sport=80 flags=RA seq=1 win=0 rtt=4.0 ms
len=46 ip=192.168.1.60 ttl=64 DF id=0 sport=80 flags=RA seq=2 win=0 rtt=3.9 ms
len=46 ip=192.168.1.60 ttl=64 DF id=0 sport=80 flags=RA seq=3 win=0 rtt=2.9 ms
len=46 ip=192.168.1.60 ttl=64 DF id=0 sport=80 flags=RA seq=4 win=0 rtt=2.4 ms
```

Figure 4.5: Example output from hping-command

4.6.2 Port Scan

The execution of the port scan attack on scanning the entire 192.168.1.0/24 range was successful, and it found open ports. This was done with the command "nmap -sT 192.168.1.0/24"; -sT specifies that it does a TCP connect port scan of specified IP/subnet.

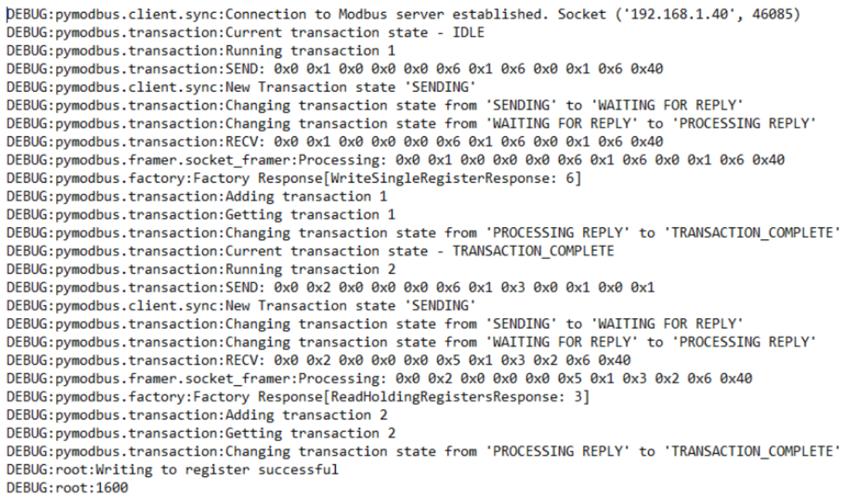


```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-11 19:53 CEST
Nmap scan report for _gateway (192.168.1.1)
Host is up (0.0013s latency).
Not shown: 998 filtered ports
PORT   STATE SERVICE
53/tcp open  domain
80/tcp open  http
MAC Address: 08:00:27:3D:C4:45 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.1.50
Host is up (0.0058s latency).
All 1000 scanned ports on 192.168.1.50 are closed
MAC Address: 08:00:27:8E:E4:94 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.1.60
Host is up (0.011s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp  open  http-proxy
20000/tcp open  dnp
MAC Address: 08:00:27:AA:33:45 (Oracle VirtualBox virtual NIC)

Nmap scan report for nodered (192.168.1.40)
Host is up (0.000052s latency).
All 1000 scanned ports on nodered (192.168.1.40) are closed

Nmap done: 256 IP addresses (4 hosts up) scanned in 6.34 seconds
```

Figure 4.6: Showing output from "nmap -sT 192.168.1.0/24" command, successfully scanning 256 addresses and identifying the four hosts that are up.

4.6.3 MITM attack

The attack is done with a script written in python. It connects to the “OpenPLCs” port 502 from the same device it usually receives change of input from. It then writes to the register that controls the allowed maximum level of the water tank.



```
DEBUG:pymodbus.client.sync:Connection to Modbus server established. Socket ('192.168.1.40', 46085)
DEBUG:pymodbus.transaction:Current transaction state - IDLE
DEBUG:pymodbus.transaction:Running transaction 1
DEBUG:pymodbus.transaction:SEND: 0x0 0x1 0x0 0x0 0x0 0x6 0x1 0x6 0x0 0x1 0x6 0x40
DEBUG:pymodbus.client.sync:New Transaction state 'SENDING'
DEBUG:pymodbus.transaction:Changing transaction state from 'SENDING' to 'WAITING FOR REPLY'
DEBUG:pymodbus.transaction:Changing transaction state from 'WAITING FOR REPLY' to 'PROCESSING REPLY'
DEBUG:pymodbus.transaction:RECV: 0x0 0x1 0x0 0x0 0x0 0x6 0x1 0x6 0x0 0x1 0x6 0x40
DEBUG:pymodbus.framer.socket_framer:Processing: 0x0 0x1 0x0 0x0 0x0 0x6 0x1 0x6 0x0 0x1 0x6 0x40
DEBUG:pymodbus.factory:Factory Response[WriteSingleRegisterResponse: 6]
DEBUG:pymodbus.transaction:Adding transaction 1
DEBUG:pymodbus.transaction:Getting transaction 1
DEBUG:pymodbus.transaction:Changing transaction state from 'PROCESSING REPLY' to 'TRANSACTION_COMPLETE'
DEBUG:pymodbus.transaction:Current transaction state - TRANSACTION_COMPLETE
DEBUG:pymodbus.transaction:Running transaction 2
DEBUG:pymodbus.transaction:SEND: 0x0 0x2 0x0 0x0 0x0 0x6 0x1 0x3 0x0 0x1 0x0 0x1
DEBUG:pymodbus.client.sync:New Transaction state 'SENDING'
DEBUG:pymodbus.transaction:Changing transaction state from 'SENDING' to 'WAITING FOR REPLY'
DEBUG:pymodbus.transaction:Changing transaction state from 'WAITING FOR REPLY' to 'PROCESSING REPLY'
DEBUG:pymodbus.transaction:RECV: 0x0 0x2 0x0 0x0 0x0 0x5 0x1 0x3 0x2 0x6 0x40
DEBUG:pymodbus.framer.socket_framer:Processing: 0x0 0x2 0x0 0x0 0x0 0x5 0x1 0x3 0x2 0x6 0x40
DEBUG:pymodbus.factory:Factory Response[ReadHoldingRegistersResponse: 3]
DEBUG:pymodbus.transaction:Adding transaction 2
DEBUG:pymodbus.transaction:Getting transaction 2
DEBUG:pymodbus.transaction:Changing transaction state from 'PROCESSING REPLY' to 'TRANSACTION_COMPLETE'
DEBUG:root:Writing to register successful
DEBUG:root:1600
```

Figure 4.7: Shows output from the setpointattack.py script, which successfully writes to a register in the water tank to change its value.

Chapter 5

Results of implementation and detection

This chapter is about the results of the implementation of an ICS combined with a security solution. The results of the general implementation of an ICS will be shown with screenshots of traffic and interfaces that are used to control the PLC. The two different security solutions will be based on the following criteria for the implementation:

- *Installation time*
- *Available documentation*
- *Being able to detect and alert on attacks*
- *Subjective difficulty of implementation*
- *Subjective difficulty of use*
- *Completeness of solution and available logs or data for correlation*

5.1 General implementation

During the implementation period different devices and solutions were tested before being able to implement a system that simulates traffic and logs that can be compared to a live environment. The system does the following:

- produces Modbus/TCP traffic that resembles traffic that can be logged and used from a live PLC

- has a PLC that runs in a loop that receives input and produces output based on a specific program
- has a HMI device that can be used as an interface to control the PLC directly
- has the possibility to install applications/forwarders that can send logs and interesting information to a security solution
- resembles a water tank with sensors and actuators that can be run attacks and different simulations on

5.2 IDS Implementation

Implementing Snort as a IDS with the topology and attacks in question is estimated to have taken 20 hours including time spent researching. The available documentation helped greatly with the implementation, as both installing the software and creating custom rules had a lot of specialized documentation in articles and educational videos from Snort. This in turn helped lower both the difficulty of use, implementation and time used for implementation. With Snort being installed and rules set up for specific attacks, alerts were coming in when the attacks applied to the rules written. With the way it was implemented it lacked the capability to detect when the setpoints of the water tank were changed outside of the allowed values. The completeness of the solution comes down to that it contains all functionality of Snort, except the IPS part because this will interfere with the availability and uptime of the ICS. It executes the tasks it is supposed to perform, with only logging network packets and no other form of data making correlation with other input difficult.

5.3 SIEM implementation

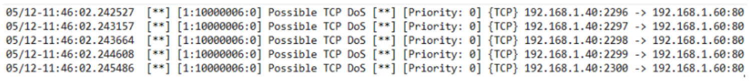
While implementing the SIEM two different solutions were tested, a modified version of the ELK stack and Malcolm. As previously mentioned Malcolm had a level of difficulty that required more resources than initially thought, so ELK stack was chosen. This was because it also had a free version that was able to send alerts. The time estimated to implement the entire ELK stack including beats are close to approximately 120 hours including the time spent researching different ways to set up and configure. Here the time spent configuring the different beats, Elasticsearch and Kibana to enable the availability of interesting logs and data was the largest portion. The documentation for the Elastic Stack is well written and makes the software easily configurable with their basic solution, as to what information you want to collect and send to Kibana. However the documentation in finding the necessary data you want to analyze and alert on, and how to use it is severely lacking and hard to figure out on your own and takes a lot of time tinkering. The subjective difficulty of implementation is not considered easy, but a carefully and well written documentation makes the installation of Elasticsearch, Kibana and the different beats easy compared to how complex it seems.

The difficulty of use in the way intended for this experiment is considered to be larger than for the installation. This is because of the amount of data in kibana combined with lacking documentation for users not using the paid version of elastic security. Setting up the alerts with the free elastalert version proved to be extremely difficult and time-consuming, and the documentation on writing rules and fetching data from elasticsearch was not understandable without extra research. This combined with the hardship of using kibana to get the correct data, keeps the difficulty of implementation high, usability low, getting alerts hard and completeness not sufficient for what the possible utilization is. The completeness of the solution is not enough to correlate events and decide whether to alert based on this. With the beats installed the ELK stack implementation collects various types of logs from the different devices that can be used for correlation. The correlation can help with either detecting security threats or helping lower the rates of false-positive alerts.

5.4 IDS detection

5.4.1 DoS

The logs collected from the IDS confirm that it detects that there is a possible DoS attack ongoing with TCP traffic.

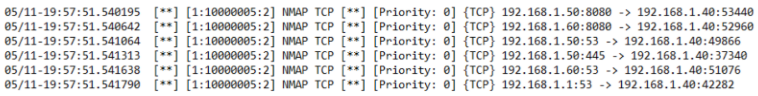


```
05/12-11:46:02.242527  [**] [1:10000006:0] Possible TCP DoS [**] [Priority: 0] {TCP} 192.168.1.40:2296 -> 192.168.1.60:80
05/12-11:46:02.243157  [**] [1:10000006:0] Possible TCP DoS [**] [Priority: 0] {TCP} 192.168.1.40:2297 -> 192.168.1.60:80
05/12-11:46:02.243664  [**] [1:10000006:0] Possible TCP DoS [**] [Priority: 0] {TCP} 192.168.1.40:2298 -> 192.168.1.60:80
05/12-11:46:02.244608  [**] [1:10000006:0] Possible TCP DoS [**] [Priority: 0] {TCP} 192.168.1.40:2299 -> 192.168.1.60:80
05/12-11:46:02.245486  [**] [1:10000006:0] Possible TCP DoS [**] [Priority: 0] {TCP} 192.168.1.40:2300 -> 192.168.1.60:80
```

Figure 5.1: Example output from Snort.

5.4.2 Port scan

The logs from Snort show that the attack was detected as a possible NMAP TCP scan.



```
05/11-19:57:51.540195  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.50:8080 -> 192.168.1.40:53440
05/11-19:57:51.540642  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.60:8080 -> 192.168.1.40:52960
05/11-19:57:51.541064  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.50:53 -> 192.168.1.40:49866
05/11-19:57:51.541313  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.50:445 -> 192.168.1.40:37340
05/11-19:57:51.541638  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.60:53 -> 192.168.1.40:51076
05/11-19:57:51.541790  [**] [1:10000005:2] NMAP TCP [**] [Priority: 0] {TCP} 192.168.1.1:53 -> 192.168.1.40:42282
```

Figure 5.2: Example output from Snort.

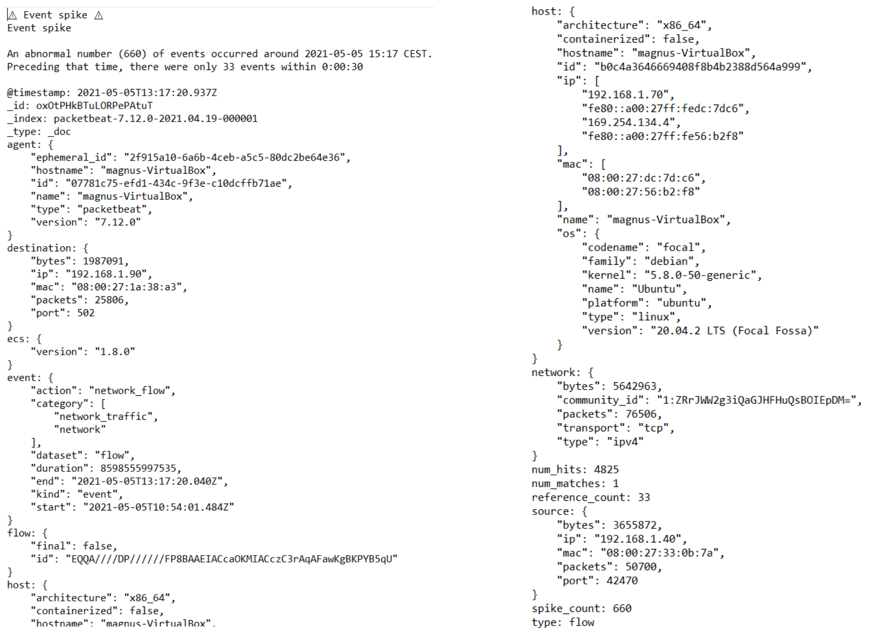
5.4.3 MITM

Snort IDS has no possibility to alert that the MITM attack is happening.

5.5 SIEM detection

5.5.1 DoS

The message sent to Telegram from the elastalert service indicates that there is a possible spike in events which can indicate a DoS attack.



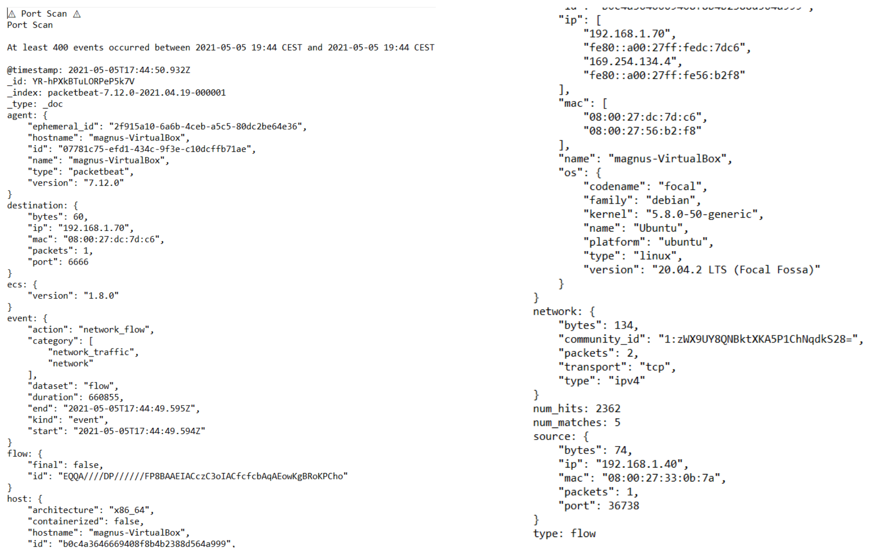
```
⚠ Event spike ⚠
Event spike

An abnormal number (660) of events occurred around 2021-05-05 15:17 CEST.
Preceding that time, there were only 33 events within 0:00:30

@timestamp: 2021-05-05T13:17:20.937Z
_id: oxOtPHkBTuLORPePAtuT
_index: packetbeat-7.12.0-2021.04.19-000001
_type: _doc
agent: {
    "ephemeral_id": "2f915a10-6a6b-4ceb-a5c5-80dc2be64e36",
    "hostname": "magnus-VirtualBox",
    "id": "07781c75-efd1-434c-9f3e-c10dcffb71ae",
    "name": "magnus-VirtualBox",
    "type": "packetbeat",
    "version": "7.12.0"
}
destination: {
    "bytes": 1987091,
    "ip": "192.168.1.90",
    "mac": "08:00:27:1a:38:a3",
    "packets": 25806,
    "port": 502
}
ecs: {
    "version": "1.8.0"
}
event: {
    "action": "network_flow",
    "category": [
        "network_traffic",
        "network"
    ],
    "dataset": "flow",
    "duration": 8598555997535,
    "end": "2021-05-05T13:17:20.040Z",
    "kind": "event",
    "start": "2021-05-05T10:54:01.484Z"
}
flow: {
    "final": false,
    "id": "EQQA////DP//////FP8BAAEIACcaOKMIACczC3rAqAFawKgBKPYB5qU"
}
host: {
    "architecture": "x86_64",
    "containerized": false,
    "hostname": "magnus-VirtualBox",
```

```
host: {
    "architecture": "x86_64",
    "containerized": false,
    "hostname": "magnus-VirtualBox",
    "id": "b0c4a3646669408f8b4b2388d564a999",
    "ip": [
        "192.168.1.70",
        "fe80::a00:27ff:fedc:7dc6",
        "169.254.134.4",
        "fe80::a00:27ff:fe56:b2f8"
    ],
    "mac": [
        "08:00:27:dc:7d:c6",
        "08:00:27:56:b2:f8"
    ],
    "name": "magnus-VirtualBox",
    "os": {
        "codename": "focal",
        "family": "debian",
        "kernel": "5.8.0-50-generic",
        "name": "Ubuntu",
        "platform": "ubuntu",
        "type": "linux",
        "version": "20.04.2 LTS (Focal Fossa)"
    }
}
network: {
    "bytes": 5642963,
    "community_id": "1:ZRrJWW2g3iQaGJHFHuQsBOIEpDM=",
    "packets": 76506,
    "transport": "tcp",
    "type": "ipv4"
}
num_hits: 4825
num_matches: 1
reference_count: 33
source: {
    "bytes": 3655872,
    "ip": "192.168.1.40",
    "mac": "08:00:27:33:0b:7a",
    "packets": 50700,
    "port": 42470
}
spike_count: 660
type: flow
```

Figure 5.3: Screenshot from Telegram with alert on an Event Spike.

5.5.2 Port scan

The message sent to Telegram from the elastalert service indicates that there is a port scan happening based on the frequency of events occurring on the network.



```
⚠ Port Scan ⚠
Port Scan

At least 400 events occurred between 2021-05-05 19:44 CEST and 2021-05-05 19:44 CEST

@timestamp: 2021-05-05T17:44:50.932Z
_id: YR-hPXkBTuLORPeP5k7V
_index: packetbeat-7.12.0-2021.04.19-000001
_type: _doc
agent: {
    "ephemeral_id": "2f915a10-6a6b-4ceb-a5c5-80dc2be64e36",
    "hostname": "magnus-VirtualBox",
    "id": "07781c75-efd1-434c-9f3e-c10dcffb71ae",
    "name": "magnus-VirtualBox",
    "type": "packetbeat",
    "version": "7.12.0"
}
destination: {
    "bytes": 60,
    "ip": "192.168.1.70",
    "mac": "08:00:27:dc:7d:c6",
    "packets": 1,
    "port": 6666
}
ecs: {
    "version": "1.8.0"
}
event: {
    "action": "network_flow",
    "category": [
        "network_traffic",
        "network"
    ],
    "dataset": "flow",
    "duration": 660855,
    "end": "2021-05-05T17:44:49.595Z",
    "kind": "event",
    "start": "2021-05-05T17:44:49.594Z"
}
flow: {
    "final": false,
    "id": "EQQA////DP//////FP8BAAEIACczC3oIACfcfcbAqAEowKgBRoKPCho"
}
host: {
    "architecture": "x86_64",
    "containerized": false,
    "hostname": "magnus-VirtualBox",
    "id": "b0c4a3646669408f8b4b2388d564a999".
```

```
    "ip": [
        "192.168.1.70",
        "fe80::a00:27ff:fedc:7dc6",
        "169.254.134.4",
        "fe80::a00:27ff:fe56:b2f8"
    ],
    "mac": [
        "08:00:27:dc:7d:c6",
        "08:00:27:56:b2:f8"
    ],
    "name": "magnus-VirtualBox",
    "os": {
        "codename": "focal",
        "family": "debian",
        "kernel": "5.8.0-50-generic",
        "name": "Ubuntu",
        "platform": "ubuntu",
        "type": "linux",
        "version": "20.04.2 LTS (Focal Fossa)"
    }
}
network: {
    "bytes": 134,
    "community_id": "1:zWX9UY8QNBktXKA5P1ChNqdkS28=",
    "packets": 2,
    "transport": "tcp",
    "type": "ipv4"
}
num_hits: 2362
num_matches: 5
source: {
    "bytes": 74,
    "ip": "192.168.1.40",
    "mac": "08:00:27:33:0b:7a",
    "packets": 1,
    "port": 36738
}
type: flow
```

Figure 5.4: Screenshot from Telegram with alert on an Port scan.

5.5.3 MITM

To implement an elastalert rule that detects a MITM attack with the specifications in “setpointattack.py”, elasticsearch has to be updated with data that correlates with the actual levels of the water tank directly from the sensors. This is not implemented yet in the current solution, but will be referenced in chapter 8.

Chapter 6

Discussion

Some of the criteria are subjective from the author of this thesis, which for reference is a student completing a masters degree in information security with previous experience on the topic. This means reproduction of the experiment may not yield the same exact results. If a SIEM for instance was supposed to be implemented in a live environment on a bigger scale the timeframe would be longer, or if the active observer has more experience it would probably be shorter. The time is estimated using an approximate from the notes and project logs. A virtual system like this will never be completely similar to a live environment, but it should still be strived for. In this sense the completeness of the solution means completeness in comparison to what is possible to accomplish with the different devices, software and solutions used.

When implementing a virtualization of an ICS over a period of five months with no previous knowledge on the subject there will be some shortcuts. Even with this the utilization of the resources available, a working simulation was completed. It produces Modbus traffic, but the OpenPLC device can be used in tandem with another solution if a different type of traffic is wanted. The topology has all devices connected, and the possibility to control the input and output of a PLC device. With limited literature available on the subject of implementing a system that resembles ICS traffic and behavior, it makes troubleshooting difficult. The documentation available from the "OpenPLCproject" combined with software from GitHub[33] made the set up easier. Without the documentation and available software the process would have been time consuming because of the learning required to even start writing a gateway and server to simulate the water tank.

Using the metrics mentioned in chapter five, the two different security solutions provide completely different pictures of what they should be used for and how hard they are to implement. When comparing implementation time spent on the ELK stack with the Snort IDS, it is clear that the first requires more effort and time to get up and going. With the available documentation being somewhat similar as of installation goes, the biggest differential here is actually using the system you have installed. Getting the information into the SIEM is not hard, but actually utilising it is extremely difficult. The IDS on the other hand is also easy to get the information it can be implemented with, but even fully utilised it does not cover the security needs of the system it is supposed to protect. Correlation of logs in an industrial environment can help with detecting anomalies and figure out if they are threats or not. This in turn means that the security system less often forces halts based on false-positives, which increases uptime. With the available logs and data from ELK stack, correlation should be possible. In this thesis I was not able to find a way to do this, and I encourage future studies to try and utilize this aspect of a SIEM implementation. The logs and data available from elasticsearch visualized in Kibana is of great benefit when wanting to secure every part of your system, it brings the possibility to track and monitor everything that happens on all your devices from commands executed to the metrics of the hardware.

My experience with the security part of ELK stack is not necessarily something which reflects the general opinion because I chose not to take advantage of the already integrated Elastic Security. Instead elastalert was chosen because of the idea that being open source and free, this was something that easily could be applied. Writing the rules for the different attacks were on a difficulty level not needed for the effect it had. The documentation was not well enough written and with bad examples as of how to proceed when rules that were different from the examples were needed. Going for the more established and complete solution of Elastic Security from elastic can make detection and alerting on threats easier.

With the DoS and port scan attack being only network-based, they can be detected with rules that apply only the information gained from network packets. With the IDS and SIEM both being able to detect and alert that suspicious activity has occurred when they are executed, the only difference between them can be how fast it is detected, and if it can be correlated with another event. The biggest difference arrives in the third attack, where executing the “setpointattack.py” script from inside the system. With the information available there is no possible way of denying the HMI to set the value of the water tanks levels. When only reviewing the network packets, this looks like a basic operation. To detect this type of attack more data and information is necessary.

Chapter 7

Conclusion

One of the most important conclusions that can be decied upon regarding the research problems is that the implementation of a SIEM is going to be time consuming and difficult if you want it to be a perfect fit for your system. An important note is that it has the possibility to be significantly more powerful than an IDS solution, solely based on the opportunities more types of logs and correlation introduces.

The other part of the experiment also points to other conclusions that can be drawn:

- The difficulty of implementing the virtualization of the ICS was not time consuming based on the fact that there was excellent documentation available on the subject
- Correct implementation that allows correlation of logs increases the complexity of implementing a SIEM
- When detecting attacks that are executed with network packets the IDS performs at a similar level as the SIEM
- The bad documentation from ElastAlert directly influences the difficulty level of getting alerts from Elasticsearch without subscribing to Elastic Security
- Man-in-the-Middle attacks are hard to detect with only the logs from network packets
- The noisiness of systems creates massive amounts of logs and data available in Kibana from Elasticsearch. The importance of clearing away the noise to be able to see critical and vital information can not be underrated

The results of implementation and attacks on both the ICS itself and the security solutions implicates that there still is a lot of room for improvement. With more time the security and traffic simulation could be more realistic and hardened, and further work and research can be recommended. Some of the areas where this applies will be mentioned in chapter 8.

Chapter 8

Further work

This chapter contains areas that can be improved and should be researched further than what was done in this thesis.

8.1 ElastAlert rules

The previously mentioned elastalert rule for detecting the MITM-attack in chapter 4.4.3.2 can be worked more on in the future. For a rule to detect the MITM-attack specified in the experiment, there are several possibilities:

- It has access to an index in elasticsearch which carries the logs from the physical sensors as to how and when they were accessed by using anomaly-based detection the timing of the usually automated water tank can seem off
- The maximum value of the water tank can be polled and sent to an elasticsearch index where if the number allowed as maximum exceeds a certain value it can fire off an alert
- It can alert if the payload message from the Node-RED machine contains setting the value of the water tank outside an accepted scope

The reason I was not capable of getting in place a rule that will alert me to an attack like this is because of limited time, knowledge, resources, and documentation available on the subject. Because of the price of using Elastic Security which probably has good documentation and support, I had to go forward with the free and open-source solution Elastalert. It does not have proper documentation available for it to be easily set up without a lot of research and trial and error. Just to implement the two basic rules that detect port scans and DoS attacks which were simpler with Snort I had to use approximately 20 hours to get them working. This should be worked on because it really has some promise to detect attacks that an IDS does not have the capabilities to.

8.2 Improving data in Kibana

The data shown from Kibana in the appendixes can be improved so that they are more visible and easier to extract important information from. Being that the ELK stack thrives and survives on being able to correlate data from a vast amount of sources in different formats a good utilization of Kibana can be helpful in securing your system.

8.3 Expanding from one PLC

The singular PLC can be expanded to multiple ones, for more accurate simulation of a system that is more similar to a DCS. This would be of more interest because it would generate more traffic and noise, hence testing what security solution and attacks will have greater resemblance to a live environment.

8.4 Integrating with other zones and subnets

A big part of the evolution in the IIoT industry is that it supports integrating IT and OT traffic. This will also need further testing to find out safe procedures and policies. Further work could be done on creating zones outside of the OT zone, to simulate attacks and traffic coming from another zone or subnet to determine values and rules.

8.5 Implementing Elastic Security

Instead of using elastalert for detection, elastic provides its own solution to security. This was not used in my thesis because of time available and cost of implementing it. This service can provide a better solution to detecting and alerting on security threats than self-written rules in elastalert. It also has direct integration in Kibana, and it can be easier to handle all your data and alerts from one interface instead of having to send them somewhere else.

Bibliography

[1] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial internet of things,” 5. 2015. (Accessed on 16/05/2021).

[2] K. Stouffer, L. S., P. V., A. M., and H. A., “Guide to industrial control systems (ics) security,” n.d. 2014. (Accessed on 16/05/2021).

[3] A. Mader, “A classification of plc models and applications,” n.d. 2000. (Accessed on 16/05/2021).

[4] C. Peshek and M. Mellish, “Recent developments and future trends in plc programming languages and programming tools for real-time control,” 5. 1993. (Accessed on 16/05/2021).

[5] M. Minas and G. Frey, “Visual plc-programming using signal interpreted petri nets,” 05. 2002. (Accessed on 16/05/2021).

[6] H. Benitez-Perez and F. Garcia-Nocetti, “Reconfigurable distributed control,” n.d. 2005.

[7] A. Daneels and W. Salter, “What is scada?,” n.d. 1999. (Accessed on 16/05/2021).

[8] C. Gong, “Human-machine interface: Design principles of visual information in human-machine interface design,” 2009.

[9] A. Swales, “Open modbus/tcp specification,” 3. 1999. (Accessed on 16/05/2021).

[10] N. Goldenberg and A. Wool, “Accurate modeling of modbus/tcp for intrusion detection in scada systems,” 6. 2013. (Accessed on 16/05/2021).

[11] A. Hahn, “Operational technology and information technology in industrial control systems,” 8. 2016. (Accessed on 16/05/2021).

[12] M. Scarfone, “Guide to intrusion detection and prevention systems(idps),” 2. 2007. (Accessed on 16/05/2021).

[13] L. L. and L. Lin, “Intrusion detection system: A comprehensive review,” 1. 2013. (Accessed on 16/05/2021).

[14] S. Wu P., “Signature based network intrusion detection system and method,” 3. 2002. (Accessed on 16/05/2021).

[15] M. Tedoro and V. Diaz-Verdejo, “Anomaly-based network intrusion detection: Techniques systems and challenges,” 2-3. 2009. (Accessed on 16/05/2021).

[16] Y. Yang and S. Mclaughlin, “Stateful intrusion detection for iec 60870-5-104 scada security,” 6. 2014. (Accessed on 16/05/2021).

[17] N. Fovino, M. Masera, A. Carcano, and A. Trombetta, “An experimental investigation of malware attacks on scada systems,” 12. 2009. (Accessed on 16/05/2021).

[18] C. Mcparland, S. A., and S. Peisert, “Monitoring security of networked control systems: It’s the physics,” 11-12. 2014. (Accessed on 16/05/2021).

[19] C. Falliere and Murchu, “W32.stuxnet dossier.” Published by Symantec, 2. 2011. (Accessed on 16/05/2021).

[20] M. Slay, “Lessons learned from the maroochy water breach,” n.d. 2008. (Accessed on 16/05/2021).

[21] S. Bhatt, P. Mandahata, and L. Zomlot, “The operational role of security information and event management systems,” 9. 2014. (Accessed on 16/05/2021).

[22] A. Kuleshov, I. Ushakov, and K. I.., “Aggregation of elastic stack instruments for collecting, storing and processing of security information and events,” n.d. 2017. (Accessed on 16/05/2021).

[23] E. B.V., “Auditbeat overview.” Webpage. (Accessed on 16/05/2021).

[24] E. B.V., “Packetbeat overview.” Webpage. (Accessed on 16/05/2021).

[25] E. B.V., “Metricbeat overview.” Webpage. (Accessed on 16/05/2021).

[26] E. B.V., “Heartbeat overview.” Webpage. (Accessed on 16/05/2021).

[27] Yelp, “Yelp elastalert.” Webpage. (Accessed on 16/05/2021).

[28] R. Montesino, S. Fenz, and W. Baluja, “Siem-based framework for security conrols automation,” 10. 2012. (Accessed on 16/05/2021).

[29] H. Mokalled, R. Catelli, V. Casola, D. Debertol, E. Meda, and R. Zunino, “The applicability of a siem solution: Requirements and evaluation,” n.d. 2019. (Accessed on 16/05/2021).

[30] UIO, “Eksperiment.” Webpage, 2. 2011. (Accessed on 16/05/2021).

[31] UIO, “Retningslinjer prosjektoppgave.” Webpage. (Accessed on 16/05/2021).

[32] VirtualBox, “Virtualbox manual.” Webpage. (Accessed on 16/05/2021).

[33] J.-M. Storm, “simulation-server-modbus-gateway.” Webpage. (Accessed on 16/05/2021).

[34] A. Thiago, “What is a plc?.” Webpage. (Accessed on 16/05/2021).

[35] R. Medlin, “Csi-siem.” Webpage. (Accessed on 16/05/2021).

[36] J. Gadge and A. A. Patil, “Port scan detection.” 2008 16th IEEE International Conference on Networks, n.d. 2008. (Accessed on 16/05/2021).

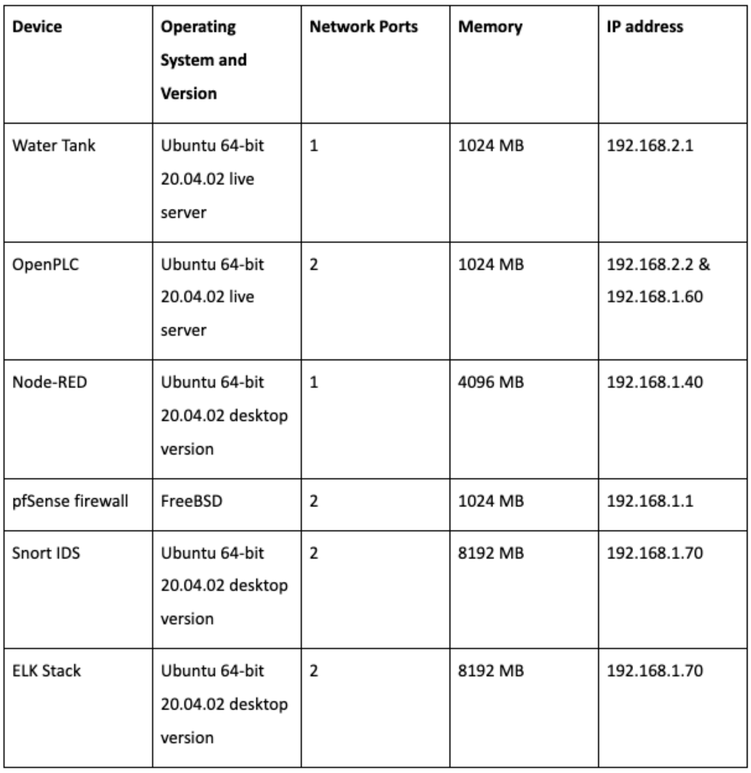
[37] NMAP, “Nmap reference guide.” Manual Pages. (Accessed on 16/05/2021).

[38] A. D. Wood and J. A. Stankovic, “Denial of service in sensor networks.” Computer, vol. 35, no. 10, 10. 2002. (Accessed on 16/05/2021).

[39] S. Sanfilippo, “hping3 package description.” Webpage. (Accessed on 16/05/2021).

[40] N. Asokan, V. Niemi, and N. K., “Man-in-the-middle in tunnelled authentication protocols.” International Workshop on Security Protocols, n.d. 2005. (Accessed on 16/05/2021).

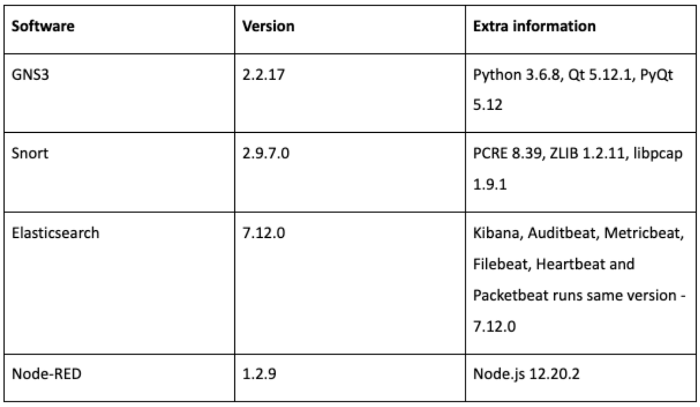
Appendix A



Device	**Operating System and Version**	**Network Ports**	**Memory**	**IP address**
Water Tank	Ubuntu 64-bit 20.04.02 live server	1	1024 MB	192.168.2.1
OpenPLC	Ubuntu 64-bit 20.04.02 live server	2	1024 MB	192.168.2.2 & 192.168.1.60
Node-RED	Ubuntu 64-bit 20.04.02 desktop version	1	4096 MB	192.168.1.40
pfSense firewall	FreeBSD	2	1024 MB	192.168.1.1
Snort IDS	Ubuntu 64-bit 20.04.02 desktop version	2	8192 MB	192.168.1.70
ELK Stack	Ubuntu 64-bit 20.04.02 desktop version	2	8192 MB	192.168.1.70

Figure 1: Overview of different devices.

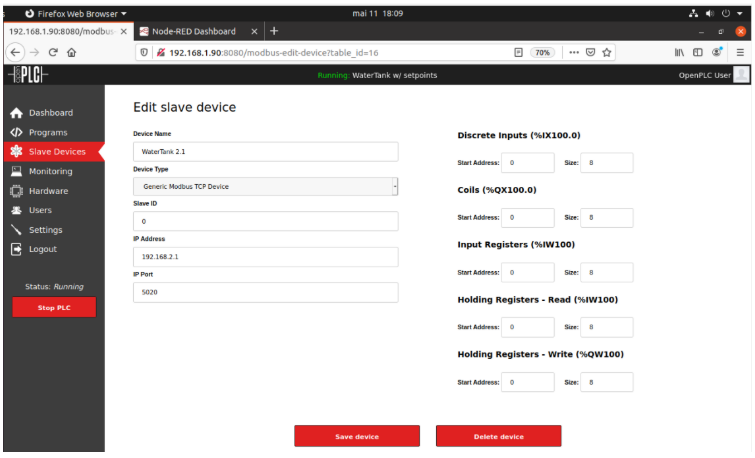
Appendix B



Software	Version	Extra information
GNS3	2.2.17	Python 3.6.8, Qt 5.12.1, PyQt 5.12
Snort	2.9.7.0	PCRE 8.39, ZLIB 1.2.11, libpcap 1.9.1
Elasticsearch	7.12.0	Kibana, Auditbeat, Metricbeat, Filebeat, Heartbeat and Packetbeat runs same version - 7.12.0
Node-RED	1.2.9	Node.js 12.20.2

Figure 2: Overview of different devices.

Appendix C



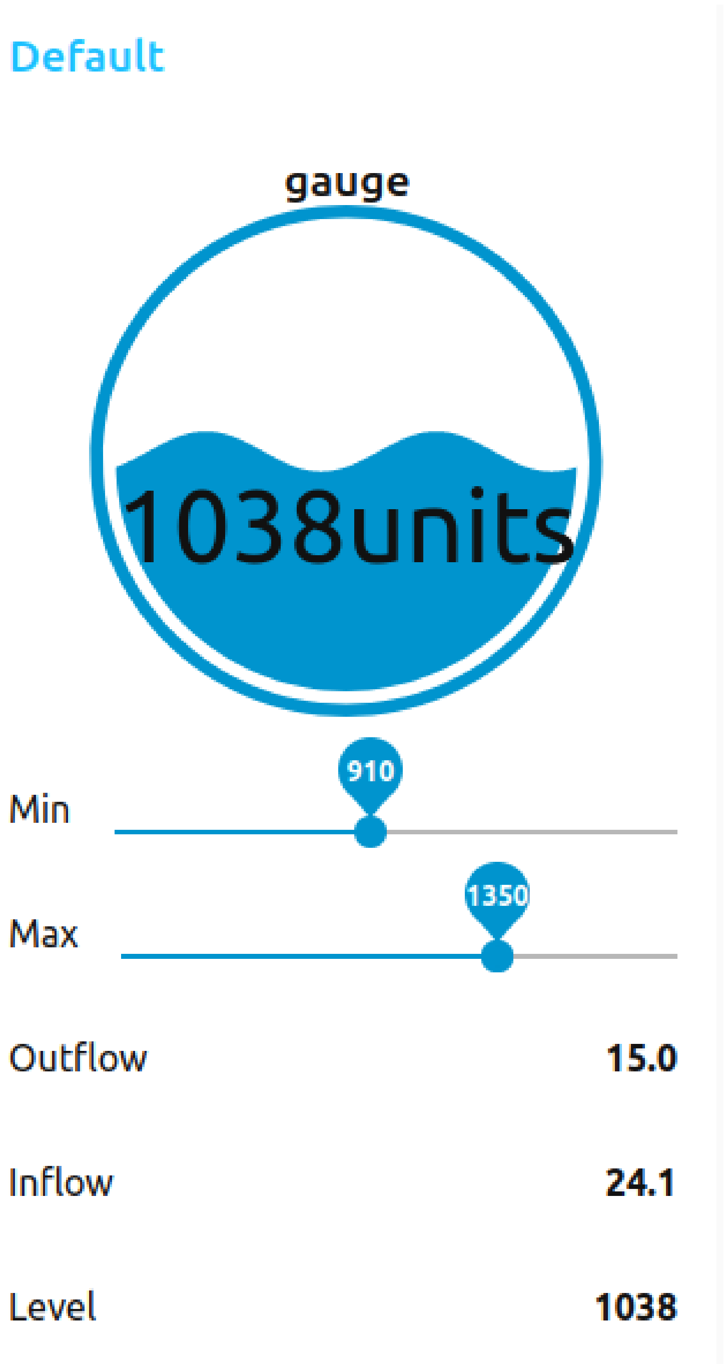
Edit slave device

Device Name: WaterTank 2.1
Device Type: Generic Modbus TCP Device
Slave ID: 0
IP Address: 192.168.2.1
IP Port: 5020

Discrete Inputs (%IX100.0) — Start Address: 0 Size: 8
Coils (%QX100.0) — Start Address: 0 Size: 8
Input Registers (%IW100) — Start Address: 0 Size: 8
Holding Registers - Read (%IW100) — Start Address: 0 Size: 8
Holding Registers - Write (%QW100) — Start Address: 0 Size: 8

Figure 3: Open PLC project GUI for adding the water tank as a slave device.

Appendix D


Default

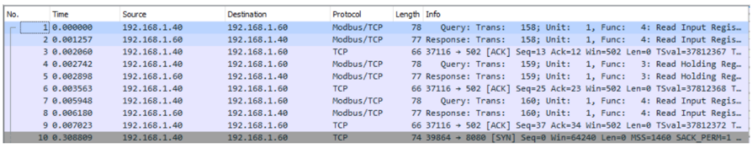
gauge

1038units

Min	910
Max	1350
Outflow	15.0
Inflow	24.1
Level	1038

Figure 4: GUI from Node-RED to select setpoints for minimum and maximum values, and show current value of the water tank.

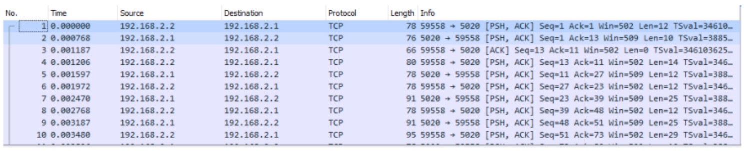
Appendix E



```
No.  Time      Source        Destination   Protocol    Length Info
   1 0.000000  192.168.1.40  192.168.1.60  Modbus/TCP      78    Query: Trans:   158; Unit:   1, Func:   4: Read Input Regis…
   2 0.001257  192.168.1.60  192.168.1.40  Modbus/TCP      77 Response: Trans:   158; Unit:   1, Func:   4: Read Input Regis…
   3 0.002060  192.168.1.40  192.168.1.60  TCP             66 37116 → 502 [ACK] Seq=13 Ack=12 Win=502 Len=0 TSval=37812367 T…
   4 0.002742  192.168.1.40  192.168.1.60  Modbus/TCP      78    Query: Trans:   159; Unit:   1, Func:   3: Read Holding Reg…
   5 0.002898  192.168.1.60  192.168.1.40  Modbus/TCP      77 Response: Trans:   159; Unit:   1, Func:   3: Read Holding Reg…
   6 0.003563  192.168.1.40  192.168.1.60  TCP             66 37116 → 502 [ACK] Seq=25 Ack=23 Win=502 Len=0 TSval=37812368 T…
   7 0.005948  192.168.1.40  192.168.1.60  Modbus/TCP      78    Query: Trans:   160; Unit:   1, Func:   4: Read Input Regis…
   8 0.006180  192.168.1.60  192.168.1.40  Modbus/TCP      77 Response: Trans:   160; Unit:   1, Func:   4: Read Input Regis…
   9 0.007023  192.168.1.40  192.168.1.60  TCP             66 37116 → 502 [ACK] Seq=37 Ack=34 Win=502 Len=0 TSval=37812372 T…
  10 0.308809  192.168.1.40  192.168.1.60  TCP             74 39864 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 …
```

Figure 5: Screenshot of WireShark-capture between the OpenPLC device and the rest of the network.

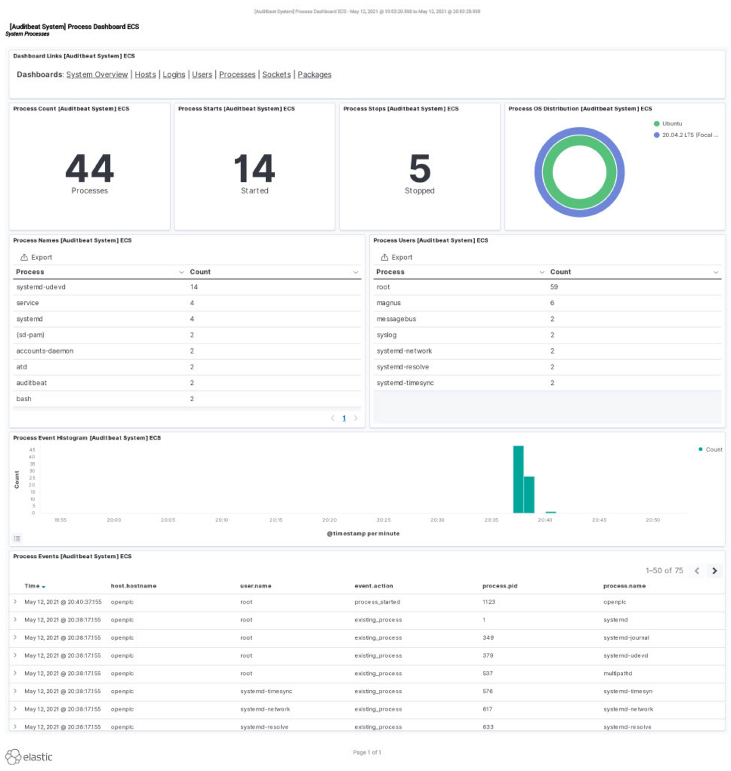
Appendix F



```
No.  Time      Source       Destination  Protocol  Length Info
   1 0.000000  192.168.2.2  192.168.2.1  TCP           78 59558 → 5020 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=12 TSval=34610…
   2 0.000768  192.168.2.1  192.168.2.2  TCP           76 5020 → 59558 [PSH, ACK] Seq=1 Ack=13 Win=509 Len=10 TSval=3885…
   3 0.001187  192.168.2.2  192.168.2.1  TCP           66 59558 → 5020 [ACK] Seq=13 Ack=11 Win=502 Len=0 TSval=346103625…
   4 0.001206  192.168.2.2  192.168.2.1  TCP           80 59558 → 5020 [PSH, ACK] Seq=13 Ack=11 Win=502 Len=14 TSval=346…
   5 0.001597  192.168.2.1  192.168.2.2  TCP           78 5020 → 59558 [PSH, ACK] Seq=11 Ack=27 Win=509 Len=12 TSval=388…
   6 0.001972  192.168.2.2  192.168.2.1  TCP           78 59558 → 5020 [PSH, ACK] Seq=27 Ack=23 Win=502 Len=12 TSval=346…
   7 0.002470  192.168.2.1  192.168.2.2  TCP           91 5020 → 59558 [PSH, ACK] Seq=23 Ack=39 Win=509 Len=25 TSval=388…
   8 0.002768  192.168.2.2  192.168.2.1  TCP           78 59558 → 5020 [PSH, ACK] Seq=39 Ack=48 Win=502 Len=12 TSval=346…
   9 0.003187  192.168.2.1  192.168.2.2  TCP           91 5020 → 59558 [PSH, ACK] Seq=48 Ack=51 Win=509 Len=25 TSval=388…
  10 0.003480  192.168.2.2  192.168.2.1  TCP           95 59558 → 5020 [PSH, ACK] Seq=51 Ack=73 Win=502 Len=29 TSval=346…
```

Figure 6: Screenshot of WireShark-capture between the OpenPLC device and the water tank device.

Appendix G



[Auditbeat System] Process Dashboard ECS - May 12, 2021 @ 19:53:20.598 to May 12, 2021 @ 20:53:20.599

[Auditbeat System] Process Dashboard ECS
System Processes

Dashboard Links [Auditbeat System] ECS

Dashboards: System Overview | Hosts | Logins | Users | Processes | Sockets | Packages

Process Count [Auditbeat System] ECS

44 Processes

Process Starts [Auditbeat System] ECS

14 Started

Process Stops [Auditbeat System] ECS

5 Stopped

Process OS Distribution [Auditbeat System] ECS

Process Names [Auditbeat System] ECS

Export

Process	Count
systemd-udevd	14
service	4
systemd	4
(sd-pam)	2
accounts-daemon	2
atd	2
auditbeat	2
bash	2

Process Users [Auditbeat System] ECS

Export

Process	Count
root	59
magnus	6
messagebus	2
syslog	2
systemd-network	2
systemd-resolve	2
systemd-timesync	2

Process Event Histogram [Auditbeat System] ECS

Process Events [Auditbeat System] ECS

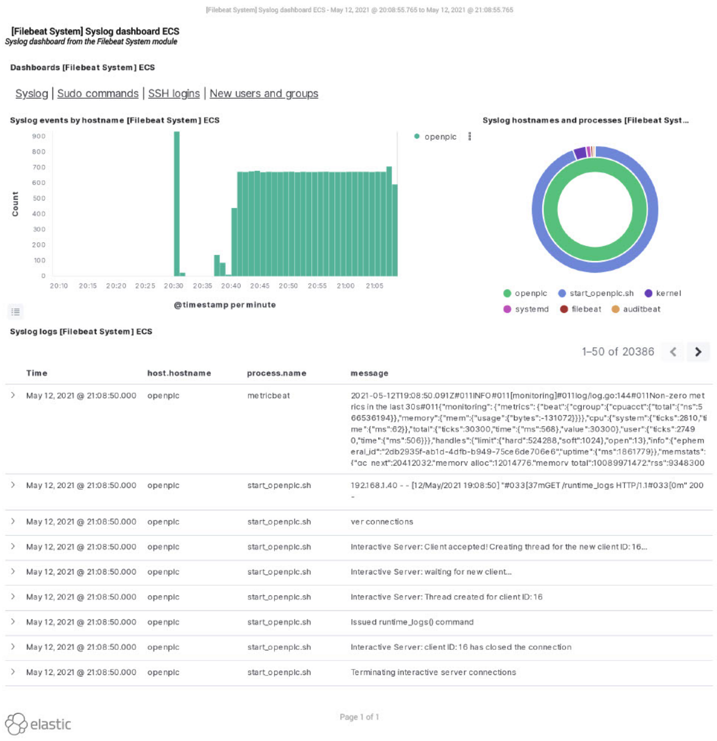
1–50 of 75

Time	host.hostname	user.name	event.action	process.pid	process.name
May 12, 2021 @ 20:40:37.155	openplc	root	process_started	1123	openplc
May 12, 2021 @ 20:38:17.155	openplc	root	existing_process	1	systemd
May 12, 2021 @ 20:38:17.155	openplc	root	existing_process	349	systemd-journal
May 12, 2021 @ 20:38:17.155	openplc	root	existing_process	379	systemd-udevd
May 12, 2021 @ 20:38:17.155	openplc	root	existing_process	537	multipathd
May 12, 2021 @ 20:38:17.155	openplc	systemd-timesync	existing_process	576	systemd-timesyn
May 12, 2021 @ 20:38:17.155	openplc	systemd-network	existing_process	617	systemd-network
May 12, 2021 @ 20:38:17.155	openplc	systemd-resolve	existing_process	633	systemd-resolve

elastic Page 1 of 1

Figure 7: Screenshot with example data collected by Auditbeat from the OpenPLC device shortly after starting up.

Appendix H



[Filebeat System] Syslog dashboard ECS - May 12, 2021 @ 20:08:55.765 to May 12, 2021 @ 21:08:55.765

[Filebeat System] Syslog dashboard ECS
Syslog dashboard from the Filebeat System module

Dashboards [Filebeat System] ECS

Syslog | Sudo commands | SSH logins | New users and groups

Syslog events by hostname [Filebeat System] ECS

Syslog hostnames and processes [Filebeat Syst...

● openplc ● start_openplc.sh ● kernel
● systemd ● filebeat ● auditbeat

Syslog logs [Filebeat System] ECS

1–50 of 20386

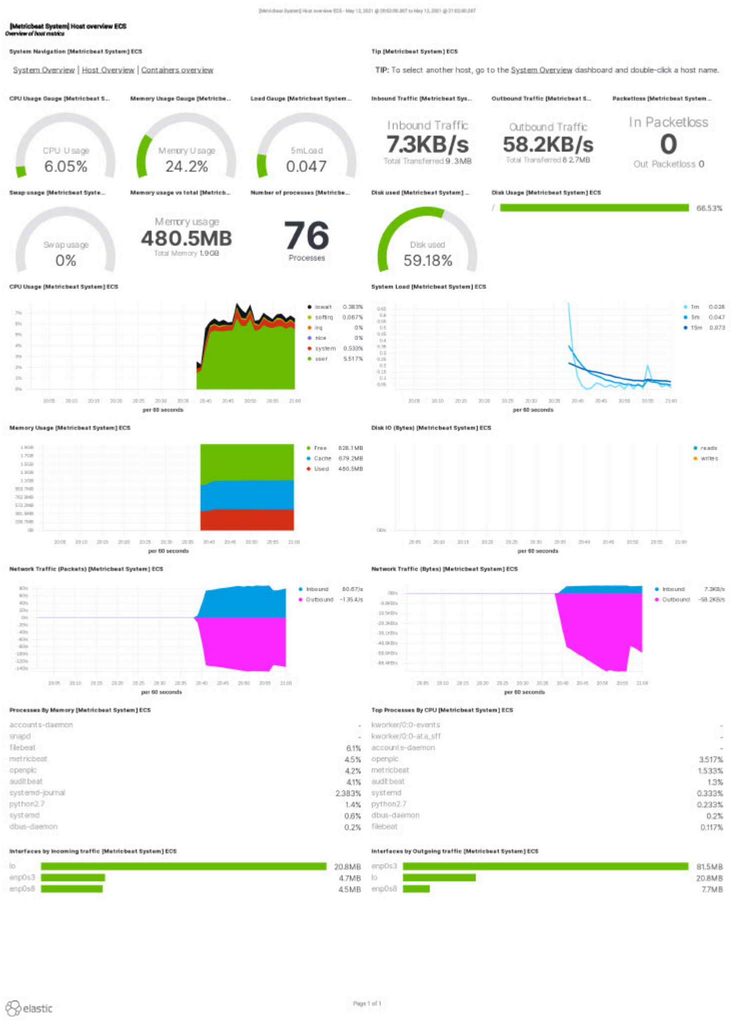
Time	host.hostname	process.name	message
May 12, 2021 @ 21:08:50.000	openplc	metricbeat	2021-05-12T19:08:50.091Z#011INFO#011[monitoring]#011log/log.go:144#011Non-zero metrics in the last 30s#011{"monitoring": {"metrics": {"beat":{"cgroup":{"cpuacct":{"total":{"ns":566536194}},"memory":{"mem":{"usage":{"bytes":-131072}}}},"cpu":{"system":{"ticks":2810,"time":{"ms":62}},"total":{"ticks":30300,"time":{"ms":568},"value":30300},"user":{"ticks":27490,"time":{"ms":506}}},"handles":{"limit":{"hard":524288,"soft":1024},"open":13},"info":{"ephemeral_id":"2db2935f-ab1d-4dfb-b949-75ce6de706e6","uptime":{"ms":1861779}},"memstats":{"gc_next":20412032,"memory_alloc":12014776,"memory_total":10089971472,"rss":9348300
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	192.168.1.40 - - [12/May/2021 19:08:50] "#033[37mGET /runtime_logs HTTP/1.1#033[0m" 200 -
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	ver connections
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Interactive Server: Client accepted! Creating thread for the new client ID: 16...
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Interactive Server: waiting for new client...
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Interactive Server: Thread created for client ID: 16
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Issued runtime_logs() command
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Interactive Server: client ID: 16 has closed the connection
May 12, 2021 @ 21:08:50.000	openplc	start_openplc.sh	Terminating interactive server connections

elastic

Page 1 of 1

Figure 8: Screenshot with example data collected by Filebeat form the OpenPLC device shortly after starting up.

Appendix I



[Metricbeat System] Host overview ECS - May 12, 2021 @ 20:02:00.397 to May 12, 2021 @ 21:02:00.397

[Metricbeat System] Host overview ECS
Overview of host metrics

System Navigation [Metricbeat System] ECS

System Overview | Host Overview | Containers overview

Tip [Metricbeat System] ECS

TIP: To select another host, go to the System Overview dashboard and double-click a host name.

CPU Usage Gauge [Metricbeat S... — CPU Usage 6.05%

Memory Usage Gauge [Metricbe... — Memory Usage 24.2%

Load Gauge [Metricbeat System... — 5mLoad 0.047

Inbound Traffic [Metricbeat Sys... — Inbound Traffic 7.3KB/s, Total Transferred 9.3MB

Outbound Traffic [Metricbeat S... — Outbound Traffic 58.2KB/s, Total Transferred 82.7MB

Packetloss [Metricbeat System... — In Packetloss 0, Out Packetloss 0

Swap usage [Metricbeat Syste... — Swap usage 0%

Memory usage vs total [Metricb... — Memory usage 480.5MB, Total Memory 1.9GB

Number of processes [Metricbe... — 76 Processes

Disk used [Metricbeat System] ... — Disk used 59.18%

Disk Usage [Metricbeat System] ECS — / 66.53%

CPU Usage [Metricbeat System] ECS

iowait	0.383%
softirq	0.067%
irq	0%
nice	0%
system	0.533%
user	5.517%

System Load [Metricbeat System] ECS

1m	0.038
5m	0.047
15m	0.073

Memory Usage [Metricbeat System] ECS

Free	828.1MB
Cache	679.2MB
Used	480.5MB

Disk IO (Bytes) [Metricbeat System] ECS — reads, writes

Network Traffic (Packets) [Metricbeat System] ECS

Inbound	80.67/s
Outbound	-135.4/s

Network Traffic (Bytes) [Metricbeat System] ECS

Inbound	7.3KB/s
Outbound	-58.2KB/s

Processes By Memory [Metricbeat System] ECS

accounts-daemon	-
snapd	-
filebeat	6.1%
metricbeat	4.5%
openplc	4.2%
auditbeat	4.1%
systemd-journal	2.383%
python2.7	1.4%
systemd	0.6%
dbus-daemon	0.2%

Top Processes By CPU [Metricbeat System] ECS

kworker/0:0-events	-
kworker/0:0-ata_sff	-
accounts-daemon	-
openplc	3.517%
metricbeat	1.533%
auditbeat	1.3%
systemd	0.333%
python2.7	0.233%
dbus-daemon	0.2%
filebeat	0.117%

Interfaces by Incoming traffic [Metricbeat System] ECS

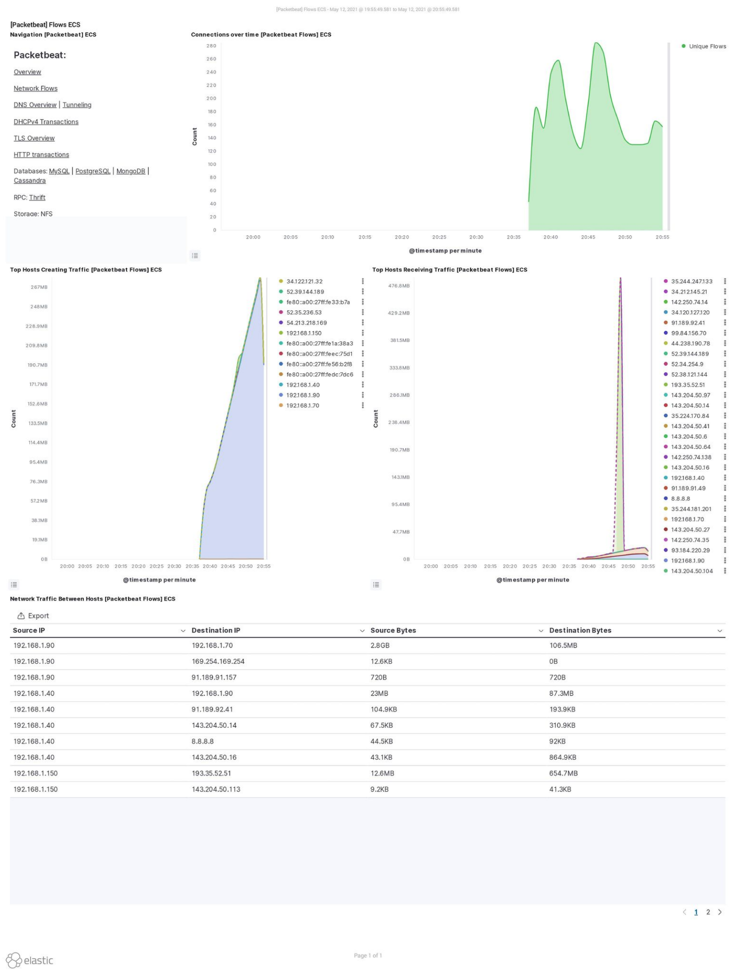
lo	20.8MB
enp0s3	4.7MB
enp0s8	4.5MB

Interfaces by Outgoing traffic [Metricbeat System] ECS

enp0s3	81.5MB
lo	20.8MB
enp0s8	7.7MB

Figure 9: Screenshot with example data collected by Metricbeat from the OpenPLC device shortly after starting up.

Appendix J



[Packetbeat] Flows ECS

Navigation [Packetbeat] ECS

Packetbeat:

Overview

Network Flows

DNS Overview | Tunneling

DHCPv4 Transactions

TLS Overview

HTTP transactions

Databases: MySQL | PostgreSQL | MongoDB | Cassandra

RPC: Thrift

Storage: NFS

Network Traffic Between Hosts [Packetbeat Flows] ECS

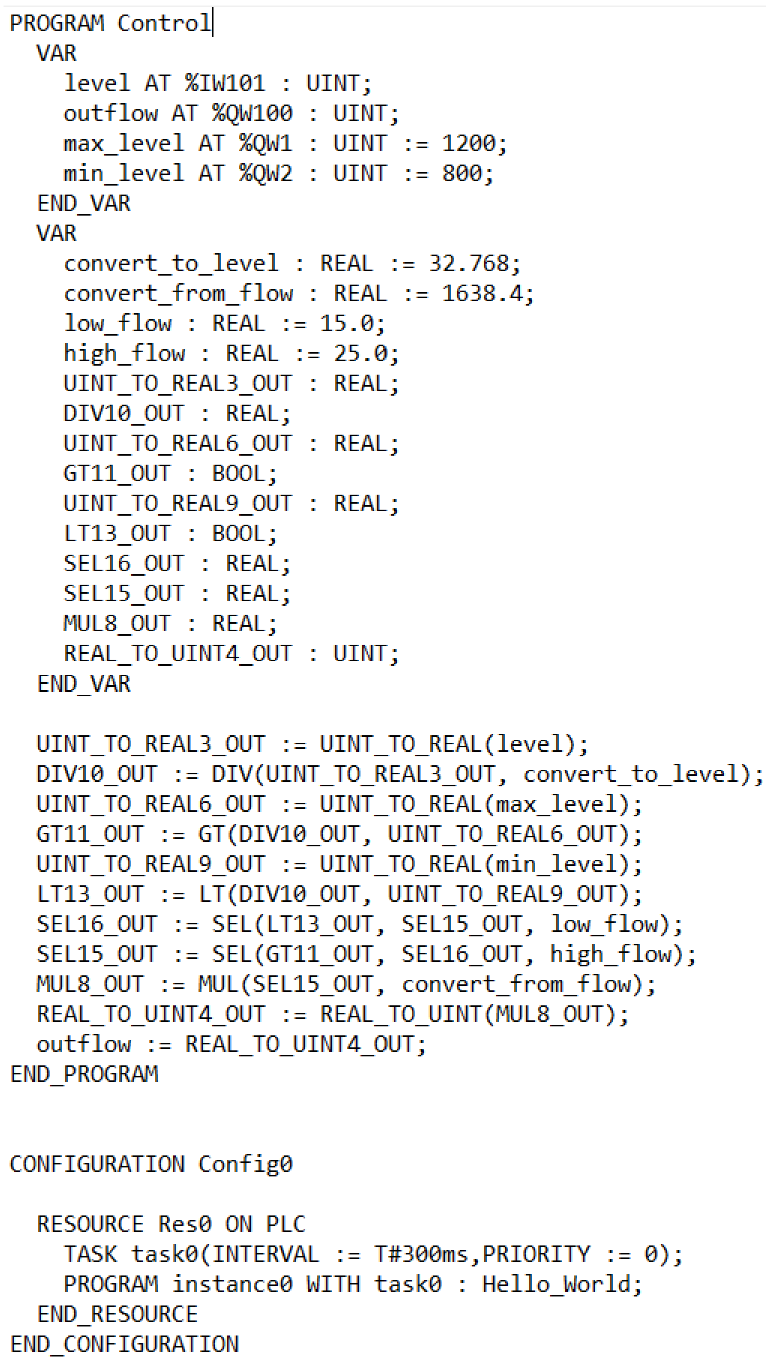
Export

Source IP	Destination IP	Source Bytes	Destination Bytes
192.168.1.90	192.168.1.70	2.8GB	106.5MB
192.168.1.90	169.254.169.254	12.6KB	0B
192.168.1.90	91.189.91.157	720B	720B
192.168.1.40	192.168.1.90	23MB	87.3MB
192.168.1.40	91.189.92.41	104.9KB	193.9KB
192.168.1.40	143.204.50.14	67.5KB	310.9KB
192.168.1.40	8.8.8.8	44.5KB	92KB
192.168.1.40	143.204.50.16	43.1KB	864.9KB
192.168.1.150	193.35.52.51	12.6MB	654.7MB
192.168.1.150	143.204.50.113	9.2KB	41.3KB

1 2

Figure 10: Screenshot with example data collected by Packetbeat from the SPAN port on OpenVSwitch after starting up.

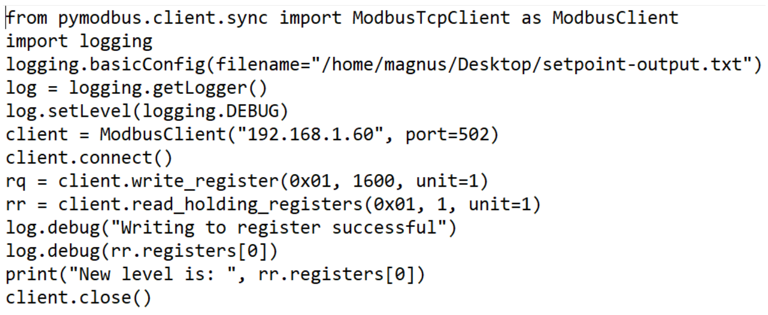
Appendix K



```
PROGRAM Control
  VAR
    level AT %IW101 : UINT;
    outflow AT %QW100 : UINT;
    max_level AT %QW1 : UINT := 1200;
    min_level AT %QW2 : UINT := 800;
  END_VAR
  VAR
    convert_to_level : REAL := 32.768;
    convert_from_flow : REAL := 1638.4;
    low_flow : REAL := 15.0;
    high_flow : REAL := 25.0;
    UINT_TO_REAL3_OUT : REAL;
    DIV10_OUT : REAL;
    UINT_TO_REAL6_OUT : REAL;
    GT11_OUT : BOOL;
    UINT_TO_REAL9_OUT : REAL;
    LT13_OUT : BOOL;
    SEL16_OUT : REAL;
    SEL15_OUT : REAL;
    MUL8_OUT : REAL;
    REAL_TO_UINT4_OUT : UINT;
  END_VAR

  UINT_TO_REAL3_OUT := UINT_TO_REAL(level);
  DIV10_OUT := DIV(UINT_TO_REAL3_OUT, convert_to_level);
  UINT_TO_REAL6_OUT := UINT_TO_REAL(max_level);
  GT11_OUT := GT(DIV10_OUT, UINT_TO_REAL6_OUT);
  UINT_TO_REAL9_OUT := UINT_TO_REAL(min_level);
  LT13_OUT := LT(DIV10_OUT, UINT_TO_REAL9_OUT);
  SEL16_OUT := SEL(LT13_OUT, SEL15_OUT, low_flow);
  SEL15_OUT := SEL(GT11_OUT, SEL16_OUT, high_flow);
  MUL8_OUT := MUL(SEL15_OUT, convert_from_flow);
  REAL_TO_UINT4_OUT := REAL_TO_UINT(MUL8_OUT);
  outflow := REAL_TO_UINT4_OUT;
END_PROGRAM


CONFIGURATION Config0

  RESOURCE Res0 ON PLC
    TASK task0(INTERVAL := T#300ms,PRIORITY := 0);
    PROGRAM instance0 WITH task0 : Hello_World;
  END_RESOURCE
END_CONFIGURATION
```

Figure 11: Structured Text program used as input for the PLC.

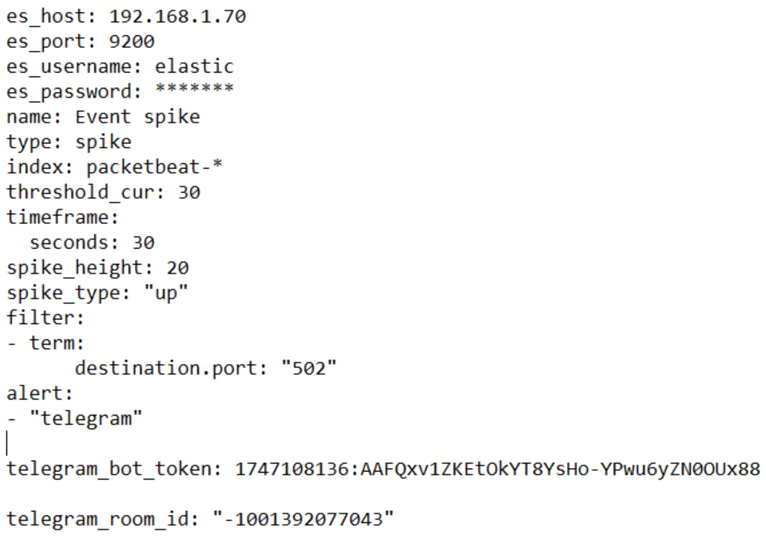
Appendix L



```
from pymodbus.client.sync import ModbusTcpClient as ModbusClient
import logging
logging.basicConfig(filename="/home/magnus/Desktop/setpoint-output.txt")
log = logging.getLogger()
log.setLevel(logging.DEBUG)
client = ModbusClient("192.168.1.60", port=502)
client.connect()
rq = client.write_register(0x01, 1600, unit=1)
rr = client.read_holding_registers(0x01, 1, unit=1)
log.debug("Writing to register successful")
log.debug(rr.registers[0])
print("New level is: ", rr.registers[0])
client.close()
```

Figure 12: Script for executing set point attack on the water tank.

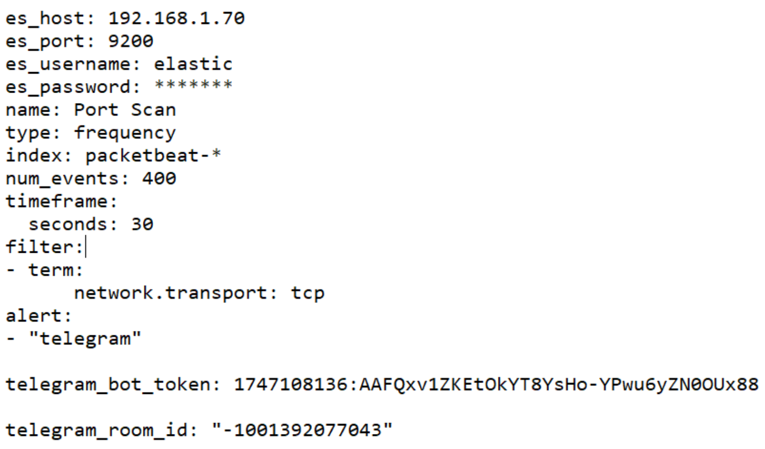
Appendix M



```
es_host: 192.168.1.70
es_port: 9200
es_username: elastic
es_password: *******
name: Event spike
type: spike
index: packetbeat-*
threshold_cur: 30
timeframe:
  seconds: 30
spike_height: 20
spike_type: "up"
filter:
- term:
      destination.port: "502"
alert:
- "telegram"

telegram_bot_token: 1747108136:AAFQxv1ZKEtOkYT8YsHo-YPwu6yZN0OUx88

telegram_room_id: "-1001392077043"
```

Figure 13: Rule for ElastAlert to alert on Port Scan attacks.

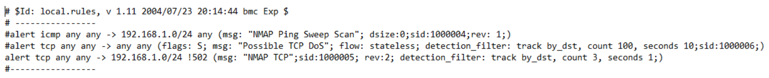
Appendix N



```
es_host: 192.168.1.70
es_port: 9200
es_username: elastic
es_password: *******
name: Port Scan
type: frequency
index: packetbeat-*
num_events: 400
timeframe:
  seconds: 30
filter:
- term:
      network.transport: tcp
alert:
- "telegram"

telegram_bot_token: 1747108136:AAFQxv1ZKEtOkYT8YsHo-YPwu6yZN0OUx88

telegram_room_id: "-1001392077043"
```

Figure 14: Rule for ElastAlert to alert on Port Scan attacks.

Appendix O



```
# $Id: local.rules, v 1.11 2004/07/23 20:14:44 bmc Exp $
# ----------------
#alert icmp any any -> 192.168.1.0/24 any (msg: "NMAP Ping Sweep Scan"; dsize:0;sid:1000004;rev: 1;)
#alert tcp any any -> any any (flags: S; msg: "Possible TCP DoS"; flow: stateless; detection_filter: track by_dst, count 100, seconds 10;sid:1000006;)
alert tcp any any -> 192.168.1.0/24 !502 (msg: "NMAP TCP";sid:1000005; rev:2; detection_filter: track by_dst, count 3, seconds 1;)
#-----------------
```

Figure 15: Rulefile in Snort with rules for DoS and Port Scan attacks.