Ref. Ares(2020)4078229 - 03/08/2020

SDN-μSense

Project No. 833955

Project acronym: SDN-microSENSE

Project title:

SDN - microgrid reSilient Electrical eNergy SystEm

Deliverable D3.3
EPES Honeypots

Programme: H2020-SU-DS-2018
Start date of project: 01.05.2019
Duration: 36 **months**

Editor: TECNALIA

Due date of deliverable: 31/07/2020 Actual submission date: 31/07/2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833955

Deliverable Description:

Deliverable Name	EPES Honeypots
Deliverable Number	D3.3
Work Package	WP3
Associated Task	T3.3
Covered Period	M1-M15
Due Date	M15
Completion Date	M15
Submission Date	31/07/2020
Deliverable Lead Partner	TECNALIA
Deliverable Author(s)	Marisa Escalante (TECNALIA) Xabier Yurrebaso (TECNALIA) Santiago de Diego (TECNALIA) Alberto Molinuevo (TECNALIA) Vasilis Machamint (8BELLS) Ruben Trapero (ATOS) Preetika Srivastava (CLS) Alba Collet (IREC) Elisavet Grigoriou (SIDROCO) Anastasios Lytos (SIDROCO) Panagiotis I. Radoglou (UOWM) Thomas Lagkas (UOWM)
Version	1.0

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

CHANGE CONTROL

DOCUMENT HISTORY

Version	Date	Change History	Author(s)	Organisation
0.0	15/05/2020	ToC	Marisa Escalante	TECNALIA
0.1	26/05/2020	ToC ready for review	Marisa Escalante and Panagiotis Radoglou	TECNALIA & UOWM
0.2	22/06/2020	First iteration of contributions	Marisa Escalante Xabier Yurrebaso Vasilis Machamint Ruben Trapero Preetika Srivastava	TECNALIA 8BELLS ATOS CLS

0.3	25/06/2020	Added contributions	Marisa Escalante Xabier Yurrebaso	TECNALIA
V0.4	26/06/2020	Added contributions	Marisa Escalante Xabier Yurrebaso Santiago de Diego Alberto Molinuevo Vasilis Machamint Alba Collet	TECNALIA 8BELLS Irec
V0.5	01/07/2020	Integration of all the contributions. Some information is missing	Marisa Escalante Xabier Yurrebaso Santiago de Diego Alberto Molinuevo Elisavet Grigoriou Anastasios Lytos Panagiotis I. Radoglou-GrammatikisThomas Lagkas	TECNALIA SIDROCO UOWM
V0.6	03/07/2020	Version ready for internal review	Marisa Escalante Xabier Yurrebaso Santiago de Diego Alberto Molinuevo Vasilis Machamint Ruben Trapero Preetika Srivastava Alba Collet Elisavet Grigoriou Anastasios Lytos Panagiotis I. Radoglou Thomas Lagkas	TECNALIA 8Bells ATOS CLS IREC SIDROCO UOWM
V07	16/07/2020	Version with the comments from reviewers	Marisa Escalante Xabier Yurrebaso Orestis Mavropoulos Elisavet Grigoriou Alba Collet	TECNALIA IREC CLS
V0.8	23/07/2020	Version including unit testing	Marisa Escalante Xabier Yurrebaso Santiago de Diego Alberto Molinuevo Elisavet Grigoriou Anastasios Lytos Panagiotis I. Radoglou Thomas Lagkas	TECNALIA SIDROCO UOWM
V1.0	30/07/2020	Final Version	Marisa Escalante	TECNALIA

DISTRIBUTION LIST

SAB APPROVAL

NAME	INSTITUTION	DATE
PROF. STEPHEN D. WOLTHUSEN	NTNU	22/07/2020

ACADEMIC AND INDUSTRIAL PARTNER REVISION

NAME	INSTITUTION	DATE
DR. DAVE RAGGETT	ERCIM (ACADEMIC)	06/07/2020
HANS CHRISTIAN BOLSTAD	SINTEF (ACADEMIC)	10/07/2020
GIANNIS LEDAKIS	UBITECH (INDUSTRIAL)	10/07/2020

TECHNICAL MANAGER REVISION

NAME	INSTITUTION	DATE
ANASTASIOS DROSOU	CERTH	06/07/2020

QUALITY MANAGER REVISION

NAME	INSTITUTION	DATE
DIMOSTHENIS IOANNIDIS	CERTH	06/07/2020

Date	Issue	Group
03/07/2020	Revision	CERTH ERIC UBITECH AYESA
29/07/2020	Acceptance	CERTH AYESA SAB
30/07/2020	Submission	AYESA

Table of contents

List of Figures

List of Tables

Acronyms

Acronym	Explanation
API	Applications Programming Interface
BACnet	Building Automation and Control Networks
CAPEC	Common Attack Pattern Enumeration and Classification
EPES	Electrical Power and Energy Systems
FTP	File Transfer Protocol
FTP	File Transfer Protocol
HTTP	Hyper-text Transfer Protocol
HTTPS	Secure HTTP
ICD	IED Capability Description
ICS	Industrial Control System
ICT	Information and Communication Technologies
IDPS	Intrusion Detection and Prevention System
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IP	Internet Protocol
IPMI	Intelligent Platform Management Interface
JSON	JavaScript Object Notation
ML	Machine Learning
NBI	North-bound Interface
PLC	Programmable Logic Controller
RAF	Risk Assessment Framework
REST	Representational State Transfer
RTU	Remote Terminal Unit
SBI	South-bound Interface
SCADA	Supervisory Control and Data Acquisition
SDN	Software Defined Networks
SIEM	Security Information and Event Management
SLD	Single Line Diagram
SMS	SDN-microSENSE
SNMP	Simple Network Management Protocol
S-RAF	SDN-microSENSE Risk Assessment Framework
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transfer Control Protocol
UA	Unknown Anomaly
UDP	User datagram protocol
VM	Virtual Machine
VPN	Virtual Private Network
XL-EPDS	Cross-layer Energy Protection and Detection System
XL-SIEM	Cross-layer SIEM

XML	Extensible Mark-up Language

Executive Summary

Deception technologies such as honeypots are being used as part of al cyber defence strategy of an organization in order to protect critical assets in the system. The greatest impact of the deception mechanisms is at the detection area. Since it is very difficult to detect zero-day and advanced attacks against production systems, the honeypots can simplify the detection process.

Honeypots comprise an additional layer of security in the SDN-microSENSE architecture. As internet-connected energy elements become more pervasive, enabling remote monitoring of elements, cyber-criminals are finding new ways to breach energy and power networks through advanced malware. The attack surface of the energy system expands when interconnected devices, such as IEDs, infrastructure and applications, are permitted to connect to the EPES network. The SDN-microSENSE honeypots are software applications designed to be attacked by hackers in order to provide them configuration information and try to keep them busy, while security auditors try to collect information about the attack practices. Honeypots assist in the development of a more holistic data collection and analysis process, since they go beyond 'traditional' network data, used as forensics, to a richer set of evidence based on the execution of actual attacks.

This document presents the honeypots (IEC 61850, IEC60870-5-104, and Modbus) and the Honeypot Manager. The honeypots, in general, play the role of deception technology within an overall cyber defence strategy of an organization, and specifically the SDN-microSENSE honeypots emulate the most relevant protocols in the Electrical Power and Energy Systems (EPES). On the other hand, the Honeypot Manager provides two main functionalities: (i) supports the automatic deployment of the different types of honeypots; and (ii) indicates to the SDN controller the required information to re-arrange the honeypots network in order to collect information for unknown anomalies.

All mentioned components have been implemented as part of T3.3 and are documented in this deliverable. Specifically, the implemented components are: (i) IEC 61850 honeypot (ii) IEC60870-5-104 honeypot (iii) Modbus honeypot and (iv) the Honeypot Manager. The functional and architectural description as well as the interface model and the deployment details are presented in the section 5 of this deliverable. This deliverable also presents a state of the art of the honeypots and the added value provided by the SDN-microSENSE developments (Section 2 and Section 3).

1 Introduction

1.1 Purpose of the document

The objective of this deliverable is to present the development and technical details of the honeypots developed during the Task 3.3: IEC 61850, IEC60870-5-104 and Modbus. The main function of honeypots in the context of the SDN-microSENSE project is to serve as a decoy mechanism to possible attackers, facilitating early detection of threats and attack patterns, and the mitigation of potential risks. The information that the honeypots capture is sent to the XL-SIEM and it is a complementary information source, usable by the Intrusion Detection and Prevention Systems (IDPS) and the risk assessment components. Honeypots can capture details on an attacker's activity on the system, which also makes it possible to learn about these attacks, which may indicate potential vulnerabilities in the real system. Insofar as all incoming activities in honeypots may be regarded as suspicious,

In the task 3.3, the Honeypot Manager is also developed. This Honeypot Manager has two main functions. On the one hand, the Honeypot Manager is responsible for processing the information about unknown anomalies (like zero-day attacks) and to indicate to the SDN controller the required information to re-arrange the honeypots network in order to collect information for unknown anomalies detected by the ML models of the XL-EPDS. On the other hand, the Honeypot Manager is responsible for automating the deployment and configuration of honeypots in the virtual machines.

The document is accompanied by the actual implementation of the following tools: (i) IEC 61850 honeypot, (ii) IEC60870-5-104 honeypot, (iii) Modbus honeypot and, (iv) the Honeypot Manager, in the form of software prototypes that are available as explained in Section 5.

1.2 Structure of the document

D3.3 is divided into the following sections:

- Section 1 is the introductory part of the report and gives the objective and the relation of the work done in this task with the other WPs.
- Section 2 provides a general vision of honeypots and how the honeypots can be classified based on their characteristics.
- Section 3 provides a State of the Art of the honeypots available in the EPES environments. This section also indicates those honeypots developed in different EU projects and finally the added value of the work done in SDN-microSENSE project
- Section 4 contains the analysis of the specifications related to the EPES honeypots defined in the deliverable D2.3 and how these specifications are solved with the work done in the task 3.3
- Section 5 contains one subsection that presents how the honeypots and Honeypot Manager fit in the overall architecture of SDN-microSENSE and the following subsections present the details of functional description, the prototype architecture, the interface model and the deployment details for each of the developed honeypots and the Honeypot Manager.
- Section 6 describes the unit testing related to the honeypots and related components.
- Section 7 concludes the document summarising the document contents and identifying the main innovations.

1.3 Relation of this deliverable with other SDN microSENSE tasks

Figure 1 depicts the relationships of the deliverable to deliverable to the other deliverables of the project .

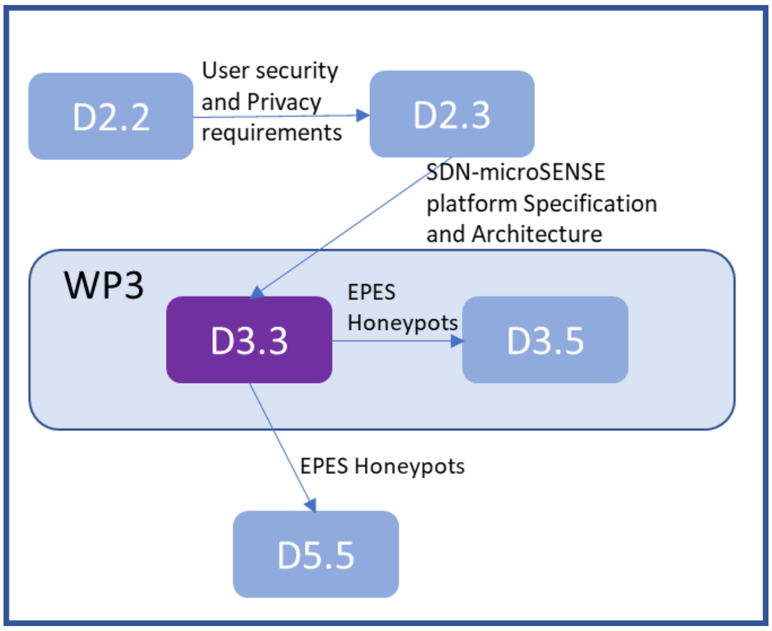

D2.2
User security and Privacy requirements
D2.3
SDN-microSENSE platform Specification and Architecture
WP3
D3.3
EPES Honeypots
D3.5
EPES Honeypots
D5.5

Figure 1. Deliverable D3.3 relationships within other SDN-Microsense Deliverables

This deliverable D3.3 collects the specifications defined for the honeypot in the deliverable D2.3 [SMS20-D23], that are based in the requirements expressed in the D2.2. How the honeypots and the Honeypot Manager developed in this task cover the specifications is explained in section 4.

The deliverable D3.3 is used to complete the D3.5 “SDN-microSENSE Risk assessment framework” because honeypots and the Honeypot Manager are part of the S-RAF. In addition, this deliverable D3.3 is used by the Deliverable D5.1 because provides information about the logs collected by the honeypots.

Section 4 explains which and how specifications of the D2.3 are covered by the work done in task 3.3 and in Section 5.1, it is explained how the components developed in the task T3.3 fit in the general SDN microSENSE architecture and the interfaces used.

2 Honeypots Overview

2.1 Honeypot Definition

Honeypots serve various purposes and have diverse capabilities, although they are commonly defined as "an information system resource whose value lies in unauthorized or illicit use of that resource" [SPI-03]. A honeypot is a computer system that is set up to act as a decoy to lure cyber-attackers, to detect and learn about new cyber threats and attack patterns and to improve the cyber security strategy of the organization [HDEF].

Security personal often use honeypots as a tool to gather intelligent on the attacker. Attackers constantly modify their methods to take advantage of different types of attacks. In some cases, it can be seen attackers using zero-day attacks against honeypot servers. If the security operator/administrator does not configure the honeypot properly, it might appear suspicious to an experienced attacker and simply avoid it. The honeypot supports the security team to understand the attacker's methodologies, learn more about known and unknown attacks and in this way to better protect the real production systems. Thus, making honeypots a very useful part of the defence system. [MAS19]. Based on this, some advantages and disadvantages of the honeypots could be identified:

Advantages:

- Attackers can be can observed in action and learn about their behaviour.
- Knowing the types of attacks being used supports the Security team in the implementation of the defences needed to be installed to protect your real systems and data from attack.
- To attack on a honeypot is likely to frustrate attackers and stop them from hacking your real computer systems.

Disadvantages[MAS19] [TIT15]:

- The capture of data takes place only when the attacker is attacking the system actively.
- If there is an attack occurring in another system, the honeypot will not be able to identify it.
- It is easy for an experienced attacker to understand if he is attacking a honeypot system or a real system. Additionally, more attacks are automated, and these automated threats will scan, attack, and exploit any device that you might find vulnerable.
- Honeypots add complexity to a network, and the more complex a network is, the harder it is to secure. The honeypot could introduce vulnerabilities that could be exploited to gain access to real systems and data.
- An attacker may use the own honeypots to distract, exploiting it as a zombie. Thus, giving him the opportunity to attack other systems within the network and compromising them.

Depending on the type of the honeypots used the advantages and disadvantages explain above can be appeared or not. In the following sections, different classifications of honeypot are presented, each security responsible of the system should choose the better type of honeypots depending on his system and need.

2.2 Honeypot classifications

The honeypots can be classified according to their physicality, operation field, location and level of interaction, as shown in Figure 2.

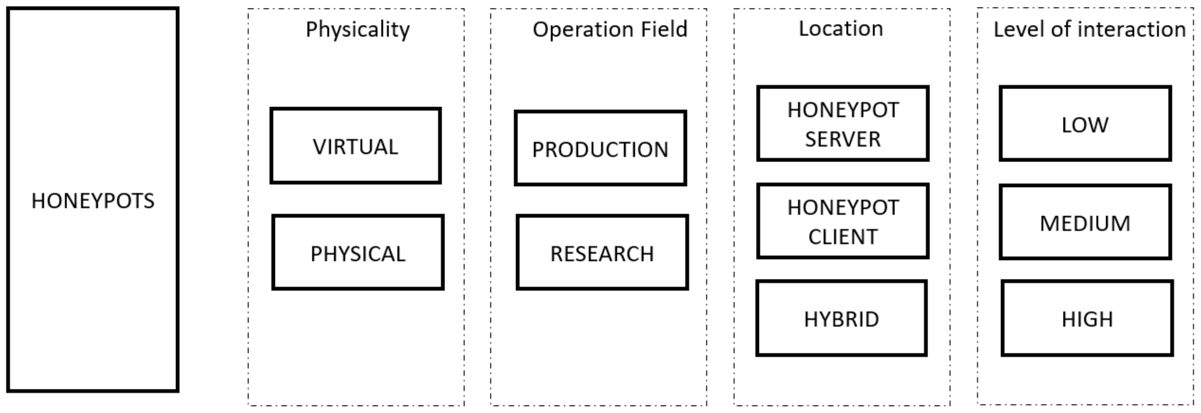


Figure 2. Honeypots classification

In the following sections detailed explanation of this classification is provided.

2.2.1 Operation field

Honeypots can be deployed and used for two different purposes [MCG+17] [NWS+16]: production and research. The purpose of a production honeypot is to assist in mitigating risk in an organization by adding value to the security measures implemented. Instead, when a honeypot is deployed for research purposes, it is designed to gather useful information for the community in order to generate intelligence about threats and attacks, with the ultimate goal of improving the protection of the companies' systems.

2.2.1.1 Research Honeypots

One of the biggest challenges that the security community faces is the lack of information about the enemy, information such as: who the threat is, why they attack, how they attack and when they attack. These kinds of questions are the ones that the security community often cannot answer. To defeat a threat, you first have to know about it. However, in the world of information security we have little such information.

Honeypots can increase the value of research by providing a platform to study the threat [SR01]. Research honeypots provide comprehensive information about attacks but are difficult to deploy. The objective is to keep them functioning with the highest level of risk associated with them, in order to expose them to a wider range of attacks and compromised situations. They are often used by research organizations and network forensic scientists to examine attacks and develop comprehensive countermeasures against threats. The purpose of this kind of honeypots is to gather a large volume of statistical and event data, most of them being used only for research purposes. They do not directly protect an organization, but they help to understand threats, develop counter measures, and address exploitable breaches, thus indirectly contributing to security. In addition, research honeypots are excellent tools for capturing automated attacks. Since these attacks are targeted at entire blocks of the network, research honeypots can quickly capture these attacks for analysis.

Learning from a research honeypot can be applied to improve the prevention, detection or reaction to an attack. However, research honeypots do not contribute significantly to security of an organization. If an organization is trying to improve the security of its production environment, it might want to consider using production honeypots, as they are easy to implement and maintain. However, if an organization, such as a university, government or large company, is interested in learning more about threat research, honeypots would be a useful research tool.

In summary, research honeypots have 3 main functions to perform the role for which they are deployed: 1) help to understand threats, develop countermeasures and address exploitable breaches, 2) capture automated attacks and 3) generate intelligence regarding malicious attacks.

Honeypots as Sensors

Honeypots can be considered as sensors, since they can gather valuable data about the attacks they suffer. This is particularly useful for detecting weaknesses and vulnerabilities in system design, as data captured from attacks can be analysed to understand the attacker's behaviour and strategies, as well as their motivations.

A number of critical aspects in the design of the honeypot itself is involved here. On the one hand, if the goal is to be able to detect and analyse the vulnerabilities and weaknesses of the services/components that will be part of the system in production, it must be exactly the same as the exposed service/system, but securing all the necessary logic of the honeypot, so that the attacker does not detect the deception. Moreover, if the purpose is to study the behaviour of the attacker (tactics and strategies), the honeypot will present certain vulnerabilities, specially designed to be exploited by the attacker.

Honeypots as Collector of Valuable Data

Research honeypots can be considered as a data collection system, where these data will not be contaminated by the noise introduced by production activities and which are usually of great value. This reduces the size of the data sets and makes data analysis less complex. Regardless of the workload, the Honeypots only need to process the traffic that goes to them or originates from them.

If the honeypots are of medium or high interaction, the volume of information increases, and the complexity associated with its analysis also increases. For example, a honeypot designed to detect a zero-day-exploit captures everything that is used against them, this means that it will be able to identify unknown strategies and zero-day-exploits. How to capture and store data in a misleading way is an eternal problem, where each author develops a different solution.

Honeypots as Tactics and Strategies Modelling

If we focus on the last stage that provides a research honeypot, we can consider it as a modelling system of tactics and strategies. In the design of the research honeypot, one of the main objectives is to provide the attacker with great flexibility, so that he can perform the required actions during his attack. A subsequent analysis of these actions will allow us to identify the tactics and strategies used.

Given the large volume of information collected, the innate complexity of today's systems, as well as the tactics and strategies used by the attacker to avoid detection, it is necessary to apply techniques and algorithms based on artificial intelligence or statistical analysis, so that we can generate the knowledge we are looking for in relation to the attack. Automating this process is one of the open challenges today.

2.2.1.2 Production Honeypots [ABH03]

A second type of honeypots in relation to their purpose, are the honeypots that are deployed in production, having an active part in defence functions of the global cyber security of an organization.

The production honeypots are characterized by their simplicity of deployment and operation and the amount of information gathered, being closely monitored and maintained, having an important part in the Information Security Management System (ISMS). Their purpose is to achieve a higher level of security in the network of a specific company by redirecting attacks.

A honeypot includes two essential elements, decoys and security programs. The decoy may be any type of information system resource, and the security program facilitates security-related functions such as attack monitoring, prevention, detection, response and profiling. In addition, the security program must be run stealthily to avoid detection.

Honeypots as Decoys [FDF+18]

A honeypot, a system designed to be compromised, will not help to keep attackers out. What will keep the attackers out are good security practices, such as disabling unnecessary or unsecured services, patching services that are required, and using strong authentication mechanisms. In fact, if a honeypot is implemented incorrectly, it can make it easier for an attacker to get in.

Honeypot is generally a "decoy" system designed to attract cyber-attackers in the interest of early detection of attack attempts, slowing down the attacker's actions and mitigating them. It is also responsible for gathering traces of real-world attacks to learn attack vectors and design cyber security systems. One of the most interesting functions that a honeypot can assume is to deceive the attacker in order to dissuade him from his goal. The idea is that the attackers will spend time and resources to attack the honeypots instead of attacking the production systems. In order to be effective, the honeypot must seem to be exactly what the attacker is looking for. When the honeypot is attacked, it can immediately notify system administrators of the presence of an intruder on the network, allowing them to take the appropriate course of action to secure it.

The most important thing, perhaps, is that this deception process will fail against the most common of attacks: automated tools and worms. These days, more and more attacks are automated. These automated threats will scan, attack, and exploit any device that you might find vulnerable.

Honeypots as Active Network Defence

Firewalls are often deployed around an organization's perimeter to block unauthorized access by filtering certain ports and content, but they are not used to assess traffic. They can block all access to a certain service in order to prevent malicious traffic, but this also blocks any benign traffic that wants to access the service. In contrast, honeypots are designed to open ports to capture as many attacks as possible for subsequent data analysis.

Although not usually considered, honeypots also provide added value to the reaction. Often, when an organization is compromised, the Incident Response Team cannot determine what has actually happened if users and systems have contaminated the data collected. Clean evidence is much more difficult to gather in this type of context; the deployment of a production honeypot facilitates this process of gathering evidence that is valid for later analysis. Another challenge faced by many organizations is that compromised systems cannot be removed from the network after an incident. As a result, incident response teams are unable to perform adequate or complete forensic analysis. A deployment of a production honeypot facilitates these processes.

Honeypots as Intrusion Detector

Intrusion detection systems are designed to detect attacks. However, IDS administrators can be overwhelmed with false positives, alerts that are erroneously generated when a sensor senses and alerts on an attack that is actually valid traffic. False positives are dangerous because system administrators receive so many alerts that they get accustomed to them, as they are falsely alerted day after day. If false positives are not effectively reduced, system administrators can simply begin to ignore alerts issued by IDS sensors. This does not mean that honeypots will never have false positives, only that they will be dramatically lower than in most IDS implementations.

Another risk that the intrusion detection systems present is false negatives, which occur when IDS systems fail to detect a valid attack. Honeypots eliminate false negatives, as they are not easily evaded or defeated by new exploits. In fact, one of the main values of honeypots is that they can detect new or unknown attacks. Administrators do not have to worry about updating the signature database or patching the anomaly detection engines. Honeypots successfully capture any attack that is presented to them. However, as discussed above, this only works if the honeypot itself is attacked.

Honeypots can simplify the detection process. Since honeypots have no production activity, all connections to and from the honeypot are suspicious by nature. By definition, each time a connection is established to the honeypot, it is most likely a scan or unauthorized attack. Each time the honeypot initiates a connection, this most likely means that the system was successfully compromised. This helps reduce both false positives and false negatives, simplifying the detection process enormously. By no means should honeypots replace your IDS systems or be the only method of detection; however, they can be a powerful tool to complement intrusion detection capabilities.

2.2.2 Level of Interaction of Honeypots

Honeypots differ in the way that they are deployed, and the sophistication of the decoy attack vectors. One way to classify the different kinds of honeypots is by their level of involvement, or interaction with host systems [WW19]. The level of interaction determines the level of penetration a malicious actor can have on the targeted system. A higher level of interaction between a honeypot and the host system means that a malicious actor can interact more critically with the system. Lower level interaction means that a malicious actor will not be able to interact with the host system in a critical manner.

There are three levels of interaction for honeypots [KGV+17]: 1) low-interaction honeypots; 2) medium-interaction honeypots [WIC06], and 3) high-interaction honeypots. Table 1 shows a relative comparison between the different levels of interaction [KMS14].

Table 1. Honeypot interaction level comparison

Level of Interaction	Benefits	Drawbacks
Low	• A malicious actor cannot critically impact the system because it has little interaction with the underlying host system. • Simple to deploy with an easy configuration. • It does not require significant resources to maintain.	• The data generated from a malicious actor are minimal and do not provide any critical details on a malicious actor's behaviour (complex threats, zero-day exploits). • Easy for a malicious actor to detect the presence of the honeypot.

Medium	• The system looks as an attractive target to a malicious actor since it is emulated to provide a more realistic setting. • Data generated from a malicious actor's actions offer better view of their behaviour.	• The risk of actual system compromised is increased since malicious actors can escape from the protected environment and impact the host system.
High	• High-interaction honeypots help in identifying unknown vulnerabilities. • The data generated offer, reflect a malicious actor's behaviour accurately.	• The risk of host system compromise is significant because malicious actors can exploit the honeypot to access the host system. • Time-consuming and complex configuration to implement.

2.2.2.1 Low-interaction honeypots

Low-interaction honeypots offer limited interaction with the actual system. A malicious actor has very limited attack vectors for compromising or impacting the actual system. Low-interaction honeypots have no operating system that a malicious actor can interact. As a result, the attack surface is minimized. The deployment and configuration of low-level honeypots is a simple task. However, due to the limited interaction between a malicious actor, it is not possible to receive any significant amount of actionable data. The main use of low-interaction honeypots is the monitoring and identification of network traffic. Network traffic can enable security analysts to identify network-based attacks, such as viruses, and worms. Low-interaction honeypots offer one or more simple services, such as Secure Shell (SSH), File Transfer Protocol (FTP), web servers, or other network daemons. Such services need to be simple and available for a malicious to interact with. All communication attempts with any particular service are logged and investigated afterward. This type of honeypots is considered as system daemons that help a network administrator to monitor any attempts of compromise on the host system in passively.

2.2.2.2 Medium-interaction honeypots

Medium-interaction honeypots try to combine the benefits of both approaches regarding to botnet detection and malware collection while removing their shortcomings [AAO13]. Medium-interaction honeypots are more advanced than low-interaction honeypots and offer additional features. The host operating system is emulated through application layer virtualization to provide a more attractive target to a malicious actor. As a result, the data generated by the behaviour of the malicious actor is more actionable than the data generated by low-interaction honeypots. Medium-interaction honeypots do not aim to fully simulate the host system's operation environment, such as implementing application protocols in detail. Medium-interaction honeypots aim to provide enough responses that known exploits wait on certain ports that will trick malicious actors into sending their payload. Once this payload has been received, the shellcode of the exploits can be extracted and analysed. This type of data is highly valuable in preventing novel attack vectors on the host system.

A medium-interaction honeypot strategy is a balanced solution [AAO13]. It provides less risk than creating a complete physical or virtualized system to divert malicious actors, but with more functionality. Medium-interaction honeypots are still not suitable for complex threats such as zero-day exploits but could reveal attack patterns for specific vulnerabilities.

2.2.2.3 High-interaction honeypots

High-interaction honeypots are the most advanced type of honeypots. Rather than emulating certain services and applications of the host system, a malicious actor is provided with a real system to compromise [PB19]. This reduces the likelihood of the honeypot's discovery by the malicious actor. It allows security analysts to learn more about the tools, tactics, and motives of malicious actors and get a better understanding of their behaviour [CAB+17]. A high-interaction honeypot is a conventional computer system, such as a commercial off-the-shelf (COTS) computer, a router, or a switch. A high-interaction honeypot can adapt to each incident, making it far less likely that the malicious actor will realize they are engaging with a decoy. The main drawback to a high-interaction honeypot is the time and effort it takes to build the decoy system, and then to maintain the monitoring of it long-term to mitigate risk for the host system. Furthermore, the level of interaction with the host system is significant, increasing the likelihood of the malicious actor compromising and exploiting the host system.

2.2.3 Physicality of Honeypots

The honeypots can be classified by their physicality [PRO04]:

- **Virtual Honeypots.** These are virtual honeypots and are simulated by a host machine that forwards the network traffic to the virtual honeypot. Usually, this kind of honeypots is high-interaction honeypots, more flexible and cost-effective. These ones are the ones used in this project.
- **Physical Honeypots:** This kind of honeypot are real machines on the network with its IP address. Physical honeypots have higher costs in comparison with the virtual one, but they are more reliable and it’s more difficult be identified as honeypots.

2.2.4 Location of Honeypots

The honeypots can be classified based on their location on the following categories [MZG12]:

- **Client honeypots** identify attacks on client applications, discover vulnerabilities of client applications and detect malicious webservers. A characteristic example is the case of a browser that visits a variety of websites in order to attack them by taking advantage of any existing vulnerability.
- **Server honeypots** serve as decoys on the network waiting for an attacker. They are designed to protect production environments by mirroring production servers and services. In case that a server honeypot is attacked then all the actions of the attacker are recorded. Thus, the administrators can be more prepared and have a better protection against future attackers.
- **Hybrid honeypots** include active modules of client honeypots into server honeypots to interact with webservers and increase the exposure of server honeypot's services.

In the SDN-microSENSE project, the objective of having honeypots as part of the Risk assessment framework is to serve as a decoy mechanism to possible attackers, facilitating early detection of threats and attack patterns, and the mitigation of potential risks. The three honeypots (IEC 61850, IEC60870-5-104 and Modbus) are classified as production, medium interaction, virtual and server honeypots. The description of the functionalities of the different honeypots are detailed in the section 5.

3 Existing honeypots for EPES

EPES and other relevant environments that handle the generation, transmission and energy metering, have become a prominent target for major cyberattacks. The convergence of IT and Operational Technology (OT) environments has complicated OT security, making Industrial Control Systems (ICS) infrastructure of EPES a target of increasingly sophisticated cyberattacks. ICS (Industrial Control Systems) are systems or devices that manage, regulate and control the behaviour of other devices or control systems used in the specific processes of an industry, such as nuclear, electrical, chemistry, oil, gas, water, etc. They combine electronic, mechanical, electrical, hydraulic, pneumatic components, among others. Some of the multiple communication protocols used by ICS are Modbus, DNP3, EIP, ICCP and CIP.

ICSs are made up of multiple types of control systems, including the Distributed Control Systems (DCS), the Process Control System (PCS), the Supervisory Control Systems and data acquisition (SCADA), the Remote Terminal Units (RTU), the Human Machine Interface (HMI), the Programmable Logic Controller etc. EPES systems that have industrial processes use ICS, SCADA, PLC, RTU, HMI and other industrial control devices, and are made up of a corporate network (web services, email, etc.), of supervision (SCADA, workstations, etc.), of control systems (HMI, PLC, RTU etc). The industrial process is supervised and controlled from a control network that allows sending and receiving information using industrial communication protocols (Modbus, DNP3, etc.) to the PLCs and RTUs through an HMI by wired or wireless means, to control industrial devices that can open or close breakers and connectors, obtain temperature, release pressure, among other functions.

In recent years, various honeypot projects have emerged to protect and secure ICS systems. While most of the honeypot projects are developed for IT environments, there are honeypots developed for OT environments, such as Conpot, among others. These honeypots have evolved and are currently implemented in the form of honeynets, entire networks of honeypots that simulate complete systems, thus allowing much more information to be gathered about attacks. it is very important to obtain all possible information about potential attackers, their methodology, potential targets and systems of interest, in order to anticipate potential attacks and increase the level of response to any occurred incidents.

A honeypot is a useful tool to improve the security of control systems. There are several existing honeypots that can be quickly and easily deployed in an industrial control system. Several state-of—the-art honeypot solutions for EPES systems are presented below.

Conpot is a low-interactive server-side ICS / SCADA honeypot maintained by the Conpot development team and The Honeynet Project [RIS13]. The main features of Conpot is that it is designed to be easy to modify, expand and deploy. The present configuration simulates a Siemens SIMATIC S7-200 PLC, with basic functions, an input / output module and a CP 443-1 communications processor, that allows connecting to SIMATIC in an ethernet network.

Another feature of Conpot is that it can be connected to a real HMI and also allows interaction with real ICS hardware, since Conpon has been created in order to work with industrial control protocols and with basic elements to build its own ICS. It supports common industrial control protocols like HTTP, Modbus, S7Comm, SNMP, CIP, Ethernet/IP, BACnet and IPMI.

It is possible to delay the response times of the services to imitate the behaviour of industrial systems when they are under high processing. In addition, it has a custom Human Machine Interface (HMI) software. These features improve emulation and expand the honeypot's interaction, increasing the points to associate and attack. This attracts cybercriminals, making them think that it is a real ICS, like a honeycomb that attracts bees; This allows obtaining more information on the methods and forms of attack to create intelligence and implement mitigation measures for these cyber-attacks.

Honeyd is a very versatile low-interaction honeypot that can simulate various TCP/IP services [PH07]. It is an open-source honeypot that allows users to configure and run multiple virtual hosts on a computer network. These virtual hosts can be configured to mimic several different types of servers, allowing the user to simulate a large number of network configurations. In addition, Honeyd is able to assume up to thousands of unused IP addresses in a network and serving requests made to them. Honeyd is a highly configurable honeypot. Hundreds of simulated virtual machines can be created on the same physical machine. It can emulate any operating system, but also different switch operating systems, routers etc. The systems emulation is not only done at the application level but also at the IP stack level.

Furthermore, it can listen to any TCP or UDP port and it is capable of handling ICMP ping requests. Another peculiarity of Honeyd is that it allows to redirect the requests made to a specific IP address and port to a server that presents this service. It also provides a fairly detailed package and service log. HoneyD can be used to hide the real system, identify threats, assess risks and improve network security. It supports IP, ICMP HTTP, FTP, MODBUS TCP and UDP protocols.

Gridpot is an open-source honeypot that simulates a power grid SCADA in a realistic way. Gridpot uses a honeypot layer and a modelling layer to add electrical components and integration between GridLAB-D and Conpot and has built-in IEC 61850 protocols for imitating largescale electric power grids [RNK+20]. This combination allows this honeypot to acquire all the data acquisition advantages of Conpot and a simulation environment with multiple models that provide great realism thanks to GridLAB-D. Researchers in [RNK+20] have found GridPot to be more successful at deception than Conpot. Gridpot supports HTTP, Modbus, S7Comm, SNMP, and IEC 61850 protocols.

Dipot [CLL+18] is a Distributed Industrial Honeypot System that monitors and captures scanning and attacking behaviours against ICS. Dipot aims to collect data without being traced by attackers and is comprised of honeypot nodes, as well as data processing and visualization modules Dipot supports HTTP, Modbus, Kamstrup, SNMP, BACnet and S7comm protocols.

SHAPE [KG15] is a honeypot specifically designed for Electric Power Substations. It can emulate any IED conforming to the IEC 61850 standard and can detect unauthorized or illicit traffic in IEC 61850-based Electric Substation automation Systems.

CryPLH [BJM+14] is a low-interaction PLC honeypot which can detect attacks against ICSs. It simulates HTTP, HTTPS, SNMP, and Siemens SIMATIC STEP7 configuration interfaces. CryPLH is also capable of detecting port scans and brute force attempts via SSH.

3.1 Honeypot solutions in research projects

There are several EU-funded and international projects that have developed honeypot solutions. One project deployed a large-scale cloud-based low-interaction honeynet for 28 days using Amazon's EC2 cloud environment. This honeynet was able to simulate Modbus, IEC-104, DNP3, ICCP, XMPP, TFTP and

SNMP protocols [SOY15]. Furthermore, researchers in [MLC19] implemented a low-interaction honeypot solution by setting up 5 AWS instances on different geographic locations. Their honeypot was able to simulate IEC 61850 MMS (and Siemens S7) Modbus TCP, ENIP, IEC 60870-5-104, DNP3 and BACnet.

In the European Research Area, a number of projects have conducted research on honeypots. The H2020 **SPEAR** (Secure and PrivatE smArt gRid) project **[**SPE19] develops an innovative platform that exploits honeypot technologies to perform incident detection, anomaly detection and provide forensic data that can be presented as legal evidence in court [DAL19]. The AMI honeypots offered in SPEAR are mainly of two types: i) RTU honeypot s that emulate all the industrial communication protocols of RTUs used in Smart Grids, and ii) NeuralPot honeypots that leverage Deep Neural Networks (DNN) to efficiently emulate industrial devices utilising the Modbus protocol.

The **SERIoT** (Secure and Safe Internet of Things) project [SER18] implements honeypots for IoT devices. The project develops low and medium interaction virtualized honeypots that can address the main typologies of IoT devices of the market. The project demonstrates their honeypot approach in real-world environment showing effectiveness against attacks on IoT devices.

The **YAKSHA** (Cybersecurity Awareness and Knowledge Systemic High-level Application) project [YAK18] focuses on IoT Security and develops the innovative concept of honeypots-as-a-service. It provides an automated framework for deploying honeypots, thus facilitating the end-users with the installation and configuration of custom honeypot solutions

The **ATENA** (Advanced Tools to assEss and mitigate the criticality of ICT compoNents and their dependencies over Critical InfrAstructures) project [ATE20] developed an Intrusion and Anomaly Detection System that integrated both existing components and new probes, such as SCADA honeypots. The project's developed detection agents included domain-specific honeypot s and Honeynets, Shadow RTUs, and other network devices.

Last but not least, the **SISSDEN** (Secure Information Sharing Sensor Delivery Event Network) project [SIS16] deployed a worldwide network of over 250 sensors to observe client to server attacks. This network used various honeypot solutions and darknets (network telescopes). 13 types of honeypots were deployed as part of the sensor network: CiscoASA, Cowrie, Conpot, Dionaea, Elasticpot, Glastopf, Heralding, Honeypy, MICROS, Rdpy, Spampot, Struts and Weblogic. These honeypots collected over 2 billion attack events over the course of the project.

3.2 Innovations of honeypots in SDN-Microsense

During the task 3.3 of SDN-microSENSE project, three honeypots that emulate IEC61850 [61850], IEC60870-5-104 [60870] and Modbus [MODBUS] protocols have been developed and also another component called Honeypot Manager that has two main objectives: 1) to process the information about unknown anomalies (like zero-day attacks) and to indicate to the SDN controller the required information to re-arrange the honeypots network in order to collect information for unknown anomalies detected by the ML models of the XL-EPDS and 2) to automate the deployment and configuration of honeypots in the virtual machines.

This section will highlight how the SDN-microSENSE honeypot technology is differentiated compared to the existing EPES honeypots and the innovation provided with respect of the state of art presented in the section above.

Honeypot IEC 61850

The adoption of the IEC 61850 [61850] standard family for substation automation is expanding rapidly worldwide. For this reason, new cybersecurity monitoring capabilities will need to be deployed into substation networks. The IEC 61850 honeypot developed in Task T3.3 comes to increase the potential of the monitoring tools and solutions of SDN-microSENSE architecture. The honeypot simulates the behaviour of a real Control IED (Intelligent Electronic Device) in charge of the control of a high/medium voltage circuit breaker in a substation. IEC 61850 honeypot design approach is based on the adaptation of the IED Capability Description (ICD) file of a real IED installed in Tecnalia's Smart Grid Cybersecurity Laboratory. However, the implemented honeypot architecture and design facilitates the compilation of new ICD files to incorporate new IEC61850 control and protection functions to the honeypot or building different honeypots for each type of IED inside the substation. The flexibility introduced into the design will allow to deceive attackers and make the believe they are controlling a real IED while the honeypot register every valuable information for further analysis. Section 5.2.1 presents a detailed explanation of this honeypot.

Honeypot IEC60870-5-104

Conpot honeypot supports the IEC 60870-104 [60870] that used to monitor energy systems, control systems and their communications. We use the RTU honeypot to emulate the behaviour of an RTU that controls devices in a smart grid substation. In the context of the SDN-microSENSE, the Conpot honeypot was extended to support more communication commands regarding the System Information in the Control direction such as Counter Interrogation command and read command.

Honeypot Modbus

The Modbus [MODBUS] honeypot developed in the context of the SDN-microSENSE is capable of emulating any device using Modbus/TCP either it is a server device like an RTU or PLC or a client device, such as an HMI. Moreover, both functionalities are supported simultaneously, thus, taking full advantage of the flexibility provided by the Modbus/TCP protocol. In particular, the Modbus honeypot emulates dynamically the Modbus/TCP network traffic generated by the real devices, utilising a Generative Adversarial Network (GAN). Finally, compared to other Modbus honeypots, the specific honeypot supports more Modbus/TCP function codes, such as function codes 22, 23 and 24.

Honeypot Manager

The Honeypot Manager developed in this task, although it is based on the one created in SPEAR project [SPE20-D43], contains relevant improvements and added functionalities. First, and regarding to the deployment functionality, the business logic is used the one defined in SPEAR, but the component itself has been developed using a different technology to facilitate the integration with the rest of the components of SDN-MicroSENSE. In SPEAR project, the Honeypot Manager was developed using Phyton-Flask technology. In the Honeypot Manager of SDN-Microsense, the business logic is developed in: 1) Spring JPA to DB access, 2) Spring REST to build the API REST required and 3) Spring Boot to mount the whole back-end application, additionally the Honeypot Manager of this project offers a front-end to support the security operator to manage the honeypots deployment.

On the other hand, this Honeypot Manager has the functionality to receive information about unknow anomalies, like for example zero-day attacks from the XL-EDPS component. This information from the XL-EDPS is processed together with the information that the Honeypot Manager has about the honeypots that are deployed or available. Finally, the Honeypot Manager indicates to the SDN controller the required information to re-arrange the honeypots network in order to collect information for these unknown anomalies.

4 Analysis of Use Case Requirements and Specifications for Honeypots

The following tables present the functional specifications and operational specification [SMS20-D23] related to the honeypots and the work done in the T3.3. The tables are completed with an explanation on how the developed honeypots and the Honeypot Manager give the solution to each specification.

Specification Name	Monitoring of Infrastructure Components and Anomaly Detection	**Specification ID**	SPEC-F1
Action	In order to get the previous, monitoring of the EPES infrastructure components and anomaly detection systems are included in the SDN-microSENSE architecture, with focus on Intrusion Detection and Prevention and Advanced Anomaly Detection. These systems provide real-time network traffic analysis and the network components monitoring, focusing in detecting deviations from what is considered the normal behaviour. This monitoring system is able to detect issues in all relevant industrial protocols as defined in the related requirements, e.g.: • Modbus TCP • EtherCAT • IEC 61850 • IEC 60850-5-101 • IEC 60850-5-102 • IEC 60850-5-104 • PROCOME The overall monitoring and detection process is performed in a continuous and automatic manner. The result of this processing is the generation of Security Events indicating a change of the EPES status, which can be related to a cyberattack or an anomaly.		
Related Requirement(s)	User Requirements: FR-UR-3, FR-UR-4, FR-UR-5, FR-UR-6, FR-UR-7, FR-UR-8, FR-UR-9, FR-UR-10, FR-UR-11, FR-UR-12, FR-UR-13, FR-UR-14 and FR-UR-15. Use Case Requirements: FR-UC1-01, FR-UC1-02, FR-UC1-03, FR-UC1-04, FR-UC1-05, FR-UC1-06, FR-UC1-07, FR-UC1-08, FR-UC1-09, FR-UC1-10, FR-UC1-11, FR-UC2-01 and FR-UC3-01. General Functional Requirements: FR-GR-04, FR-GR-06, FR-GR-08, FR-GR-09, FR-GR-10 and FR-GR-12.		
Honeypot task contribution	Three honeypots have been developed to decoy potential attacks using the following industrial protocols: • IEC 61850 • Modbus • IEC 60850-5-104		

[1] Codes here are the requirement codes in Deliverable D2.2 [SMS20-D22].

	Honeypot Manager, moreover, to deploy the honeypots selected by the security operator in the network, is in charge to get from the XL-SIEM information regarding the zero-day attacks detected by Nightwatch. It collects this information, analyses it and send to the SDN controller information to forward traffic network to the honeypots in order to get more information of these zero days attacks.

Specification Name	Security Events Processing	**Specification ID**	SPEC-F2
Action	The processing of the security events generated from the Intrusion Detection and Prevention and the Advanced Anomaly Detection systems is processed in real time by a specialized SIEM (Security Information and Event Management) system in the architecture, which generates the Security Alerts towards the SDN-microSENSE Risk Assessment Framework (S-RAF) system for further processing.		
Related Requirement(s)	User's Requirement: FR-UR-16 Use Case Requirements: FR-UC1-01, FR-UC1-02, FR-UC1-03, FR-UC1-04, FR-UC1-05, FR-UC1-06, FR-UC1-07, FR-UC1-08, FR-UC1-09, FR-UC1-10, FR-UC1-11, FR-UC2-01 and FR-UC3-01. General Functional Requirements: FR-GR-03, FR-GR-04 and FR-GR-12.		
Honeypot task contribution	Honeypots developed get information of the cyber-attacks received and send to the XL- SIEM in order to be analysed. The information collected by the honeypots are analysed by the XL-SIEM engine in the same way that analyses the information from other sensors.		

Specification Name	Alerts Categorisation and Incidents Generation	**Specification ID**	SPEC-F4
Action	Alerts generated from the SIEM are forwarded towards a Risk Level Assessment component to be further processed and categorized. The risk assessment component is to conduct an analysis on the identified alerts coming for the XL-SIEM with respect to the existing assets and vulnerabilities of the infrastructure. Attacks that target vulnerable assets will be reported as incidents and will modify the risk models accordingly. For the case when the alerts are categorized as incidents, the following actions are performed by the Risk Level Assessment component: • Incident is recorded in an anonymous repository of incidents (ARIEC) • Incident is forwarded to the SDN-microSENSE self-healing which takes the necessary counter-measure actions on the EPES infrastructure.		

	The risks level categorization will be based in a meta-model covering existing IT and industrial devices (including older legacy SCADA and ICS devices and IoT components), electric power networks and related software systems, and energy-related personnel, stakeholders and processes. The risk assessment model will consider vulnerabilities and threats that target diverse energy equipment and services from AMIs, RTUs or SCADAS to IT systems. To provide this functionality, the Risk Assessment framework consists of two main functional blocks: i. The Risk Level Assessment block itself, ii. The Vulnerabilities Management block, and iii. An associated honeypots Management module The Vulnerabilities Management block is a kind of helper for the Risk Level Assessment block: the second one queries the first one to get information about the vulnerability of the different components in the infrastructure. Honeypots are used to emulate smart IT and industrial devices, in order to apply protection and detection of possible entry points, so as to hide the actual infrastructure to potential attackers. Thus, the honeypot s Management module is a key component for identifying security vulnerabilities, including IT (IoT devices, gateways and communication devices) and energy systems and devices (smart meters and SCADA & ICS components). Honeypots are intended to be operating as an additional level of security in the SDN-microSENSE architecture. This Risk Assessment framework will be deployed as an SDN application. This way, suspicious data flows can be automatically diverted towards the honeypot's infrastructure if necessary.
Related Requirement(s)	User Requirements: FR-UR-01, FR-UR-02 and FR-UR-16. Use Cases Requirements: FR-UC1-01, FR-UC1-02, FR-UC1-03, FR-UC1-04, FR-UC1-05, FR-UC1-06, FR-UC1-07, FR-UC1-08, FR-UC1-09, FR-UC1-10, FR-UC1-11, FR-UC2-01 and FR-UC3-01. Non-Functional Requirements: NFR-DPT-03, NFR-DPT-28, NFR-SEC-16, NFR-SEC-20, NFR-TST-01 and NFR-TST-02. Organizational Requirements: OR-GR-02, OR-GR-05, OR-GR-10, OR-GR-12 and OR-GR-15.
Honeypot task contribution	In this task, three honeypots to emulate three industrial protocols have been developed. These honeypots collect the information of the cyberattacks. The information of the logs collected by the honeypots are

	sent to the S-RAF component through the XL-SIEM that is in charge of analysing the honeypot logs. During this task, a new functionality executed by Honeypot Manager is developed. With this functionality it is possible to get more information of the zero-day attacks (identified by nightwatch), forwarding the traffic network regarding these zero-day attacks to the honeypots and analysing the logs produced. This task is done through the SDN- Controller. The Honeypot Manager receives the information from the XL-SIEM about zero-days attacks and analyses this information with the knowledge that it has about the honeypots that are deployed in the network. Based on this analysis, it forwards the information to the SDN-Controller to send the traffic network to the honeypot that is able to get more information.

Specification Name	Resilience and Reliability	**Specification ID**	SPEC-OP1
Action	A common and effective approach to address this problem is to make redundant the critical system components, which ensures the correct operation even in case of failures. To get this, all the elements in SDN-microSENSE architecture have the possibility of being deployed in a redundant way following different approaches, ranging from the ability to deploy nodes with active-standby policies to more sophisticated options such as automatic fail-over. Regular backup of the systems comprising SDN-microSENSE (SPEC-OP4) also help on this, as it allows the system to be restored to a safe operating point. Also, for the SDN Controller (a critical component in the system) a specific Synchronisation and Coordination Service (SCS) has been issued as part of the architecture for providing distributed synchronization and group services for the SDN-C components. Also, because its main purpose and design, we can tell SDN-microSENSE is reliable by design, because it has been designed to detect certain potential failures (because cyberattacks or other causes) and initiate the necessary preventive actions to handle them.		
Related Requirement(s)	Resilience Requirements: NFR-RES-01, NFR-RES-02 and NFR-RES-03. Reliability Requirements: NFR-RL-03, NFR-RL-04 and NFR-RL-05. Other Requirements: NFR-SDNSEC-12		
Honeypot task contribution	The Honeypot Manager allows to deploy the honeypots that the security operator decided. This implies that any honeypot can be deployed following a redundant strategy. It is also important to take into account that honeypots are not considered as critical components, because they are not directly involved in the correct operation.		

Specification Name	Data security by design	**Specification ID**	SPEC-OP3
Action	In order to achieve this requirement, the SDN-microSENSE system adopts security by design approach and provides several tools that are geared towards the protection of the system. Here, the core functions of the XL-EPDS are designed to detect, prevent and record security incidents. These are implemented through the Security Information and Event Management (SIEM) function, which is intended to monitor continuously, control and correlate the operations that take place in the Infrastructure Plane of the platform, an Intrusion Detection and Prevention System (SS-IDPS) function, to provide specification-based detection techniques, and having the capability to detect zero-day attacks; an Anomaly Detection function, to apply anomaly detection methods for detecting anomalies and identify risks based on them and an Anonymous Repository of Incidents (ARIEC), which stores and anonymously share information about the detected incidents with other EPES. In addition, SDN-microSENSE also incorporates the S-RAF framework, that consists of the honeypots Manager and the Risk Level Assessment module to enhance the security of the system. It establishes the necessary risk priorities and security policies, identifies possible threats and vulnerabilities and determines the corresponding risk levels. Furthermore, the various functionalities on the PPF as described in the previous table, are all geared towards the enhancement of the security of the SDN-microSENSE system.		
Related Requirement(s)	Data security requirements (NFR-SEC). SDN-specific Security Requirements (NFR-SDNSEC).		
Honeypot task contribution	Honeypots are intended to be operating as an additional level of security in the SDN-microSENSE architecture because all the traffic that the honeypots capture can be considered as attack, so all the information gathered by the honeypots is useful to improve the security of the network.		

Constraint Name	Authentication	**Constraint ID**	CONS-T2
Action	To overcome this constraint, the SDN-microSENSE architecture uses authentication for all connections among parts of the system and the equipment used, while for sensitive information and session expiration multifactor authentication controls are implemented. Also, authentication is set on all connections among the system components. Specifically, the PPFs Access Control Manager is used for authentication procedures for all users and systems involved in the architecture.		
Related Requirement(s)	NFR-SEC-03, NFR-SEC-04, NFR-SEC-09, NFR-SEC-10, NFR-SEC-18, NFR-SDNSEC-01 and NFR-DPT-26.		

Honeypot task contribution	The Honeypot Manager has an authentication process implemented. At this point of the project the PPFs Access control manager is not implemented, but in the integration task, the authentication process that the Honeypot Manager has implemented will be adapted to the one offered by the PPFs Access Control Manager

In the Deliverable D2.2 "User & Stakeholder, Security and Privacy Requirements" [SMS20-D22], different types of attacks are defined to be detected by the technologies developed in the project. The honeypots developed in this task could collect information about some of these attacks, mainly those related to the communications . The table relates the potential attacks that could be detected by the honeypots, the CAPEC id and the code of the requirement form D2.2.

Table 2. Types of attacks detected by honeypots

Attacks type	**CAPEC id**	**Requirements** [SMS20-D22]
Denial-of-service attack	CAPEC-125: https://capec.mitre.org/data/definitions/125.html	FR-UC1-03, FR-UC1-06, FR-UC1-10, FR-UC3-01, FR-UR-03
Protocol Analysis	CAPEC-192: https://capec.mitre.org/data/definitions/192.html	FR-UR-13, FR-UR-07, FR-UR-10
Unsolicited content injection attack	CAPEC-148 https://capec.mitre.org/data/definitions/148.html	FR-UC1-02, FR-UC1-05, FR-UC1-09, FR-UC3-01, FR-UR-05

5 Architecture and Detailed design

5.1 Architecture overview

This section presents how the components developed in this task fit in the general architecture of SDN-Microsense [SMS20-D23].

The components developed in the task 3.3, the honeypots and the Honeypot Manager, are two components of the SDN- microSENSE architecture. The Honeypot Manager (Section 5.2.4) is placed in the application plane and the honeypots developed in this task (sections 5.2.1, 5.2.2, 5.2.3) are placed in the Data/Infrastructure plane.

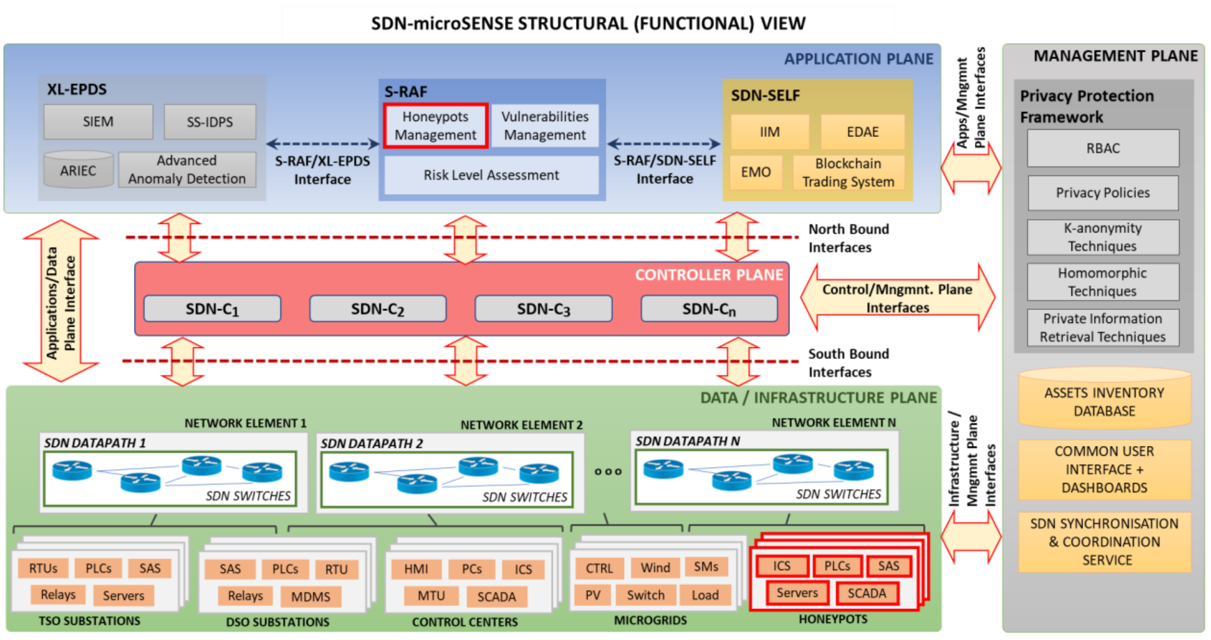


Figure 3. SDN-microSENSE Architecture. Structural View.

Figure 4 shows the general architecture of the honeypots and the Honeypot Manager and how interact with other components of the architecture.

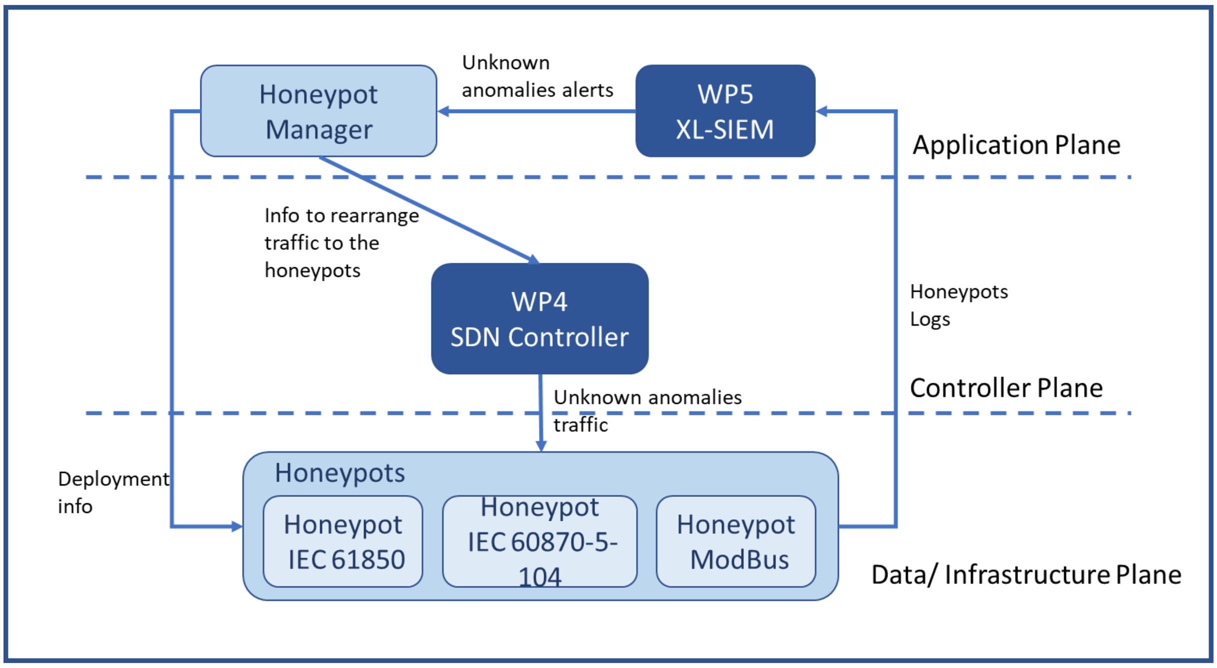


Figure 4. General view of the components and the relationships with other components.

The Honeypot Manager interacts with other tools in the SDN-microSENSE ecosystem (see Figure 5):

- It communicates with **XL-EPDS** interface to gather the information of the unknown anomalies, for example, the zero-day attacks provided by the XL-SIEM through the Application Plane Inner Interfaces [SMS20-D23]. The interface used (S-RAF/XL-EPDS-01) is the one provided by XL-SIEM that provides necessary alerts metadata for processing the unknow anomalies alerts. These alerts are sent through RabbitMQ.
- It communicates with the **SDN-controller**. The Honeypot Manager provides information in order to forward network traffic to the deployed honeypots about the unknown anomalies through the SDN- Controller. This is done using the North Bound interfaces. This interface (S-RAF_NBI-1) allows to request to the SDN Controller (SDN-C) how network packets should be forwarded to the deployed honeypots, so that they can collect more information regarding the attacks or the anomalies.

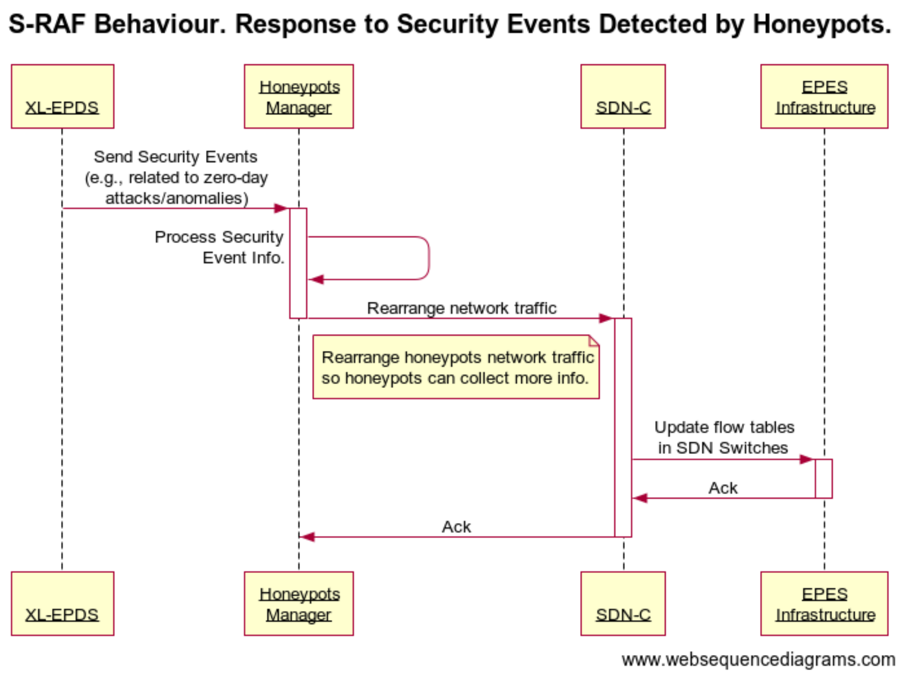

S-RAF Behaviour. Response to Security Events Detected by Honeypots.
XL-EPDS
Honeypots Manager
SDN-C
EPES Infrastructure
Send Security Events (e.g., related to zero-day attacks/anomalies)
Process Security Event Info.
Rearrange network traffic
Rearrange honeypots network traffic so honeypots can collect more info.
Update flow tables in SDN Switches
Ack
Ack
www.websequencediagrams.com

Figure 5. Response to Security Events detected in honeypots. [SMS20-D23]

The honeypots interact mainly with two other components in the SDN-microSENSE ecosystem, XL-SIEM and SDN controller:

- Honeypots send their logs to the **XL-EPDS** to allow analysing this information. Honeypots use an application Plane Inner Interfaces (S-RAF/XL-EPDS-03) to send to the XL-SIEM the captured logs. These logs are parser by the XL-SIEM and send to the Advanced anomaly detection.
- **SDN-controller** communicates with the honeypots[2] using the South-bound interface (see Figure 3). This interface allows the SDN controllers to define the behaviour at the Infrastructure/Data Plane elements. The most common API for this Southbound Interface is OpenFlow [ONF-15].

5.2 Functional and Components view

5.2.1 IEC 61850 Honeypot

5.2.1.1 Functional description

The IEC Strategic Group 3 on Smart Grid includes the IEC 61850 communication protocol in their Smart Grid IEC reference architecture document [IECSTD] as one central communication standard. As stated in that reference architecture, IEC 61850 [61850] is the standard selected for the communications between field devices and systems inside a substation. Even if the different extensions of the IEC 61850 consider the use of the standard also in communications between substations and control centres, the reality is that such implementation is not widespread today.

Hereunder are listed some of the main features of this standard:

[2] The EPES Infrastructure components also include honeypots

- Communication profiles are based on existing IEC/IEEE/ISO/OSI communication standards.
- All the protocols used are open and support self-descriptive devices. New functionalities can be added.
- Data objects are defined by the standards related to the needs of the electric power industry.
- Communications syntax and semantics are based on the use of common data objects related to the power system.
- Communications services can be mapped to different state-of-the-art protocols.
- Considers the implications of the substation being one node in the power grid.
- Machine readable language is used to specify complete topology of an electrical system, the generated and consumed information and the information flow between all IEDs.

Since the first edition of the standard, published between 2003 and 2005, the original parts of the IEC61850 series have been updated and extended. Original protection, control and monitoring functions included in the first edition were extended in future editions to include measurement and power quality information.

IEC 61850 adoption by substation operators (TSO and DSO primarily) has been gradual since the publication of the standard. Nowadays, at least in Europe, IEC 61850 is the main standard for substation automation. IEC 61850 is being installed in most of new substations automation. In the case of existing substations legacy protection and control equipment is being replaced by modern IED supporting this standard.

The strong adoption of the IEC 61850 into substation automation, from cybersecurity perspective, requires the development of new threat monitoring and prevention techniques that will contribute to the securitization of substation assets and by extension of the overall grid. In this context of a growing deployment of IEC 61850 compliant IEDs it is interesting to have specific IEC61850 protocol honeypots between network monitoring tools to detect and analyse APT targeting this kind of devices.

The IEC 61850 honeypot functionality developed in this task simulates the behaviour of a real IED in charge of the control and protection of a high/medium voltage circuit breaker in a substation. In Figure 6, two different types of circuit breakers that could be found in an electrical substation are shown.

Figure 6. Types of circuit breakers

High and medium voltage circuit breakers are electromechanical switching devices which connect and break current circuits (operating currents and fault currents) and carry the nominal current in closed position. Control, protection and measurement functionalities of this switchgear are carried out by IEDs that could be wired point-to-point to them or, in the case of modern substations, a single fiberoptic process bus could substitute those direct connections. IEDs are connected to the substation bus to be monitored and controlled by a substation SCADA or RTU by the IEC 61850 standard. The Figure 7 depicts the simplified network architecture of an IEC 61850 substation and IED connection to circuit breakers.

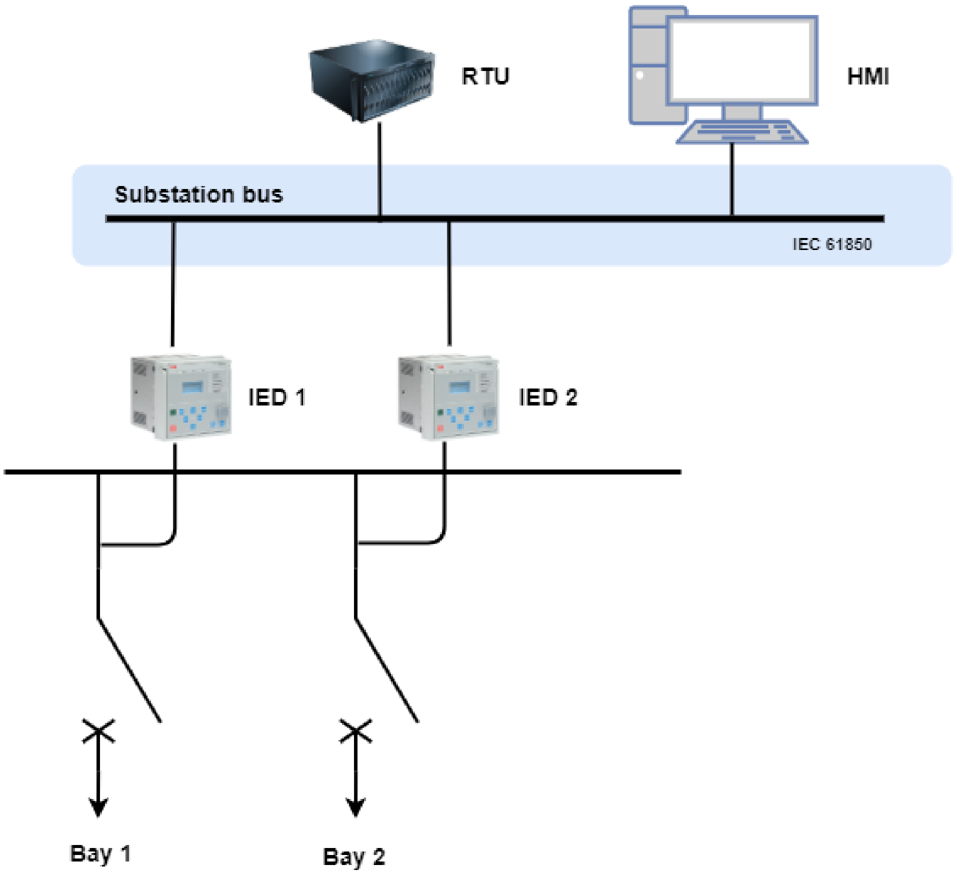


Figure 7. Simplified network architecture of an IEC 61850 substation and IED connection to circuit breakers.

SDN-microSENSE IEC 61850 honeypot includes the following functional characteristics:

- Integrates control functionalities that could be found in an operative IEC 61850 IED controlling a circuit breaker.
- Includes simple Control operation interlocking capabilities.
- Integrates the IEC 61850 model of a real IED to make it more attractive for attackers and make them believe that they are dealing with a real IED.
- Easily deployable to simulate a complete substation bus in which different IEDs control different bay positions.
- Buffered and unbuffered reports with different datasets are defined to simulate the interactions that normally take place in a substation between IEDs, RTUs and the SCADA.

IEC 61850 Configuration Description Language

Part 6 of IEC 61850 standard [61850-6] specifies the format of the file that describes communication related IED configurations and IED parameters, communications system configurations, switch yard (function) structures, and the relations between them. In a nutshell, the purpose of this file format is to describe IED capabilities and could be used as part of IED configuration data. The language that IEC 61850 has defined is based on XML version 1.0 and is called System Configuration Language (SCL). Files containing IED capability descriptions are called ICD or CID file.

The goal of the honeypot described in this section is to represent, as accurately as possible, the behaviour and functionalities of a real IED. Therefore, the ICD file that has been used to create honeypot's IEC61850 model is based on the ICD file of a fully commercial IED installed in Tecnalia's Smart Grid Cybersecurity Laboratory. This approach allows to focus only in the monitorization of IEC 61850 standard singularities. This ICD file has been simplified to leave only those elements of and IED model required to control the operation of a Breaker. Figure 8 shows a single line diagram (SLD) representation and a high-level diagram of the logical device and nodes included in the ICD file used to build the honeypot.

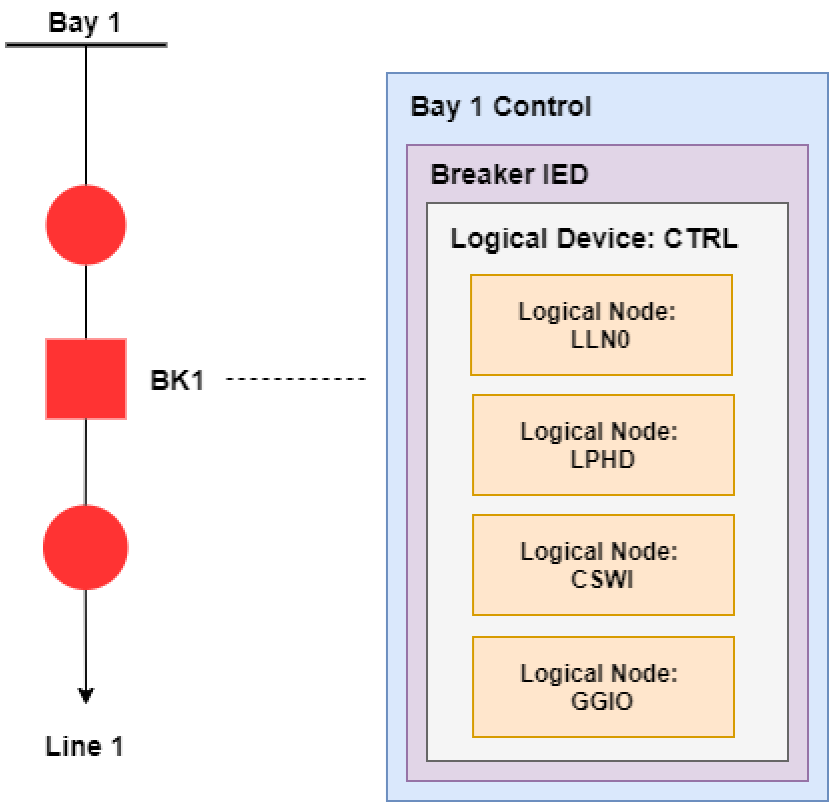


Figure 8. IEC 61850 functional hierarchy of the honeypot

Figure 9 aims to represent what an attacker would believe that is happening into a SLD of a real substation while interacting with the honeypot. The honeypot, in its interaction with the attacker, will react to received operation requests and update accordingly every data attribute involved in the operation of its IEC61850 model. By doing this, the attacker is deceived and believes that he is controlling a real electrical equipment while the honeypot is registering every interaction with it.

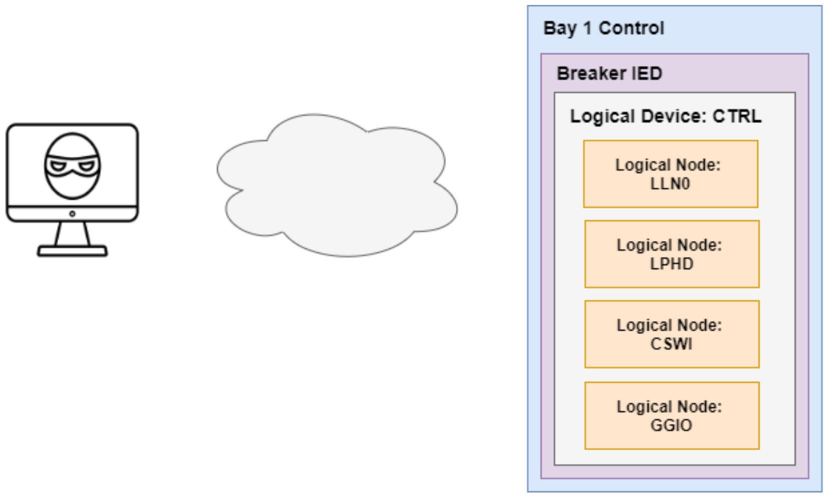


Figure 9. Attacker view of the honeypot

The functionality of the honeypot could easily be expanded by compiling a new ICD files which contain more control and protection elements. This will create a new IEC 61850 model to be included into the honeypot building process (explained in "Technical description" section). The honeypot will need to incorporate the required functions to deal with the objects and data attributes introduced by the expanded model. Figure 10 shows an expanded representation of the high-level diagram shown in Figure 8 and Figure 9 to show data objects associated to each of the Logical nodes of the model:

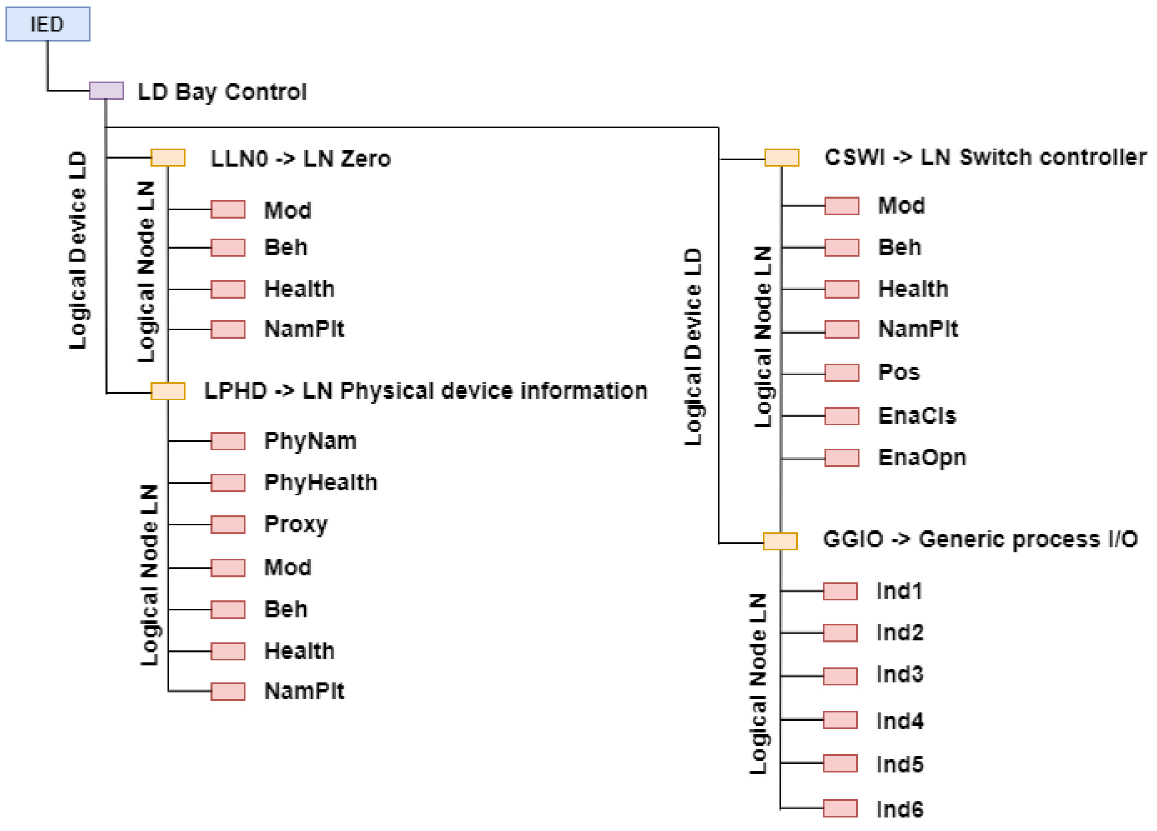


Figure 10. Organization of LDs, LNs, DOs and DAs of honeypot

- A set of logical nodes belong to a LD.
- The LN LLN0 is a special logical node per LD and contains for example the data sets, the various control blocks.
- The LN LPHD is a special logical node per LD and contains data objects that describe the status of the physical device (the IED).
- Each logical node represents a function and contains a number of data objects (DO).
- Each DO includes a number of data attributes (DA).

5.2.1.2 Technical and deployment details

The IEC 61850 honeypot software and all its auxiliary modules and libraries have been written in C language. The honeypot application is built upon ***libIEC61850*** open source library [LIB61850-20]. The use of this library has facilitated the integration of IEC 61850 standard complexities. Honeypot application integrates with the library and defines all the control logic required to deal with different data objects and attributes involved in the operation. Hereunder are listed IEC 61850 features provided by the library integrated by the honeypot:

- IEC 61850/MMS server: this server provided by the library is integrated into the honeypot software which facilitates the task of dealing with the complexities of the IEC 61850 MMS communications protocol.
- Support for buffered and unbuffered reports
- Data access service (get data, set data)
- All data set services (get values, set values, browse)

Source code is structured as a Makefile project to allow its compilation for several target architectures (i386, x64 or ARM) and OS (Linux and Windows). There is a principal Makefile with different targets to facilitate every task related to honeypot software building process. Building from scratch overall honeypot software simply requires running “make all” on this Makefile. It will recursively call to all Makefiles of the subsidiary modules and libraries.

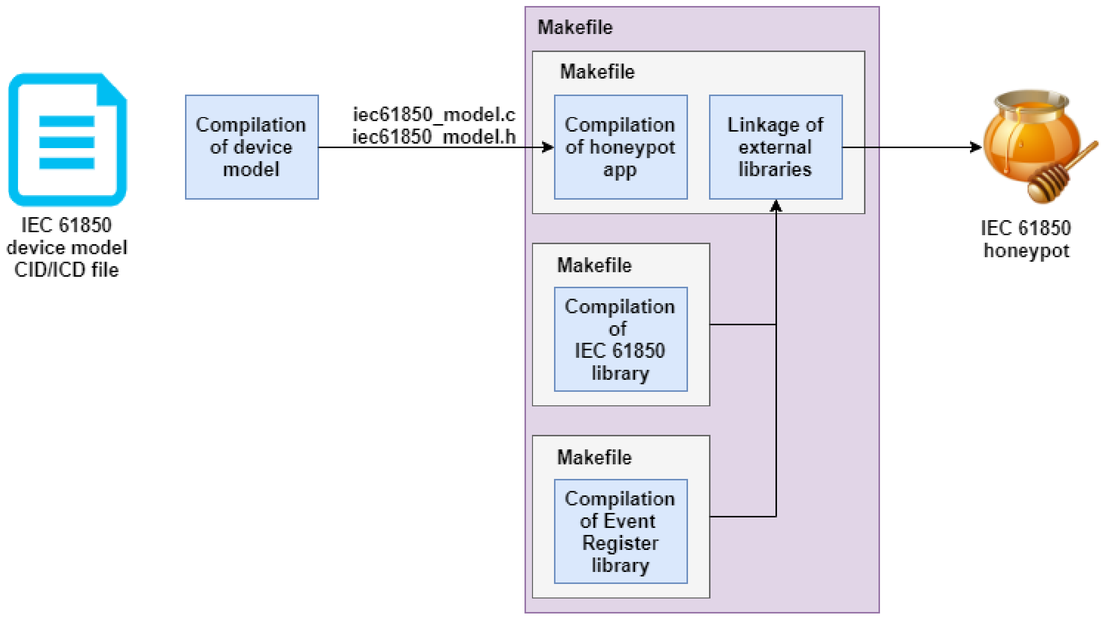

IEC 61850
device model
CID/ICD file
Compilation of device model
iec61850_model.c
iec61850_model.h
Makefile
Makefile
Compilation of honeypot app
Linkage of external libraries
Makefile
Compilation of IEC 61850 library
Makefile
Compilation of Event Register library
IEC 61850
honeypot

Figure 11. Honeypot software building process

As it could be noted from the previous image there is a first step needed before honeypot software building. IEC 61850 device model file with CID or ICD extension must be compiled before honeypot building process. Model compilation is performed by ***genmodel tool*** provided by libIEC61850 library. Model compilation gives as a result a .c file and a .h extension file. The resulting files need to be incorporated to the honeypot source code structure. This process needs to be done once unless the model is changed.

Despite libIEC61850 library is designed according to edition 2 of the IEC 61850 standard series it is compatible to edition 1. In the current version of the honeypot Edition 1 CID/ICD device model has been used. The use of legacy edition of IEC 61850 might make the honeypot more attractive to attackers as they could believe that they are dealing with a legacy device.

Event Registration

The main goal of a honeypot is to register appropriately the different actions performed by attackers. For event registration, a logging library has been specifically developed for the project. Once the honeypot detects an action that shall be recorded for later analysis it calls the library to store it conveniently. The purpose of using this library is to decouple from the honeypot application's source code the details of event management and, specifically, where those events are stored. With this approach future event publishing capabilities and storing locations (Databases, MQTT sending, etc) could be easily integrated into the honeypot without impacting its functionality. For each event the following information is passed to the library:

Table 3. Information stored for each event

Variable	Format	Description
eventID	String	Identificator of the event
Name	String	Name of the event
timestamp	String	Timestamp with the exact time in which the event occurred. The format of the timestamp is yyyy-MM-dd'T'HH:mm:ss*SSSZ
parameters	Array of parameters	An array of parameter structures with additional information related to the event

Each parameter is represented by key-value pair. In Table 4, it is described the format of one parameter associated to an event:

Table 4. Format of parameter information associated to an event

Variable	Format	Description
key	String	String which represents the unique identifier of the parameter
value	String	Value of the parameter

Event information is stored into files in JSON format to be processed and analysed later by the appropriated tools. File or files with event information are stored by default in */var/log* folder of the operative system. The JSON format has been selected for events because its usage is widespread among other honeypot solutions, such as the well-known Conpot [CON20].

Concretely, The IEC 61850 honeypot described in this section, registers the events listed in the Table 5 related to IEC61850/MMS communications standard:

Table 5. Events listed

			Parameters	
eventId	**name**	**timestamp**	**key**	**value**
0001	New connection	YYYY-MM-DDThh:mm:ssZ	ip	Ip of connected client-
0002	Connection closed	YYYY-MM-DDThh:mm:ssZ	ip	Ip of disconnected client
0003	Control operation	YYYY-MM-DDThh:mm:ssZ	ip	Ip of client sending control operation
			ctlNum	ctlNum attribute send by the client
			orCat	Originator category provided by the client
			ln	Logical Node
			dataObject	Requested Control object
			ctlVal	Control Value
0004	Read Operation	YYYY-MM-DDThh:mm:ssZ	ip	Ip of client sending control operation
			Ln	Logical node owning requested data object
			dataObject	Data object owning requested data attribute
			fc	Functional Constraint
0005	Write Operation	YYYY-MM-DDThh:mm:ssZ	ip	Ip of client sending control operation
			Ln	Logical node owning requested data object
			dataObject	Data object owning requested data attribute
			dataAttribute	Requested data attribute

The honeypot software could easily integrate additional event registrations. During pilot demonstrators new monitoring requirements might be identified.

5.2.1.3 Prototype architecture and components description

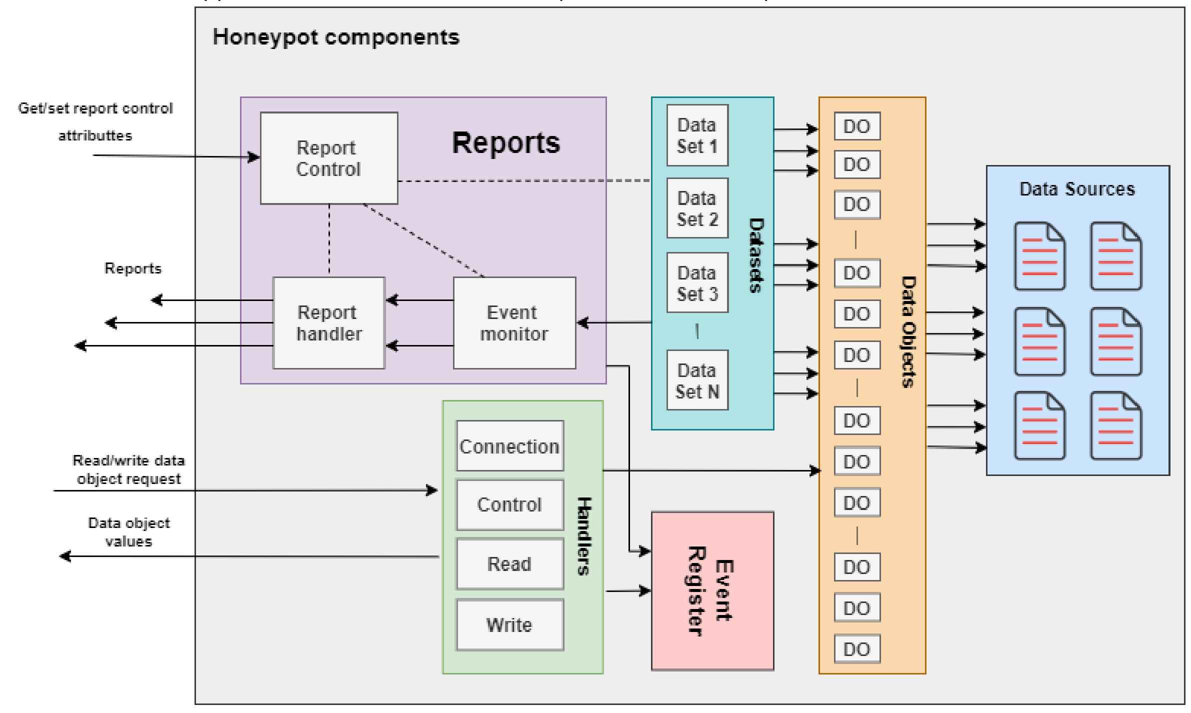


Figure 12. IEC 61850 Components diagram

The Figure 12 represents the component diagram for the IEC61850 honeypot. Following an explanation of each component is presented

- Reports Block

This is the block in charge of the of Report Control functionality. IEC 61850 devices could generate reports based on certain parameters and conditions specified by device model. Low level report control functionality is provided by the lib61850 library and the honeypot application defines which reports will be made available to clients connecting to the honeypot. Every report is associated to a dataset. The Reports Block included into the honeypot architecture supports the two kind of reports included in the IEC61850:

- ***Buffered reports:*** these reports are used to communicate internal events (triggered by different changes into the device: data-change, data-update, etc.) which need to issue an immediate sending of reports or to buffer the events for its transmission. The main feature of this type of report is that it provides the Sequence-Of-Events (SOE) functionality. It guarantees that values of data objects are not lost due to transport flow control or loss of connection.
- ***Unbuffered reports:*** this type of reports could be generated under the same events as the buffered reports, but unbuffered reports do not guarantee that events are transmitted and may be lost if no association exists, or if the data flow transport is not fast enough. The sending of the unbuffered reports is done on a best-efforts basis.

The different reports that the honeypot will serve could not be defined dynamically. Therefore, its definition is made statically at ICD/CID file compilation time as shown in Figure 11. ICD/CID file used for honeypot development contains the description, type and datasets associated to the different reports the honeypot will serve.

- Datasets Block

The information exchange modelling described in IEC61850-7-2 defines the permission to group data objects and data attributes into Datasets. Datasets are key elements of IEC61850 standard reporting functionality described before. Data sets are tied to Logical Nodes, but their member variables can be located at different logical nodes and even different logical devices.

- Handlers

Handlers architectural block's main goal is to register every received IEC61850/MMS request received by the honeypot and implement the required logic to make attackers believe that are dealing with a real IEC 61850 device. Honeypot defines several handler modules to manage each type of MMS access:

- Connection handler

  This is the module that registers new and closed IEC61850/MMS connections on port 102 of the honeypot. In the current version of the honeypot no authentication control is performed as in the majority of IEDs installed on real substations. Nevertheless, this handler is where authentication control could be located in future versions of the honeypot to register attacker's interaction with the authentication module for later investigation.

- Control handler

  Whenever a control operation is received this handler is called by the honeypot. Control handler, apart from registering the control operation request for later analysis, performs additional checks on the controlled object, such as remote/local condition, the command blocked condition, etc. IEC 61850 defines several control models that an IED could implement. In the case of this version of the honeypot "Direct control with normal security" control model has been implemented. An IED control logic could be extremely complex depending on the number and type of Logical Nodes managed. Despite the control logic of the first version of the honeypot is simple it performs the basic operative controls over CSWI logical node as: position reached, interlocking, etc. Probably, during use case test, it might be necessary to include into the honeypot new control capabilities or control models. Control handler design approach will let easy integration of those new capabilities.

- Read Handler

  This is the module in charge of registering every data attribute read request received by the honeypot. This handler could also integrate access control to Data Objects and Data Attributes in future versions.

- Write Handler

  Write access handler captures every IEC61850/MMS write operation on the honeypot and registers it for further analysis and later communication to XL-SIEM communication.

- Event Register

This is the component that the honeypot uses to register the events that need to be recorded for deeper analysis. As described in honeypot technical information this component is based on a proprietary library specifically written for IEC 61850 honeypot which registers events to a log file. Besides, external tools could be used to process event file and extract information to different security analysis tools. This is the approach followed for the interface with the XL-SIEM.

- Data Objects

The IEC61850 standard defines a set of Logical Nodes (LN) which represent from a communication point of view a process function with several Data Objects (DO) associated. The number of DO and data attributes per DO is defined by the used LN type. Logical nodes and the associated DO included into the honeypot have been introduced in section 5.2.1.1.

- Data Sources

Previously presented Data Objects represent information signals that, in real substations, may be routed to SCADA systems, another substation level IEDs or bay control IEDs. In honeypot case, these real signals do not necessarily exist and the existence of data sources to read/write data from/to becomes necessary. The version of the honeypot described in this document makes a static assignation, at initialization time, for Data Attributes values associated to Data Objects and updates those values in memory while the attacker interacts with the honeypot. However, in the case of being necessary the honeypot might be extended to read/write to other data source type.

As it has been described in this section the honeypot is composed of different blocks that work together to build its complete functionality. Each component has been designed to allow extensions in its functionality to include new lessons that will be learnt during pilot demonstrators. These pilot demonstrators could bring an advanced electrical knowledge from utilities and operators which the honeypot might benefit from.

5.2.2 IEC60870-5-104 Honeypot

5.2.2.1 Functional description

IEC60870-5-104 honeypot is based on Conpot honeypot [CON20], which is an open-source low interaction honeypot focused on industrial systems. We use RTU honeypot to emulate the behaviour or real RTU that control devices located in smart grid substations. The RTU can use a variety of industrial communication protocols, but in this case, we focus on IEC 60870-5-104 (IEC104). This protocol is used to monitor energy systems, control systems and their associated communications. It is an asynchronous serial protocol, and it is used for tele-controlling. It adds TCP/IP capabilities and enables connectivity with LAN networks. An RTU device is able to work as a Master or as a Slave in real production systems. Thus, an RTU honeypot emulates the same behaviour.

5.2.2.2 Technical description

Conpot honeypot acts as a Master and provides multiple templates that simulate forms. Table 6 describes the five Conpot templates; the first column has the name of the of the template, the second one is the list of the corresponding ports, and the last column has a short description of the template.

Table 6. Conpot Templates [CON20]

Template	Port	Description

Default template	21, 69, 80, 102, 161, 502, 44818, 47808	It simulates an electric-power plant using Siemens SIMATIC S7-200 Programmable Logic Controllers that communicate with at least two slaves
Guardian_ast	10001	It simulates the Guardian AST tank-monitoring system typically used in gas stations
IPMI	623	It simulates a basic system using the Intelligent Platform Management Interface (IPMI).
Kamstrup_382	1025, 50100	It simulates a system for the Kamstrup 382 electricity meter
IEC 104	2404	It simulates the behaviour of a Siemens S7-300 PLC communicating with the IEC-104 protocol.

5.2.2.3 Prototype architecture and components description

In order to implement the RTU honeypot, some modifications have been performed in Conpot honeypot. Figure 13 presents the main components of Conpot honeypot, and it is highlighted with a red line the component that has been modified. Conpot is executed in the terminal, and it receives a variety of parameters to config the *configuration file* (conpot.cfg) and defines the *template* that is going to be used. This template specifies the type of industrial device to simulate and defines the *protocols* it will use to communicate. If an attacker recognizes the honeypot, it will be identified as an Industrial Control System device due to the type of web interface and the protocols it uses. The interaction between the attacker and the Conpot will be recorded in a *Log*. During that process, the Conpot generates a *response* and send it to the *attacker*.

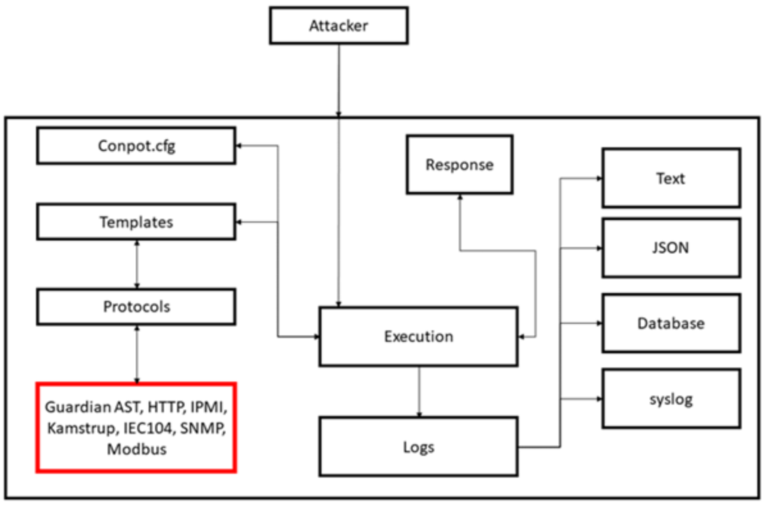


Figure 13. Overview of the Conpot functionality [COH]

IEC COMMUNICATION

As shown in Figure 14, IEC 101/104 communication is exchanged between the controlled and the controlling station [CRW04].

- **Controlled station** or “RTU Slave” is monitored or commanded by a “RTU Master”/ “Controlling Station”.

- **Controlling station** or “RTU Master” is a station where a control of controlled stations is performed (SCADA).

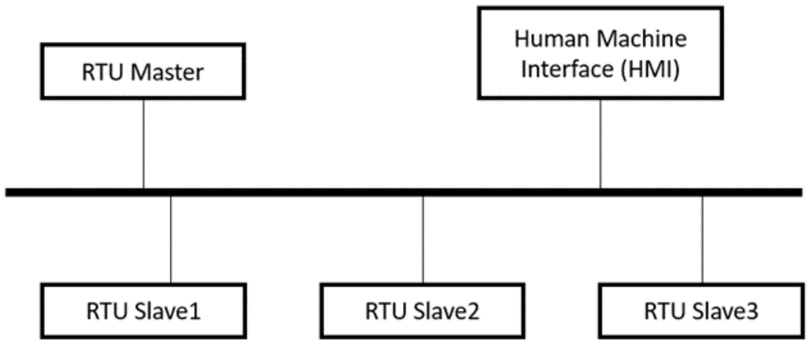


Figure 14. IEC104 topology [MRG19]

IEC 101/104 defines 3 modes of direction:

1. **Monitor Direction** is a direction of transmission from controlled station (RTU) to the controlling station (PC).
2. **Control Direction** is a direction of transmission from controlling station, typical a SCADA system to the controlled station i.e. an RTU.
3. **Reversed Direction** is a direction when monitored station is sending commands and controlling station is sending data in monitor direction.

Figure 15 presents a topology of IEC104 router connected with 104 SCADA monitoring system using IEC104 protocol over TCP/IP and an RTU IEC101.

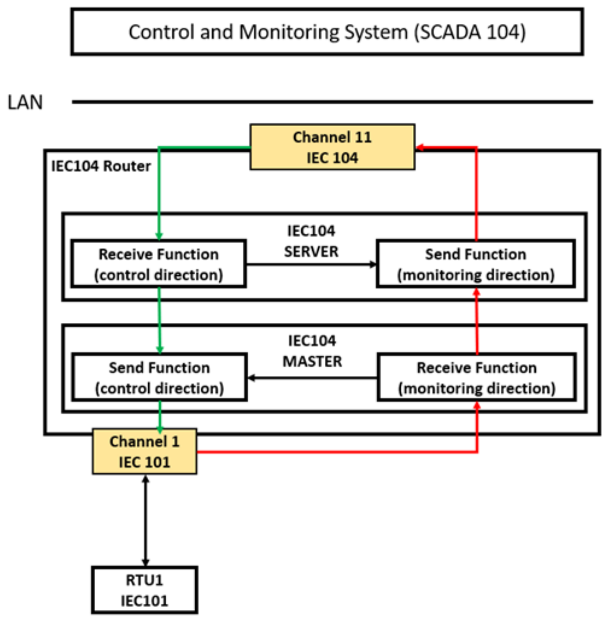


Figure 15. Network topology of SCADA monitoring system [TR-IEC104]

IEC104 is implemented in the application layer of the TCP/IP stack using Application Protocol Data Unit (APDU) and Application Service Data Unit (ASDU). Based on APDU Control field, three APDU formats are defined:

1. **i-frame** to transmit data. It includes two parts (a) fixed-length ASDU header and (b) a variable-length list of information objects.
2. **s-frame** for supervisory operations.
3. **u-frame** to transmit unnumbered control functions (test frame, start transfer, stop transfer)

The ASDU header includes (a) ASDU type (TypeID), (b) the number of transmitted objects, (c) the Cause of Transmission (COT) and (d) an ASDU address (station address).

The ASDU Type Identification Field (TypeID): The range of numbers 1 to 127 is used for standard definitions from IEC 60870-5-101 standard. The range 128 to 135 is reserved for routing of messages. The numbers 136 up to 255 are for special use. In the range of standard type definitions, there are presently 58 specific types defined. These are grouped as shown in the Table 7. The Table includes for each group of processes the range of the TypeIDs, the name of the Group, the TypeID, the Code of it, a short description and in last two columns, we mentioned if this TypeID (Conpot command) is implemented by Conpot or by SDN-microSENSE..

Table 7. The definition of TypeID numbers for process and system information in monitor and control direction

Defined TypeID	Group	TypeID	Code	Description	Conpot support	Extended by SDN_microSENSE
1 – 40	Process information in monitoring direction	1	M_SP_NA_1	Single point information	YES	
		2	M_SP_TA_1	Single point information with time tag	YES	
		3	M_DP_NA_1	Double point information	YES	
		4	M_DP_TA_1	Double point information with time tag	YES	
		11	M_ME_NB_1	Measured value, scaled value	YES	
		12	M_ME_TB_1	Measured value, scaled value with time tag	YES	
		13	M_ME_NC_1	Measured value, short floating point value	YES	
		14	M_ME_TC_1	Measured value, short floating	YES	

				point value with time tag		
		30	M_SP_TB_1	Single point information with time tag CP56Time2a	YES	
		31	M_DP_TB_1	Double point information with time tag CP56Time2a	YES	
		35	M_ME_TE_1	Measured value, scaled value with time tag CP56Time2a	YES	
		36	M_ME_TF_1	Measured value, short floating point value with time tag CP56Time2a	YES	
45 – 51	Process information in control direction	45	C_SC_NA_1	Single command	YES	
		46	C_DC_NA_1	Double command	YES	
		49	C_SE_NB_1	Setpoint command, scaled value	YES	
		50	C_SE_NC_1	Setpoint command, short floating point value	YES	
70	System information in monitor direction					
100 – 106	System information in control direction	100	C_IC_NA_1	General-Interrogation command	YES	
		101	C_CI_NA_1	Counter interrogation command	NO	YES
		102	C_RD_NA_1	Read command	NO	YES
		102	C_CS_NA_1	Clock synchronizat	NO	YES

				ion command		
		104	C_TS_NB_1	Test command	NO	YES
		105	C_RP_NC_1	Reset process command	NO	YES
		106	C_CD_NA_1	Delay acquisition command	NO	YES
		107	C_TS_TA_1	Test command with time tag CP56Time2a		
110 – 113	Parameter in control direction				NO	
120 - 136	File transfer				NO	

In the following section there are more details about the new TypeID (i.e. the Conpot commands) that we are implemented by SDN-microSENSE. For each command, it is mentioned the Name of the command, the information object type, the Code, the Sequence (SQ) bit that specifies the method of addressing the information objects and the valid Cause of Transmission (COT) codes that are supported (Control or Monitor).

1. **Counter interrogation command**
   Information Object Type: 101
   Code: C_CI_NA_1
   Valid with SQ: 0
   Information Object for SQ = 0 (Sequence of Information Objects)
   Single information object only

Information Object Address	IOA = 0
CP8	QCC (Qualifier of counter command)

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Control	8	Deactivation
Monitor	7	Activation Confirmation
Monitor	9	Deactivation confirmation
Monitor	10	Activation termination
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission

Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

2. **Read command**

Information Object Type: 102

Code: C_RD_NA_1

Valid with SQ: 0

Information Object for SQ = 0 (Sequence of Information Objects)

Single information object only

Information Object Address	IOA = 0

There is only the IoA with the data unit identifier for this command.

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
	5	Request

Information Object Address	IOA = 0
CP56Time2a	Seven-octet binary time

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Monitor	3	Spontaneous
Monitor	7	Activation Confirmation
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission
Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

3. **Clock synchronization command**

Information Object Type: 103

Code: C_CS_NA_1

Valid with SQ: 0

Information Object for SQ = 0 (Sequence of Information Objects)

Single information object only

Information Object Address	IOA = 0

CP56Time2a	Seven-octet binary time

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Monitor	3	Spontaneous
Monitor	7	Activation Confirmation
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission
Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

4. **Test command**

   Information Object Type: 104

   Code: C_TS_NB_1

   Valid with SQ: 0

   Information Object for SQ = 0 (Sequence of Information Objects)

   Single information object only

Information Object Address								IOA = 0
1	0	1	0	1	0	1	0	FBP (Fixed test pattern)
0	1	0	1	0	1	0	1	

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Monitor	7	Activation Confirmation
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission
Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

5. **Reset process command**

   Information Object Type: 105

   Code: C_RP_NC_1

   Valid with SQ: 0

   Information Object for SQ = 0 (Sequence of Information Objects)

   Single information object only

Information Object Address	IOA = 0
UI8	QRP (Qualifier of reset process command)

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Monitor	7	Activation Confirmation
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission
Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

6. **Delay acquisition command**

   Information Object Type: 106

   Code: C_CD_NA _1

   Valid with SQ: 0

   Information Object for SQ = 0 (Sequence of Information Objects)

   Single information object only

Information Object Address	IOA = 0
CP16Time2a	Two-octet binary time

Valid Cause of Transmission (CoT) codes

Direction	Code number	Code Description
Control	6	Activation
Monitor	7	Activation Confirmation
Monitor	44	Unknown type identification
Monitor	45	Unknown cause of transmission
Monitor	46	Unknown common address of ASDU
Monitor	47	Unknown information object address

5.2.3 Modbus Honeypot

5.2.3.1 Functional description

Modbus is an application-layer protocol used for connecting and controlling industrial systems [PSL+20]. The first version of Modbus was released in 1979 and nowadays is adopted by multiple energy-related stakeholders due to its simplicity. Modbus is coming with two versions known as a) Modbus RS485 or Modbus Serial and b) Modbus TCP/IP. The first one follows a master/slave communication paradigm, while the second one takes full advantage of the TCP/IP stack and thus uses the server/client model. Modbus supports a variety of operations interpreted into particular function codes. Although many critical infrastructures adopt Modbus, it is characterised by severe cybersecurity issues since it does not comprise sufficient authentication and authorisation mechanisms. Consequently, potential cyberattackers can execute a plethora of cyberattacks [HCS08] [RSL+20] against Modbus, including unauthorised access attacks, Man in the Middle (MITM), Modbus-related reconnaissance attacks, Modbus-related DoS/DDoS, replay attacks, false data injection and data modification attacks.

Based on the aforementioned remarks, the Modbus honeypot aims to imitate both server and client devices using Modbus/TCP, thus misleading potential cyber attackers and hiding the real assets. The implementation of the Modbus honeypot relies on Conpot, a Python-based software framework for implementing Industrial Control System (ICS) honeypots [DSI+19] . However, it is noteworthy that Conpot does not support all Modbus/TCP function codes and cannot impersonate efficiently the responses of the client devices since the response Modbus/TCP packets include a predefined payload value, which reveals the honeypot identity.

Therefore, in the context of SDN-microSENSE, the Modbus honeypot enhances the current functionalities provided by Conpot based on three main research and development pillars:

- **Emulating both server and client Modbus entities**: The Modbus honeypot has the ability to emulate both entities operating like a server, such as an RTU or client entities like an HMI.
- **Supporting more Modbus/TCP function codes not available in Conpot**. The current version of Conpot does not support adequately all Modbus function codes. In the context of the SDN-microSENSE project, the Modbus honeypot adopts Conpot in order to emulate the server side of the Modbus TCP/IP communication, thereby incorporating more Modbus function codes.
- **Imitating the Modbus/TCP behaviour of real assets**: Both sides of the Modbus honeypot (i.e., server and client) have the ability to mimic the behaviour of real devices, generating similar Modbus/TCP network traffic.

5.2.3.2 Technical description

Based on the previous research and development pillars, concerning the server-side, the Modbus honeypot relies on Conpot supporting the Modbus function codes provided in Table 7. It is worth mentioning that function codes 22, 23 and 24 are not supported by the current Conpot version, and they are implemented using the modbus-Tk [VKS15], PyModbus [COL17] and Scapy Python [RRM+18] libraries. More information about these Modbus function codes is provided in [MODBUS]. On the other hand, the client-side of the Modbus honeypot is implemented through a Python script, which also uses the aforementioned Python programming libraries.

Table 7. Modbus honeypot Function Codes

Function Code	Name	Explanation	Supported by Conpot
FC01 (0x01)	Read Coils	It requests the status of a discrete coil (1-bit registers with read/write permissions).	√
FC02 (0x02)	Read Discrete Inputs	It requests the status of discrete inputs (1-bit registers with read-only permission)	√
FC03 (0x03)	Read Holding Registers	It requests the value of 16-bit holding registers (read/write permissions)	√

FC04 (0x04)	Read Input Registers	It requests the value of 16-bit input registers (read-only permission)	√
FC05 (0x05)	Write Single Coil	It changes the value of a single coil.	√
FC06 (0x06)	Write Single Register	It changes the value of a single holding register.	√
FC15 (0x0F)	Write Multiple Coils	It changes multiple coil addresses in a single command.	√
FC16 (0x10)	Write Multiple registers	It changes multiple holding registers in a single command.	√
FC17 (0x11)	Report Slave ID (Serial Line only)	It reads various information related to the remote device, such as its status and the type description.	√
FC22 (0x16)	Mask Write Register	It changes the content of a holding register based on the use of AND and/or OR logical masks and the holding register’s current content.	X
FC23 (0x17)	Read/Write Multiple registers	It operates as both read and write operations in a single Modbus command. The write process preceded the read process.	X
FC24 (0x18)	Read FIFO Queue	It reads the content of a register’s First-In-First-Out (FIFO) queue	X
FC43/13 (0x2B / 0x0D)	CANopen General Reference Request and Response PDU	It encloses all services used to read or write the records a CAN-Open Device Object Dictionary as well as to monitor and handle the CANopen system.	√
FC43/14 (0x2B / 0x0E)	Read Device Identification	It is used to read information related to the functional and physical description of the remote device.	√

Regarding the third research and development pillar mentioned above, the imitation of the Modbus payload values is carried out via a Generative Adversarial Network (GAN) [YU19] [PYY+19]. Figure 15 shows in an abstract way the operation flow of the Modbus honeypot, which operates as a server. In particular, a PCAP file including the Modbus/TCP network traffic of the real asset is used to extract the respective Modbus/TCP addresses and values from the Modbus/TCP response packets (i.e., those packets that are returned as answers to the corresponding Modbus/TCP request packets). The Modbus values are used for the training of GAN, which then can generate similar Modbus/TCP payload values. On the other side, the Modbus addresses are stored in a configuration file, which indicates to Conpot how the Modbus/TCP payload values generated by GAN will be correlated with the respective Modbus addresses. Thus, Conpot, part of the Modbus honeypot is capable of emulating an asset operating as a server, generating similar Modbus/TCP payload values.

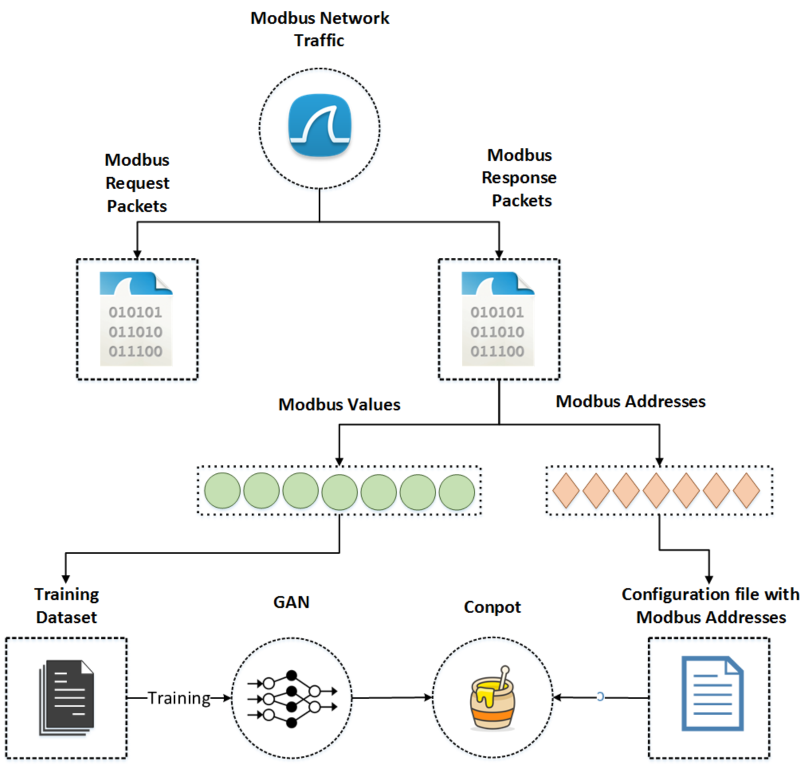


Figure 16. Modbus honeypot Operation Flow as Server

Figure 16 depicts the structure of GAN used for generating Modbus/TCP payload values when Modbus honeypot operates as a server. It consists of three main modules, namely a) Input Module, b) Generator Module and c) Discriminator Module. The Input Module generates noise data that are used by the Generator Module to produce the Modbus/TCP payload values imitating the real ones. Finally, the Discriminator Module is used only during the training process and classifies the values generated by the generator (during the training process) as real or fake, thus optimising the effectiveness of the Generator Module.

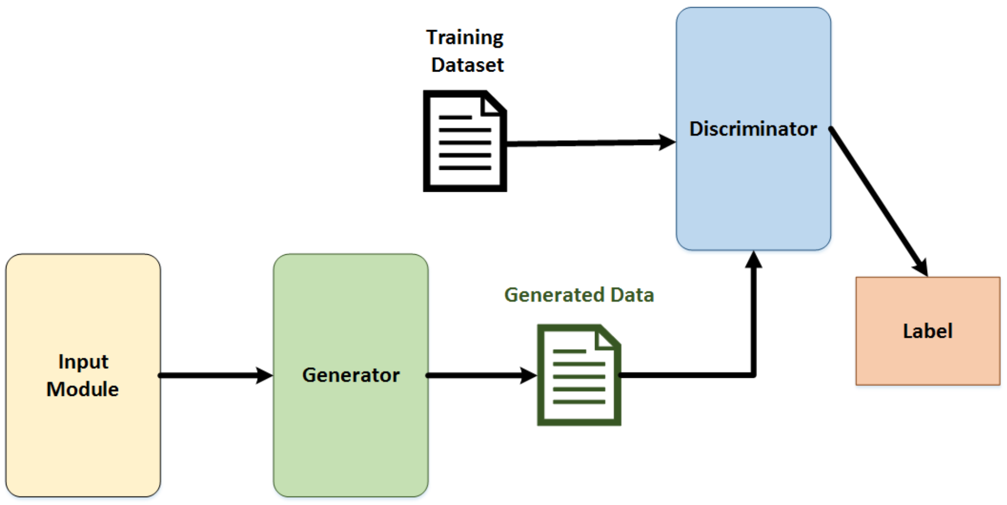


Figure 17. Modbus honeypot GAN Structure

Figure 17 shows the operation flow when the Modbus honeypot operates as a client. Similarly to the server operation, a PCAP file including the Modbus/TCP network traffic of the real asset is used to extract in a CSV file the Modbus/TCP function codes, the starting addresses and the quantity of the requested information (e.g., registers, coils, discrete inputs). Then, this data is used to construct and transmit the Modbus/TCP requests to specific targets.

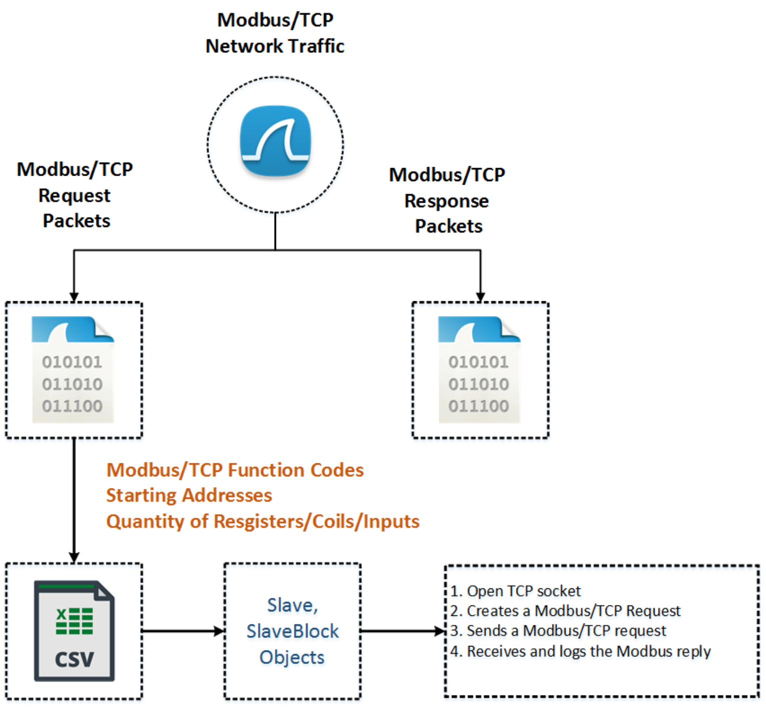

Modbus/TCP Network Traffic
Modbus/TCP Request Packets
Modbus/TCP Response Packets
010101 011010 011100
010101 011010 011100
Modbus/TCP Function Codes
Starting Addresses
Quantity of Resgisters/Coils/Inputs
CSV
Slave, SlaveBlock Objects
1. Open TCP socket
2. Creates a Modbus/TCP Request
3. Sends a Modbus/TCP request
4. Receives and logs the Modbus reply

Figure 18. Modbus honeypot Operation Flow as Client

5.2.3.3 Prototype architecture and components description

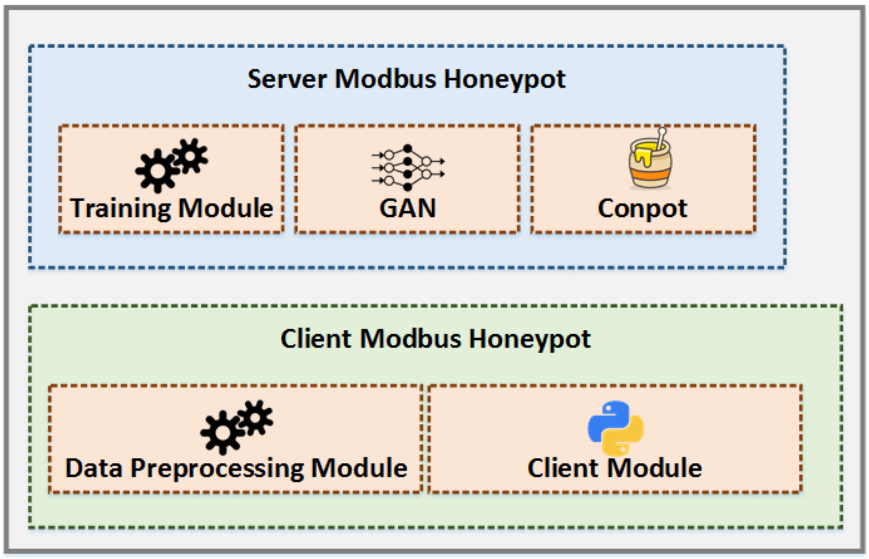


Figure 19. Modbus honeypot Architecture

Figure 19 presents the overall architecture of the Modbus honeypot, which can operate either as a server or client. In the first case where the Modbus honeypot operates as a server, it is composed of three components:

- **Training Module**: The Training Module undertakes the training of GAN incorporated into Conpot, so that it can generate Modbus/TCP payload values similar to the real assets. The previous section and specifically Figure 16 illustrates and explains how the Modbus honeypot works when it emulates a server device.
- **GAN**: GAN is responsible for generating Modbus/TCP payload values similar to those generated by the corresponding real asset. In the previous section, Figure 17 depicts the structure of GAN. Moreover, the previous section explains in detail how the proposed GAN works.
- **Conpot**: Conpot interacts with the respective communication parties, transmitting the Modbus/TCP response packets. It encloses the appropriate data structures and characteristics for emulating a server device.

On the other side, when the Modbus honeypot operates as a client it consists of the following components:

- **Data Pre-processing Module**: The Data Pre-processing Module receives a PCAP file including Modbus network traffic data and extracts a CSV file containing the Modbus function codes, the starting addresses, and the quantity of the information values for the corresponding Modbus data object (e.g., register, discrete input, coil). The operation flow of the Modbus honeypot used as a client is described in the previous section. In particular, shows how this operation flow is carried out.
- **Client Module**: The CSV file generated by the Data Pre-processing Module is used by the Client Module in order to construct Modbus/TCP request packets similar to those of the real device.

5.2.4 Honeypot Manager

5.2.4.1 Functional description

The Honeypot Manager is a component that belongs to the S-RAF component in the application layer (See Figure 4) and it has two main functionalities in the SDN-microSENSE project:

1. Provides means to develop in an automatic way those honeypots that the security operator or manager decide to deploy in the network.
2. Processes the information about unknown anomalies (like zero-day attacks) provided by the XL-SIEM and to indicate to the SDN controller the required information to re-arrange the honeypots network in order to collect information for these unknown anomalies detected by the ML models of the XL-EPDS.

Figure 20 shows the logic process followed by the Honeypot Manager to carry out its two main functionalities.

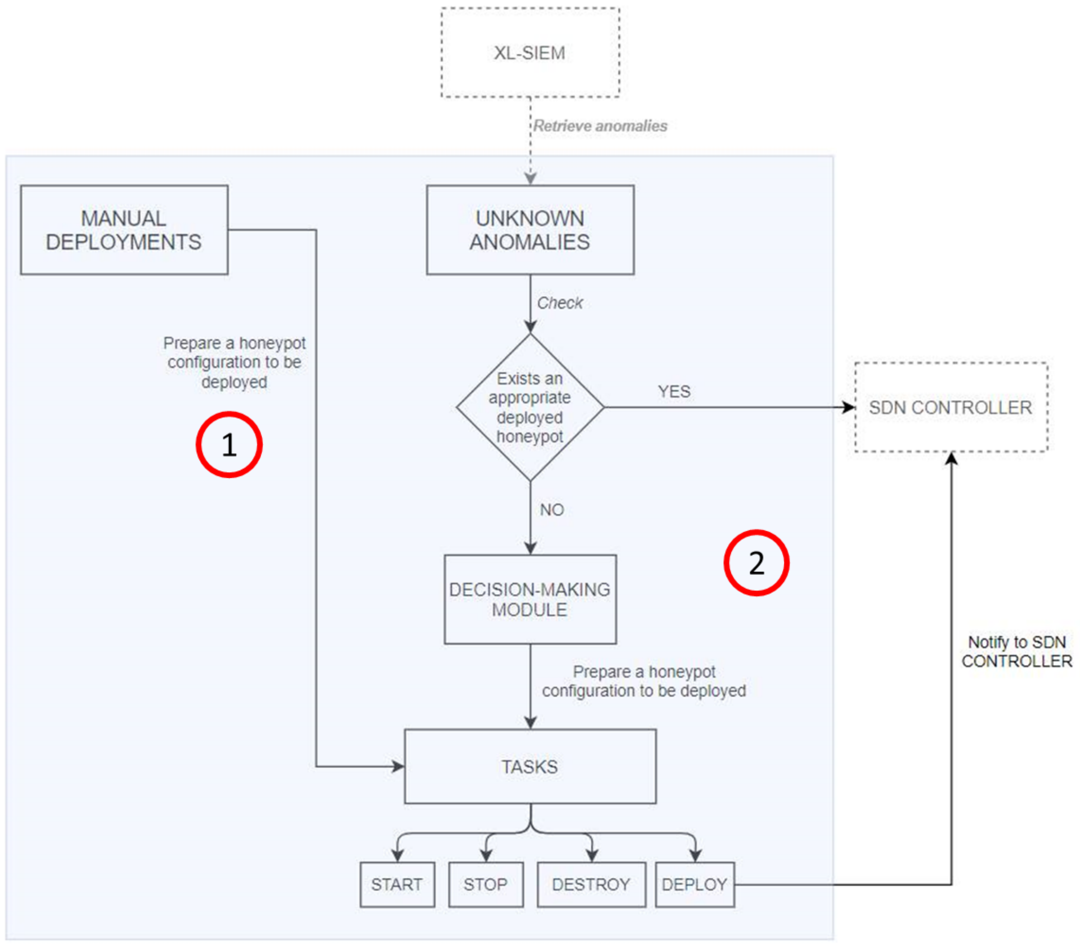


Figure 20. Honeypot Manager – Main functionalities

Honeypot Manager – Deployment functionality

The deployment functionality is based on the business logic defined in the SPEAR project [SPE19] but with significant changes. The first change is the technology used, it has been decided to change the technology to do the Honeypot Manager more extensible and scalable. The second and more important change is that it has been included the functions from which that the security operator /manager in charge of the security actions of the network can select the honeypot that want to deploy in the network as an additional level of security.

This functionality is responsible for automating and optimizing the honeypot deployment. Multiple actions are performed related to define a specific honeypot configuration:

- It handles the lifecycle of the virtual machines in which the honeypots will be deployed. It creates, configures and destroys virtual machine instances within the infrastructure. Each of the honeypots planned to be installed in the EPES infrastructure as security mechanism are deployed in a separate virtual machine instance.
- It handles the lifecycle of the honeypots to be deployed as security mechanism in the EPES infrastructure; it installs, configures, starts and stops the corresponding honeypots. There is a catalogue of different honeypots that can be planned to deploy.

Honeypot Manager – Unknown anomalies functionality

This functionality has the final objective of obtaining more information of how the IDEs can react in case of an unknown attack (zero-day attack).

In order to gather this information, the Honeypot Manager analyse the information received from the XL-SIEM about zero-day attack alerts. The process for creating zero-day attack alerts involves several components, as depicted in Figure 21.

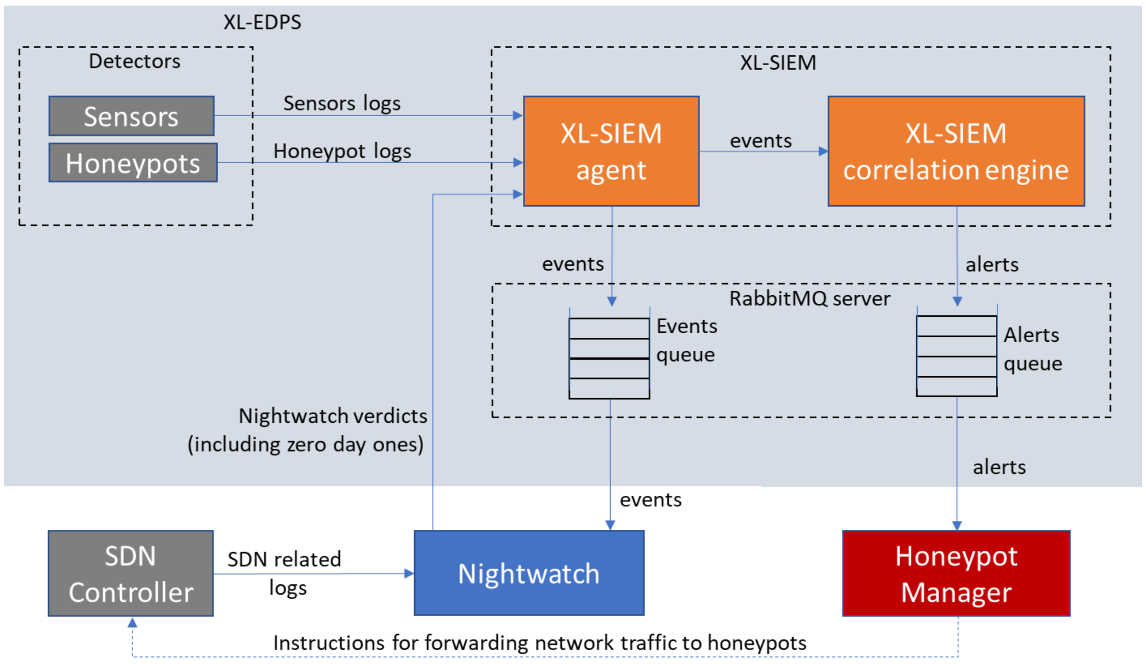


Figure 21. General overview of the Honeypot Manager and its interaction with other SDN-microSENSE components

Zero-days attacks are detected by the Nightwatch component (XL-EDPS component), using information taken from the SDN Controller and from detectors (security sensors and honeypots) deployed in the infrastructure to monitor. Nightwatch receives events in real time through a RabbitMQ queue where the XL-SIEM export the normalised events. Additionally, Nightwatch receives logs from the SDN controller related to network metrics (i.e., transmission rate). Both, events and SDN controller logs are used by the Nightwatch component to infer the occurrence of zero-day attacks. These positive occurrences are injected back in the XL-EDPS which will be used by the XL-SIEM to generate alerts associated to these zero-day attacks verdicts. These alerts generated by the XL-SIEM, including zero-day attack ones, are exported to a RabbitMQ queue and are consumed by the Honeypot Manager.

To optimize the deployment of components and to simplify the usage of interfaces, the Nightwatch detector reuses the communication channel established between detectors and the XL-SIEM: sensors and honeypots logs are sent to the XL-SIEM agent.

Once the Honeypot Manager receives the unknown anomalies alerts, it analyses them in order to obtain information and to check if there is a possibility to rearrange the network traffic to one of the honeypots that are deployed in the EPES network. Two possibilities could be found:

- The first possibility is that a honeypot that could receive traffic is already deployed in the infrastructure, so in this case the Honeypot Manager utilises the REST API of the SDN controller in order to instruct the transparent redirection of traffic, originally destined to the real device, towards the honeypot. The Honeypot Manager has already configured appropriately the honeypot to share the same network information with the real device, i.e. same MAC and same IP address. In this way, the attacker will assume that he communicates with the real device, although he communicates with the honeypot.
- The second possibility is that in the honeypot catalogue of the Honeypot Manager, there is a honeypot capable to emulate the protocol related to the unknown anomaly, however this honeypot has not been deployed in the EPES network. In this case, the Honeypot Manager a) informs the Security Administrator that a honeypot with specific configurations needs to be deployed, b) the Security Administrator decides to deploy this honeypot, c) the Honeypot Manager deploys the honeypot and d) once the honeypot is running, the Honeypot Manager sends the appropriate REST API commands to the SDN controller in order to redirect the traffic, similarly to the first case.

5.2.4.2 Technical description

Back-end:

The Honeypot Manager's back-end consists of a Spring Boot application, managed by Maven, and contains the following modules:

- Database: A MySQL database is accessed using Spring JPA Repository technology.
- REST API: A definition of REST services using Spring REST technology.
- Security: All REST services are secured through a role-based system. In addition, to access each of the REST services it is necessary to be authenticated through JSON Web Tokens (JWT technology).

Front-end:

Honeypot Manager's front-end consists of a web application developed in HTML5 + CSS3 technologies. The selected scripting language to perform all the operations is AngularJS.

5.2.4.3 Prototype architecture and components description

The Honeypot Manager tool has been designed following the high-level architecture shown in Figure 22.

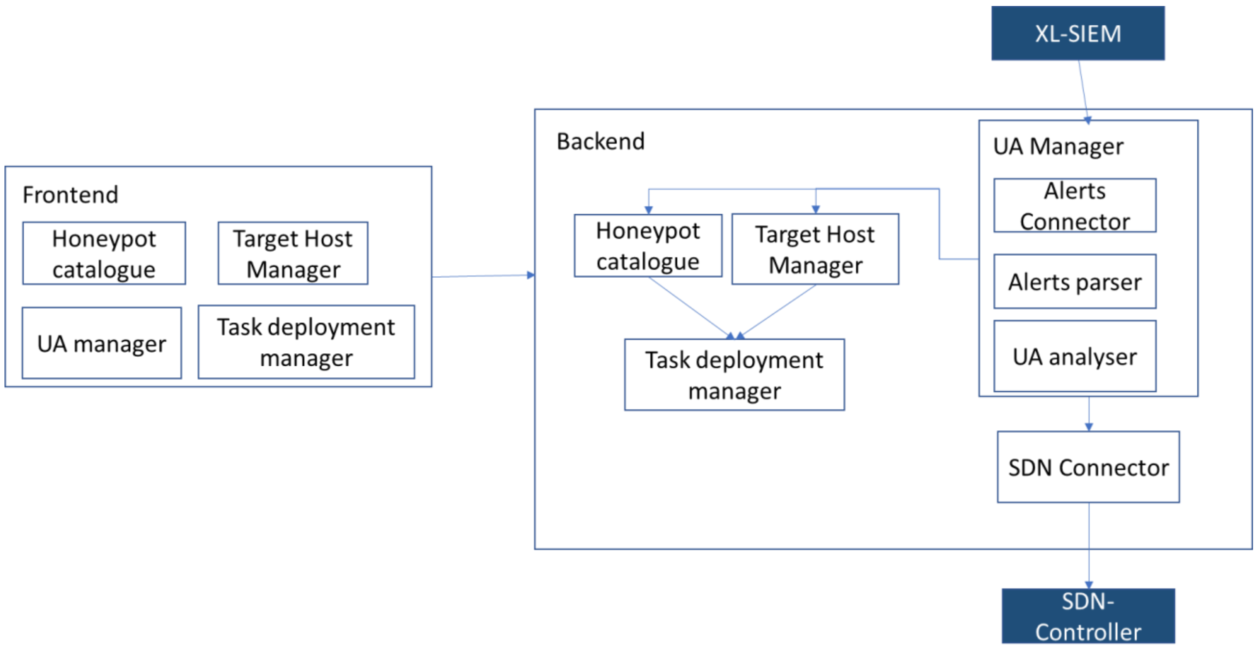


Figure 22. Honeypot Manager - High Level Architecture

The main components are:

- The **front-end** is composed by four subcomponents in charge of presenting to the Security operator all the information that could be useful for managing the honeypots deployed in the EPES network.
- The **back-end** component oversees the execution of the two main functionalities of this tool: deployment of the honeypots and analysing the unknown anomalies (UA) to rearrange the traffic to the honeypots. The back-end is also responsible to manage the communication with the XL- SIEM and the SDN-controller.

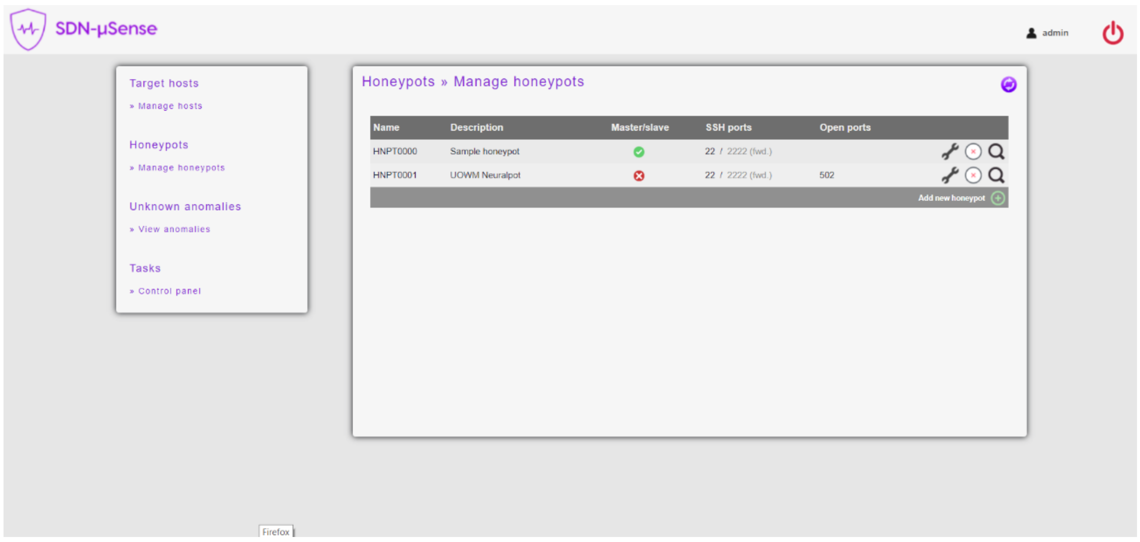
Following, these two components are described in detail:

Front-end

This component has the main objective of presenting to the security operator all the information available about the honeypot deployed in the EPES network. Additionally, through this front end, the security operator can select and configure the honeypots to be deployed and manage (run and stop) the virtual machines where the honeypots are deployed.

The main subcomponents are:

1. Honeypot catalogue



Honeypots » Manage honeypots

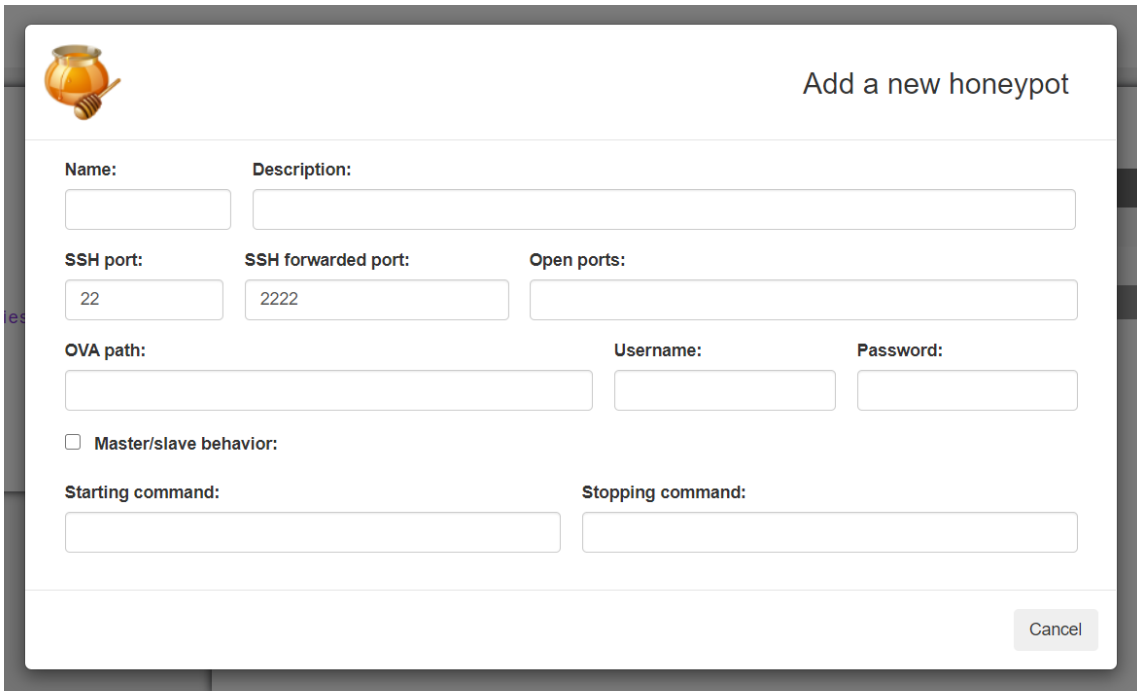
Name	Description	Master/slave	SSH ports	Open ports
HNPT0000	Sample honeypot	✓	22 / 2222 (fwd.)	
HNPT0001	UOWM Neuralpot	✗	22 / 2222 (fwd.)	502

Add new honeypot

Figure 23. Main Screen - Honeypot catalogue

This section shows the honeypot catalogue. The following options are available:

- Add/edit honeypot: It allows to insert/update a honeypot through the following dialog:

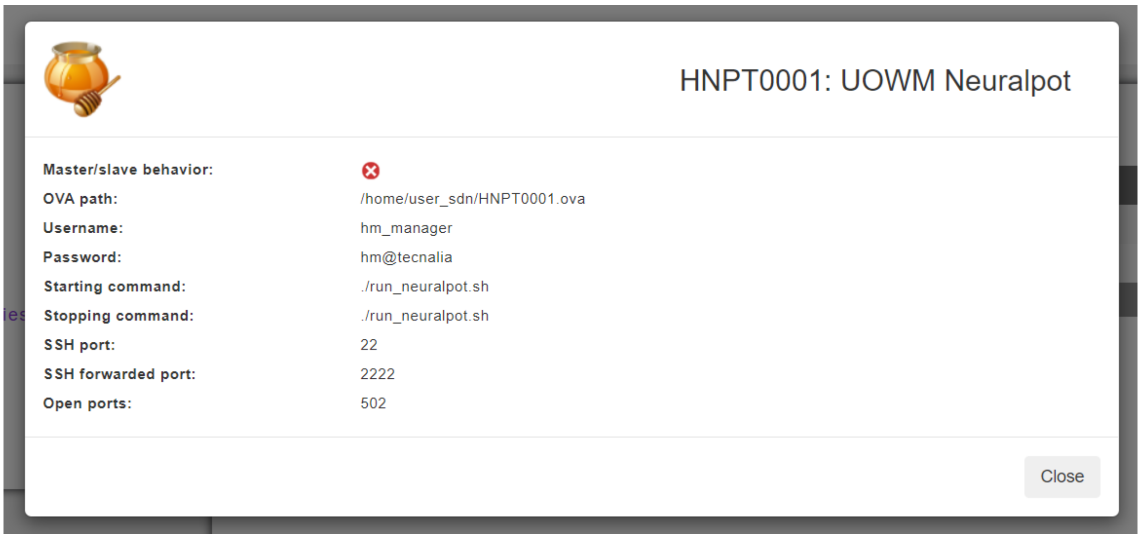


Add a new honeypot

Name:
Description:
SSH port: 22
SSH forwarded port: 2222
Open ports:
OVA path:
Username:
Password:
☐ Master/slave behavior:
Starting command:
Stopping command:

Figure 24. Add/edit option - Honeypot catalogue

- View detail: It allows to consult all the information stored of a honeypot:

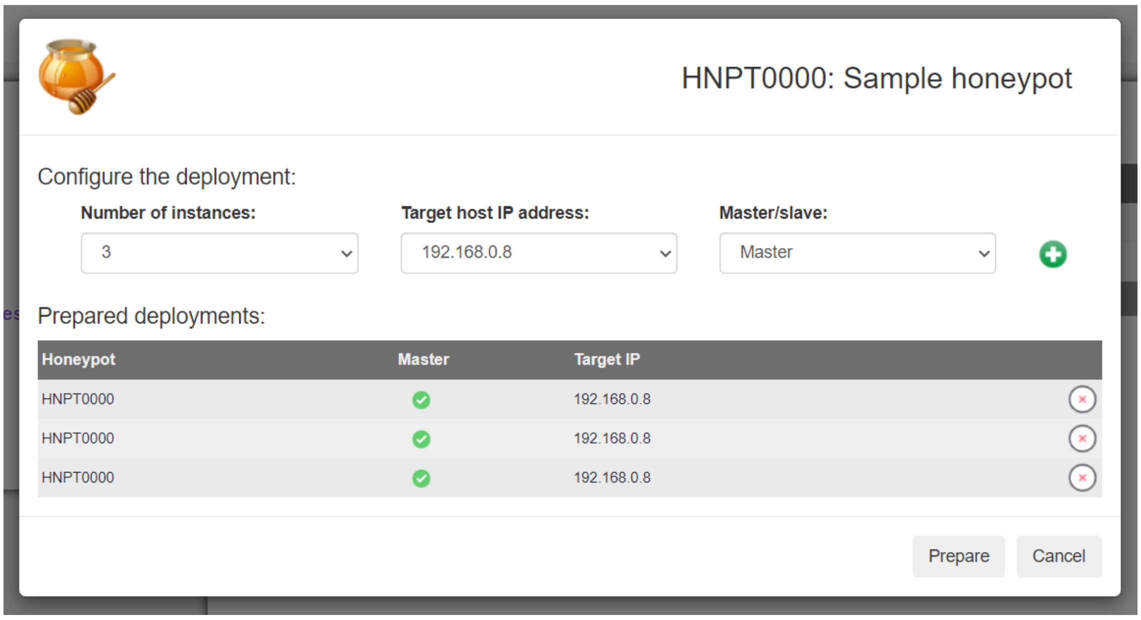


HNPT0001: UOWM Neuralpot

Master/slave behavior:	✗
OVA path:	/home/user_sdn/HNPT0001.ova
Username:	hm_manager
Password:	hm@tecnalia
Starting command:	./run_neuralpot.sh
Stopping command:	./run_neuralpot.sh
SSH port:	22
SSH forwarded port:	2222
Open ports:	502

Figure 25. View option - Honeypot catalogue

- Delete honeypot: It allows to remove a honeypot from the database when it is not being used by a task.
- Prepare for deploy: This option allows to prepare a honeypot to be deployed. Through this dialog:



HNPT0000: Sample honeypot

Configure the deployment:

Number of instances: 3
Target host IP address: 192.168.0.8
Master/slave: Master

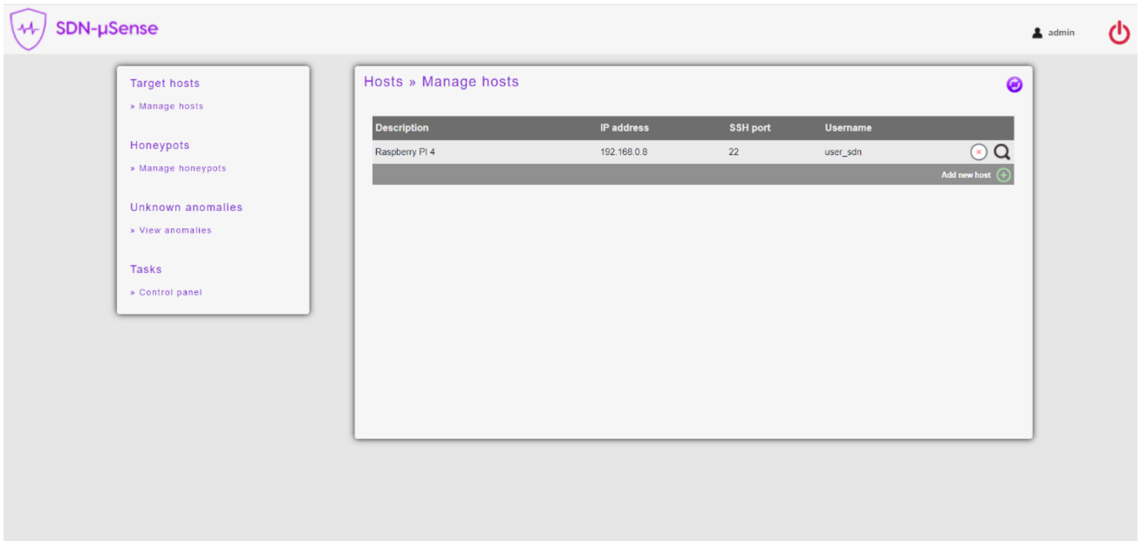
Prepared deployments:

Honeypot	Master	Target IP
HNPT0000	✓	192.168.0.8
HNPT0000	✓	192.168.0.8
HNPT0000	✓	192.168.0.8

Figure 26. Prepare deployment option - Honeypot catalogue

The security operator chooses the honeypots and where he wants to deploy them. Once the configuration is selected, it is converted to deployment tasks.

2. Target host manager.

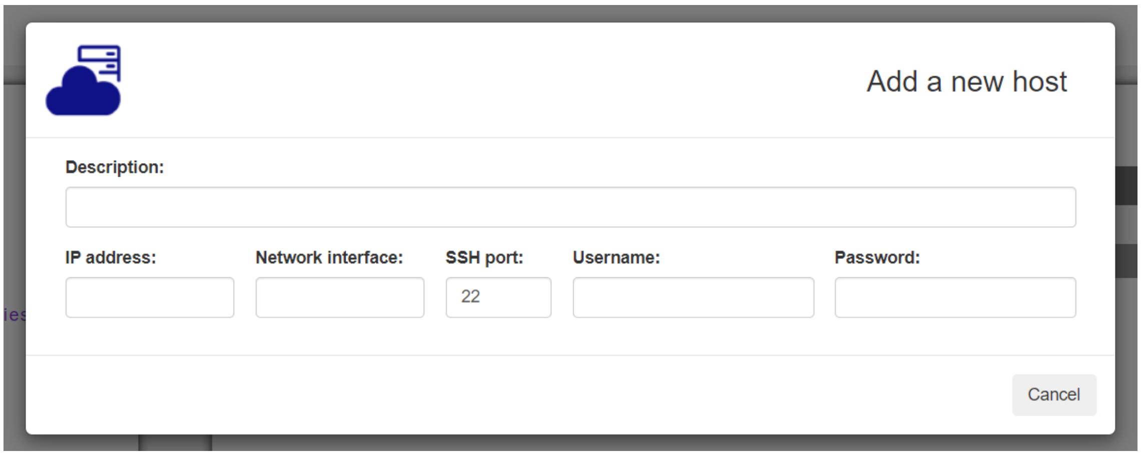


Description	IP address	SSH port	Username
Raspberry PI 4	192.168.0.8	22	user_sdn

Figure 27. Main Screen - Target host manager

The target host manager shows the list of the stored host machines. The following actions are available:

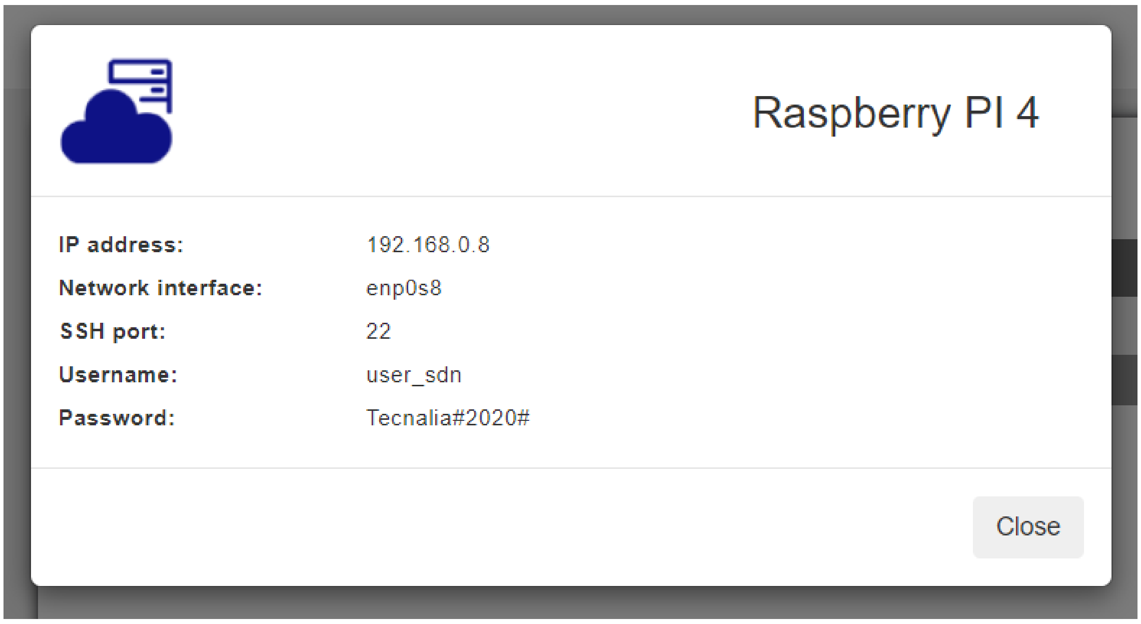
- Add/edit host: It allows to insert/update a host through the following dialog:



Description:				
IP address:	Network interface:	SSH port:	Username:	Password:
		22		

Figure 28. Add/edit host option - Target host manager

- View detail: It allows to consult all the information stored of a host:



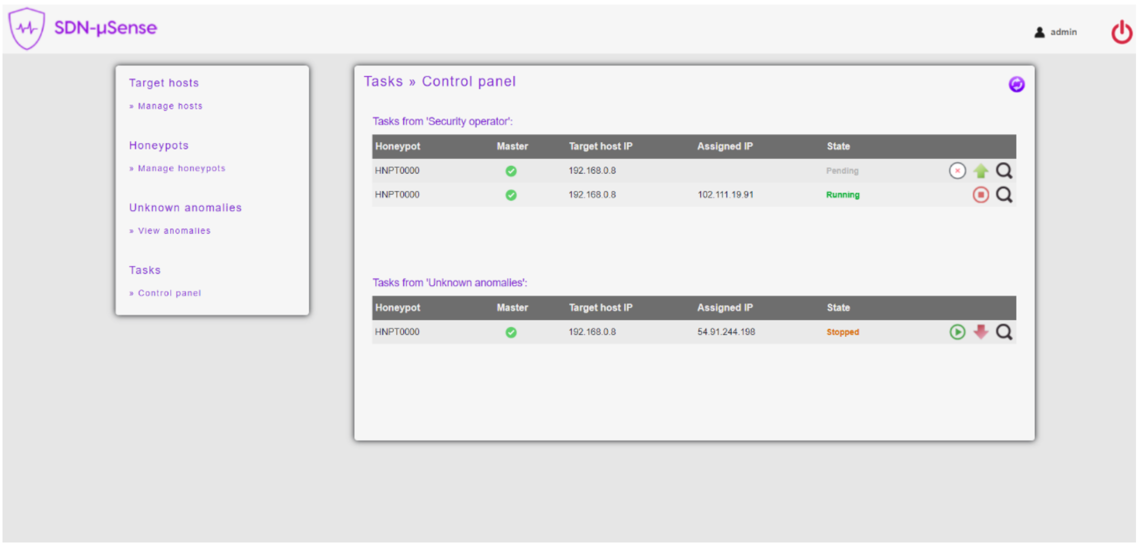
Raspberry PI 4

IP address:	192.168.0.8
Network interface:	enp0s8
SSH port:	22
Username:	user_sdn
Password:	Tecnalia#2020#

Figure 29. View host option - Target host manager

- Delete host: It allows to remove a host from the database.

3. Task deployment manager



Tasks » Control panel

Tasks from 'Security operator':

Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8		Pending
HNPT0000	✓	192.168.0.8	102.111.19.91	Running

Tasks from 'Unknown anomalies':

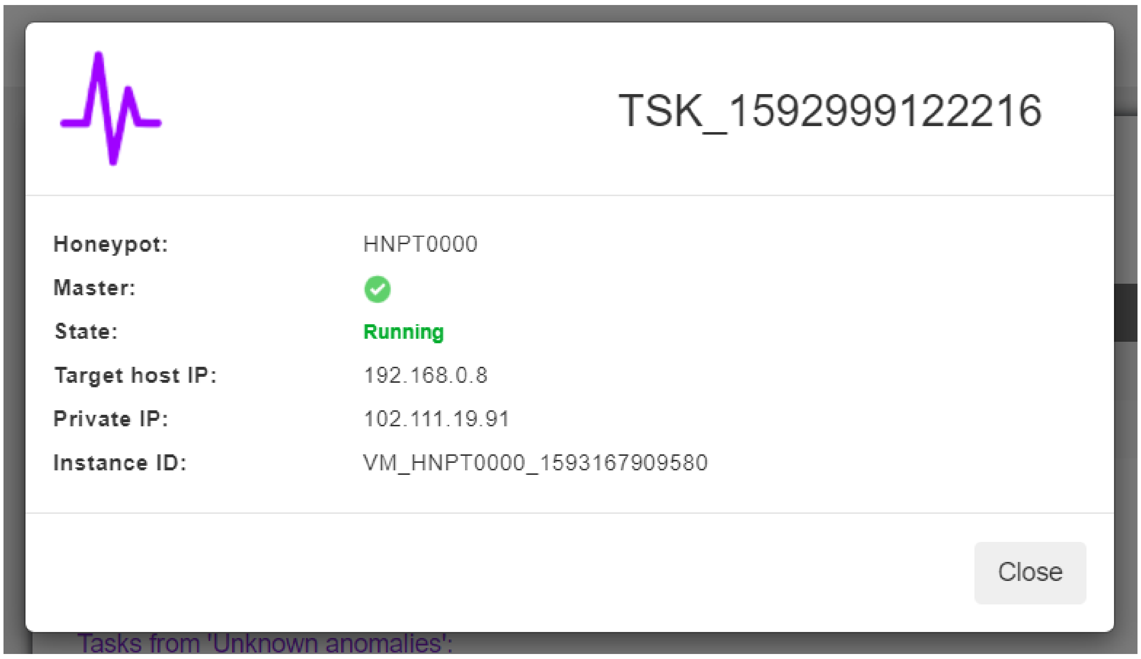
Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8	54.91.244.198	Stopped

Figure 30. Main Screen – Task deployment manager

This section manages the prepared deployment tasks. There are two tables: the first one shows the prepared tasks by the selection of the security operator, and the second one shows the tasks prepared based on the analysis done by the Unknown Anomalies component.

The following options are available:

- View detail: It allows to consult all the information about the task



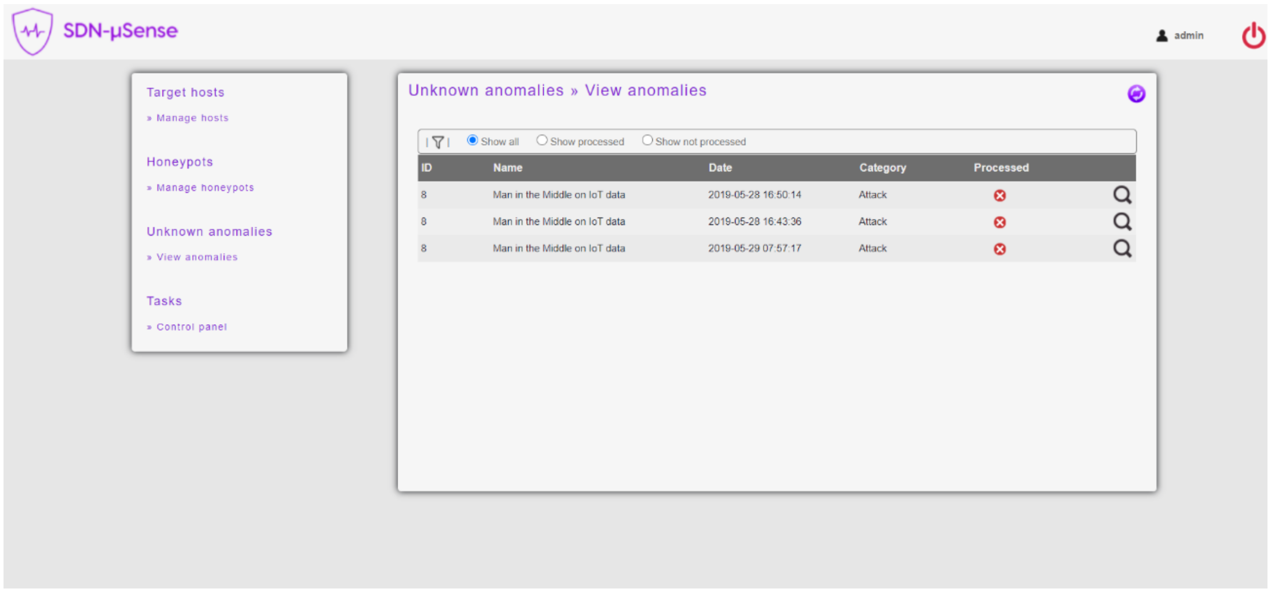
TSK_1592999122216

Honeypot:	HNPT0000
Master:	✓
State:	Running
Target host IP:	192.168.0.8
Private IP:	102.111.19.91
Instance ID:	VM_HNPT0000_1593167909580

Figure 31. View Option – Task deployment manager

- Delete task: It allows to remove a task from the database. This option is only available when a task is “Pending” or “Finished”.
- Deploy task: When the task state is “Pending”, this option allows to deploy a honeypot.
- Undeploy task: When the task state is “Stopped”, this option allows to undeploy a honeypot.
- Start task: When a task state is “Stopped”, this option allows to re-start the honeypot.
- Stop task: When a task state is “Running”, this option allows to stop the honeypot.

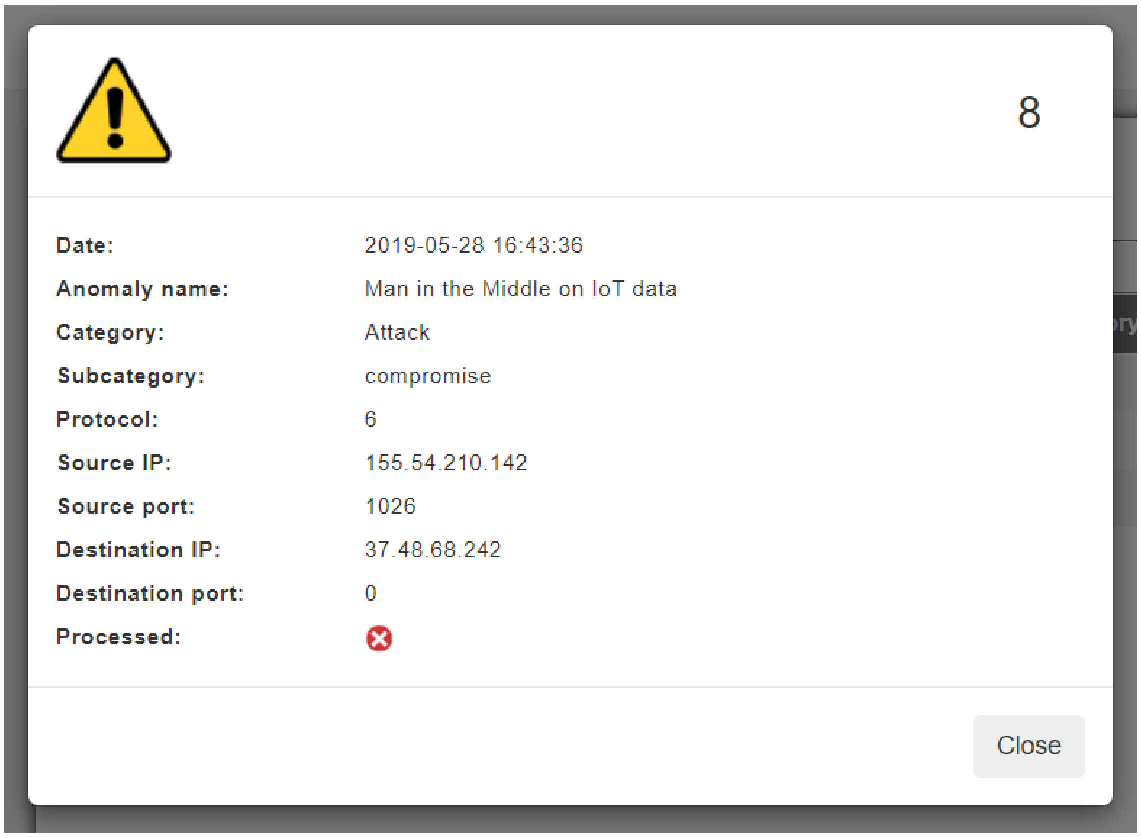
4. Unknown Anomalies manager



ID	Name	Date	Category	Processed
8	Man in the Middle on IoT data	2019-05-28 16:50:14	Attack	❌
8	Man in the Middle on IoT data	2019-05-28 16:43:36	Attack	❌
8	Man in the Middle on IoT data	2019-05-29 07:57:17	Attack	❌

Figure 32. Main Screen – Unknown Anomalies manager

This section shows all the unknown anomalies retrieved from the XL-SIEM component. The list of anomalies can be filtered by processed/no processed, and the detail of each alarm can be consulted through the dialog:


8

Date:	2019-05-28 16:43:36
Anomaly name:	Man in the Middle on IoT data
Category:	Attack
Subcategory:	compromise
Protocol:	6
Source IP:	155.54.210.142
Source port:	1026
Destination IP:	37.48.68.242
Destination port:	0
Processed:	❌

Figure 33. Unknown Anomaly details – Unknown Anomalies manager

Once an anomaly is processed, it is allowed to remove it from the database.

Back-end

This component is in charge of 1) deploying the honeypots selected and 2) analysing unknown anomalies.

Honeypot deployment

1. **Honeypot catalogue**. This component is responsible for managing the honeypots registered and available to be deployed. It manages the following information:
   - Name/description: The name and a short description of each honeypot.
   - OVA file: The OVA virtual machine file containing the honeypot.
   - Username/password: The credentials to access the virtual machine containing the honeypot.
   - Master/slave behaviour: If a group of honeypots can behave like master slaves, communicating with each other.
   - Starting/stopping commands: The commands needed to start and/or stop a honeypot.
   - SSH port: The port configured to access the honeypot by SSH.
   - SSH forwarded port: The configured port, corresponding to the host machine, through which the Honeypot Manager will connect to the honeypot when it does not yet have an assigned IP address.
   - Open ports: The rules that specify which ports should be opened. For example, port 502 for MODBUS.
2. **Target host manager**. Honeypots are deployed on virtual machines. And these virtual machines run on a physical machine. This component is in charge of managing the information about these physical machines:
   - Description: A short description of the host machine.
   - IP address: The IP address assigned to the physical machine.
   - Network interface: The network interface used to establish a bridge between the host and the honeypots virtual machines.
   - SSH port: Its configured SSH port.
   - Username/password: The credentials to access the host machine.
3. **Task deployment manager.** Honeypot Manager calls "task" to a prepared deployment of a honeypot in a host machine. This is, after choosing a honeypot of the catalogue and selecting a host machine, a deployment task is prepared. A task can be "Pending", "Running", "Stopped" and "Finished, and the most relevant information it collects is:
   - Honeypot: The selected honeypot.
   - Master/slave: If the honeypot behaves as master or slave (if supported).
   - State: The current state of the task: "Pending", "Running", "Stopped" or "Finished". This state is mostly used to know if a honeypot is running or not.
   - Host IP: The IP address corresponding to the host machine.
   - Honeypot IP: The IP address assigned to the virtual machine containing the honeypot.

Unknown Anomalies manager

This component is composed by three main subcomponents

1. **Alert connector**. This component retrieves the alerts form the XL-SIEM correlation engine. This is done through a RabbitMQ server provided by the XL- SIEM. A detailed explanation on how this is done could be found in the section regarding the interfaces.
2. **Alert parser**. The alerts retrieved from the XL-SIEM should be converted to an object that could be analysed by the UA Analyser subcomponent. The retrieved alert has more information that the one that it is required by the UA Analyser, so this parser is in charge to read all the information provided by the alert, select the relevant fields of the alert: PLUGIN_SID , SID_NAME, CATEGORY, SUBCATEGORY, PROTOCOL, SRC_IP, DST_IP, SRC_PORT and DST_PORT and transform this information in an object to be understandable by the UA Analyser.
3. **UA Analyser**. The analyser with the information coming from the alert and the information that the information available in Honeypot Manager, i.e.: honeypot deployed, honeypots in the catalogue, type of attack etc. has two possibilities:
    - There is a deployed honeypot which can received the traffic rearrange with the information of the unknown anomalies, in this case the UA Analyser calls to the SDN connector.
    - There is a honeypot available in the catalogue, but it is not deployed in the EPES network. In this case, through the component UA manager of the front-end informs to the security manager of this option and if the decision is to deploy this honeypot, the process to deploy automatically the honeypot starts. Once the new honeypot is deployed and running, the UA Analyser calls to the SDN connector to rearrange the traffic to it.
4. **SDN Connector.** This connector is responsible to carry out all the actions required to rearrange the traffic to the selected honeypot. The SDN connector receives the required information to implement the redirection from a) the Honeypot Manager, including IP and/or MAC addresses of both the attacker and the honeypot as well as b) the topology from the SDN controller in order to specify the switch port that the honeypot is connected to. More details about the specific REST API commands are provided in the Interfaces section.

5.3 Interfaces model

5.3.1 Honeypots: IEC 61850, IEC60870-5-104, Modbus

As mentioned in section 5.1, the honeypots interact mainly with two other components in the SDN-microSENSE ecosystem, XL-SIEM and SDN controller.

Figure 34 shows graphically these interfaces to other SDN-Microsense architecture components:

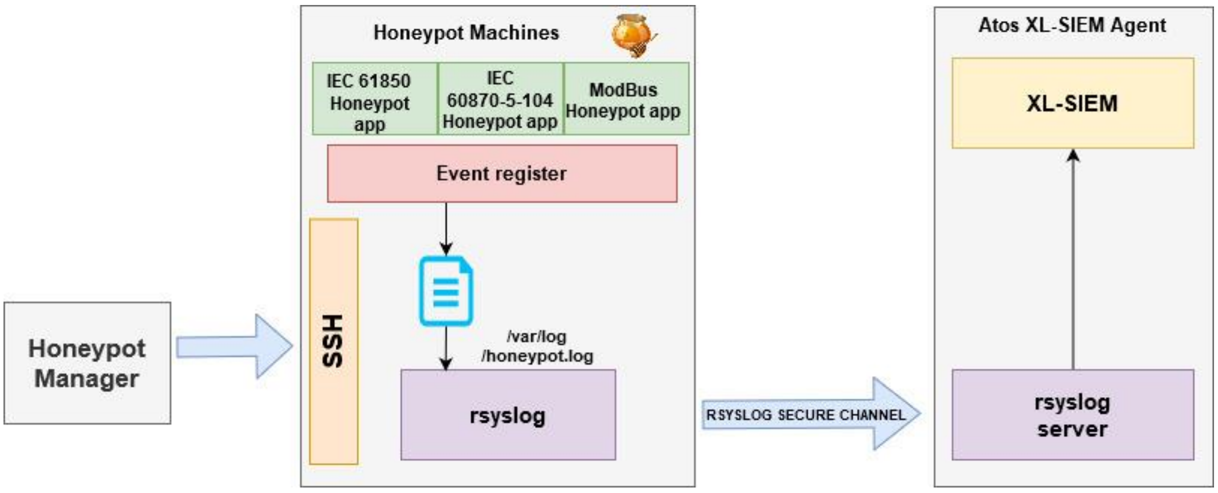


Figure 34. Interface model

SDN-Microsense Honeypot Manager component uses SSH communication protocol to establish a secure connection with honeypot machines where there are the honeypots. This connection is required to perform the following tasks related to honeypot virtual machine management:

- Create, start, stop and destroy of honeypot virtual machine.
- Network interface management.
- Monitorization of virtual machine status.

Furthermore, each honeypots machine has a direct connection with XL-SIEM Agent component developed by ATOS in the context of WP5.

XL-SIEM exposes different interfaces to communicate security events and incidents. In the cases of the developed honeypots, the rsyslog interface exposed by XL-SIEM agent has been selected for event communication. Rsyslog is run into honeypot machine to recollect inputs to event file ***/var/log/honeypot.log*** and output the results to XL-SIEM by means of a secure connection. The connection is secured by TLS protocol. Rsyslog could deal with different input formats. As XL-SIEM does not define a specific message format for events the following JSON message format has been defined:

- IEC61850 logs:

{"eventId":"0001","name":"New connection","timestamp":"Tue Mar 19 22:04:00 4448757", "parameters":{"ip":"10.0.0.3","param2":"Another param"}}

Detailed description of each parameter is given in "Technical description" section. Rsyslog Server deployed in XL-SIEM agent is listening for incoming events on port 5200.

- IEC60870-5-104 logs

{"timestamp": "2020-06-22T16:44:34.656128", "sensorid": "default", "id": "245cc5ac-ba77-4677-4276-876976583791", "src_ip": "192.168.1.55", "src_port": 60839, "dst_ip": "192.168.1.135", "dst_port": 2404, "data_type": "iec104", "request": null, "response": null, "event_type": "STARTDT act"}

- Modbus logs

{"timestamp": "2019-11-25T13:34:53.902098", "sensorid": "default", "id": "167b36dc-af68-4935-8393-490972134866", "src_ip": "122.228.19.80", "src_port": 60824, "dst_ip": "83.212.75.106", "dst_port": 5020, "public_ip": "83.212.75.106", "data_type": "modbus", "request": null, "response": null, "event_type": "NEW_CONNECTION"}

5.3.2 Honeypot Manager

Additionally to the interface mentioned in the section above, the Honeypot Manager as commented in in the section 5.1, Honeypot Manager interacts with **XL-EPDS** to retrieve information of the unknown anomalies (S-RAF/XL-EPDS-01) and **SDN-controller** to provides information to forward network traffic to the deployed honeypots (S-RAF_NBI-1). This section explains how these communications are done.

XL-EPDS (S-RAF/XL-EPDS-01)

As shown in Figure 35, the Honeypot Manager receives alerts from the XL-SIEM correlation engine. This is done through a RabbitMQ server that manages the subscription of consumers to these alerts. This RabbitMQ server is working on a fan-out mode, which means that an exchange queue is used by the XL-SIEM to export alerts as shown in Figure 35. When a component wants to consume such as alerts it must create a new queue and attach it to the exchange queue used by the XL-SIEM. This is a good working mode because in the SDN-microSENSE project it would be more than one component consuming these alerts, and it is a good way to guarantee that all alerts are delivered to all components without losing information. If all components consuming alerts were attached to the same queue just one consumer would get an alert because messages are deleted from queues when consumed.

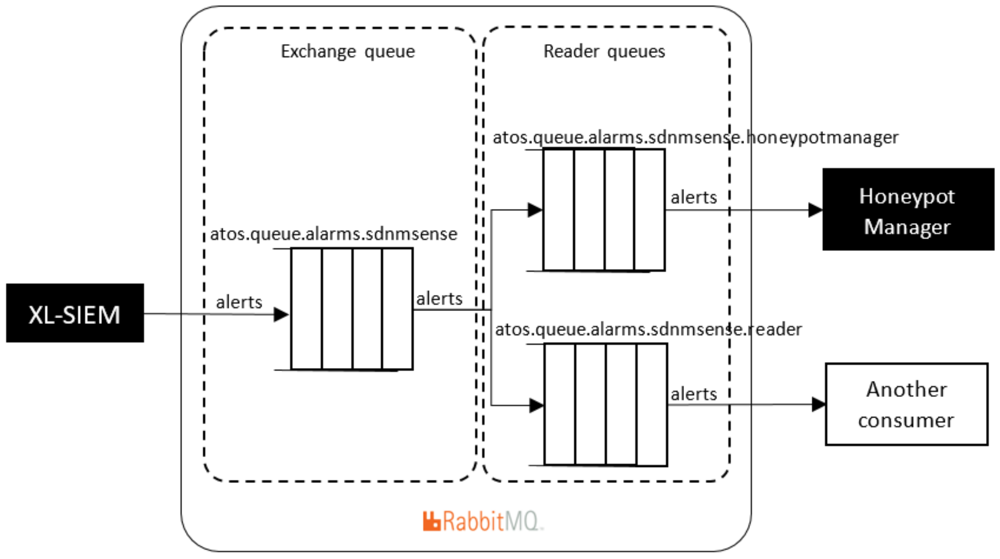
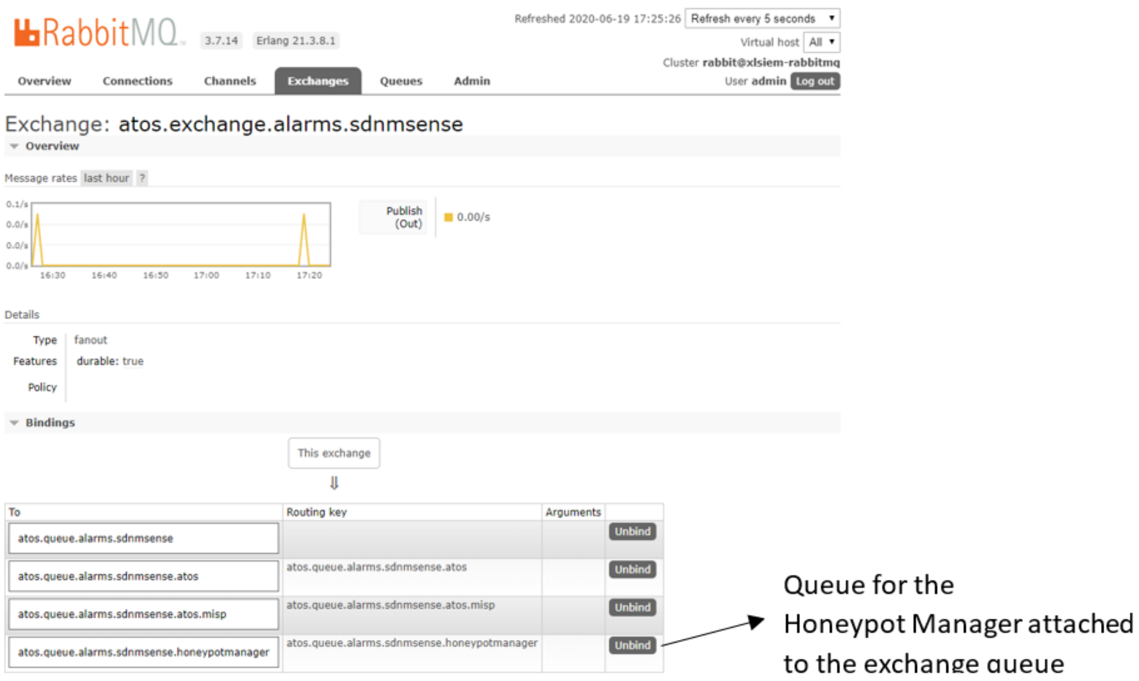

Exchange queue
Reader queues
atos.queue.alarms.sdnmsense.honeypotmanager
alerts
Honeypot Manager
atos.queue.alarms.sdnmsense
XL-SIEM
alerts
alerts
atos.queue.alarms.sdnmsense.reader
alerts
Another consumer
RabbitMQ

Figure 35. Fan-out queue configuration used to feed the Honeypot Manager

Figure 36 shows a screenshot of the RabbitMQ control panel showing the exchange queue for the XL-SIEM alerts and the Honeypot Manager queue blinded to the exchange queue.



Exchange: atos.exchange.alarms.sdnmsense

Details

Type	fanout
Features	durable: true
Policy	

To	Routing key	Arguments	
atos.queue.alarms.sdnmsense			Unbind
atos.queue.alarms.sdnmsense.atos	atos.queue.alarms.sdnmsense.atos		Unbind
atos.queue.alarms.sdnmsense.atos.misp	atos.queue.alarms.sdnmsense.atos.misp		Unbind
atos.queue.alarms.sdnmsense.honeypotmanager	atos.queue.alarms.sdnmsense.honeypotmanager		Unbind

Figure 36. RabbitMQ exchange queue for alerts

SDN-controller (S-RAF_NBI-1)

The Honeypot Manager communicates with the SDN controller via the SDN connector to insert specific rules that manage traffic of honeypots. The SDN connector interacts with SDN-C for the following reasons:

- To retrieve the topology
- To initialise the communications of a honeypot
- To redirect traffic towards a honeypot
- To revert the traffic redirection

When the Honeypot Manager deploys a new honeypot that has the same network configurations with a real device, by default all traffic towards the honeypot should be blocked. To implement this, first, the SDN connector obtains the network topology via the topology REST API by issuing the HTTP GET command **/v1.0/topology/hosts.** This command can be executed to the first available SDN-C. A response example is provided bellow:

```
[
  {
    "mac": "0c:85:e2:13:c9:00",
    "ipv4": [
      "172.16.1.1"
    ],
    "ipv6": [
      "fe80::e85:e2ff:fe13:c900"
      ],
      "port": {
        "dpid": "0000aae8061f594b",
        "port_no": "00000003",
        "hw_addr": "3e:db:af:ef:64:b0",
        "name": "eth3"
      }
    },
    {
      "mac": "00:50:79:66:68:03",
      "ipv4": [],
      "ipv6": [
        "::"
      ],
      "port": {
        "dpid": "00004a4b56428b4e",
        "port_no": "00000003",
        "hw_addr": "46:25:b6:ac:a5:f1",
        "name": "eth3"
      }
    }
]
```

By searching for the JSON object that has the MAC address of the deployed honeypot (mac field), two properties are obtained:

- The datapath id (dpid, SDN switch identification), to which the honeypot is connected. Future commands should be addressed to this datapath id.
- The switch port number that the honeypot occupies (port_no)

If the honeypot's MAC address has not been found by asking the first available SDN-C, then iteratively all SDN-C´s are being asked. The list of all available SDN controllers is obtained from the Synchronisation and Coordination Service (SCS).

Moreover, the SDN connector searches the above JSON objects for the attacker's MAC or IP address to obtain the switch port that they are connect to.

After the required information has been obtained (datapath id, attacker's port, honeypot's port), the SDN Connector retrieves the master controller for the specific datapath id, via SCS, and executes the HTTP POST command **/stats/flowentry/add** with the following JSON data included in the body:

```
{
  "dpid": "00004a4b56428b4e"
  "priority": 10,
  "cookie_id": 10,
  "table_id": 100
  "match": {
    "in_port": 6,
      "eth_src": "46:25:b6:ac:a5:f1",
    },
    "actions": []
}
```

The JSON object is explained as follows:

- dpid refers to the datapath id obtained via the topology REST API
- priority determines the priority level of the flow. Greater priority implies that the flow will be looked up first by the switch. The value 10 is reserved by SDN-C for this type of flow.
- cookie_id is a number that the Honeypot Manager always must include in any command to SDN-C. This number is used internally by SDN-C to track and manage all instructions inserted by a particular SDN app. 10 is the number reserved for commands inserted by the Honeypot Manager.
- table_id is the table ID to insert the flow. The Honeypot Manager should always insert new flows to table 100.
- In the match field, in_port specifies the port obtained also by the topology REST API and eth_src matches the MAC address of the honeypot. This filter matches all ingress traffic generated by the honeypot.
- Finally, the empty actions field implies that the matches packets/frames will be dropped. In other words, this means that the honeypot will be able to receive traffic, but its responses are dropped.

Next step is to redirect traffic originated from the attacker to the honeypot. To achieve this, the SDN connector asks again about the master controller and then issues the HTTP POST command **/stats/flowentry/add** with the following content. Note that the port in actions refers to the honeypot's port and has been obtained via the topology API:

```
{
    "dpid": "00004a4b56428b4e"
    "priority": 9,
    "cookie_id": 10,
    "table_id": 100
    "match": {
      "eth_dst": "46:25:b6:ac:a5:f1",   //honeypot MAC
      "eth_src": "00:25:b6:ac:a5:f0",   //attacker MAC
    },
    "actions": [{
       "type": "OUTPUT"
       "port": 6
    }]
}
```

Right after, the SDN connector inserts the appropriate flow to redirect the honeypot's replies to the attacker. Note that the port included in actions refers to the attacker's port obtained via the topology API:

```
{
    "dpid": "00004a4b56428b4e"
    "priority": 9,
    "cookie_id": 10,
    "table_id": 100
    "match": {
      "eth_dst": "00:25:b6:ac:a5:f0",   // attacker MAC
      "eth_src": "46:25:b6:ac:a5:f1",   // honeypot MAC
    },
    "actions": [{
       "type": "OUTPUT"
       "port": 1
    }]
}
```

Finally, if the SDN Connector wants to revoke any of the aforementioned commands, then the SDN Connector should issue an HTTP POST command to **/stats/flowentry/delete_strict** with the exact same body content as the flow that the connector intends to delete.

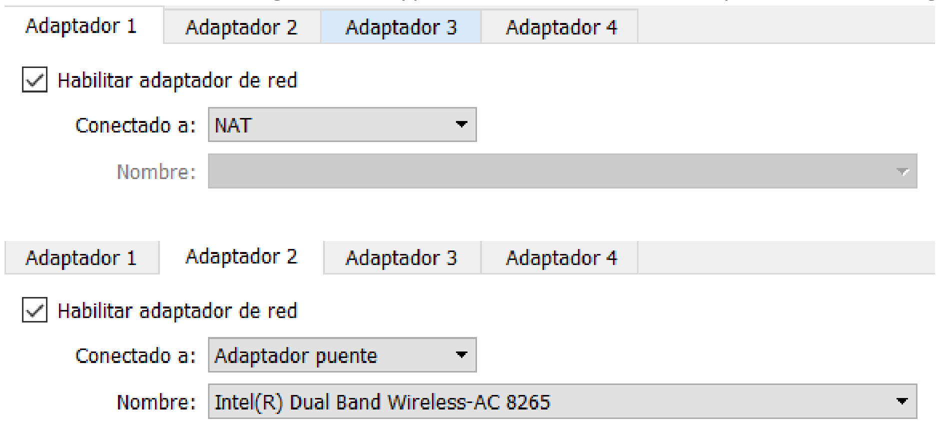
5.4 Deployment details

5.4.1 Honeypots: IEC 61850, IEC60870-5-104, Modbus

These honeypots are deployed using Honeypot Manager. Following the description on how this deployment is done.

Prerequisites:

- Oracle VirtualBox 6.0.
- SSH port (usually 22) and 2200 open to the IP address of the machine containing the Honeypot Manager.
- The OVA file containing the honeypot must have a "NAT adapter" and a "Bridge adapter":



☑ Habilitar adaptador de red
Conectado a: NAT
Nombre:

☑ Habilitar adaptador de red
Conectado a: Adaptador puente
Nombre: Intel(R) Dual Band Wireless-AC 8265

- The OVA file containing the honeypot must have the Netplan configured that picks up the Bridge adapter connected to the DHCP server. For example:

Steps to deploy a honeypot:

The steps described in Figure 37 have to be followed:

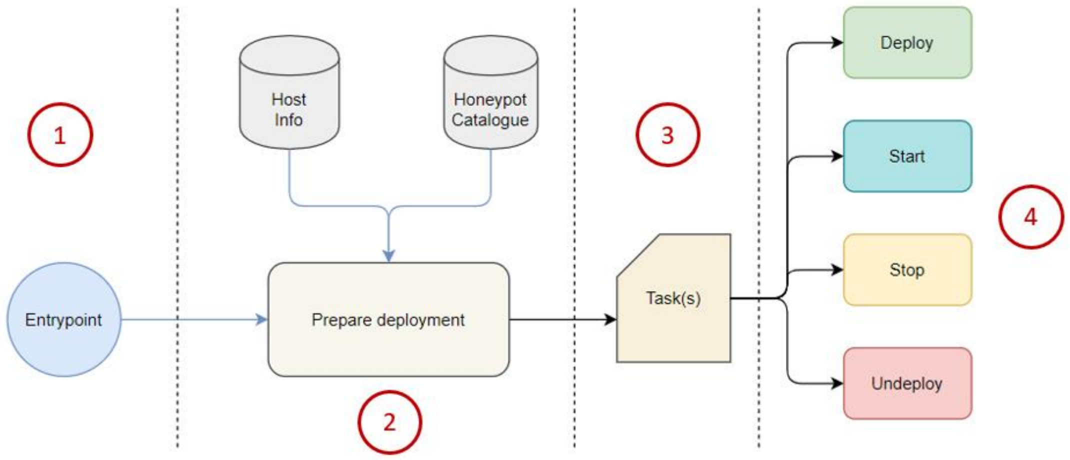


Figure 37. Steps to deploy a honeypot using the Honeypot Manager

Step 1: Access to the Honeypot Manager:

Login in the Honeypot Manager front-end as administrator or as Security Operator (Figure 38).

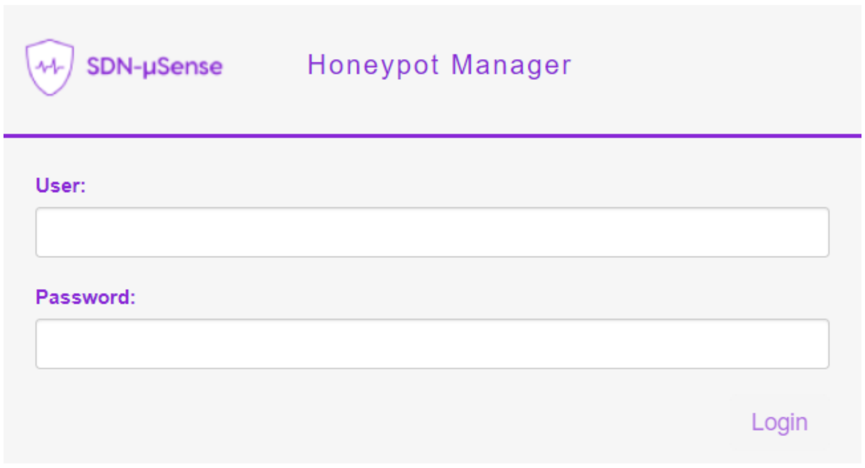
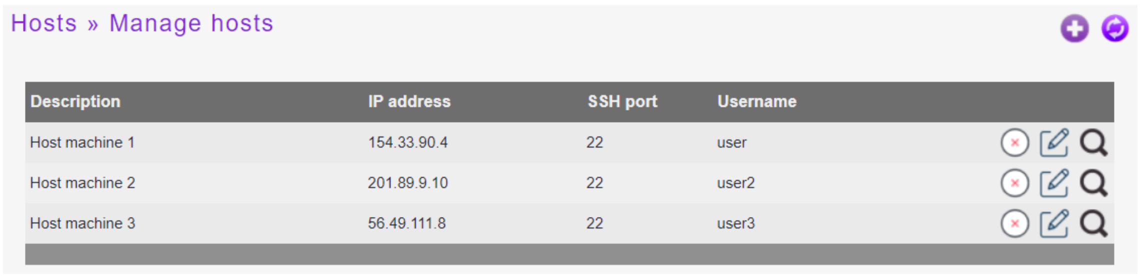


Figure 38. Login Screen – Honeypot Manager

Step 2: Prepare the deployment:

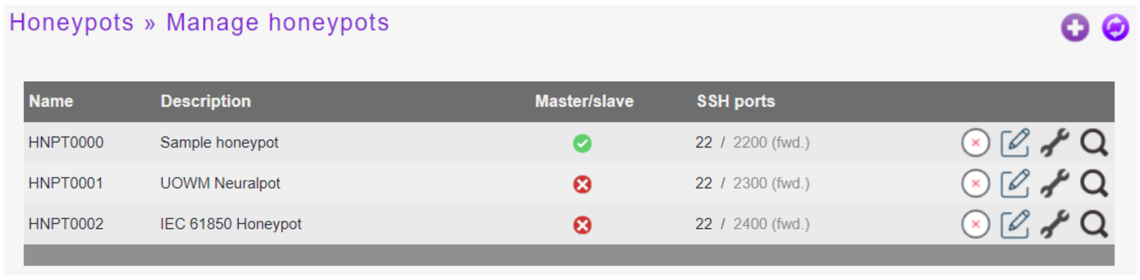
- Check that there are Hosts available, and that there are honeypots in the catalogue.



Hosts » Manage hosts

Description	IP address	SSH port	Username
Host machine 1	154.33.90.4	22	user
Host machine 2	201.89.9.10	22	user2
Host machine 3	56.49.111.8	22	user3

Figure 39. Manage host – Honeypot Manager



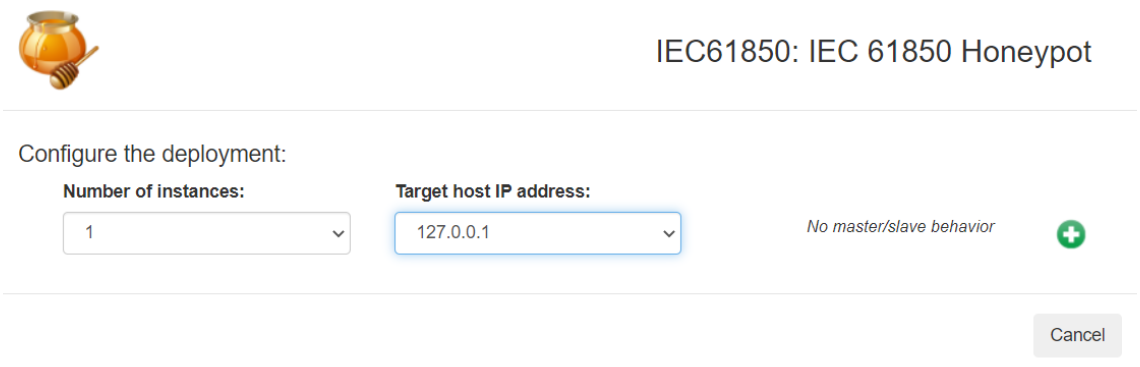
Honeypots » Manage honeypots

Name	Description	Master/slave	SSH ports
HNPT0000	Sample honeypot	✓	22 / 2200 (fwd.)
HNPT0001	UOWM Neuralpot	✗	22 / 2300 (fwd.)
HNPT0002	IEC 61850 Honeypot	✗	22 / 2400 (fwd.)

Figure 40. Manage Honeypots – Honeypot Manager

- In the "Manage honeypots" click on the "tool" icon to configure a new deployment

- Select the number of instances to deploy, the target IP address, and click the ⊕ button.



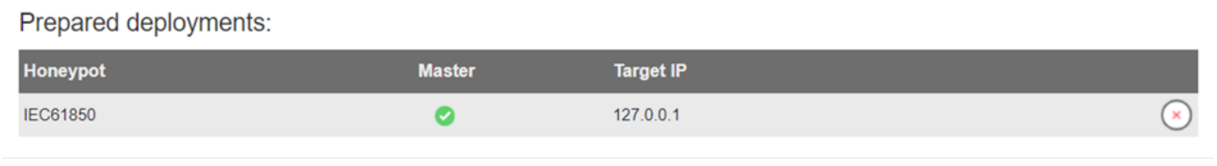
IEC61850: IEC 61850 Honeypot

Configure the deployment:

Number of instances:	Target host IP address:	
1	127.0.0.1	*No master/slave behavior*

Figure 41. Deployment configuration – Honeypot Manager

- The new deployments will be added to the list of prepared deployments:

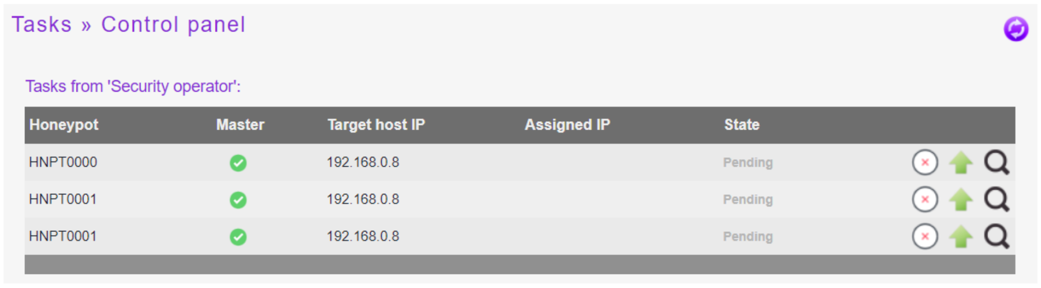


Prepared deployments:

Honeypot	Master	Target IP
IEC61850	✓	127.0.0.1

Figure 42. Deployments prepared (example)– Honeypot Manager

- When the configuration is done, click on "Prepare" button. It will appear as a "Task" in the Task Control Panel screen:



Tasks » Control panel

Tasks from 'Security operator':

Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8		Pending
HNPT0001	✓	192.168.0.8		Pending
HNPT0001	✓	192.168.0.8		Pending

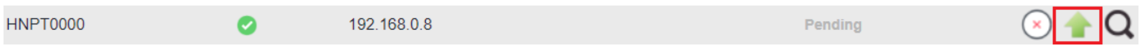
Figure 43. Task management screen– Honeypot Manager

Step 3: Manage the Task(s) generated:

In the Task list, select the honeypot Task to be deployed (Figure 43). In this screen, the Security operator can view the Task detail, remove Tasks, and perform one of these operations on honeypots: deploy, start, stop and undeploy.

Step 4: Deploy the honeypot:

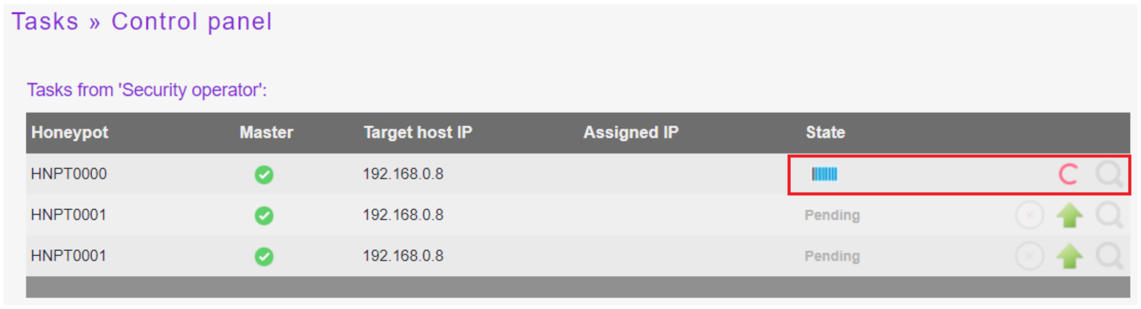
Once a deployment Task is selected, click on the “Deploy” button:



HNPT0000	✓	192.168.0.8	Pending

Figure 44. Start deployment– Honeypot Manager

The deployment will start:



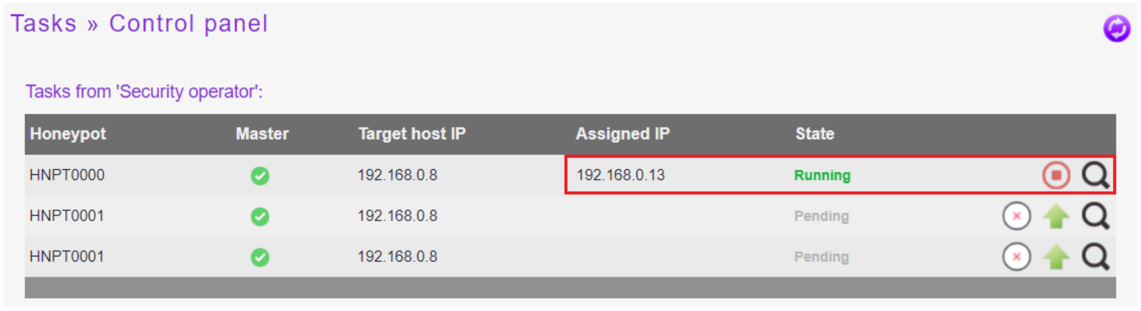
Tasks » Control panel

Tasks from 'Security operator':

Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8		
HNPT0001	✓	192.168.0.8		Pending
HNPT0001	✓	192.168.0.8		Pending

Figure 45. Deployment in progress– Honeypot Manager

Once the deployment is completed, the Task will change its state to “Running”, and the assigned IP address to the honeypot will be shown:



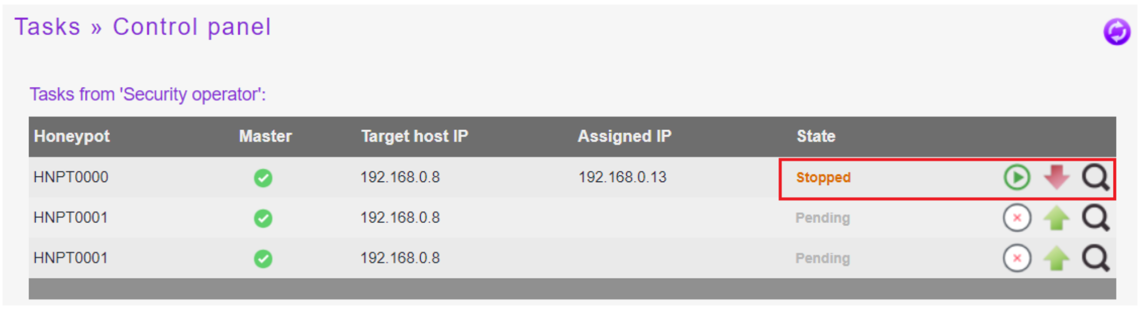
Tasks » Control panel

Tasks from 'Security operator':

Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8	192.168.0.13	Running
HNPT0001	✓	192.168.0.8		Pending
HNPT0001	✓	192.168.0.8		Pending

Figure 46. Honeypot running– Honeypot Manager

If a Task is “Running”, a “Stop” button appears, clicking on it, the machine will be stopped:


Tasks » Control panel

Tasks from 'Security operator':

Honeypot	Master	Target host IP	Assigned IP	State
HNPT0000	✓	192.168.0.8	192.168.0.13	Stopped
HNPT0001	✓	192.168.0.8		Pending
HNPT0001	✓	192.168.0.8		Pending

Figure 47. Honeypot status management– Honeypot Manager

If a Task is “Stopped”, the options of “Restart” and “Undeploy” will appear:

- Restart: the honeypot will be restarted, and the Task will change its state to “Running”.
- Undeploy: the honeypot will be destroyed, and the Task will change its state to “Finished”.

5.4.2 Honeypot Manager

Prerequisites

- A computer with Linux Ubuntu 10.04 installed.
- Docker server installed.
- SSH port (usually 22) open to the IP address of the machine that will host the deployed honeypots.

Honeypot Manager installation:

Create a new directory in “/home/*<user>*”, and copy inside the four dockerized components of the Honeypot Manager:

- sdn.honeypot.manager.mysql.server
- sdn.honeypot.manager.mysql.client
- sdn.honeypot.manager.backend.server
- sdn.honeypot.manager.frontend.server

Build Docker images:

Build MySQL Server Docker image:

```
cd /home/<user>/SDN_HPM/honeypot.manager.mysql.server
sudo docker build -f Dockerfile -t sdn/honeypot.manager.mysql.server .
```

Build MySQL Client Docker image:

```
cd /home/<user>/SDN_HPM/honeypot.manager.mysql.client
sudo docker build -f Dockerfile -t sdn/honeypot.manager.mysql.client .
```

Build Backend Docker image:

```
cd /home/vlad/SDN_HPM/honeypot.manager.backend.server
sudo docker build -f Dockerfile -t sdn/honeypot.manager.backend.server .
```

Build Frontend Docker image:

```
cd /home/vlad/SDN_HPM/honeypot.manager.frontend.server/src/main/docker
sudo docker build -f Dockerfile -t sdn/honeypot.manager.frontend.server .
```

Create a Docker network:

In order for the different components to communicate with each other, it is necessary to configure a Docker network:

```
sudo docker network create --driver=bridge sdn_hm_network > /dev/null 2>&1 || true
```

Run Docker images:

Run MySQL Server Docker image:

```
sudo docker run -d -p 3306:3306 --name sdn.hm.mysql.server --env
MYSQL_ROOT_PASSWORD=sdn_hm_pwd --network sdn_hm_network
sdn/honeypot.manager.mysql.server
```

Run MySQL Client Docker image:

```
sudo docker run -d --name sdn.hm.mysql.client --restart=no --env MYSQL_DB_USER=root --
env MYSQL_DB_PASSWORD= sdn_hm_pwd --env MYSQL_DB_NAME=sdn_hm_manager --
env MYSQL_DB_HOST=sdn.hm.mysql.server --network sdn_hm_network
sdn/honeypot.manager.mysql.client
```

Run Backend Docker image:

```
sudo docker run -d -p 10001:8080 --name sdn.hm.backend.server --network
sdn_hm_network sdn/honeypot.manager.backend.server
```

Run Frontend Docker image:

```
sudo docker run -d -p 82:80 --name sdn.hm.frontend.server --network sdn_hm_network
sdn/honeypot.manager.frontend.server
```

Check the installation:

In order to check if the Honeypot Manager is properly running, type in a web browser:

```
http://<host>:82
```

The login screen appears and to access as **Administrator** should use the following credentials:

- User: admin
- Password: sdn_hm_admin

To access as a Security Operator:

- User: operator
- Password: sdn_hm_operator

6 SDN-Microsense Assessment: Unit Testing

The testing strategy defined in the D2.4, "Pilot, Demonstration & Evaluation Strategy" [SMS20-D24] defines the Unit Testing Phase as the phase where the operation of each SDN-microSENSE component in terms of the corresponding technical specifications defined in D2.3 is tested. The unit tests will be implemented in the relevant technical deliverables in WP3, WP4 and WP5 aiming to cover the respective technical specifications defined in D2.3. If there are implementation constraints or necessary interconnections with other components, the corresponding unit tests can also be implemented in D7.4 named Early Prototype, which will describe the first version of the integrated SDN-microSENSE platform.

In this deliverable unit test cases have been implemented for the components developed within task 3.3, namely the components previously described in sections 5: IEC 61850 honeypot, IEC 60870-5-104 honeypot, Modbus honeypot and Honeypot Manager. The unit test cases are referencing the specification defined in D2.3 [SMS20-D23]; those specifications have been previously elicited from the user, security and privacy requirements defined in D2.2 [SMS20-D22].

The details of the performed test cases and their results are shown below.

6.1 IEC 61850 Honeypot

6.1.1 Unit testing scenario

Figure 48 shows the testing scenario deployed in Tecnalia's Smart Grid Cybersecurity laboratory to validate IEC 61850 honeypot application functionality. As it has been described in the technical description section of the honeypot, the application could be easily built for any target platform. Thus, a Raspberry Pi 4, connected to laboratory's switch, has been selected to deploy honeypot application. As client application, laboratory's existing SCADA has been used to stablish connections and request different actions against the honeypot using IEC 61850/MMS communications standard. An output connection to the internet is open to allow the honeypot to communicate the registered events to the XL-SIEM.

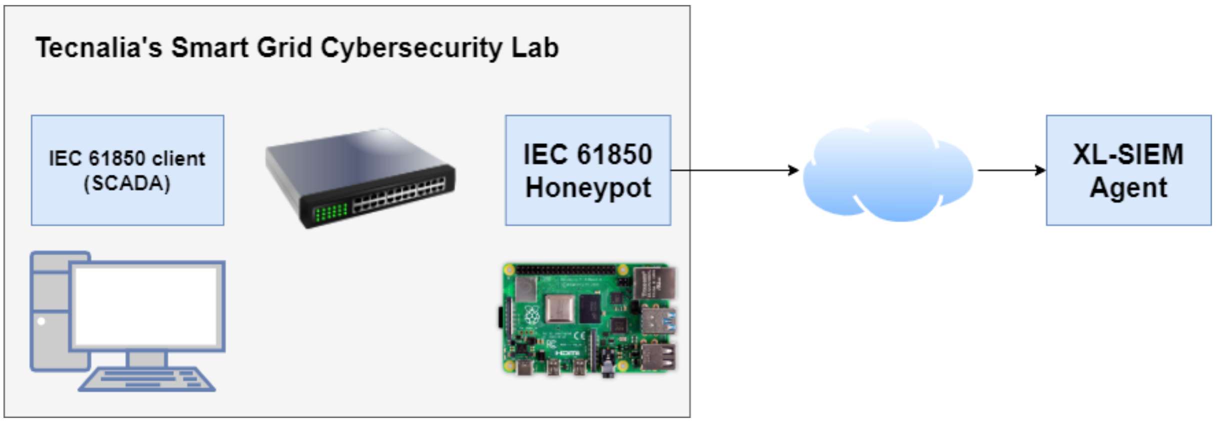
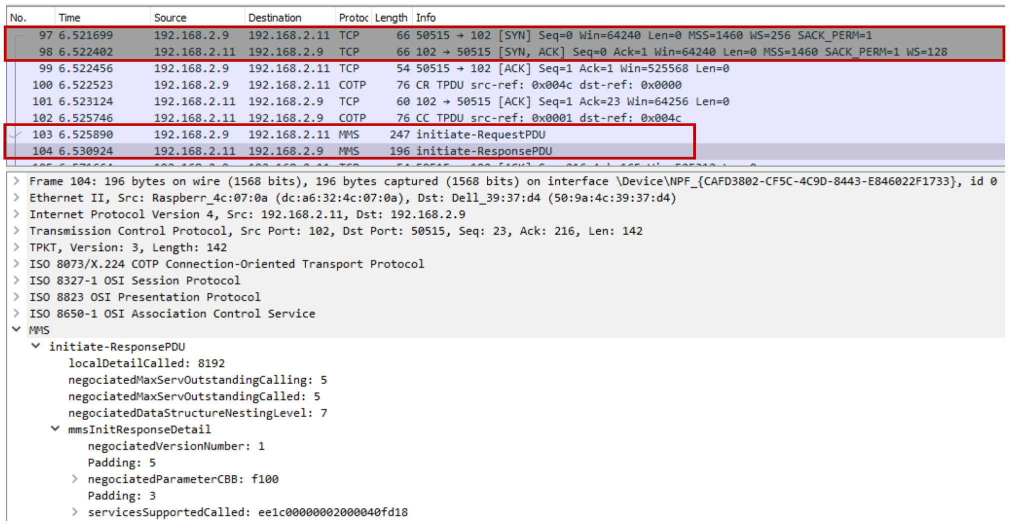
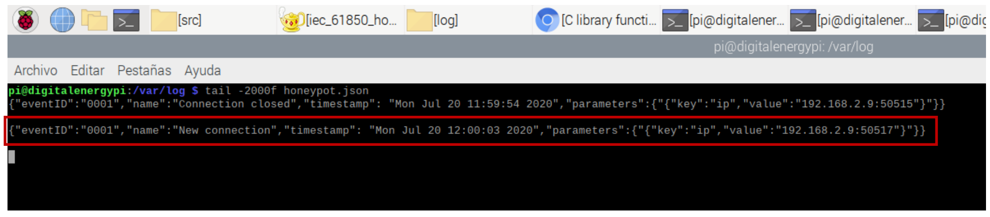


Figure 48. IEC 61850 testing scenario

6.1.2 Unit tests

Test Case ID	HP_61850_01	**Component**	IEC 61850 honeypot
Description	The honeypot registers new connection events when it receives new connection attempts directed to TCP port number 102		
SPEC ID	SPEC-F4	**Priority**	High
Prepared by	TECNALIA	**Tested by**	TECNALIA
Pre-condition(s)	• SCADA has network connectivity with IEC 61850 honeypot • IEC 61850 Honeypot must be up and listening on port 102 • Rsyslog daemon must be installed and running		
Test steps			
1	IEC 61850 client attempts to open a new MMS connection on port 102.		
2	Test if the connection has been successfully established		
3	Test if the honeypot has correctly registered the event on /var/log/honeypot.log		
Input data	N/A		
Result	• **Step 1 and Step 2:**  • **Step 3** 		

Step 1 and Step 2:

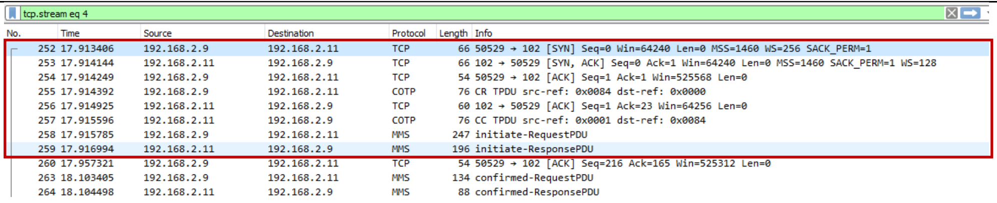
```
No.  Time        Source        Destination   Protoc Length Info
 97 6.521699    192.168.2.9   192.168.2.11  TCP     66 50515 → 102 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 98 6.522402    192.168.2.11  192.168.2.9   TCP     66 102 → 50515 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
 99 6.522456    192.168.2.9   192.168.2.11  TCP     54 50515 → 102 [ACK] Seq=1 Ack=1 Win=525568 Len=0
100 6.522523    192.168.2.9   192.168.2.11  COTP    76 CR TPDU src-ref: 0x004c dst-ref: 0x0000
101 6.523124    192.168.2.11  192.168.2.9   TCP     60 102 → 50515 [ACK] Seq=1 Ack=23 Win=64256 Len=0
102 6.525746    192.168.2.11  192.168.2.9   COTP    76 CC TPDU src-ref: 0x0001 dst-ref: 0x004c
103 6.525890    192.168.2.9   192.168.2.11  MMS    247 initiate-RequestPDU
104 6.530924    192.168.2.11  192.168.2.9   MMS    196 initiate-ResponsePDU

> Frame 104: 196 bytes on wire (1568 bits), 196 bytes captured (1568 bits) on interface \Device\NPF_{CAFD3802-CF5C-4C9D-8443-E846022F1733}, id 0
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 50515, Seq: 23, Ack: 216, Len: 142
> TPKT, Version: 3, Length: 142
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
> ISO 8650-1 OSI Association Control Service
v MMS
  v initiate-ResponsePDU
      localDetailCalled: 8192
      negociatedMaxServOutstandingCalling: 5
      negociatedMaxServOutstandingCalled: 5
      negociatedDataStructureNestingLevel: 7
    v mmsInitResponseDetail
        negociatedVersionNumber: 1
        Padding: 5
      > negociatedParameterCBB: f100
        Padding: 3
      > servicesSupportedCalled: ee1c00000002000040fd18
```

Step 3

```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 11:59:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50515"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:00:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}
```

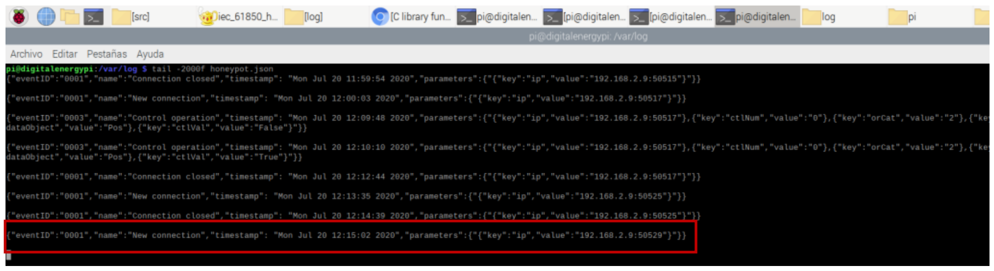
Test Case Result	Achieved

Test Case ID		HP_61850_02	**Component**	IEC 61850 honeypot
Description		The honeypot registers every control operation on CSWI logical node position which would indicate that an attacker is trying to open/close the circuit breaker.		
SPEC ID		SPEC-F4	**Priority**	High
Prepared by		TECNALIA	**Tested by**	TECNALIA
Pre-condition(s)		• SCADA has network connectivity with IEC 61850 honeypot • IEC 61850 Honeypot must be up and listening on port 102 • Rsyslog daemon must be installed and running • When the honeypot app starts the status of the breaker is closed		
Test steps				
1	IEC 61850 client opens a new MMS connection on port 102.			
2	Test if the connection has been successfully established			
3	Test if the honeypot has correctly registered the event on /var/log/honeypot.log			
4	Test if the Circuit Breaker status is correctly displayed on the SCADA's HMI			
5	Test to open the Circuit Breaker sending control operation on CSWI position			
6	Test if the response of the honeypot to the Control Operation is SUCCESS			
7	Test if the honeypot has registered a *"Control Operation"* event with ID 0003 on /var/log/honeypot.log			
8	Test to close the Circuit Breaker sending control operation on CSWI position			
9	Test if the response of the honeypot to the Control Operation is SUCCESS			
10	Test if the honeypot has registered a *"Control Operation"* event with ID 0003 on /var/log/honeypot.log			
Input data		N/A		
Result		• **Step 1 and Step 2**		



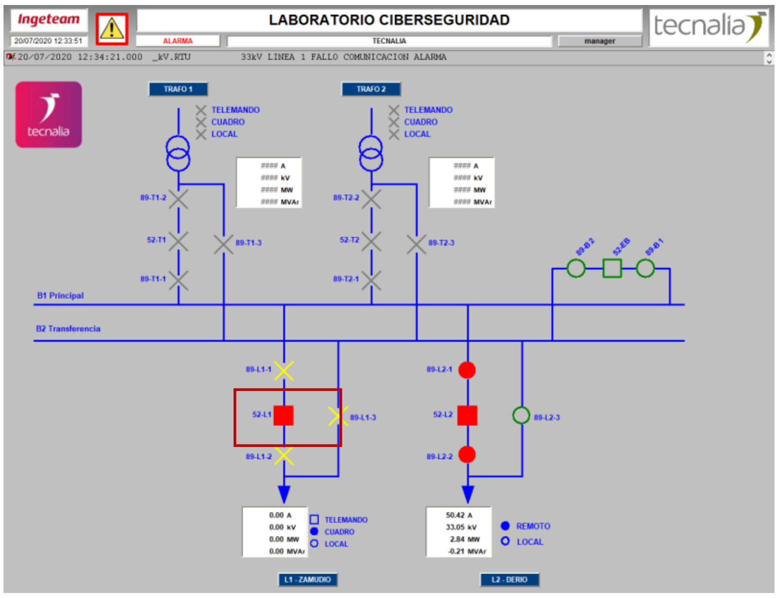
```
No.  Time       Source        Destination   Protocol Length Info
 252 17.913406  192.168.2.9   192.168.2.11  TCP         66 50529 → 102 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 253 17.914144  192.168.2.11  192.168.2.9   TCP         66 102 → 50529 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
 254 17.914249  192.168.2.9   192.168.2.11  TCP         54 50529 → 102 [ACK] Seq=1 Ack=1 Win=525568 Len=0
 255 17.914392  192.168.2.9   192.168.2.11  COTP        76 CR TPDU src-ref: 0x0084 dst-ref: 0x0000
 256 17.914925  192.168.2.11  192.168.2.9   TCP         60 102 → 50529 [ACK] Seq=1 Ack=23 Win=64256 Len=0
 257 17.915596  192.168.2.11  192.168.2.9   COTP        76 CC TPDU src-ref: 0x0001 dst-ref: 0x0084
 258 17.915785  192.168.2.9   192.168.2.11  MMS        247 initiate-RequestPDU
 259 17.916994  192.168.2.11  192.168.2.9   MMS        196 initiate-ResponsePDU
 260 17.957321  192.168.2.9   192.168.2.11  TCP         54 50529 → 102 [ACK] Seq=216 Ack=165 Win=525312 Len=0
 263 18.103405  192.168.2.9   192.168.2.11  MMS        134 confirmed-RequestPDU
 264 18.104498  192.168.2.11  192.168.2.9   MMS         88 confirmed-ResponsePDU
```

- **Step 3**

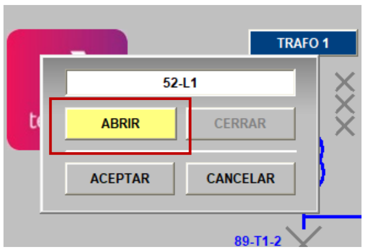


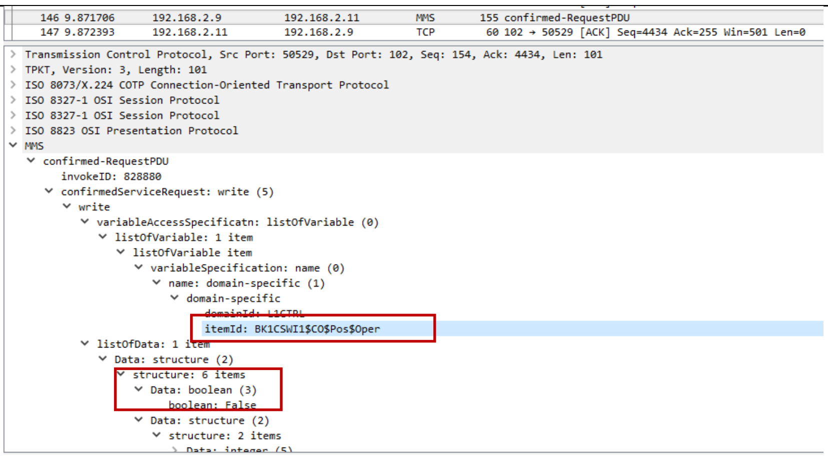
```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 11:59:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50515"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:00:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:09:48 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"ke
dataObject","value":"Pos"},{"key":"ctlVal","value":"False"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:10:10 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"ke
dataObject","value":"Pos"},{"key":"ctlVal","value":"True"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:12:44 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:13:35 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:14:39 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:15:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"}"}}
```

- **Step 4**



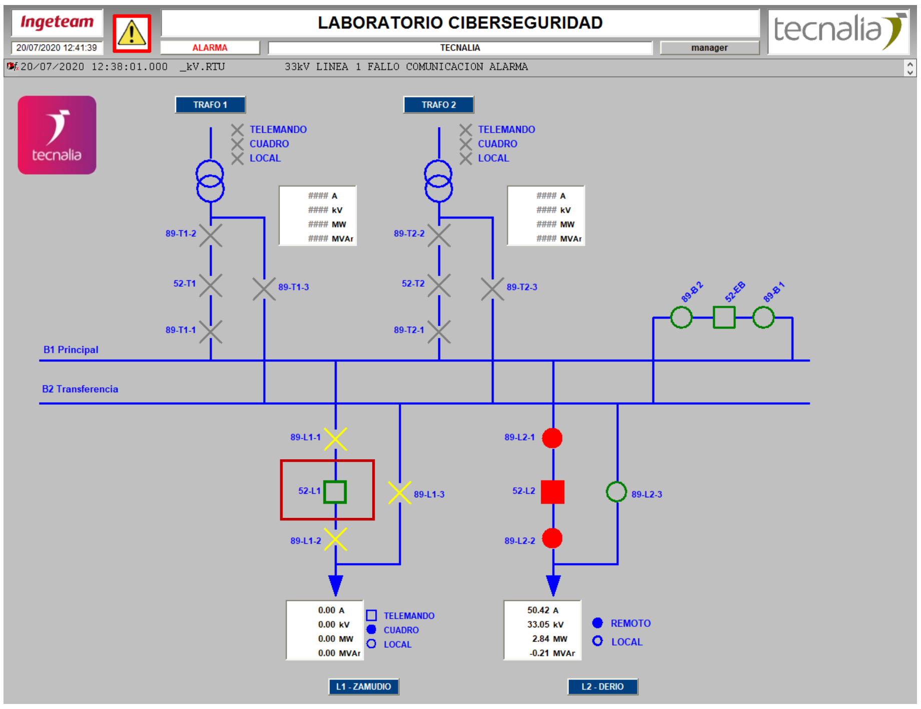
- **Step 5**

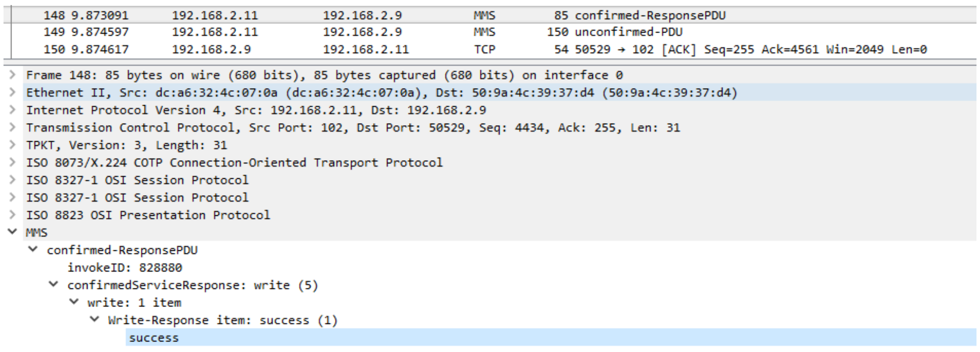




```
 146 9.871706      192.168.2.9        192.168.2.11       MMS      155 confirmed-RequestPDU
 147 9.872393      192.168.2.11       192.168.2.9        TCP       60 102 → 50529 [ACK] Seq=4434 Ack=255 Win=501 Len=0

> Transmission Control Protocol, Src Port: 50529, Dst Port: 102, Seq: 154, Ack: 4434, Len: 101
> TPKT, Version: 3, Length: 101
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
   v confirmed-RequestPDU
        invokeID: 828880
      v confirmedServiceRequest: write (5)
         v write
            v variableAccessSpecificatn: listOfVariable (0)
               v listOfVariable: 1 item
                  v listOfVariable item
                     v variableSpecification: name (0)
                        v name: domain-specific (1)
                           v domain-specific
                                domainId: L1CTRL
                                itemId: BK1CSWI1$CO$Pos$Oper
            v listOfData: 1 item
               v Data: structure (2)
                  v structure: 6 items
                     v Data: boolean (3)
                          boolean: False
                     v Data: structure (2)
                        v structure: 2 items
                           > Data: integer (5)
```

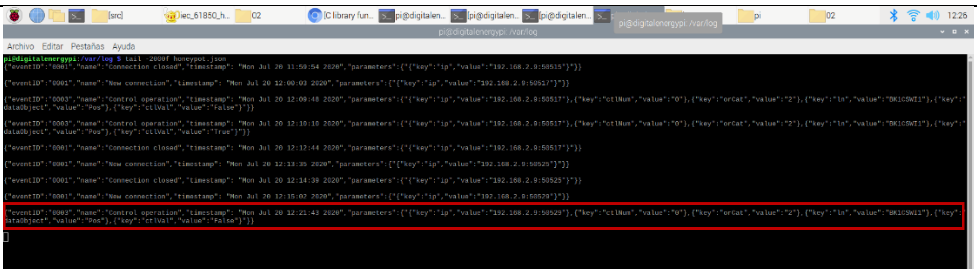
- **Step 6**





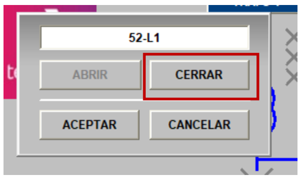
```
 148 9.873091      192.168.2.11       192.168.2.9        MMS        85 confirmed-ResponsePDU
 149 9.874597      192.168.2.11       192.168.2.9        MMS       150 unconfirmed-PDU
 150 9.874617      192.168.2.9        192.168.2.11       TCP        54 50529 → 102 [ACK] Seq=255 Ack=4561 Win=2049 Len=0

> Frame 148: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
> Ethernet II, Src: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a), Dst: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 50529, Seq: 4434, Ack: 255, Len: 31
> TPKT, Version: 3, Length: 31
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
   v confirmed-ResponsePDU
        invokeID: 828880
      v confirmedServiceResponse: write (5)
         v write: 1 item
            v Write-Response item: success (1)
                 success
```

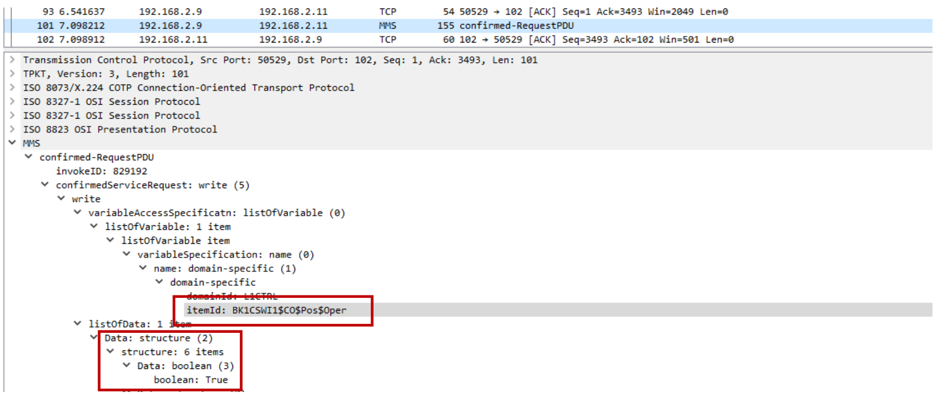
- **Step 7**



```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 11:59:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50515"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:00:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:09:48 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"
dataObject","value":"Pos"},{"key":"ctlVal","value":"False"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:10:10 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"
dataObject","value":"Pos"},{"key":"ctlVal","value":"True"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:12:44 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:13:35 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:14:39 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:15:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:21:43 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"
dataObject","value":"Pos"},{"key":"ctlVal","value":"False"}"}}
```

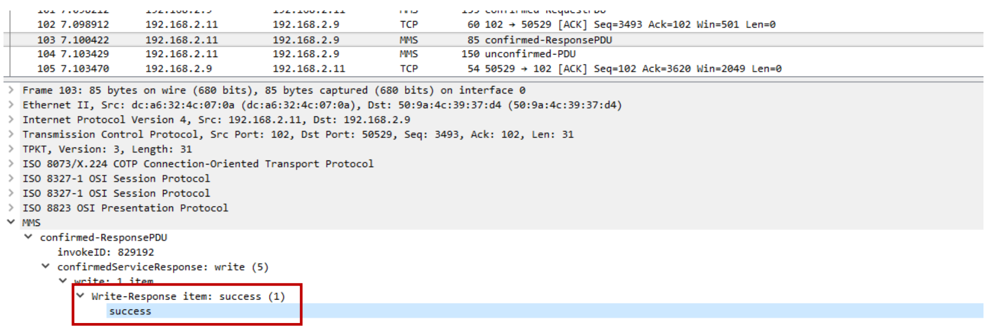
- **Step 8**





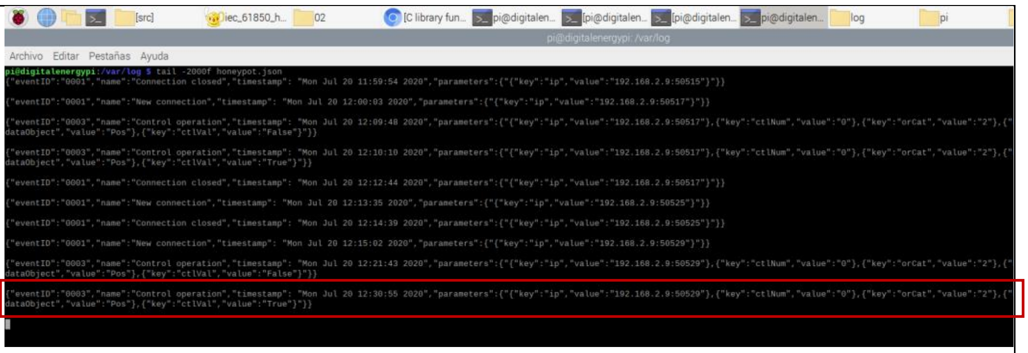
```
 93 6.541637   192.168.2.9    192.168.2.11   TCP    54 50529 → 102 [ACK] Seq=1 Ack=3493 Win=2049 Len=0
101 7.098212   192.168.2.9    192.168.2.11   MMS   155 confirmed-RequestPDU
102 7.098912   192.168.2.11   192.168.2.9    TCP    60 102 → 50529 [ACK] Seq=3493 Ack=102 Win=501 Len=0

> Transmission Control Protocol, Src Port: 50529, Dst Port: 102, Seq: 1, Ack: 3493, Len: 101
> TPKT, Version: 3, Length: 101
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-RequestPDU
      invokeID: 829192
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: True
```

- **Step 9**



```
102 7.098912   192.168.2.11   192.168.2.9    TCP    60 102 → 50529 [ACK] Seq=3493 Ack=102 Win=501 Len=0
103 7.100422   192.168.2.11   192.168.2.9    MMS    85 confirmed-ResponsePDU
104 7.103429   192.168.2.11   192.168.2.9    MMS   150 unconfirmed-PDU
105 7.103470   192.168.2.9    192.168.2.11   TCP    54 50529 → 102 [ACK] Seq=102 Ack=3620 Win=2049 Len=0

> Frame 103: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
> Ethernet II, Src: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a), Dst: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 50529, Seq: 3493, Ack: 102, Len: 31
> TPKT, Version: 3, Length: 31
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-ResponsePDU
      invokeID: 829192
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: success (1)
            success
```

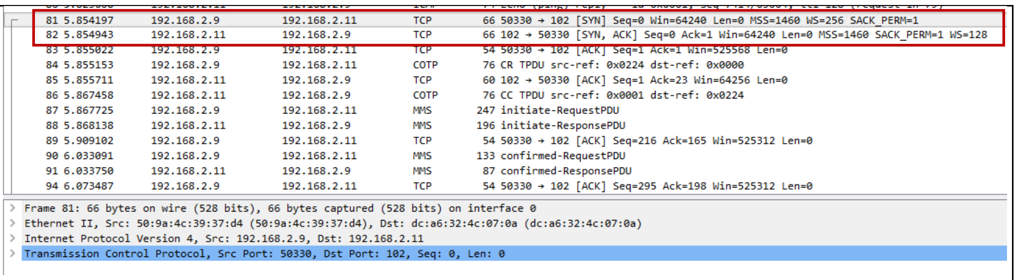
- **Step 10**

```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 11:59:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50515"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:00:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:09:48 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"
dataObject","value":"Pos"},{"key":"ctlVal","value":"False"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:10:10 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"
dataObject","value":"Pos"},{"key":"ctlVal","value":"True"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:12:44 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50517"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:13:35 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 12:14:39 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50525"}"}}

{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 12:15:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:21:43 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"
dataObject","value":"Pos"},{"key":"ctlVal","value":"False"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Mon Jul 20 12:30:55 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50529"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"
dataObject","value":"Pos"},{"key":"ctlVal","value":"True"}"}}
```

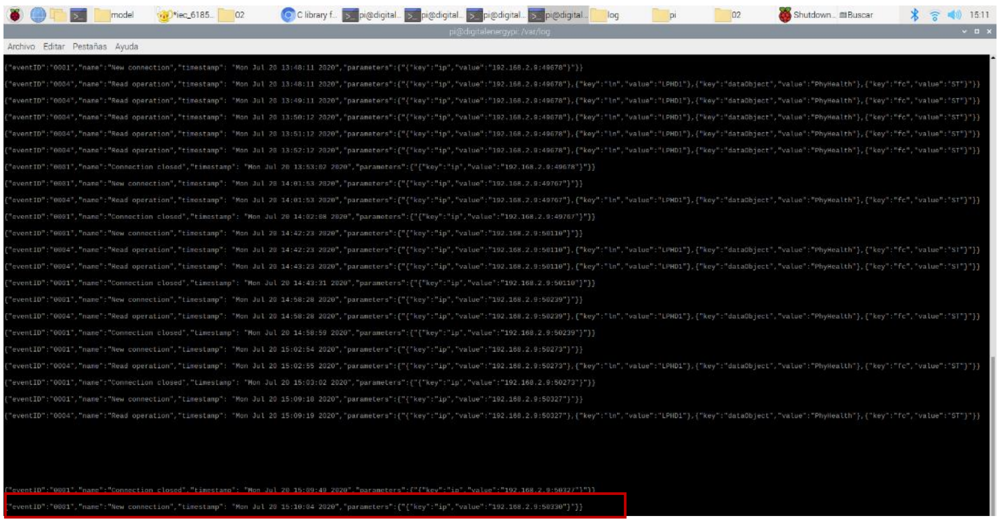
Test Case Result	Achieved

Test Case ID	HP_61850_03	**Component**	IEC 61850 honeypot
Description	The honeypot registers MMS read operations on data objects and attributes.		
SPEC ID	SPEC-F4	**Priority**	High
Prepared by	TECNALIA	**Tested by**	TECNALIA
Pre-condition(s)	• SCADA has network connectivity with IEC 61850 honeypot • IEC 61850 Honeypot must be up and listening on port 102 • Rsyslog daemon must be installed and running • On honeypot initialization PhyHealth Data object stVal is set to “Ok”		
Test steps			
1	IEC 61850 client opens a new MMS connection on port 102.		
2	Test if the connection has been successfully established		
3	Test if the honeypot has correctly registered the event on /var/log/honeypot.log		
4	IEC 61850 sends read request to retrieve status data attribute of PhyHealth Data Object on LPHD Logical Node.		
5	Test if honeypot returns the correct value for PhyHealth Data Object that has been established in initialization time		
6	Test if the honeypot has registered a “Read Operation” event with ID 0004 on /var/log/honeypot.log		
Input data	N/A		
Result	• **Step 1 and Step 2**		



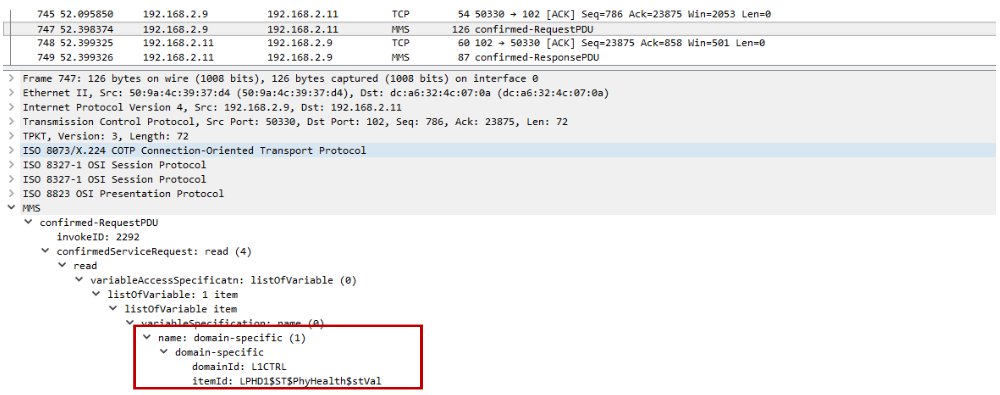
```
 81 5.854197    192.168.2.9     192.168.2.11    TCP     66 50330 → 102 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
 82 5.854943    192.168.2.11    192.168.2.9     TCP     66 102 → 50330 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
 83 5.855022    192.168.2.9     192.168.2.11    TCP     54 50330 → 102 [ACK] Seq=1 Ack=1 Win=525568 Len=0
 84 5.855153    192.168.2.9     192.168.2.11    COTP    76 CR TPDU src-ref: 0x0224 dst-ref: 0x0000
 85 5.855711    192.168.2.11    192.168.2.9     TCP     60 102 → 50330 [ACK] Seq=1 Ack=23 Win=64256 Len=0
 86 5.867458    192.168.2.11    192.168.2.9     COTP    76 CC TPDU src-ref: 0x0001 dst-ref: 0x0224
 87 5.867725    192.168.2.9     192.168.2.11    MMS    247 initiate-RequestPDU
 88 5.868138    192.168.2.11    192.168.2.9     MMS    196 initiate-ResponsePDU
 89 5.909102    192.168.2.9     192.168.2.11    TCP     54 50330 → 102 [ACK] Seq=216 Ack=165 Win=525312 Len=0
 90 6.033091    192.168.2.9     192.168.2.11    MMS    133 confirmed-RequestPDU
 91 6.033750    192.168.2.11    192.168.2.9     MMS     87 confirmed-ResponsePDU
 94 6.073487    192.168.2.9     192.168.2.11    TCP     54 50330 → 102 [ACK] Seq=295 Ack=198 Win=525312 Len=0

> Frame 81: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
> Ethernet II, Src: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4), Dst: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.9, Dst: 192.168.2.11
> Transmission Control Protocol, Src Port: 50330, Dst Port: 102, Seq: 0, Len: 0
```

- **Step 3**



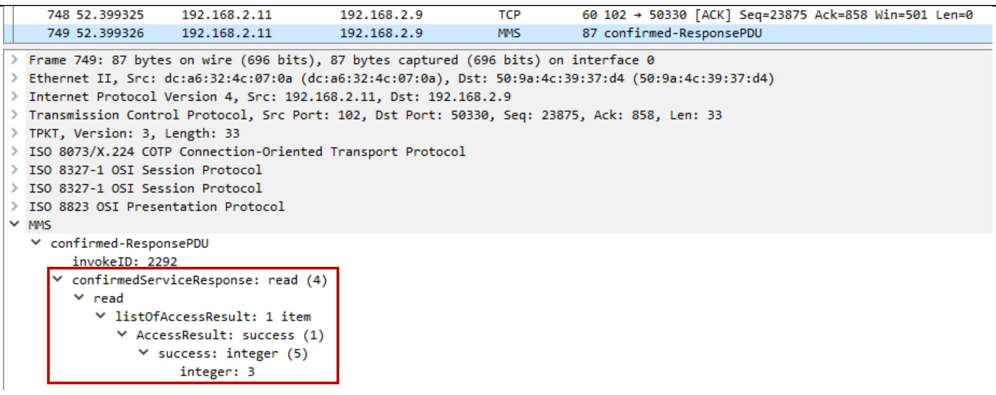
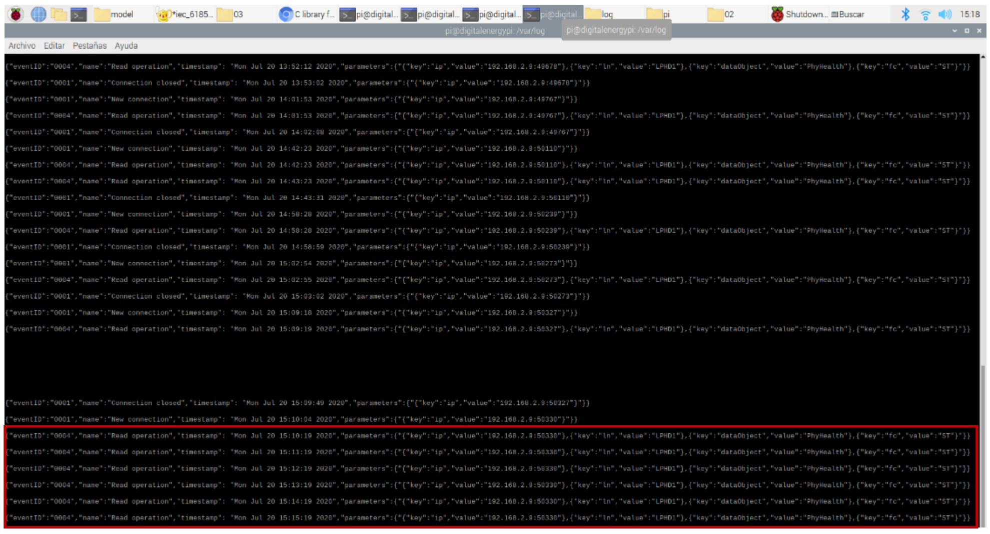
```
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 13:48:11 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:48:11 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:49:11 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:50:12 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:51:12 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:52:12 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 13:53:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:01:53 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:01:53 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:02:08 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:42:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:42:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:43:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:43:31 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:58:28 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:58:28 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:58:59 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:02:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:02:55 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 15:03:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:09:18 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:09:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}



{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 15:09:49 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:10:04 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"}"}}
```

- **Step 4**



```
745 52.095850    192.168.2.9     192.168.2.11    TCP     54 50330 → 102 [ACK] Seq=786 Ack=23875 Win=2053 Len=0
747 52.398374    192.168.2.9     192.168.2.11    MMS    126 confirmed-RequestPDU
748 52.399325    192.168.2.11    192.168.2.9     TCP     60 102 → 50330 [ACK] Seq=23875 Ack=858 Win=501 Len=0
749 52.399326    192.168.2.11    192.168.2.9     MMS     87 confirmed-ResponsePDU

> Frame 747: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
> Ethernet II, Src: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4), Dst: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.9, Dst: 192.168.2.11
> Transmission Control Protocol, Src Port: 50330, Dst Port: 102, Seq: 786, Ack: 23875, Len: 72
> TPKT, Version: 3, Length: 72
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-RequestPDU
      invokeID: 2292
    v confirmedServiceRequest: read (4)
      v read
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: LPHD1$ST$PhyHealth$stVal
```

- **Step 5**

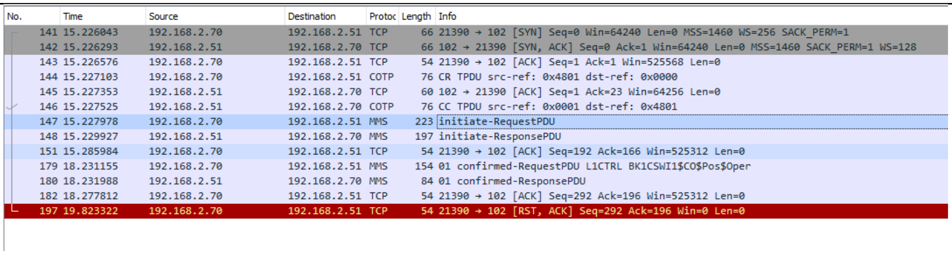
	
	Step 6 
Test Case Result	Achieved

```
748 52.399325   192.168.2.11   192.168.2.9   TCP   60 102 → 50330 [ACK] Seq=23875 Ack=858 Win=501 Len=0
749 52.399326   192.168.2.11   192.168.2.9   MMS   87 confirmed-ResponsePDU

> Frame 749: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
> Ethernet II, Src: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a), Dst: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 50330, Seq: 23875, Ack: 858, Len: 33
> TPKT, Version: 3, Length: 33
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
    v confirmed-ResponsePDU
        invokeID: 2292
      v confirmedServiceResponse: read (4)
          v read
            v listOfAccessResult: 1 item
              v AccessResult: success (1)
                v success: integer (5)
                    integer: 3
```

```
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 13:52:12 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 13:53:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49678"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:01:53 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:01:53 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:02:08 2020","parameters":{"{"key":"ip","value":"192.168.2.9:49767"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:42:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:42:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:43:23 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:43:31 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50110"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 14:58:28 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 14:58:28 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 14:58:59 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50239"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:02:54 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:02:55 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 15:03:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50273"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:09:18 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:09:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}




{"eventID":"0001","name":"Connection closed","timestamp": "Mon Jul 20 15:09:49 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50327"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Mon Jul 20 15:10:04 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:10:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:11:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:12:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:13:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:14:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Mon Jul 20 15:15:19 2020","parameters":{"{"key":"ip","value":"192.168.2.9:50330"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

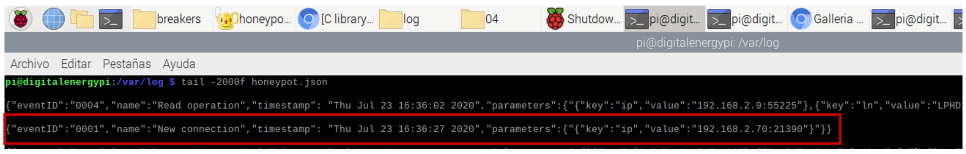
Test Case ID	HP_61850_04	**Component**	IEC 61850 honeypot
Description	Test case to verify honeypot's control operation interlocking logic.		
SPEC ID	SPEC-F4	**Priority**	High
Prepared by	TECNALIA	**Tested by**	TECNALIA
Pre-condition(s)	• SCADA has network connectivity with IEC 61850 honeypot • IEC 61850 Honeypot must be up and listening on port 102 • EnaCls and EnaOpn Data Object values in CSWI are True at initialization time. • When the honeypot app starts the status of the breaker is closed		
Test steps			

1	IEC 61850 client opens a new MMS connection on port 102.
2	Test if the connection has been successfully established
3	Test if the honeypot has correctly registered the event on /var/log/honeypot.log
4	Send control operation on CSWI to change Circuit Breaker position to closed
5	Test if the honeypot rejects the operation due to “Object Access Denied” cause.
6	Test if the honeypot has registered a “Control Operation” event with ID 0003 and the cause for the denial on /var/log/honeypot.log
7	Send control operation on CSWI to change Circuit Breaker position to open
8	Test if the response of the honeypot to the ‘Control Operation’ is SUCCESS
9	Test if the honeypot has registered a “Control Operation” event with ID 0003 and the cause for the denial on /var/log/honeypot.log
10	Test if Circuit Breaker position status has changed in SCADA’s HMI to opened
11	Send control operation on CSWI to change Circuit Breaker position to opened
12	Test if the honeypot rejects the operation due to “Object Access Denied” cause
13	Test if the honeypot has registered a “Control Operation” event with ID 0003 and the cause for the denial on /var/log/honeypot.log
14	Change EnaCls to FALSE state
15	Send control operation on CSWI to change Circuit Breaker position to closed
16	Test if the honeypot rejects the operation due to “Object Access Denied” cause
17	Test if the honeypot has registered a “Control Operation” event with ID 0003 and the cause for the denial on /var/log/honeypot.log
18	Change EnaOpn to FALSE state
19	Send control operation on CSWI to change Circuit Breaker position to opened
20	Test if the honeypot rejects the operation due to “Object Access Denied” cause
21	Test if the honeypot has registered a “Control Operation” event with ID 0003 and the cause for the denial on /var/log/honeypot.log
Input data	N/A
Result	• **Step 1 and Step 2**



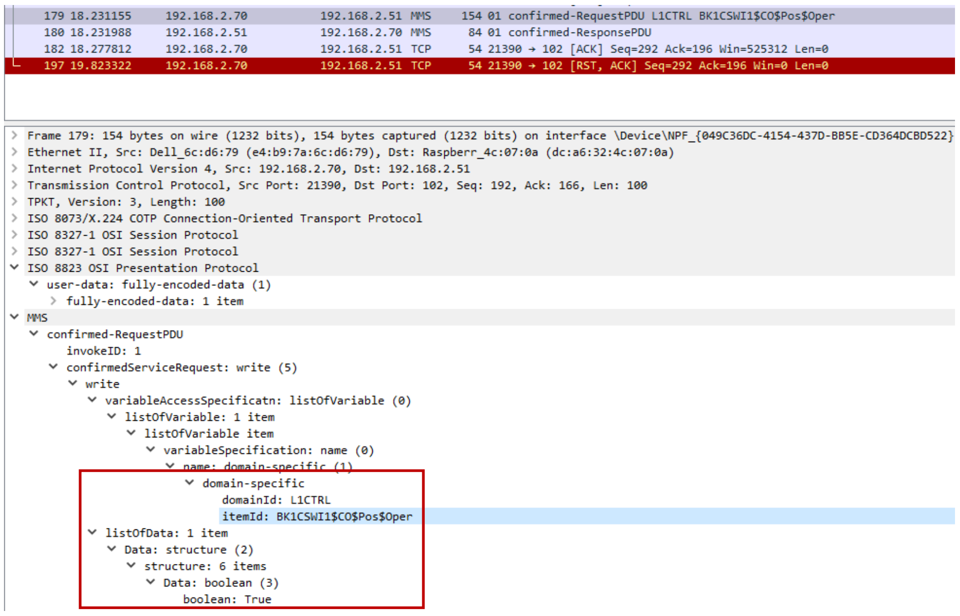
```
No.  Time       Source        Destination   Protoc Length Info
141 15.226043  192.168.2.70  192.168.2.51  TCP    66 21390 → 102 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
142 15.226293  192.168.2.51  192.168.2.70  TCP    66 102 → 21390 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
143 15.226576  192.168.2.70  192.168.2.51  TCP    54 21390 → 102 [ACK] Seq=1 Ack=1 Win=525568 Len=0
144 15.227103  192.168.2.70  192.168.2.51  COTP   76 CR TPDU src-ref: 0x4801 dst-ref: 0x0000
145 15.227353  192.168.2.51  192.168.2.70  TCP    60 102 → 21390 [ACK] Seq=1 Ack=23 Win=64256 Len=0
146 15.227525  192.168.2.51  192.168.2.70  COTP   76 CC TPDU src-ref: 0x0001 dst-ref: 0x4801
147 15.227978  192.168.2.70  192.168.2.51  MMS   223 initiate-RequestPDU
148 15.229927  192.168.2.51  192.168.2.70  MMS   197 initiate-ResponsePDU
151 15.285984  192.168.2.70  192.168.2.51  TCP    54 21390 → 102 [ACK] Seq=192 Ack=166 Win=525312 Len=0
179 18.231155  192.168.2.70  192.168.2.51  MMS   154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
180 18.231988  192.168.2.51  192.168.2.70  MMS    84 01 confirmed-ResponsePDU
182 18.277812  192.168.2.70  192.168.2.51  TCP    54 21390 → 102 [ACK] Seq=292 Ack=196 Win=525312 Len=0
197 19.823322  192.168.2.70  192.168.2.51  TCP    54 21390 → 102 [RST, ACK] Seq=292 Ack=196 Win=0 Len=0
```

- **Step 3**



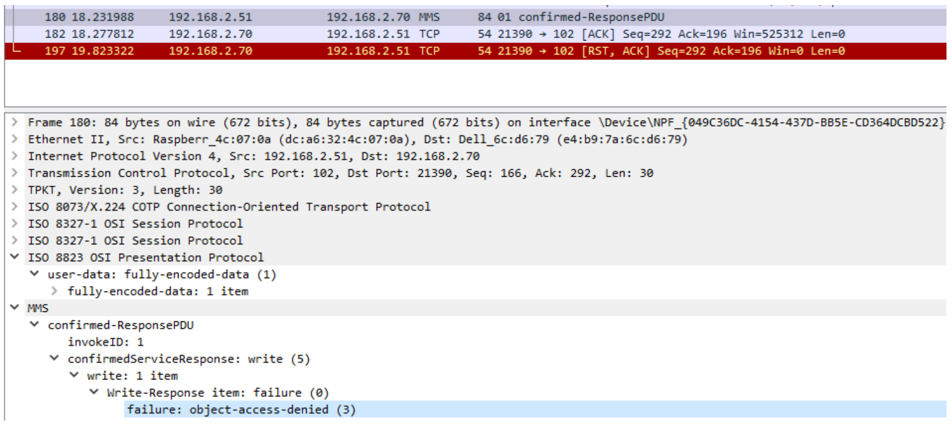
```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:36:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 16:36:27 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21390"}"}}
```

- **Step 4**



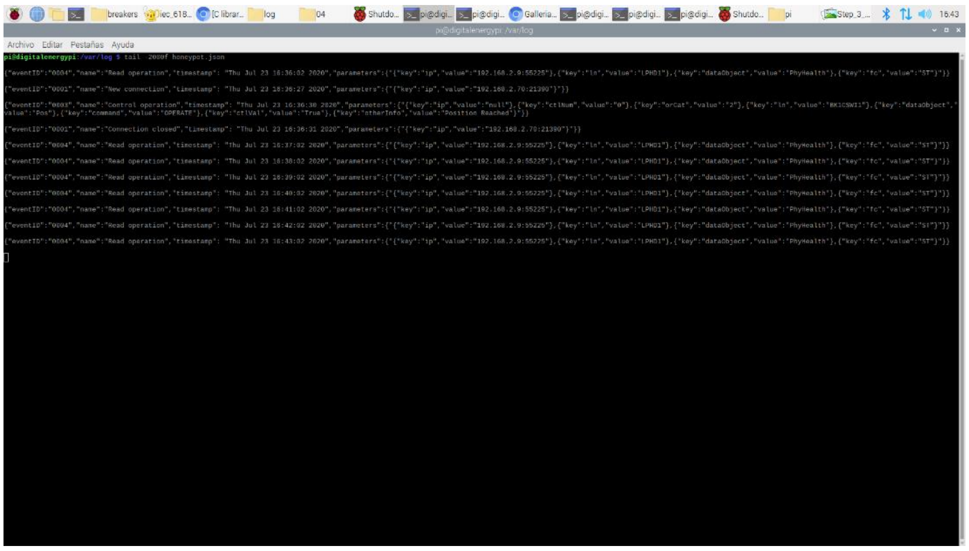
```
179 18.231155   192.168.2.70   192.168.2.51 MMS   154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
180 18.231988   192.168.2.51   192.168.2.70 MMS    84 01 confirmed-ResponsePDU
182 18.277812   192.168.2.70   192.168.2.51 TCP    54 21390 → 102 [ACK] Seq=292 Ack=196 Win=525312 Len=0
197 19.823322   192.168.2.70   192.168.2.51 TCP    54 21390 → 102 [RST, ACK] Seq=292 Ack=196 Win=0 Len=0

> Frame 179: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79), Dst: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.70, Dst: 192.168.2.51
> Transmission Control Protocol, Src Port: 21390, Dst Port: 102, Seq: 192, Ack: 166, Len: 100
> TPKT, Version: 3, Length: 100
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-RequestPDU
      invokeID: 1
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: True
```

- **Step 5**



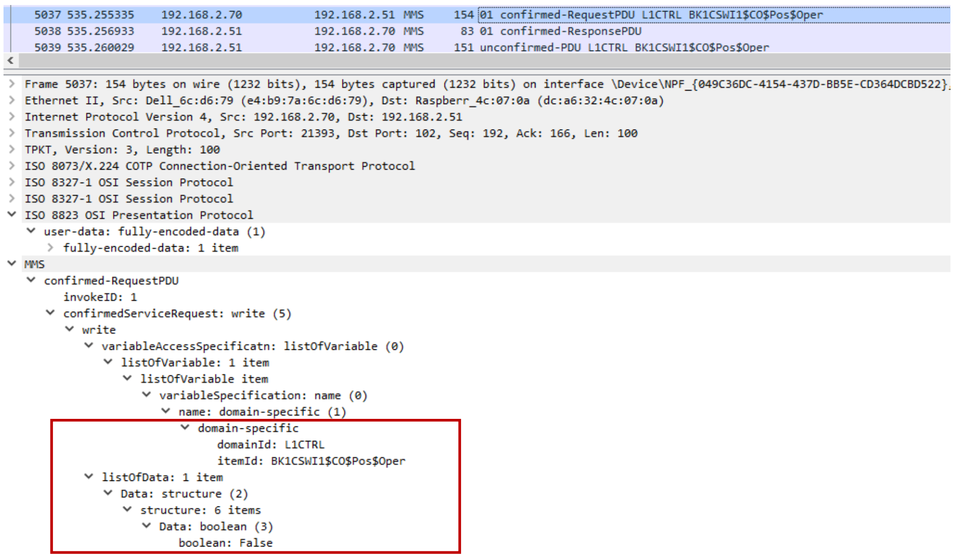
```
180 18.231988   192.168.2.51   192.168.2.70 MMS    84 01 confirmed-ResponsePDU
182 18.277812   192.168.2.70   192.168.2.51 TCP    54 21390 → 102 [ACK] Seq=292 Ack=196 Win=525312 Len=0
197 19.823322   192.168.2.70   192.168.2.51 TCP    54 21390 → 102 [RST, ACK] Seq=292 Ack=196 Win=0 Len=0

> Frame 180: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79)
> Internet Protocol Version 4, Src: 192.168.2.51, Dst: 192.168.2.70
> Transmission Control Protocol, Src Port: 102, Dst Port: 21390, Seq: 166, Ack: 292, Len: 30
> TPKT, Version: 3, Length: 30
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-ResponsePDU
      invokeID: 1
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: failure (0)
            failure: object-access-denied (3)
```

- **Step 6**



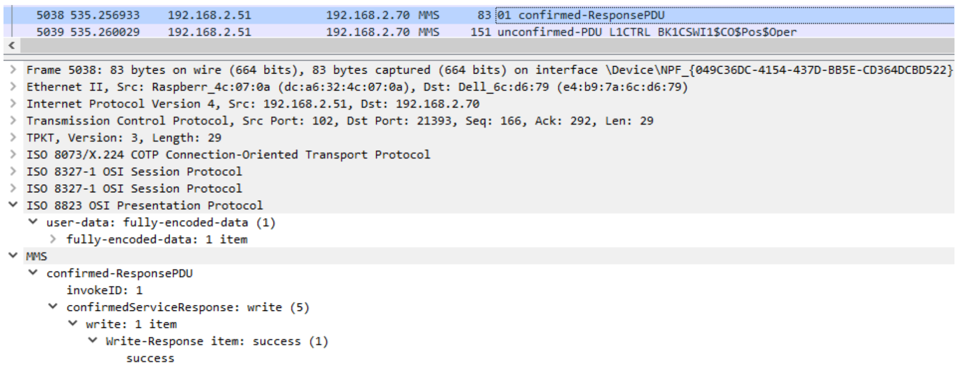
```
pi@digitalenergypi:/var/log $ tail -2000f honeypot.json
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:36:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 16:36:27 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21390"}"}}
{"eventID":"0003","name":"Control operation","timestamp": "Thu Jul 23 16:36:30 2020","parameters":{"{"key":"ip","value":"null"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"dataObject","
value":"Pos"},{"key":"command","value":"OPERATE"},{"key":"ctlVal","value":"True"},{"key":"otherInfo","value":"Position Reached"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 16:36:31 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21390"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:37:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:38:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:39:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:40:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:41:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:42:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:43:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

- **Step 7**



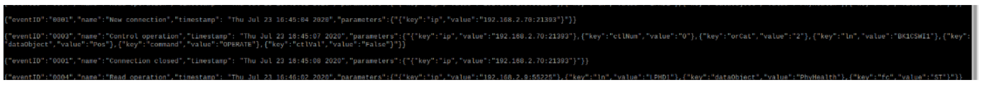
```
5037 535.255335   192.168.2.70        192.168.2.51  MMS    154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
5038 535.256933   192.168.2.51        192.168.2.70  MMS     83 01 confirmed-ResponsePDU
5039 535.260029   192.168.2.51        192.168.2.70  MMS    151 unconfirmed-PDU L1CTRL BK1CSWI1$CO$Pos$Oper

> Frame 5037: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79), Dst: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.70, Dst: 192.168.2.51
> Transmission Control Protocol, Src Port: 21393, Dst Port: 102, Seq: 192, Ack: 166, Len: 100
> TPKT, Version: 3, Length: 100
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-RequestPDU
      invokeID: 1
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: False
```

- **Step 8**



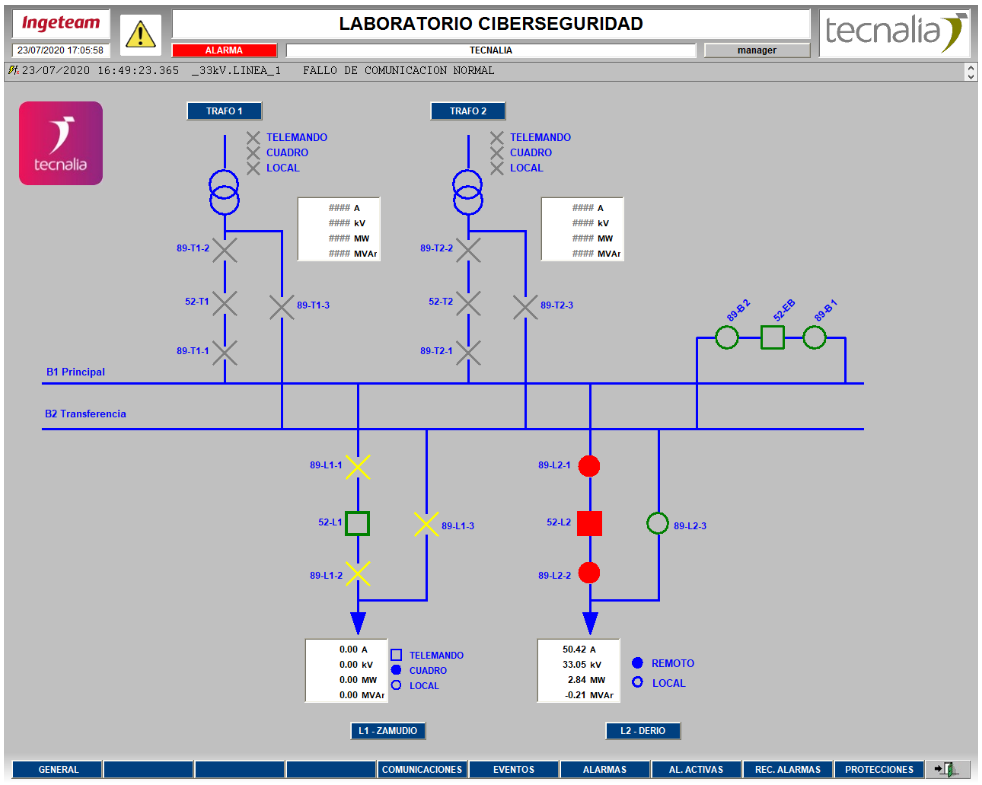
```
5038 535.256933    192.168.2.51        192.168.2.70  MMS      83 01 confirmed-ResponsePDU
5039 535.260029    192.168.2.51        192.168.2.70  MMS     151 unconfirmed-PDU L1CTRL BK1CSWI1$CO$Pos$Oper

> Frame 5038: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79)
> Internet Protocol Version 4, Src: 192.168.2.51, Dst: 192.168.2.70
> Transmission Control Protocol, Src Port: 102, Dst Port: 21393, Seq: 166, Ack: 292, Len: 29
> TPKT, Version: 3, Length: 29
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-ResponsePDU
      invokeID: 1
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: success (1)
            success
```

- **Step 9**

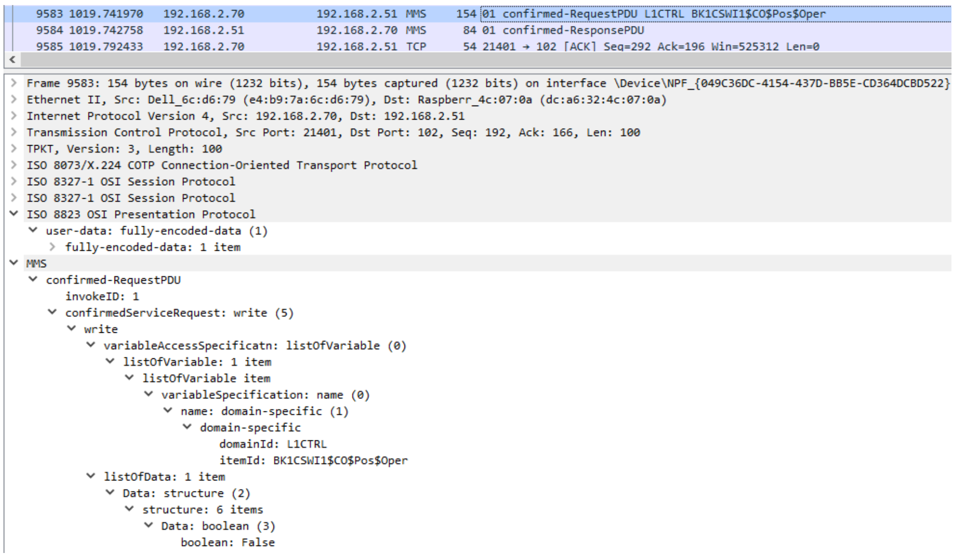


```
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 16:45:04 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21393"}"}}

{"eventID":"0003","name":"Control operation","timestamp": "Thu Jul 23 16:45:07 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21393"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":
"dataObject","value":"Pos"},{"key":"command","value":"OPERATE"},{"key":"ctlVal","value":"False"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 16:45:08 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21393"}"}}

{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 16:46:02 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

- **Step 10**

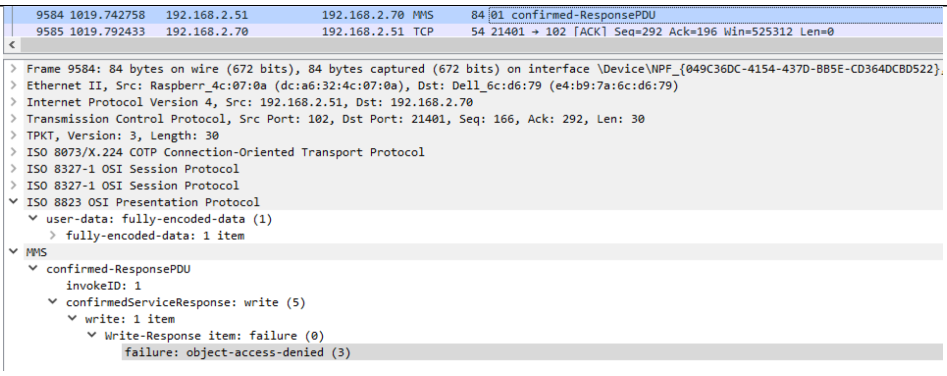


- **Step 11**



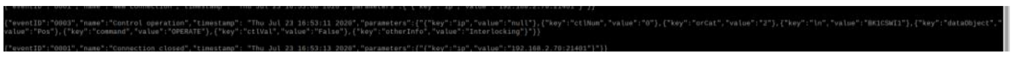
```
9583 1019.741970   192.168.2.70         192.168.2.51  MMS    154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
9584 1019.742758   192.168.2.51         192.168.2.70  MMS     84 01 confirmed-ResponsePDU
9585 1019.792433   192.168.2.70         192.168.2.51  TCP     54 21401 → 102 [ACK] Seq=292 Ack=196 Win=525312 Len=0

> Frame 9583: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79), Dst: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.70, Dst: 192.168.2.51
> Transmission Control Protocol, Src Port: 21401, Dst Port: 102, Seq: 192, Ack: 166, Len: 100
> TPKT, Version: 3, Length: 100
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-RequestPDU
      invokeID: 1
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: False
```

- **Step 12**



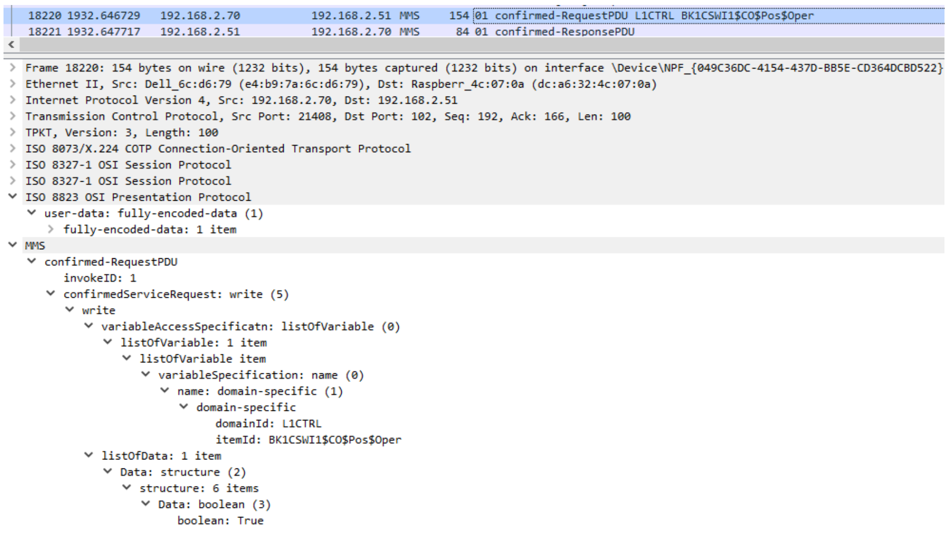
```
9584 1019.742758   192.168.2.51          192.168.2.70  MMS     84 01 confirmed-ResponsePDU
9585 1019.792433   192.168.2.70          192.168.2.51  TCP     54 21401 → 102 [ACK] Seq=292 Ack=196 Win=525312 Len=0

> Frame 9584: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522},
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79)
> Internet Protocol Version 4, Src: 192.168.2.51, Dst: 192.168.2.70
> Transmission Control Protocol, Src Port: 102, Dst Port: 21401, Seq: 166, Ack: 292, Len: 30
> TPKT, Version: 3, Length: 30
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-ResponsePDU
      invokeID: 1
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: failure (0)
            failure: object-access-denied (3)
```

- **Step 13**



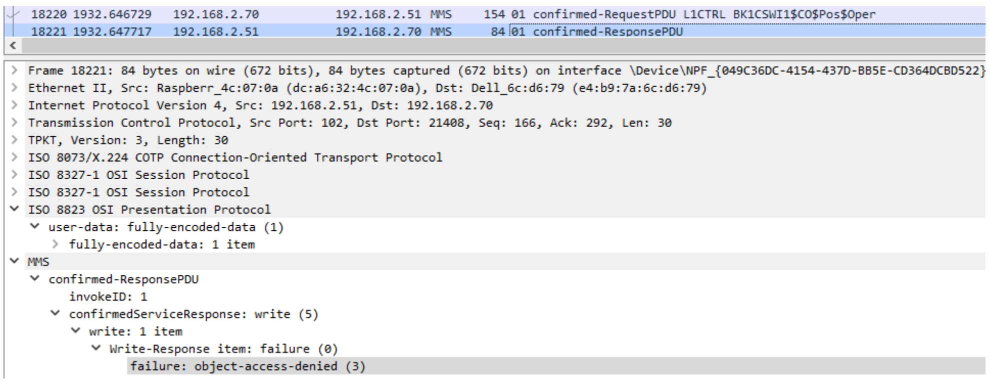
```
{"eventID":"0003","name":"Control operation","timestamp": "Thu Jul 23 16:53:11 2020","parameters":[{"key":"ip","value":"null"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"dataObject","value":"Pos"},{"key":"command","value":"OPERATE"},{"key":"ctlVal","value":"False"},{"key":"otherInfo","value":"Interlocking"}"}}

{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 16:53:13 2020","parameters":[{"key":"ip","value":"192.168.2.70:21401"}"}}
```

- **Step 15**



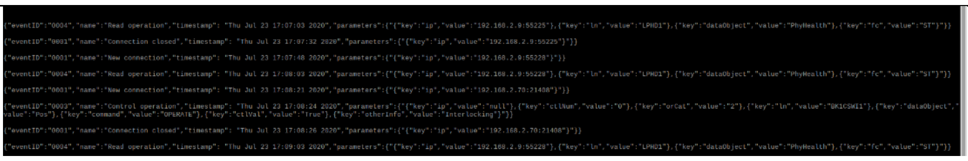
```
18220 1932.646729   192.168.2.70          192.168.2.51  MMS    154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
18221 1932.647717   192.168.2.51          192.168.2.70  MMS     84 01 confirmed-ResponsePDU

> Frame 18220: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79), Dst: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.70, Dst: 192.168.2.51
> Transmission Control Protocol, Src Port: 21408, Dst Port: 102, Seq: 192, Ack: 166, Len: 100
> TPKT, Version: 3, Length: 100
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-RequestPDU
      invokeID: 1
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: True
```

- **Step 16**



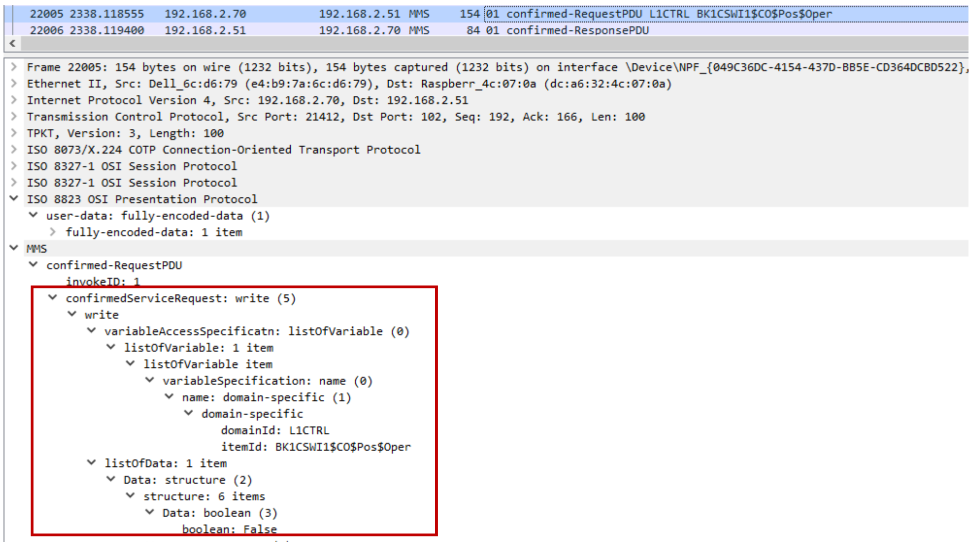
```
18220 1932.646729   192.168.2.70          192.168.2.51  MMS    154 01 confirmed-RequestPDU L1CTRL BK1CSWI1$CO$Pos$Oper
18221 1932.647717   192.168.2.51          192.168.2.70  MMS     84 01 confirmed-ResponsePDU

> Frame 18221: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79)
> Internet Protocol Version 4, Src: 192.168.2.51, Dst: 192.168.2.70
> Transmission Control Protocol, Src Port: 102, Dst Port: 21408, Seq: 166, Ack: 292, Len: 30
> TPKT, Version: 3, Length: 30
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-ResponsePDU
      invokeID: 1
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: failure (0)
            failure: object-access-denied (3)
```

- **Step 17**



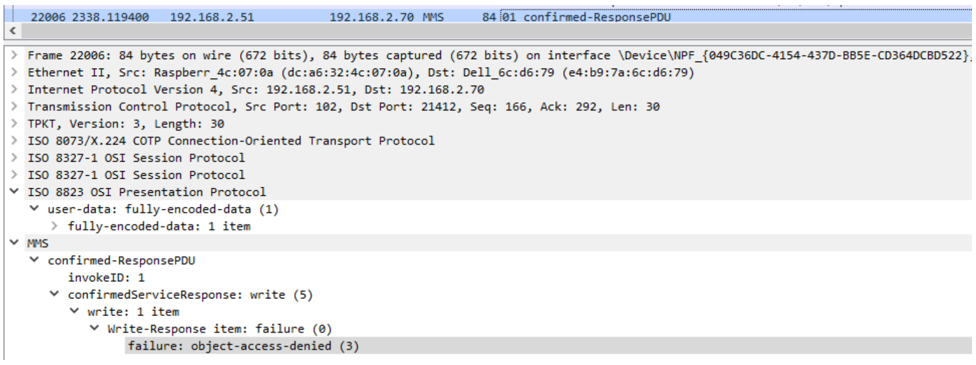
```
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:07:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 17:07:32 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55225"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 17:07:48 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55228"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:08:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55228"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 17:08:21 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21408"}"}}
{"eventID":"0003","name":"Control operation","timestamp": "Thu Jul 23 17:08:24 2020","parameters":{"{"key":"ip","value":"null"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"dataObject","
value":"Pos"},{"key":"command","value":"OPERATE"},{"key":"ctlVal","value":"True"},{"key":"otherInfo","value":"Interlocking"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 17:08:26 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21408"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:09:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55228"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

- **Step 19**



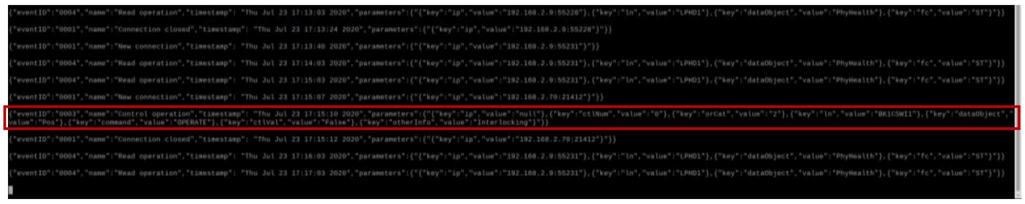
```
22005 2338.118555   192.168.2.70          192.168.2.51  MMS    154 01 confirmed-RequestPDU  L1CTRL  BK1CSWI1$CO$Pos$Oper
22006 2338.119400   192.168.2.51          192.168.2.70  MMS     84 01 confirmed-ResponsePDU

> Frame 22005: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522},
> Ethernet II, Src: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79), Dst: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.70, Dst: 192.168.2.51
> Transmission Control Protocol, Src Port: 21412, Dst Port: 102, Seq: 192, Ack: 166, Len: 100
> TPKT, Version: 3, Length: 100
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-RequestPDU
      invokeID: 1
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          v listOfVariable: 1 item
            v listOfVariable item
              v variableSpecification: name (0)
                v name: domain-specific (1)
                  v domain-specific
                      domainId: L1CTRL
                      itemId: BK1CSWI1$CO$Pos$Oper
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: False
```

- **Step 20**



```
22006 2338.119400   192.168.2.51          192.168.2.70  MMS     84 01 confirmed-ResponsePDU

> Frame 22006: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface \Device\NPF_{049C36DC-4154-437D-BB5E-CD364DCBD522}
> Ethernet II, Src: Raspberr_4c:07:0a (dc:a6:32:4c:07:0a), Dst: Dell_6c:d6:79 (e4:b9:7a:6c:d6:79)
> Internet Protocol Version 4, Src: 192.168.2.51, Dst: 192.168.2.70
> Transmission Control Protocol, Src Port: 102, Dst Port: 21412, Seq: 166, Ack: 292, Len: 30
> TPKT, Version: 3, Length: 30
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
v ISO 8823 OSI Presentation Protocol
  v user-data: fully-encoded-data (1)
    > fully-encoded-data: 1 item
v MMS
  v confirmed-ResponsePDU
      invokeID: 1
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: failure (0)
            failure: object-access-denied (3)
```

- **Step 21**



```
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:13:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55228"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 17:13:24 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55228"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 17:13:40 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55231"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:14:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55231"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:15:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55231"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 17:15:07 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21412"}"}}
{"eventID":"0003","name":"Control operation","timestamp": "Thu Jul 23 17:15:10 2020","parameters":{"{"key":"ip","value":"null"},{"key":"ctlNum","value":"0"},{"key":"orCat","value":"2"},{"key":"ln","value":"BK1CSWI1"},{"key":"dataObject","
value":"Pos"},{"key":"command","value":"OPERATE"},{"key":"ctlVal","value":"False"},{"key":"otherInfo","value":"Interlocking"}"}}
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 17:15:12 2020","parameters":{"{"key":"ip","value":"192.168.2.70:21412"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:16:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55231"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:17:03 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55231"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

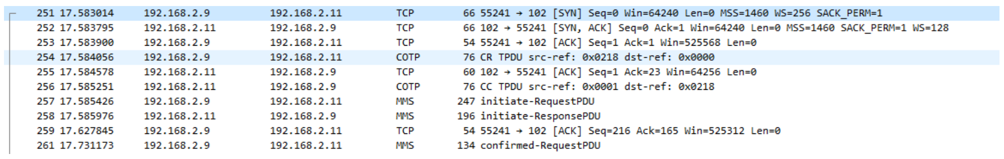
Test Case Result	Achieved

Test Case ID	HP_61850_05	Component	IEC 61850 honeypot
Description	Test case to verify if the honeypot is able to deal with Buffered and Unbuffered reports.		
SPEC ID	SPEC-F4	**Priority**	High
Prepared by	TECNALIA	**Tested by**	TECNALIA
Pre-condition(s)	• SCADA has network connectivity with IEC 61850 honeypot • IEC 61850 Honeypot must be up and listening on port 102 • Rsyslog daemon must be installed and running • One buffered and one unbuffered report are defined into honeypot ICD file. The datasets are configured to be correctly interpreted by SCADA HMI.		

Test steps	
1	IEC 61850 client opens a new MMS connection on port 102.
2	Test if the connection has been successfully established
3	Test if the honeypot has correctly registered the event on /var/log/honeypot.log
4	Test if the Circuit Breaker status is correctly displayed on the SCADA's HMI
5	Test if unbuffered Reports are periodically transmitted from honeypot to SCADA
6	Send control operation on CSWI to change Circuit Breaker position to opened
7	Test if a Buffered Report is generated by the honeypot informing the SCADA of circuit breaker status change

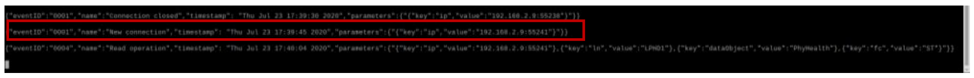
Input data	N/A
Result	

- **Step 1 and Step 2**



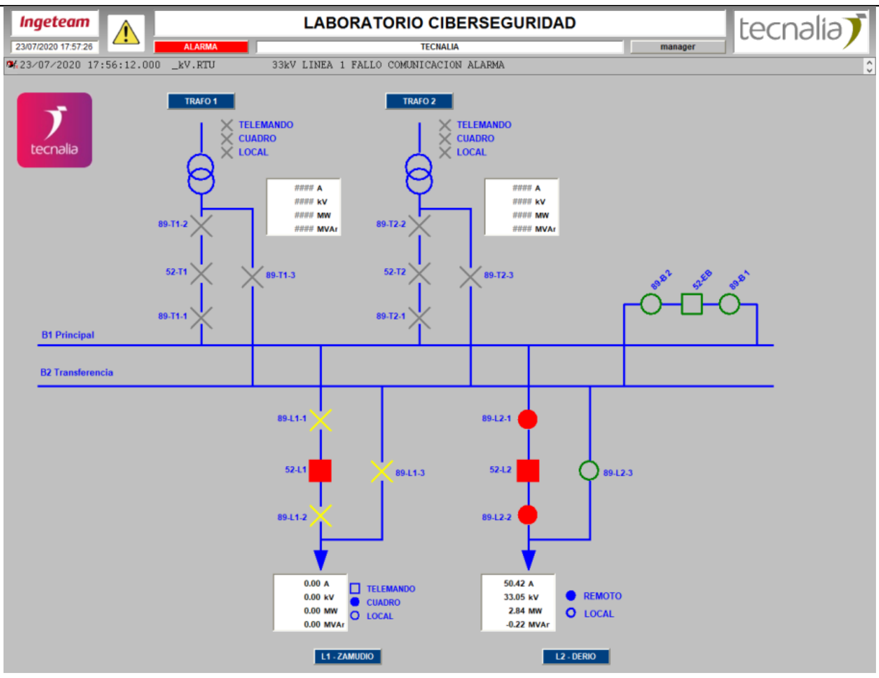
```
251 17.583014   192.168.2.9    192.168.2.11   TCP    66 55241 → 102 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
252 17.583795   192.168.2.11   192.168.2.9    TCP    66 102 → 55241 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
253 17.583900   192.168.2.9    192.168.2.11   TCP    54 55241 → 102 [ACK] Seq=1 Ack=1 Win=525568 Len=0
254 17.584056   192.168.2.9    192.168.2.11   COTP   76 CR TPDU src-ref: 0x0218 dst-ref: 0x0000
255 17.584578   192.168.2.11   192.168.2.9    TCP    60 102 → 55241 [ACK] Seq=1 Ack=23 Win=64256 Len=0
256 17.585251   192.168.2.11   192.168.2.9    COTP   76 CC TPDU src-ref: 0x0001 dst-ref: 0x0218
257 17.585426   192.168.2.9    192.168.2.11   MMS   247 initiate-RequestPDU
258 17.585976   192.168.2.11   192.168.2.9    MMS   196 initiate-ResponsePDU
259 17.627845   192.168.2.9    192.168.2.11   TCP    54 55241 → 102 [ACK] Seq=216 Ack=165 Win=525312 Len=0
261 17.731173   192.168.2.9    192.168.2.11   MMS   134 confirmed-RequestPDU
```

- **Step 3**

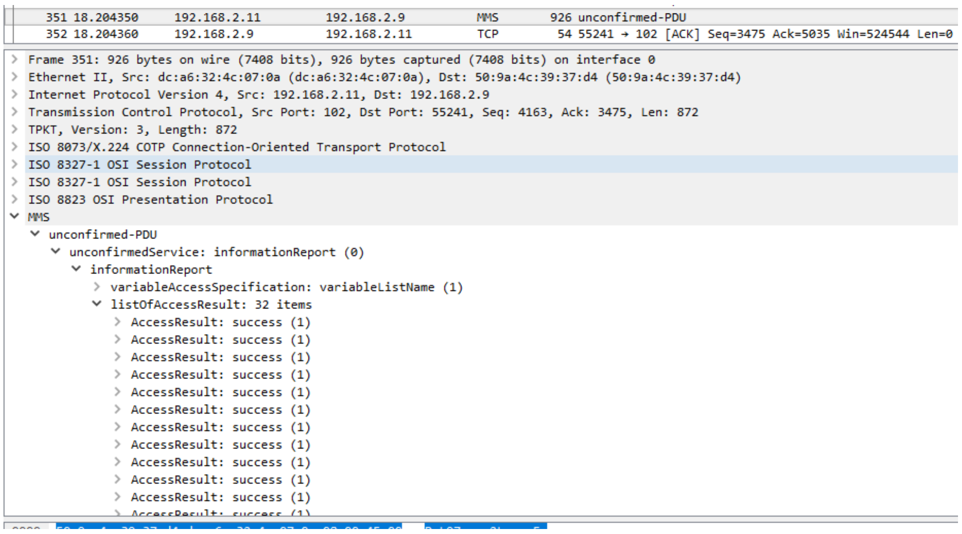


```
{"eventID":"0001","name":"Connection closed","timestamp": "Thu Jul 23 17:39:30 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55238"}"}}
"eventID":"0001","name":"New connection","timestamp": "Thu Jul 23 17:39:45 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55241"}"}}
{"eventID":"0004","name":"Read operation","timestamp": "Thu Jul 23 17:40:04 2020","parameters":{"{"key":"ip","value":"192.168.2.9:55241"},{"key":"ln","value":"LPHD1"},{"key":"dataObject","value":"PhyHealth"},{"key":"fc","value":"ST"}"}}
```

- **Step 4**

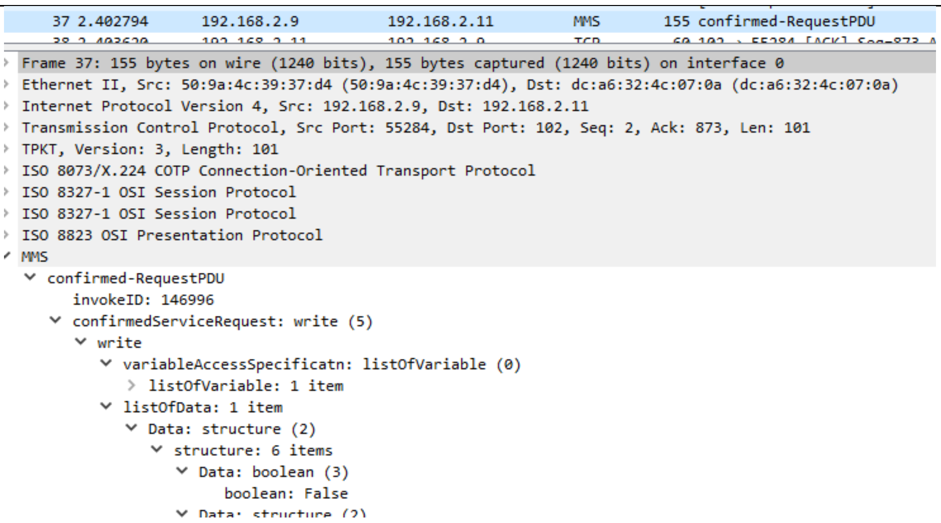


- **Step 5**

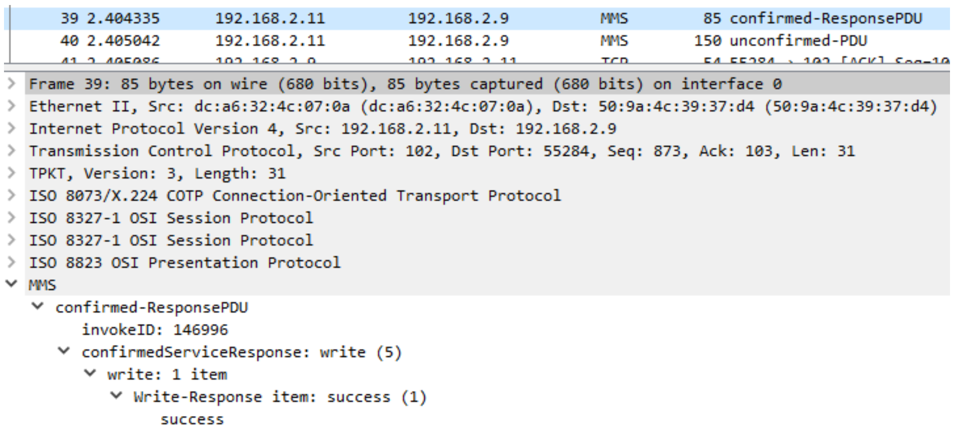


```
351 18.204350    192.168.2.11      192.168.2.9       MMS      926 unconfirmed-PDU
352 18.204360    192.168.2.9       192.168.2.11      TCP       54 55241 → 102 [ACK] Seq=3475 Ack=5035 Win=524544 Len=0

> Frame 351: 926 bytes on wire (7408 bits), 926 bytes captured (7408 bits) on interface 0
> Ethernet II, Src: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a), Dst: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 55241, Seq: 4163, Ack: 3475, Len: 872
> TPKT, Version: 3, Length: 872
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v unconfirmed-PDU
    v unconfirmedService: informationReport (0)
      v informationReport
        > variableAccessSpecification: variableListName (1)
        v listOfAccessResult: 32 items
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
          > AccessResult: success (1)
```

- **Step 6**

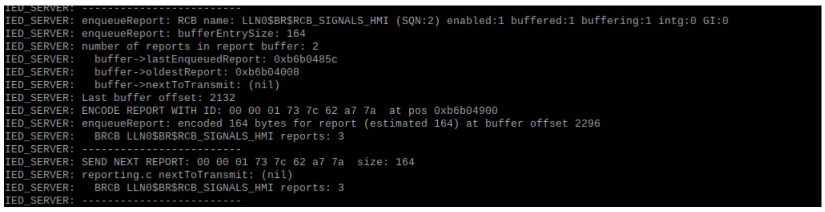


```
   37 2.402794      192.168.2.9        192.168.2.11       MMS       155 confirmed-RequestPDU
> Frame 37: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface 0
> Ethernet II, Src: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4), Dst: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a)
> Internet Protocol Version 4, Src: 192.168.2.9, Dst: 192.168.2.11
> Transmission Control Protocol, Src Port: 55284, Dst Port: 102, Seq: 2, Ack: 873, Len: 101
> TPKT, Version: 3, Length: 101
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-RequestPDU
      invokeID: 146996
    v confirmedServiceRequest: write (5)
      v write
        v variableAccessSpecificatn: listOfVariable (0)
          > listOfVariable: 1 item
        v listOfData: 1 item
          v Data: structure (2)
            v structure: 6 items
              v Data: boolean (3)
                  boolean: False
```

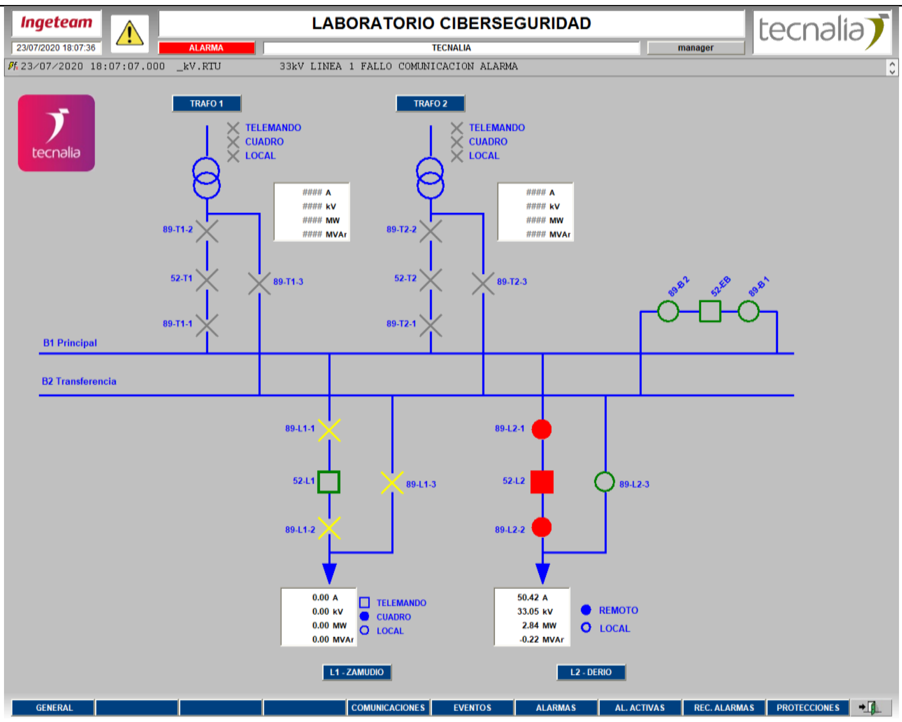


```
   39 2.404335      192.168.2.11       192.168.2.9        MMS        85 confirmed-ResponsePDU
   40 2.405042      192.168.2.11       192.168.2.9        MMS       150 unconfirmed-PDU
> Frame 39: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
> Ethernet II, Src: dc:a6:32:4c:07:0a (dc:a6:32:4c:07:0a), Dst: 50:9a:4c:39:37:d4 (50:9a:4c:39:37:d4)
> Internet Protocol Version 4, Src: 192.168.2.11, Dst: 192.168.2.9
> Transmission Control Protocol, Src Port: 102, Dst Port: 55284, Seq: 873, Ack: 103, Len: 31
> TPKT, Version: 3, Length: 31
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
v MMS
  v confirmed-ResponsePDU
      invokeID: 146996
    v confirmedServiceResponse: write (5)
      v write: 1 item
        v Write-Response item: success (1)
            success
```

- **Step 7**



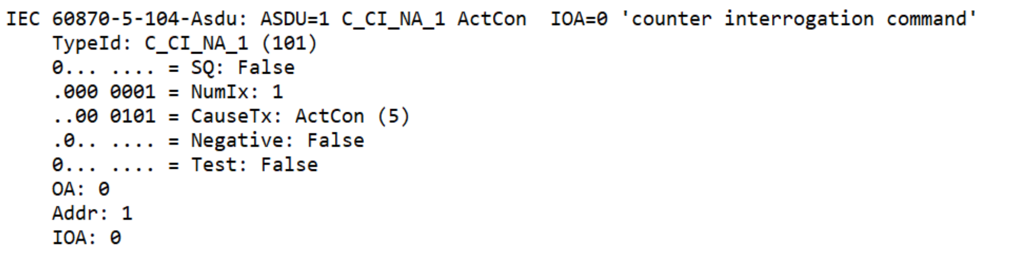
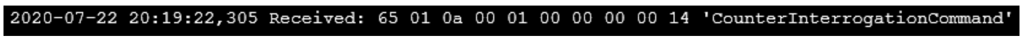
```
IED_SERVER: enqueueReport: RCB name: LLN0$BR$RCB_SIGNALS_HMI (SQN:2) enabled:1 buffered:1 buffering:1 intg:0 GI:0
IED_SERVER: enqueueReport: bufferEntrySize: 164
IED_SERVER: number of reports in report buffer: 2
IED_SERVER:   buffer->lastEnqueuedReport: 0xb6b0485c
IED_SERVER:   buffer->oldestReport: 0xb6b04008
IED_SERVER:   buffer->nextToTransmit: (nil)
IED_SERVER: Last buffer offset: 2132
IED_SERVER: ENCODE REPORT WITH ID: 00 00 01 73 7c 62 a7 7a  at pos 0xb6b04900
IED_SERVER: enqueueReport: encoded 164 bytes for report (estimated 164) at buffer offset 2296
IED_SERVER:   BRCB LLN0$BR$RCB_SIGNALS_HMI reports: 3
IED_SERVER: ------------------------
IED_SERVER: SEND NEXT REPORT: 00 00 01 73 7c 62 a7 7a  size: 164
IED_SERVER: reporting.c nextToTransmit: (nil)
IED_SERVER:   BRCB LLN0$BR$RCB_SIGNALS_HMI reports: 3
IED_SERVER: ------------------------
```

	 Ingeteam 23/07/2020 18:07:36 LABORATORIO CIBERSEGURIDAD ALARMA TECNALIA manager tecnalia TRAFO 1 TRAFO 2 TELEMANDO CUADRO LOCAL #### A #### kV #### MW #### MVAr 89-T1-2 52-T1 89-T1-3 89-T1-1 89-T2-2 52-T2 89-T2-3 89-T2-1 89-B2 52-EB 89-B1 B1 Principal B2 Transferencia 89-L1-1 52-L1 89-L1-3 89-L1-2 89-L2-1 52-L2 89-L2-3 89-L2-2 0.00 A 0.00 kV 0.00 MW 0.00 MVAr 50.42 A 33.05 kV 2.84 MW -0.22 MVAr REMOTO L1 - ZAMUDIO L2 - DERIO GENERAL COMUNICACIONES EVENTOS ALARMAS AL. ACTIVAS REC. ALARMAS PROTECCIONES 23/07/2020 18:07:07.000 _kV.RTU 33kV LINEA 1 FALLO COMUNICACION ALARMA
Test Case Result	Achieved

6.2 IEC 60870-5-104 Honeypot

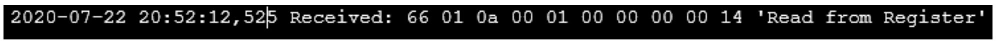
Test Case ID	HP_60870_104_01	**Component**	IEC 60870-5-104 honeypot
Description	The Master sends a Counter Interrogation command to RTU (IEC104 honeypot) using I-format to performed information transfer. The RTU supports the Counter Interrogation Mode A (Counter freeze without reset). The Counter Interrogation command freezes totals at common time.		
SPEC ID	SPEC-F1, SPEC-F4	**Priority**	Medium
Prepared by	SID	**Tested by**	SID
Pre-condition(s)	• There is a network connection with IEC 60870-5-104 honeypot • IEC 60870-5-104 honeypot must be up and listening on port 2404 • Install and execute the Rsyslog daemon.		
Test steps			
1	IEC4 client (Slave) attempts to connect with the Master.		

2	Test if the Master sends an "Activation" command, with IOA=0. The element includes information for the Counter Interrogation command.
3	Test if the RTU receives the "Counter Interrogation command" from the Master. The length of the APDU is 20 (in decimal). The "Type identification" is C_CI_NA_1 (Counter interrogation command). The number of objects is 1. The related "Counter interrogation request qualifiers" is set to 5 (general counter interrogation) which references every counter in the RTU. Note that, the initial value of the Counter is 0.
4	Test if the RTU responds with the "Activation Confirmation". It supports Mode A, the Sequence number (SQ) for the IOA is increased. The RTU changes the counter value to 1. The RTU sends the Counter Interrogation value in the response to the Master.
5	Test if the RTU sends the "Activation termination" for the Counter interrogation command.
6	Test if the IEC104 honeypot has registered this event on the “iec104_logs.log”
Input data	N/A

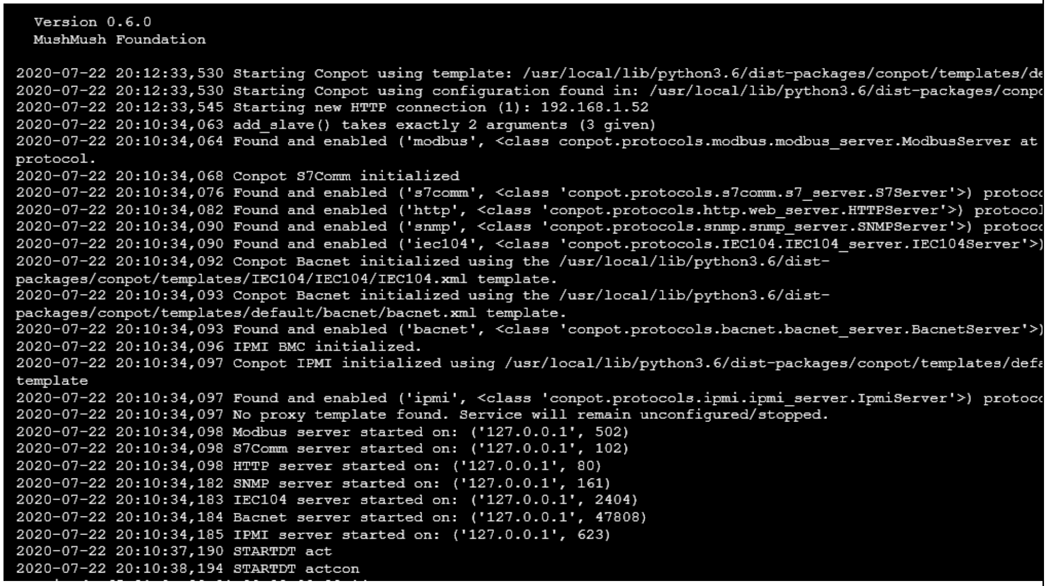
Result	The “Counter Interrogation command” is sent.  `IEC 60870-5-104-Asdu: ASDU=1 C_CI_NA_1 ActCon  IOA=0 'counter interrogation command'`
	`    TypeId: C_CI_NA_1 (101)`
	`    0... .... = SQ: False`
	`    .000 0001 = NumIx: 1`
	`    ..00 0101 = CauseTx: ActCon (5)`
	`    .0.. .... = Negative: False`
	`    0... .... = Test: False`
	`    OA: 0`
	`    Addr: 1`
	`    IOA: 0`
	
	It was received by the RTU
	 `2020-07-22 20:19:22,305 Received: 65 01 0a 00 01 00 00 00 00 14 'CounterInterrogationCommand'`

Test Case Result	Achieved

Test Case ID	HP_60870_104_02	**Component**	IEC 60870-5-104 honeypot
Description	The Master sends a Read Command (C_RD_NA_1) to RTU (IEC104 honeypot) using I-format to perform the information transfer. The “Read command" is called with the an IOA address in order to have each value of a register. It responds by the value from the corresponding IEC104 register/bit.		
SPEC ID	SPEC-F1, SPEC-F4	**Priority**	High
Prepared by	SID	**Tested by**	SID

Pre-condition(s)	• There is a network connection with IEC 60870-5-104 honeypot • IEC 60870-5-104 honeypot must be up and listening on port 2404 • Install and execute the Rsyslog daemon.
Test steps	
1	IEC4 client (Slave) attempts to connect with the Master.
2	Test if the Master sends an "Activation" command, with IOA=0. The element includes information for the “Read command”.
3	Test if the RTU receives the "Read command" from the Master
4	Test if more than one value is requested, then the meter responds with a sequence of information elements ASDU with the SQ bit set to 1.
5	Test if a single value is request, then the meter responds to a read request with a sequence of information objects ASDU with the SQ bit in the variable structure qualifier set to 0.
6	Test if the RTU sends the "Activation termination" for the “Read Command”.
7	Test if the IEC104 honeypot has registered this event on the “iec104_logs.log”
Input data	N/A
Result	The RTU receives the command to Read.  `2020-07-22 20:52:12,525 Received: 66 01 0a 00 01 00 00 00 00 14 'Read from Register'`
Test Case Result	Achieved

Test Case ID	HP_60870_104_03	**Component**	IEC 60870-5-104 honeypot
Description	The IEC 60870-5-104 port is open.		
SPEC ID	FR-UR-02, FR-UR-03	**Priority**	High
Prepared by	SID	**Tested by**	SID
Pre-condition(s)	• There is a network connection with IEC 60870-5-104 honeypot • IEC 60870-5-104 honeypot must be up and listening on port 2404 • Install and execute the Rsyslog daemon.		
Test steps			
1	Test if the port 2404 is open.		

Input data	N/A
Result	We can observe that the IEC104 server is running and the port 2404 is opened. There is no error. 
Test Case Result	Achieved

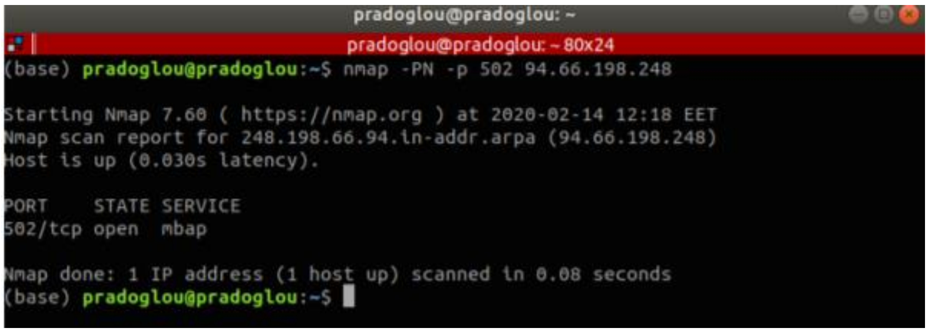
```
   Version 0.6.0
   MushMush Foundation

 2020-07-22 20:12:33,530 Starting Conpot using template: /usr/local/lib/python3.6/dist-packages/conpot/templates/de
 2020-07-22 20:12:33,530 Starting Conpot using configuration found in: /usr/local/lib/python3.6/dist-packages/conpo
 2020-07-22 20:12:33,545 Starting new HTTP connection (1): 192.168.1.52
 2020-07-22 20:10:34,063 add_slave() takes exactly 2 arguments (3 given)
 2020-07-22 20:10:34,064 Found and enabled ('modbus', <class conpot.protocols.modbus.modbus_server.ModbusServer at
 protocol.
 2020-07-22 20:10:34,068 Conpot S7Comm initialized
 2020-07-22 20:10:34,076 Found and enabled ('s7comm', <class 'conpot.protocols.s7comm.s7_server.S7Server'>) protoco
 2020-07-22 20:10:34,082 Found and enabled ('http', <class 'conpot.protocols.http.web_server.HTTPServer'>) protocol
 2020-07-22 20:10:34,090 Found and enabled ('snmp', <class 'conpot.protocols.snmp.snmp_server.SNMPServer'>) protoco
 2020-07-22 20:10:34,090 Found and enabled ('iec104', <class 'conpot.protocols.IEC104.IEC104_server.IEC104Server'>)
 2020-07-22 20:10:34,092 Conpot Bacnet initialized using the /usr/local/lib/python3.6/dist-
 packages/conpot/templates/IEC104/IEC104/IEC104.xml template.
 2020-07-22 20:10:34,093 Conpot Bacnet initialized using the /usr/local/lib/python3.6/dist-
 packages/conpot/templates/default/bacnet/bacnet.xml template.
 2020-07-22 20:10:34,093 Found and enabled ('bacnet', <class 'conpot.protocols.bacnet.bacnet_server.BacnetServer'>)
 2020-07-22 20:10:34,096 IPMI BMC initialized.
 2020-07-22 20:10:34,097 Conpot IPMI initialized using /usr/local/lib/python3.6/dist-packages/conpot/templates/defa
 template
 2020-07-22 20:10:34,097 Found and enabled ('ipmi', <class 'conpot.protocols.ipmi.ipmi_server.IpmiServer'>) protoco
 2020-07-22 20:10:34,097 No proxy template found. Service will remain unconfigured/stopped.
 2020-07-22 20:10:34,098 Modbus server started on: ('127.0.0.1', 502)
 2020-07-22 20:10:34,098 S7Comm server started on: ('127.0.0.1', 102)
 2020-07-22 20:10:34,098 HTTP server started on: ('127.0.0.1', 80)
 2020-07-22 20:10:34,182 SNMP server started on: ('127.0.0.1', 161)
 2020-07-22 20:10:34,183 IEC104 server started on: ('127.0.0.1', 2404)
 2020-07-22 20:10:34,184 Bacnet server started on: ('127.0.0.1', 47808)
 2020-07-22 20:10:34,185 IPMI server started on: ('127.0.0.1', 623)
 2020-07-22 20:10:37,190 STARTDT act
 2020-07-22 20:10:38,194 STARTDT actcon
```

6.3 Modbus Honeypot

Test Case ID	HP_Modbus_01	**Component**	Modbus honeypot
Description	This unit test aims to check and verify that the Modbus/TCP service is used normally by the VM where the Modbus honeypot operates as a server (e.g., RTU). To this end, the nmap tool was used in order to scan and identify which network services are executed by the VM in which the Modbus honeypot operates.		
Spec ID	SPEC-F4	**Priority**	High
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypots is executed, utilising the following commands.		

```
1.    cd Documents
```

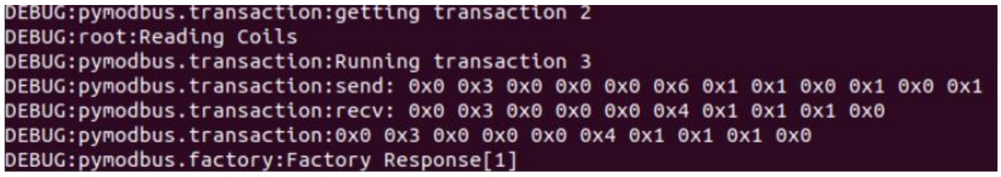
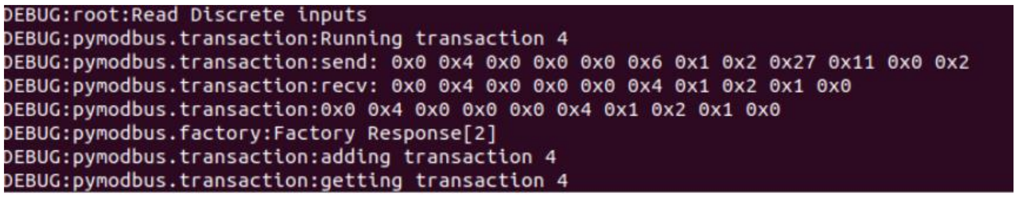
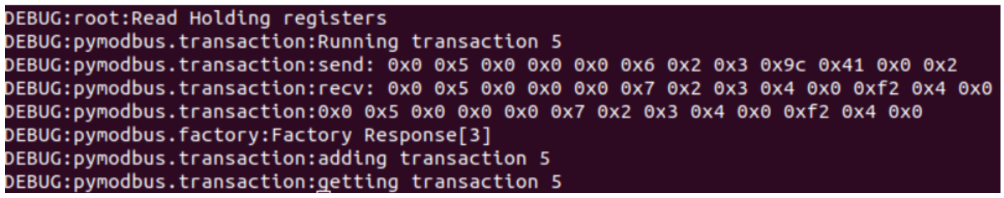
	2. ./run-modbus_honeypot.sh
3	Nmap is used in order to check and verify that the Modbus service successfully operates in the VM, which hosts the Modbus honeypot. The following command is used. nmap -PN -p 502 99.66.198.248
Input data	-

Result	The following image shows that the Modbus/TCP is used by the Modbus honeypot.
	
	pradoglou@pradoglou: ~
	pradoglou@pradoglou: ~ 80x24
	`(base) pradoglou@pradoglou:~$ nmap -PN -p 502 94.66.198.248`
	
	`Starting Nmap 7.60 ( https://nmap.org ) at 2020-02-14 12:18 EET`
	`Nmap scan report for 248.198.66.94.in-addr.arpa (94.66.198.248)`
	`Host is up (0.030s latency).`
	
	`PORT    STATE SERVICE`
	`502/tcp open  mbap`
	
	`Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds`
	`(base) pradoglou@pradoglou:~$`
	Figure 49. Nmap execution which validates that the Modbus/TCP service is used by the Modbus honeypot which operates as a server.

Test Case Result	Achieved

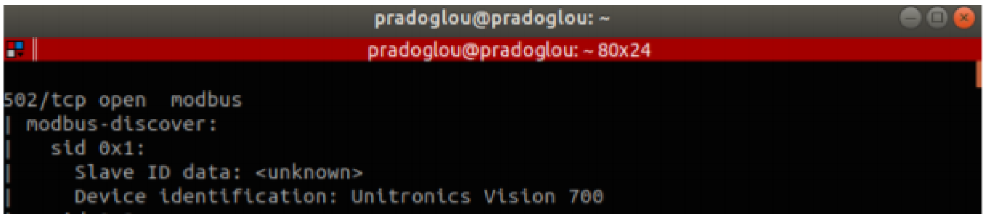
Test Case ID	HP_Modbus_02	**Component**	Modbus honeypot
Description	This unit test aims to check the capability of the Modbus honeypot to operate as a client (MTU/HMI). To this end, Modbus/TCP request packets are transmitted, including a) function code 01 (Read Coils), b) function code 02 (Read Discrete Input) and c) function code 03 (Read Holding Register).		
Spec ID	SPEC-F4	**Priority**	High
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypot is executed in the client mode (e.g., MTU/HMI), using the following command:		

	python3 modbus_hmi.py
3	The Modbus honeypot generates the following Modbus/TCP request packets. 1. Modbus/TCP request packet with the function code 01 (Read Coils) 2. Modbus/TCP request packet with the function code 02 (Read Discrete Inputs) 3. Modbus/TCP request packet with the function code 03 (Read Holding Registers)
Input data	-

Result	The following image shows the generation and transmission of the Modbus/TCP request packet with the function code 01 (Read Coils).
	
	`DEBUG:pymodbus.transaction:getting transaction 2`
	`DEBUG:root:Reading Coils`
	`DEBUG:pymodbus.transaction:Running transaction 3`
	`DEBUG:pymodbus.transaction:send: 0x0 0x3 0x0 0x0 0x0 0x6 0x1 0x1 0x0 0x1 0x0 0x1`
	`DEBUG:pymodbus.transaction:recv: 0x0 0x3 0x0 0x0 0x0 0x4 0x1 0x1 0x1 0x0`
	`DEBUG:pymodbus.transaction:0x0 0x3 0x0 0x0 0x0 0x4 0x1 0x1 0x1 0x0`
	`DEBUG:pymodbus.factory:Factory Response[1]`
	Figure 50. Generation and Transmission of the Modbus/TCP request packet with the function code 01 (Read Coils).
	The following image depicts the generation and transmission of the Modbus/TCP request packet with the function code 02 (Read Discrete Inputs)
	
	`DEBUG:root:Read Discrete inputs`
	`DEBUG:pymodbus.transaction:Running transaction 4`
	`DEBUG:pymodbus.transaction:send: 0x0 0x4 0x0 0x0 0x0 0x6 0x1 0x2 0x27 0x11 0x0 0x2`
	`DEBUG:pymodbus.transaction:recv: 0x0 0x4 0x0 0x0 0x0 0x4 0x1 0x2 0x1 0x0`
	`DEBUG:pymodbus.transaction:0x0 0x4 0x0 0x0 0x0 0x4 0x1 0x2 0x1 0x0`
	`DEBUG:pymodbus.factory:Factory Response[2]`
	`DEBUG:pymodbus.transaction:adding transaction 4`
	`DEBUG:pymodbus.transaction:getting transaction 4`
	Figure 51. Generation and Transmission of the Modbus/TCP request packet with the function code 02 (Read Discrete Inputs).
	The following image shows the generation and transmission of the Modbus/TCP request packet with the function code 03 (Read Holding Registers)
	
	`DEBUG:root:Read Holding registers`
	`DEBUG:pymodbus.transaction:Running transaction 5`
	`DEBUG:pymodbus.transaction:send: 0x0 0x5 0x0 0x0 0x0 0x6 0x2 0x3 0x9c 0x41 0x0 0x2`
	`DEBUG:pymodbus.transaction:recv: 0x0 0x5 0x0 0x0 0x0 0x7 0x2 0x3 0x4 0x0 0xf2 0x4 0x0`
	`DEBUG:pymodbus.transaction:0x0 0x5 0x0 0x0 0x0 0x7 0x2 0x3 0x4 0x0 0xf2 0x4 0x0`
	`DEBUG:pymodbus.factory:Factory Response[3]`
	`DEBUG:pymodbus.transaction:adding transaction 5`
	`DEBUG:pymodbus.transaction:getting transaction 5`
	Figure 52. Generation and Transmission of the Modbus/TCP request packet with the function code 03 (Read Holding Register).

Test Case Result	Achieved

Test Case ID	HP_Modbus_03	**Component**	Modbus honeypot
Description	This unit test aims to check and verify that the Modbus honeypot (operating as a client) can be seen as a real PLC, which utilises the Modbus protocol. To this end,		

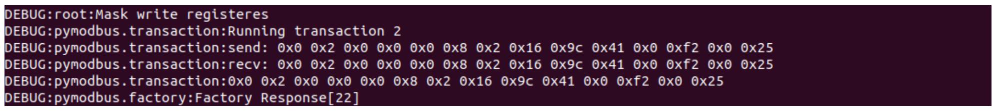
	the Modbus honeypot was configured to emulate the characteristics of real Unitronics Vision 700 PLC. The Nmap Scripting Engine (NSE) was used to execute some diagnostic tests against the Modbus honeypot.		
Spec ID	SPEC-F4	**Priority**	High
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypots is executed, utilising the following commands. 1. cd Documents 2. ./run-modbus_honeypot.sh		
3	The following Nmap NSE command is used. nmap -Pn –script modbus-discover.nse –script-args=‘modbus-discover.aggresive=true‘ -p 502 94.66.198.248		
Input data	-		
Result	The following image shows that the Modbus honeypot was recognised as a Unitronics Vision 700 PLC.  pradoglou@pradoglou: ~ pradoglou@pradoglou: ~ 80x24 502/tcp open modbus | modbus-discover: | sid 0x1: | Slave ID data: <unknown> | Device identification: Unitronics Vision 700 **Figure 53. Recognition of the Modbus honeypot (operating as a client) as a Unitronics Vision 700 PLC.**		
Test Case Result	Achieved		

Test Case ID	HP_Modbus_04	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus function code 22 (Mask Write Register) is successfully supported by the Modbus honeypot in the client-mode (e.g., MTU/HMI). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium
Prepared by	UOWM	**Tested by**	UOWM

Pre-condition(s)	-
Test steps	
1	The Modbus honeypot was deployed successfully in a VM.
2	The Modbus honeypot is executed in the client-mode (e.g., MTU/HMI), using the following command: `python3 modbus_client.py`
3	A Modbus/TCP request packet with the function code 22 is generated.
Input data	-

Result

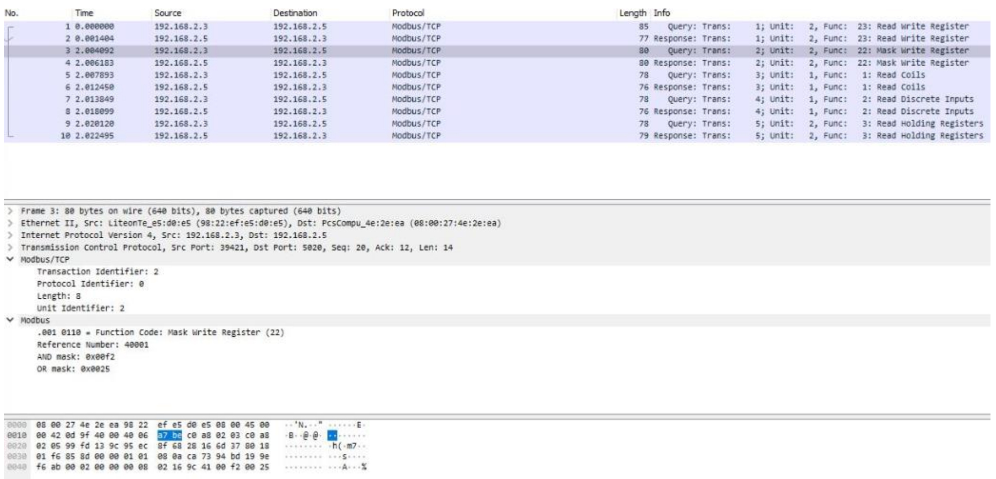
The following image depicts the generation and transmission of the Modbus/TCP request packet with the function code 22.



```
DEBUG:root:Mask write registeres
DEBUG:pymodbus.transaction:Running transaction 2
DEBUG:pymodbus.transaction:send: 0x0 0x2 0x0 0x0 0x0 0x8 0x2 0x16 0x9c 0x41 0x0 0xf2 0x0 0x25
DEBUG:pymodbus.transaction:recv: 0x0 0x2 0x0 0x0 0x0 0x8 0x2 0x16 0x9c 0x41 0x0 0xf2 0x0 0x25
DEBUG:pymodbus.transaction:0x0 0x2 0x0 0x0 0x0 0x8 0x2 0x16 0x9c 0x41 0x0 0xf2 0x0 0x25
DEBUG:pymodbus.factory:Factory Response[22]
```

Figure 54. Generation and transmission of the Modbus/TCP request packet with the function code 22 (Mask Write Register).

Subsequently, the following image illustrates the request Modbus/TCP packet with the function code 22 (Mask Write Register), using Wireshark.



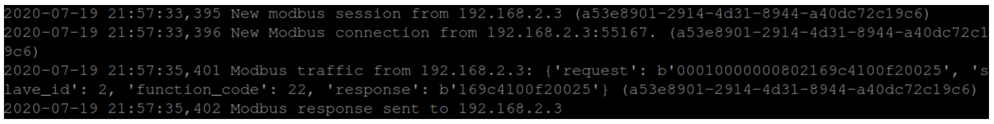
```
No.  Time      Source       Destination  Protocol    Length Info
   1 0.000000  192.168.2.3  192.168.2.5  Modbus/TCP  85    Query: Trans:   1; Unit:  2, Func: 23: Read Write Register
   2 0.001404  192.168.2.5  192.168.2.3  Modbus/TCP  77 Response: Trans:   1; Unit:  2, Func: 23: Read Write Register
   3 2.004092  192.168.2.3  192.168.2.5  Modbus/TCP  80    Query: Trans:   2; Unit:  2, Func: 22: Mask Write Register
   4 2.006183  192.168.2.5  192.168.2.3  Modbus/TCP  80 Response: Trans:   2; Unit:  2, Func: 22: Mask Write Register
   5 2.007893  192.168.2.3  192.168.2.5  Modbus/TCP  78    Query: Trans:   3; Unit:  1, Func:  1: Read Coils
   6 2.012450  192.168.2.5  192.168.2.3  Modbus/TCP  76 Response: Trans:   3; Unit:  1, Func:  1: Read Coils
   7 2.013849  192.168.2.3  192.168.2.5  Modbus/TCP  78    Query: Trans:   4; Unit:  1, Func:  2: Read Discrete Inputs
   8 2.018099  192.168.2.5  192.168.2.3  Modbus/TCP  76 Response: Trans:   4; Unit:  1, Func:  2: Read Discrete Inputs
   9 2.020120  192.168.2.3  192.168.2.5  Modbus/TCP  78    Query: Trans:   5; Unit:  2, Func:  3: Read Holding Registers
  10 2.022495  192.168.2.5  192.168.2.3  Modbus/TCP  79 Response: Trans:   5; Unit:  2, Func:  3: Read Holding Registers

> Frame 3: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
> Ethernet II, Src: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5), Dst: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea)
> Internet Protocol Version 4, Src: 192.168.2.3, Dst: 192.168.2.5
> Transmission Control Protocol, Src Port: 39421, Dst Port: 5020, Seq: 20, Ack: 12, Len: 14
v Modbus/TCP
    Transaction Identifier: 2
    Protocol Identifier: 0
    Length: 8
    Unit Identifier: 2
v Modbus
    .001 0110 = Function Code: Mask Write Register (22)
    Reference Number: 40001
    AND mask: 0x00f2
    OR mask: 0x0025

0000  08 00 27 4e 2e ea 98 22  ef e5 d0 e5 08 00 45 00   ··'N.··" ······E·
0010  00 42 0d 9f 40 00 40 06  a7 be c0 a8 02 03 c0 a8   ·B··@·@· ········
0020  02 05 99 fd 13 9c 95 ec  8f 68 28 16 6d 37 80 18   ········ ·h(·m7··
0030  01 f6 85 8d 00 00 01 01  08 0a ca 73 94 bd 19 9e   ········ ···s····
0040  f6 ab 00 02 00 00 00 08  02 16 9c 41 00 f2 00 25   ········ ···A···%
```

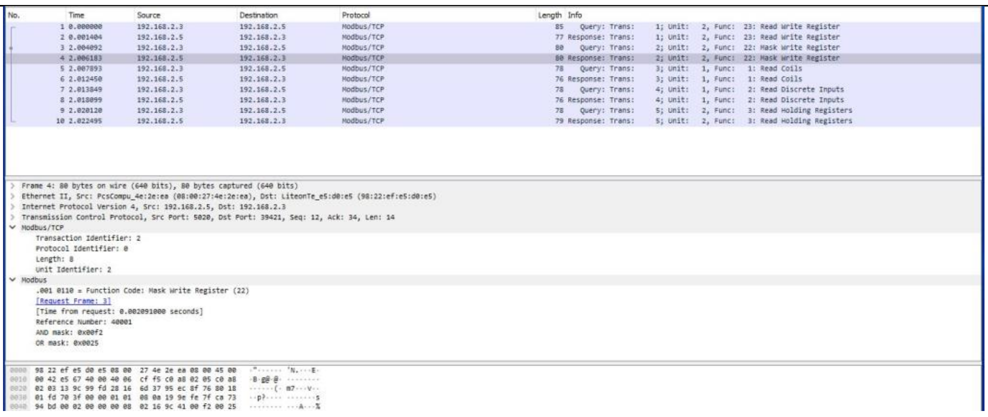
Figure 55. Modbus/TCP function code 22 (Mask Write Register) request packet in Wireshark.

Test Case Result	Achieved

Test Case ID	HP_Modbus_05	**Component**	Modbus honeypot

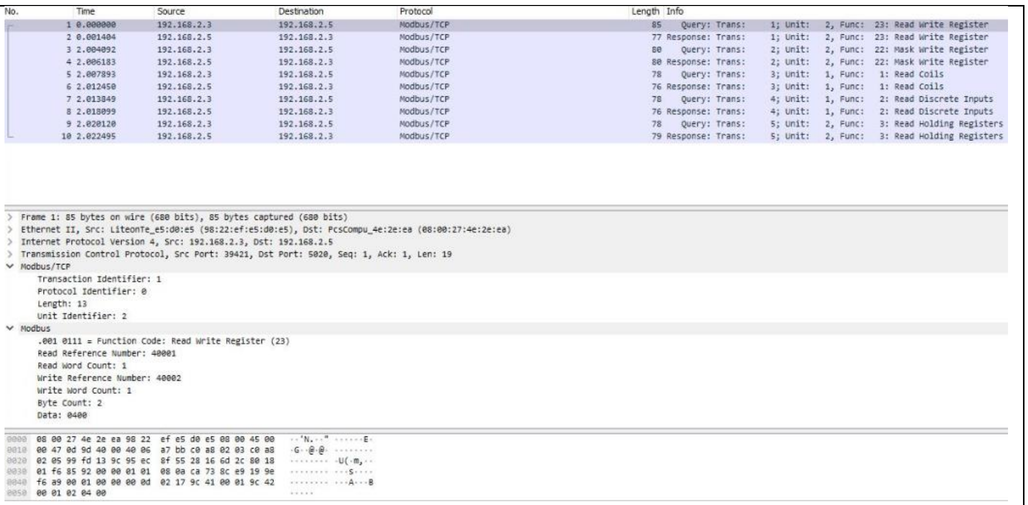
Description	This unit test aims to validate that the Modbus function code 22 (Mask Write Register) is successfully supported by the Modbus honeypot as a server (e.g., RTU). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypots is executed in the server-mode (e.g., RTU), using the following commands: 1. cd Documents 2. ./run-modbus_honeypot.sh		
3	A Modbus/TCP request packet with the function code 22 (Mask Write Register) is sent to the Modbus honeypot.		
4	The Modbus honeypot responds appropriately to the Modbus/TCP request packet with the function code 22 (Mask Write Register) with an appropriate Modbus/TCP response packet.		
Input data	Modbus/TCP request packet with the function code 22 (Mask Write Register).		

Result	The following image depicts the response of the Modbus honeypot.
	
	`2020-07-19 21:57:33,395 New modbus session from 192.168.2.3 (a53e8901-2914-4d31-8944-a40dc72c19c6)`
	`2020-07-19 21:57:33,396 New Modbus connection from 192.168.2.3:55167. (a53e8901-2914-4d31-8944-a40dc72c19c6)`
	`2020-07-19 21:57:35,401 Modbus traffic from 192.168.2.3: {'request': b'000100000008021 69c4100f20025', 'slave_id': 2, 'function_code': 22, 'response': b'169c4100f20025'} (a53e8901-2914-4d31-8944-a40dc72c19c6)`
	`2020-07-19 21:57:35,402 Modbus response sent to 192.168.2.3`
	Figure 56. Response of the Modbus honeypot, which operates as a server in a Modbus/TCP function 22 (Mask Write Register) request packet.
	Subsequently, the following image illustrates the respective response Modbus/TCP packet, using Wireshark.		

	 Figure 57. Modbus/TCP function code 22 (Mask Write Register) response packet generated by the Modbus honeypot.
Test Case Result	Achieved

```
No.  Time        Source        Destination   Protocol    Length Info
   1 0.000000    192.168.2.3   192.168.2.5   Modbus/TCP      85    Query: Trans:    1; Unit:   2, Func:  23: Read Write Register
   2 0.001404    192.168.2.5   192.168.2.3   Modbus/TCP      77 Response: Trans:    1; Unit:   2, Func:  23: Read Write Register
   3 2.004092    192.168.2.3   192.168.2.5   Modbus/TCP      80    Query: Trans:    2; Unit:   2, Func:  22: Mask Write Register
   4 2.006183    192.168.2.5   192.168.2.3   Modbus/TCP      80 Response: Trans:    2; Unit:   2, Func:  22: Mask Write Register
   5 2.007893    192.168.2.3   192.168.2.5   Modbus/TCP      78    Query: Trans:    3; Unit:   1, Func:   1: Read Coils
   6 2.012450    192.168.2.5   192.168.2.3   Modbus/TCP      76 Response: Trans:    3; Unit:   1, Func:   1: Read Coils
   7 2.013849    192.168.2.3   192.168.2.5   Modbus/TCP      78    Query: Trans:    4; Unit:   1, Func:   2: Read Discrete Inputs
   8 2.018099    192.168.2.5   192.168.2.3   Modbus/TCP      76 Response: Trans:    4; Unit:   1, Func:   2: Read Discrete Inputs
   9 2.020120    192.168.2.3   192.168.2.5   Modbus/TCP      78    Query: Trans:    5; Unit:   2, Func:   3: Read Holding Registers
  10 2.022495    192.168.2.5   192.168.2.3   Modbus/TCP      79 Response: Trans:    5; Unit:   2, Func:   3: Read Holding Registers

> Frame 4: 80 bytes on wire (640 bits), 80 bytes captured (640 bits)
> Ethernet II, Src: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea), Dst: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5)
> Internet Protocol Version 4, Src: 192.168.2.5, Dst: 192.168.2.3
> Transmission Control Protocol, Src Port: 5020, Dst Port: 39421, Seq: 12, Ack: 34, Len: 14
v Modbus/TCP
    Transaction Identifier: 2
    Protocol Identifier: 0
    Length: 8
    Unit Identifier: 2
v Modbus
    .001 0110 = Function Code: Mask Write Register (22)
    [Request Frame: 3]
    [Time from request: 0.002091000 seconds]
    Reference Number: 40001
    AND mask: 0x00f2
    OR mask: 0x0025

0000  98 22 ef e5 d0 e5 08 00  27 4e 2e ea 08 00 45 00   ·"······ 'N.···E·
0010  00 42 e5 67 40 00 40 06  cf f5 c0 a8 02 05 c0 a8   ·B·g@·@· ········
0020  02 03 13 9c 99 fd 28 16  6d 37 95 ec 8f 76 80 18   ······(· m7···v··
0030  01 fd 70 3f 00 00 01 01  08 0a 19 9e fe 7f ca 73   ··p?···· ·······s
0040  94 bd 00 02 00 00 00 08  02 16 9c 41 00 f2 00 25   ········ ···A···%
```

Test Case ID	HP_Modbus_06	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus function code 23 (Read/Write Multiple Registers) is successfully supported by the Modbus honeypot in the client-mode (e.g., MTU/HMI). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypot is executed in the client-mode (e.g., MTU/HMI), using the following command: `python3 modbus_client.py`		
3	A Modbus/TCP request packet with the function code 23 (Read/Write Multiple Registers) is generated.		
Input data	-		
Result	The following image illustrates the request Modbus/TCP packet with the function code 23 (Read/Write Multiple Registers), using Wireshark.		



```
No.   Time        Source        Destination   Protocol      Length Info
  1 0.000000    192.168.2.3   192.168.2.5   Modbus/TCP       85    Query: Trans:   1; Unit:  2, Func: 23: Read Write Register
  2 0.001404    192.168.2.5   192.168.2.3   Modbus/TCP       77 Response: Trans:   1; Unit:  2, Func: 23: Read Write Register
  3 2.004092    192.168.2.3   192.168.2.5   Modbus/TCP       80    Query: Trans:   2; Unit:  2, Func: 22: Mask Write Register
  4 2.006183    192.168.2.5   192.168.2.3   Modbus/TCP       80 Response: Trans:   2; Unit:  2, Func: 22: Mask Write Register
  5 2.007893    192.168.2.3   192.168.2.5   Modbus/TCP       78    Query: Trans:   3; Unit:  1, Func:  1: Read Coils
  6 2.012450    192.168.2.5   192.168.2.3   Modbus/TCP       76 Response: Trans:   3; Unit:  1, Func:  1: Read Coils
  7 2.013849    192.168.2.3   192.168.2.5   Modbus/TCP       78    Query: Trans:   4; Unit:  1, Func:  2: Read Discrete Inputs
  8 2.018099    192.168.2.5   192.168.2.3   Modbus/TCP       76 Response: Trans:   4; Unit:  1, Func:  2: Read Discrete Inputs
  9 2.020120    192.168.2.3   192.168.2.5   Modbus/TCP       78    Query: Trans:   5; Unit:  2, Func:  3: Read Holding Registers
 10 2.022495    192.168.2.5   192.168.2.3   Modbus/TCP       79 Response: Trans:   5; Unit:  2, Func:  3: Read Holding Registers

> Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits)
> Ethernet II, Src: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5), Dst: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea)
> Internet Protocol Version 4, Src: 192.168.2.3, Dst: 192.168.2.5
> Transmission Control Protocol, Src Port: 39421, Dst Port: 5020, Seq: 1, Ack: 1, Len: 19
v Modbus/TCP
    Transaction Identifier: 1
    Protocol Identifier: 0
    Length: 13
    Unit Identifier: 2
v Modbus
    .001 0111 = Function Code: Read Write Register (23)
    Read Reference Number: 40001
    Read Word Count: 1
    Write Reference Number: 40002
    Write Word Count: 1
    Byte Count: 2
    Data: 0400

0000  08 00 27 4e 2e ea 98 22  ef e5 d0 e5 08 00 45 00   ··'N.··" ······E·
0010  00 47 0d 9d 40 00 40 06  a7 bb c0 a8 02 03 c0 a8   ·G··@·@· ········
0020  02 05 99 fd 13 9c 95 ec  8f 55 28 16 6d 2c 80 18   ········ ·U(·m,··
0030  01 f6 85 92 00 00 01 01  08 0a ca 73 8c e9 19 9e   ········ ···s····
0040  f6 a9 00 01 00 00 00 0d  02 17 9c 41 00 01 9c 42   ········ ···A···B
0050  00 01 02 04 00                                     ·····
```

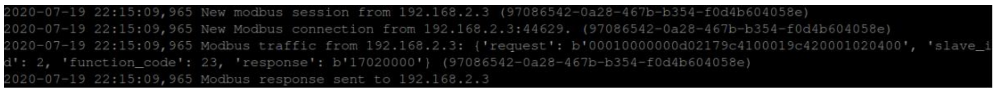
Figure 58. Modbus/TCP function code 23 (Read/Write Multiple Registers) request packet in Wireshark.

Test Case Result	Achieved

Test Case ID	HP_Modbus_07	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus function code 23 (Read/Write Multiple Registers) is successfully supported by the Modbus honeypot as a server (e.g., RTU). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			

1	The Modbus honeypot was deployed successfully in a VM.
2	The Modbus honeypots is executed in the server-mode (e.g., RTU), using the following commands: 1. cd Documents 2. ./run-modbus_honeypot.sh
3	A Modbus/TCP request packet with the function code 23 (Read/Write Multiple Registers) is sent to the Modbus honeypot.
4	The Modbus honeypot responds appropriately to the Modbus/TCP request packet with the function code 23 (Read/Write Multiple Registers) with an appropriate Modbus/TCP response packet.

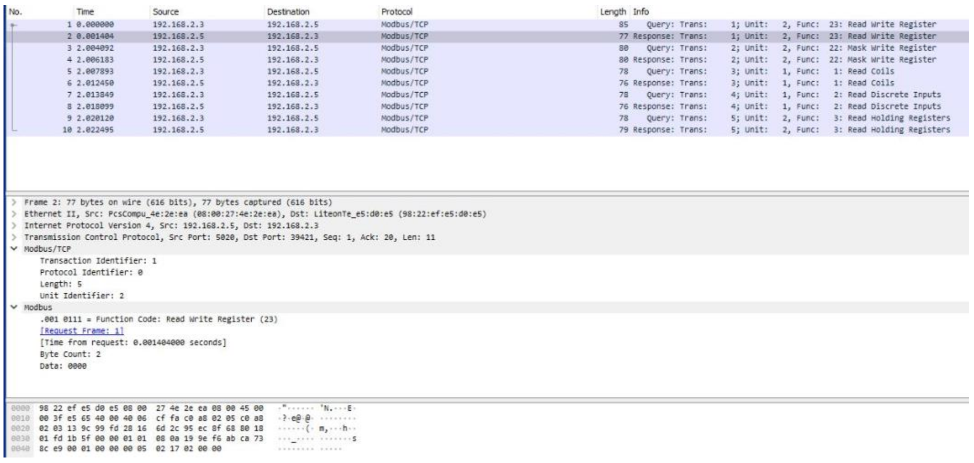
Input data	Modbus/TCP request packet with the function code 23 (Read/Write Multiple Registers)
Result	The following image depicts the response of the Modbus honeypot.



```
2020-07-19 22:15:09,965 New modbus session from 192.168.2.3 (97086542-0a28-467b-b354-f0d4b604058e)
2020-07-19 22:15:09,965 New Modbus connection from 192.168.2.3:44629. (97086542-0a28-467b-b354-f0d4b604058e)
2020-07-19 22:15:09,965 Modbus traffic from 192.168.2.3: {'request': b'00010000000d02179c4100019c420001020400', 'slave_i
d': 2, 'function_code': 23, 'response': b'17020000'} (97086542-0a28-467b-b354-f0d4b604058e)
2020-07-19 22:15:09,965 Modbus response sent to 192.168.2.3
```

Figure 59. Response of the Modbus honeypot, which operates as a server in a Modbus/TCP function 23 (Read/Write Multiple Registers) request packet.

Subsequently, the following image illustrates the respective response Modbus/TCP packet, using Wireshark.



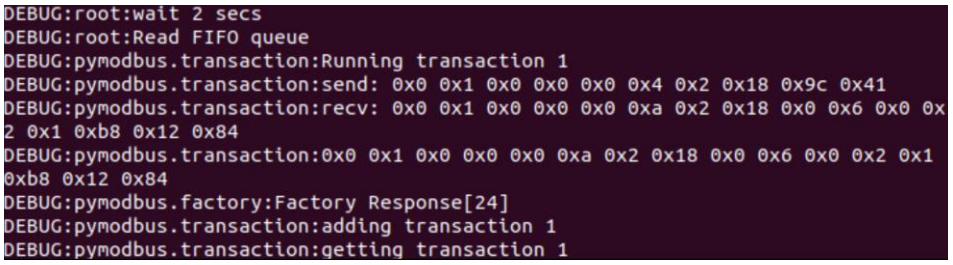
```
No.   Time       Source        Destination   Protocol     Length Info
    1 0.000000   192.168.2.3   192.168.2.5   Modbus/TCP   85     Query: Trans:   1; Unit:   2, Func:  23: Read Write Register
    2 0.001404   192.168.2.5   192.168.2.3   Modbus/TCP   77  Response: Trans:   1; Unit:   2, Func:  23: Read Write Register
    3 2.004092   192.168.2.3   192.168.2.5   Modbus/TCP   80     Query: Trans:   2; Unit:   2, Func:  22: Mask Write Register
    4 2.006183   192.168.2.5   192.168.2.3   Modbus/TCP   80  Response: Trans:   2; Unit:   2, Func:  22: Mask Write Register
    5 2.007893   192.168.2.3   192.168.2.5   Modbus/TCP   78     Query: Trans:   3; Unit:   1, Func:   1: Read Coils
    6 2.012450   192.168.2.5   192.168.2.3   Modbus/TCP   76  Response: Trans:   3; Unit:   1, Func:   1: Read Coils
    7 2.013849   192.168.2.3   192.168.2.5   Modbus/TCP   78     Query: Trans:   4; Unit:   1, Func:   2: Read Discrete Inputs
    8 2.018099   192.168.2.5   192.168.2.3   Modbus/TCP   76  Response: Trans:   4; Unit:   1, Func:   2: Read Discrete Inputs
    9 2.020120   192.168.2.3   192.168.2.5   Modbus/TCP   78     Query: Trans:   5; Unit:   2, Func:   3: Read Holding Registers
   10 2.022495   192.168.2.5   192.168.2.3   Modbus/TCP   79  Response: Trans:   5; Unit:   2, Func:   3: Read Holding Registers

> Frame 2: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)
> Ethernet II, Src: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea), Dst: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5)
> Internet Protocol Version 4, Src: 192.168.2.5, Dst: 192.168.2.3
> Transmission Control Protocol, Src Port: 5020, Dst Port: 39421, Seq: 1, Ack: 20, Len: 11
v Modbus/TCP
    Transaction Identifier: 1
    Protocol Identifier: 0
    Length: 5
    Unit Identifier: 2
v Modbus
    .001 0111 = Function Code: Read Write Register (23)
    [Request Frame: 1]
    [Time from request: 0.001404000 seconds]
    Byte Count: 2
    Data: 0000

0000  98 22 ef e5 d0 e5 08 00  27 4e 2e ea 08 00 45 00   ·"······ 'N.···E·
0010  00 3f e5 65 40 00 40 06  cf fa c0 a8 02 05 c0 a8   ·?·e@·@· ········
0020  02 03 13 9c 99 fd 28 16  6d 2c 95 ec 8f 68 80 18   ······(· m,···h··
0030  01 fd 1b 5f 00 00 01 01  08 0a 19 9e f6 ab ca 73   ···_···· ·······s
0040  8c e9 00 01 00 00 00 05  02 17 02 00 00            ········ ·····
```

Figure 60. Modbus/TCP function code 23 (Read/Write Multiple Registers) response packet generated by the Modbus honeypot.

Test Case Result	Achieved

Test Case ID	HP_Modbus_08	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus function code 24 (Read FIFO Queue) is successfully supported by the Modbus honeypot in the client-mode (e.g., MTU/HMI). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		

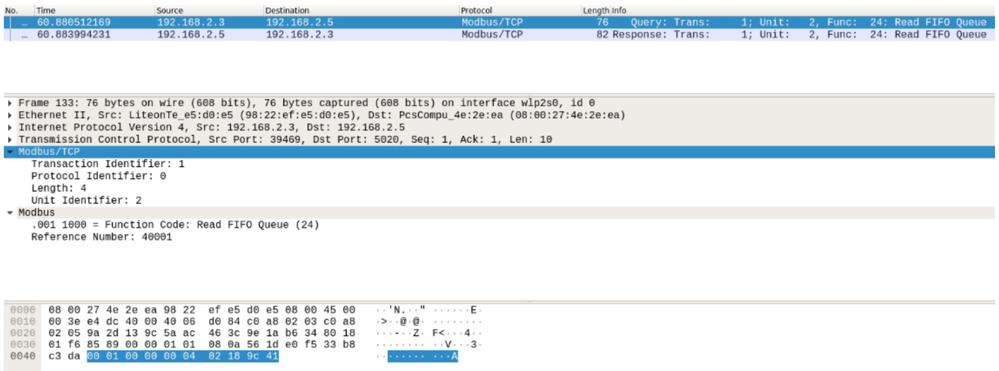
2	The Modbus honeypot is executed in the client-mode (e.g., MTU/HMI), using the following command: `python3 modbus_client.py`
3	A Modbus/TCP request packet with the function code 24 (Read FIFO Queue) is generated.
Input data	-
Result	The following image illustrates the request Modbus/TCP packet with the function code 24 (Read FIFO Queue), which is generated by the Modbus honeypot.



```
DEBUG:root:wait 2 secs
DEBUG:root:Read FIFO queue
DEBUG:pymodbus.transaction:Running transaction 1
DEBUG:pymodbus.transaction:send: 0x0 0x1 0x0 0x0 0x0 0x4 0x2 0x18 0x9c 0x41
DEBUG:pymodbus.transaction:recv: 0x0 0x1 0x0 0x0 0x0 0xa 0x2 0x18 0x0 0x6 0x0 0x
2 0x1 0xb8 0x12 0x84
DEBUG:pymodbus.transaction:0x0 0x1 0x0 0x0 0x0 0xa 0x2 0x18 0x0 0x6 0x0 0x2 0x1
0xb8 0x12 0x84
DEBUG:pymodbus.factory:Factory Response[24]
DEBUG:pymodbus.transaction:adding transaction 1
DEBUG:pymodbus.transaction:getting transaction 1
```

Figure 61. Modbus/TCP function code 24 request packet generated by the Modbus honeypot.

The following image illustrates the request Modbus/TCP packet with the function code 24 (Read FIFO Queue), using Wireshark.



```
No.  Time          Source        Destination   Protocol    Length Info
...  60.880512169  192.168.2.3   192.168.2.5   Modbus/TCP    76    Query: Trans:    1; Unit:   2, Func:  24: Read FIFO Queue
...  60.883994231  192.168.2.5   192.168.2.3   Modbus/TCP    82 Response: Trans:    1; Unit:   2, Func:  24: Read FIFO Queue

▸ Frame 133: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface wlp2s0, id 0
▸ Ethernet II, Src: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5), Dst: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea)
▸ Internet Protocol Version 4, Src: 192.168.2.3, Dst: 192.168.2.5
▸ Transmission Control Protocol, Src Port: 39469, Dst Port: 5020, Seq: 1, Ack: 1, Len: 10
▾ Modbus/TCP
    Transaction Identifier: 1
    Protocol Identifier: 0
    Length: 4
    Unit Identifier: 2
▾ Modbus
    .001 1000 = Function Code: Read FIFO Queue (24)
    Reference Number: 40001

0000  08 00 27 4e 2e ea 98 22  ef e5 d0 e5 08 00 45 00   ··'N.··" ······E·
0010  00 3e e4 dc 40 00 40 06  d0 84 c0 a8 02 03 c0 a8   ·>··@·@· ········
0020  02 05 9a 2d 13 9c 5a ac  46 3c 9e 1a b6 34 80 18   ···-··Z· F<···4··
0030  01 f6 85 89 00 00 01 01  08 0a 56 1d e0 f5 33 b8   ········ ··V···3·
0040  c3 da 00 01 00 00 00 04  02 18 9c 41               ········ ···A
```

Figure 62. Modbus/TCP function code 24 (Read FIFO Queue) request packet generated by the Modbus honeypot in Wireshark.

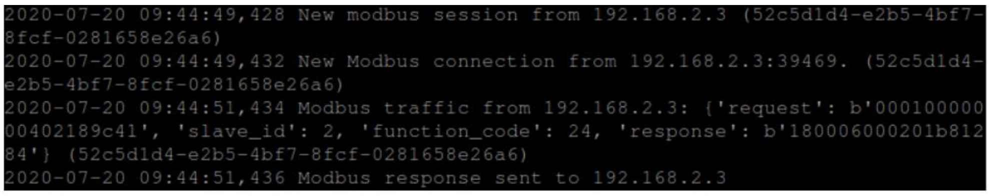
Test Case Result	Achieved

Test Case ID	HP_Modbus_09	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus function code 24 (Read FIFO Queue) is successfully supported by the Modbus honeypot as a server (e.g., RTU). The specific function code is not supported by the existing honeypots.		
Spec ID	SPEC-F4	**Priority**	Medium

Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypot is executed in the server-mode (e.g., RTU), using the following commands: 1. cd Documents 2. ./run-modbus_honeypot.sh		
3	A Modbus/TCP request packet with the function code 24 (Read FIFO Queue) is sent to the Modbus honeypot.		
4	The Modbus honeypot responds appropriately to the Modbus/TCP request packet with the function code 24 (Read FIFO Queue) with an appropriate Modbus/TCP response packet.		
Input data	Modbus/TCP request packet with the function code 24 (Read FIFO Queue)		

Result

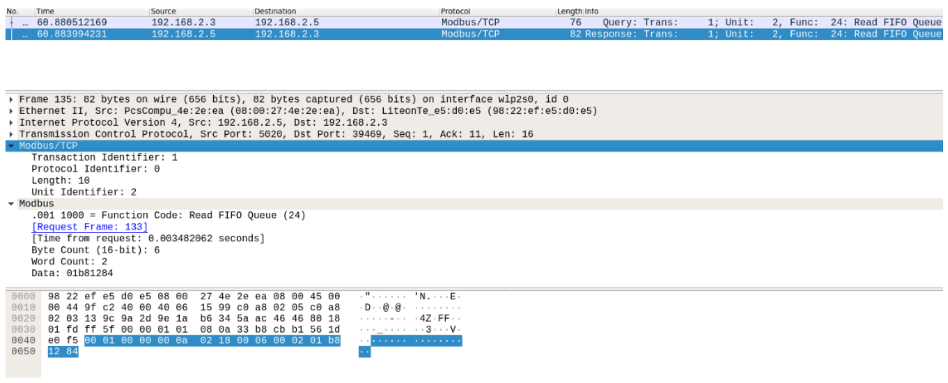
The following image depicts the response of the Modbus honeypot.



```
2020-07-20 09:44:49,428 New modbus session from 192.168.2.3 (52c5d1d4-e2b5-4bf7-
8fcf-0281658e26a6)
2020-07-20 09:44:49,432 New Modbus connection from 192.168.2.3:39469. (52c5d1d4-
e2b5-4bf7-8fcf-0281658e26a6)
2020-07-20 09:44:51,434 Modbus traffic from 192.168.2.3: {'request': b'000100000
00402189c41', 'slave_id': 2, 'function_code': 24, 'response': b'180006000201b812
84'} (52c5d1d4-e2b5-4bf7-8fcf-0281658e26a6)
2020-07-20 09:44:51,436 Modbus response sent to 192.168.2.3
```

Figure 63. Response of the Modbus honeypot, which operates as a server in a Modbus/TCP function 24 (Read FIFO Queue) request packet.

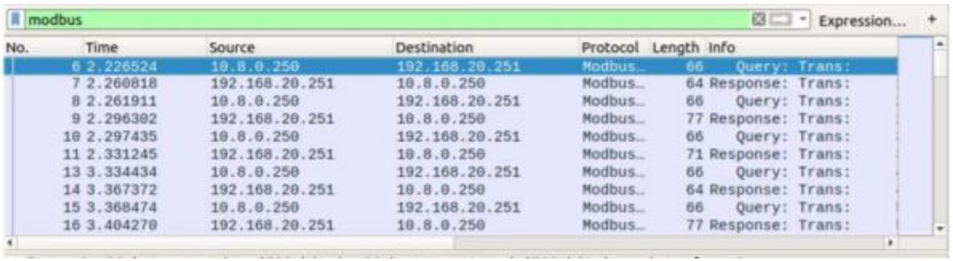
Subsequently, the following image illustrates the respective response Modbus/TCP packet, using Wireshark.



```
No.  Time          Source       Destination  Protocol    Length Info
… 60.880512169  192.168.2.3  192.168.2.5  Modbus/TCP     76    Query: Trans:     1; Unit:   2, Func:  24: Read FIFO Queue
… 60.883994231  192.168.2.5  192.168.2.3  Modbus/TCP     82 Response: Trans:     1; Unit:   2, Func:  24: Read FIFO Queue

▸ Frame 135: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface wlp2s0, id 0
▸ Ethernet II, Src: PcsCompu_4e:2e:ea (08:00:27:4e:2e:ea), Dst: LiteonTe_e5:d0:e5 (98:22:ef:e5:d0:e5)
▸ Internet Protocol Version 4, Src: 192.168.2.5, Dst: 192.168.2.3
▸ Transmission Control Protocol, Src Port: 5020, Dst Port: 39469, Seq: 1, Ack: 11, Len: 16
▾ Modbus/TCP
    Transaction Identifier: 1
    Protocol Identifier: 0
    Length: 10
    Unit Identifier: 2
▾ Modbus
    .001 1000 = Function Code: Read FIFO Queue (24)
    [Request Frame: 133]
    [Time from request: 0.003482062 seconds]
    Byte Count (16-bit): 6
    Word Count: 2
    Data: 01b81284

0000  98 22 ef e5 d0 e5 08 00  27 4e 2e ea 08 00 45 00   ·"······ 'N.···E·
0010  00 44 9f c2 40 00 40 06  15 99 c0 a8 02 05 c0 a8   ·D··@·@· ········
0020  02 03 13 9c 9a 2d 9e 1a  b6 34 5a ac 46 46 80 18   ·····-·· ·4Z·FF··
0030  01 fd ff 5f 00 00 01 01  08 0a 33 b8 cb b1 56 1d   ···_···· ··3···V·
0040  e0 f5 00 01 00 00 00 0a  02 18 00 06 00 02 01 b8   ········ ········
0050  12 84                                              ··
```

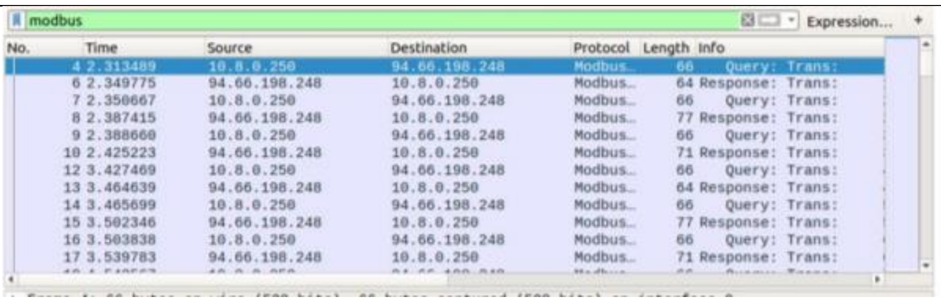
Figure 64. Modbus/TCP function code 24 (Read FIFO Queue) response packet generated by the Modbus honeypot.

Test Case Result	Achieved

Test Case ID	HP_Modbus_10	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus/TCP packets generated by the Modbus honeypot (which operates as a server) are similar to those of the real device which is emulated. To this end, the network traffic generated by the Modbus honeypot was compared with the network traffic of a real Unitronics Vision 700 PLC.		
Spec ID	SPEC-F4	**Priority**	High
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypot is executed in the server-mode (e.g., RTU), using the following commands: 1. cd Documents 2. ./run-modbus_honeypot.sh		
3	The same Modbus/TCP request packets are transmitted to the Modbus honeypot and the real Unitronics Vision 700 PLC.		
Input data	Appropriate Modbus/TCP request packets were transmitted to the Modbus honeypot and the real Unitronics Vision 700 PLC.		
Result	The following images show that the Modbus network traffic generated by the Modbus honeypot is similar to that produced by the real Unitronics Vision 700 PLC. 		

```
modbus                                                              Expression...  +
No.   Time       Source           Destination      Protocol  Length Info
    6 2.226524   10.8.0.250       192.168.20.251   Modbus…   66     Query: Trans:
    7 2.260818   192.168.20.251   10.8.0.250       Modbus…   64 Response: Trans:
    8 2.261911   10.8.0.250       192.168.20.251   Modbus…   66     Query: Trans:
    9 2.296302   192.168.20.251   10.8.0.250       Modbus…   77 Response: Trans:
   10 2.297435   10.8.0.250       192.168.20.251   Modbus…   66     Query: Trans:
   11 2.331245   192.168.20.251   10.8.0.250       Modbus…   71 Response: Trans:
   13 3.334434   10.8.0.250       192.168.20.251   Modbus…   66     Query: Trans:
   14 3.367372   192.168.20.251   10.8.0.250       Modbus…   64 Response: Trans:
   15 3.368474   10.8.0.250       192.168.20.251   Modbus…   66     Query: Trans:
   16 3.404270   192.168.20.251   10.8.0.250       Modbus…   77 Response: Trans:
```

Figure 65. The Modbus/TCP network traffic generated by the Modbus honeypot

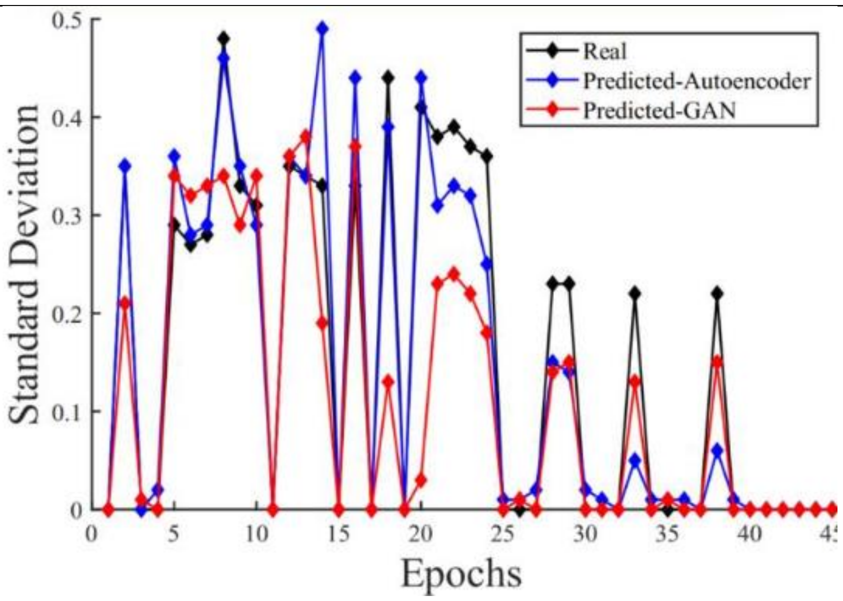
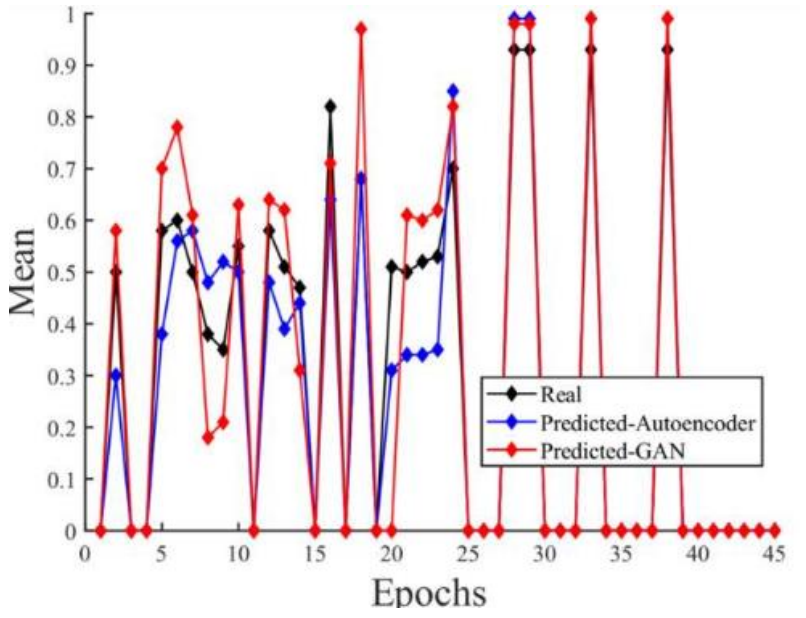


```
modbus                                                   Expression...  +
No.   Time      Source          Destination     Protocol  Length Info
    4 2.313489  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
    6 2.349775  94.66.198.248   10.8.0.250      Modbus…   64 Response: Trans:
    7 2.350667  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
    8 2.387415  94.66.198.248   10.8.0.250      Modbus…   77 Response: Trans:
    9 2.388660  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
   10 2.425223  94.66.198.248   10.8.0.250      Modbus…   71 Response: Trans:
   12 3.427469  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
   13 3.464639  94.66.198.248   10.8.0.250      Modbus…   64 Response: Trans:
   14 3.465699  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
   15 3.502346  94.66.198.248   10.8.0.250      Modbus…   77 Response: Trans:
   16 3.503838  10.8.0.250      94.66.198.248   Modbus…   66        Query: Trans:
   17 3.539783  94.66.198.248   10.8.0.250      Modbus…   71 Response: Trans:
```

Figure 66. The Modbus/TCP network traffic generated by the real Unitronics Vision 700 PLC

Test Case Result	Achieved

Test Case ID	HP_Modbus_11	**Component**	Modbus honeypot
Description	This unit test aims to validate that the payload of the Modbus/TCP packets generated by the Modbus honeypot (which operates as a server) is similar to the payload of the Modbus/TCP packets produced by a real industrial device. To this end, a real Unitronics Vision 700 PLC was used.		
Spec ID	SPEC-F4	**Priority**	High
Prepared by	UOWM	**Tested by**	UOWM
Pre-condition(s)	-		
Test steps			
1	The Modbus honeypot was deployed successfully in a VM.		
2	The Modbus honeypot is executed in the server-mode (e.g., RTU), using the following commands: 1. cd Documents 2. ./run-modbus_honeypot.sh		
3	The same Modbus/TCP request packets are transmitted to the Modbus honeypot and the real Unitronics Vision 700 PLC.		
Input data	Appropriate Modbus/TCP request packets were transmitted to the Modbus honeypot and the real Unitronics Vision 700 PLC.		
Result	The following figures show that the payload of the Modbus/TCP packets generated by the Modbus honeypot is similar to the payload of the Modbus/TCP packets produced by the real Unitronics Vision 700 PLC.		

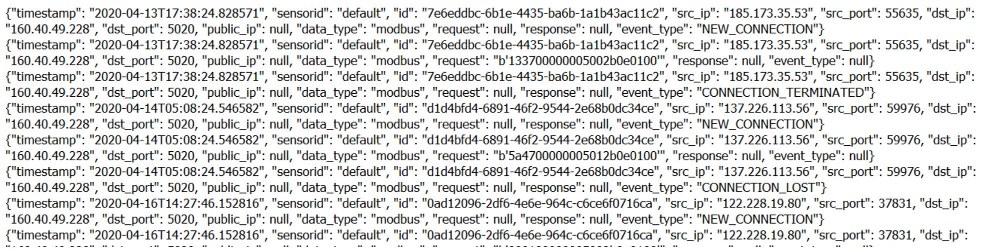
	 **Figure 67. Standard deviation of the data against the number of epochs.**  **Figure 68. Arithmetic mean of the data against the number of epochs.**
Test Case Result	Achieved

Test Case ID	HP_Modbus_12	**Component**	Modbus honeypot
Description	This unit test aims to validate that the Modbus honeypot (operating as a server) can generate normally logs if an entity (e.g., potential cyberattacker) interracts with it.		
Spec ID	SPEC-F4	**Priority**	High

Prepared by	UOWM		**Tested by**	UOWM
Pre-condition(s)	-			
Test steps				
1	The Modbus honeypot was deployed successfully in a VM.			

2	The Modbus honeypot is executed in the server-mode (e.g., RTU), using the following commands:
	`1. cd Documents`
	`2. ./run-modbus_honeypot.sh`			

3	The following Nmap NSE command is executed against the Modbus honeypot. `nmap -Pn –script modbus-discover.nse –script-args=‘modbus-discover.aggresive=true‘ -p 502 94.66.198.248`			
Input data	-			

Result	The following image shows the logs generated by the Modbus honeypot.
	
	`{"timestamp": "2020-04-13T17:38:24.828571", "sensorid": "default", "id": "7e6eddbc-6b1e-4435-ba6b-1a1b43ac11c2", "src_ip": "185.173.35.53", "src_port": 55635, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": null, "response": null, "event_type": "NEW_CONNECTION"}`
	`{"timestamp": "2020-04-13T17:38:24.828571", "sensorid": "default", "id": "7e6eddbc-6b1e-4435-ba6b-1a1b43ac11c2", "src_ip": "185.173.35.53", "src_port": 55635, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": "b'133700000005002b0e0100'", "response": null, "event_type": null}`
	`{"timestamp": "2020-04-13T17:38:24.828571", "sensorid": "default", "id": "7e6eddbc-6b1e-4435-ba6b-1a1b43ac11c2", "src_ip": "185.173.35.53", "src_port": 55635, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": null, "response": null, "event_type": "CONNECTION_TERMINATED"}`
	`{"timestamp": "2020-04-14T05:08:24.546582", "sensorid": "default", "id": "d1d4bfd4-6891-46f2-9544-2e68b0dc34ce", "src_ip": "137.226.113.56", "src_port": 59976, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": null, "response": null, "event_type": "NEW_CONNECTION"}`
	`{"timestamp": "2020-04-14T05:08:24.546582", "sensorid": "default", "id": "d1d4bfd4-6891-46f2-9544-2e68b0dc34ce", "src_ip": "137.226.113.56", "src_port": 59976, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": "b'5a4700000005012b0e0100'", "response": null, "event_type": null}`
	`{"timestamp": "2020-04-14T05:08:24.546582", "sensorid": "default", "id": "d1d4bfd4-6891-46f2-9544-2e68b0dc34ce", "src_ip": "137.226.113.56", "src_port": 59976, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": null, "response": null, "event_type": "CONNECTION_LOST"}`
	`{"timestamp": "2020-04-16T14:27:46.152816", "sensorid": "default", "id": "0ad12096-2df6-4e6e-964c-c6ce6f0716ca", "src_ip": "122.228.19.80", "src_port": 37831, "dst_ip": "160.40.49.228", "dst_port": 5020, "public_ip": null, "data_type": "modbus", "request": null, "response": null, "event_type": "NEW_CONNECTION"}`
	`{"timestamp": "2020-04-16T14:27:46.152816", "sensorid": "default", "id": "0ad12096-2df6-4e6e-964c-c6ce6f0716ca", "src_ip": "122.228.19.80", "src_port": 37831, "dst_ip":`
	Figure 69. Modbus honeypot (server) logs.			

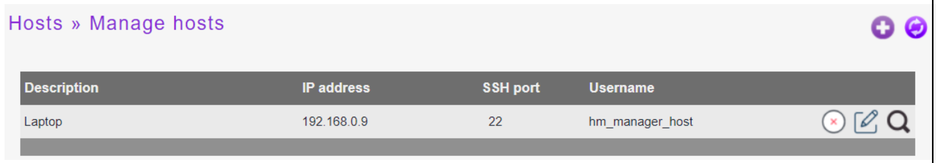
Test Case Result	Achieved			

6.4 Honeypot Manager

Test Case ID	HM_01	**Component**	Honeypot Manager
Description	Deployment with the Honeypot Manager the IEC61850 Honeypot		
SPEC ID	SPEC-OP1, SPEC-OP3	**Priority**	Medium
Prepared by	TECNALIA	**Tested by**	TECNALIA

Pre-condition(s)

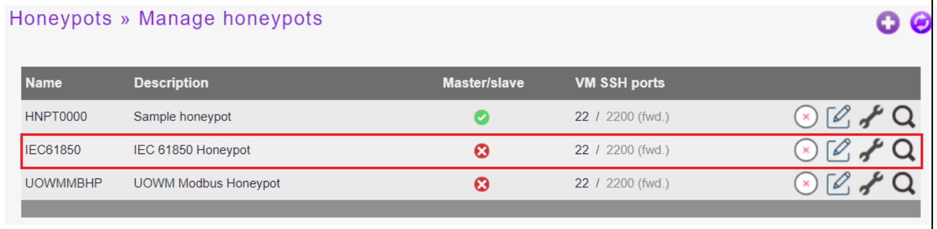
- A target host to contain the VM with the honeypot has to be registered in the application.



Hosts » Manage hosts

Description	IP address	SSH port	Username
Laptop	192.168.0.9	22	hm_manager_host

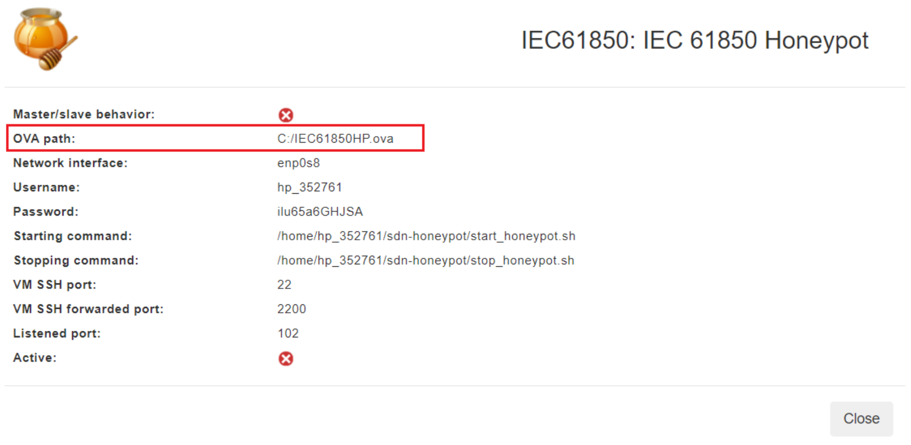
- The honeypot IEC61850 has to be registered in the application.



Honeypots » Manage honeypots

Name	Description	Master/slave	VM SSH ports
HNPT0000	Sample honeypot	✓	22 / 2200 (fwd.)
IEC61850	IEC 61850 Honeypot	✗	22 / 2200 (fwd.)
UOWMMBHP	UOWM Modbus Honeypot	✗	22 / 2200 (fwd.)

- The OVA file corresponding to the honeypot IEC61850 must exist in the path indicated in the honeypot info.



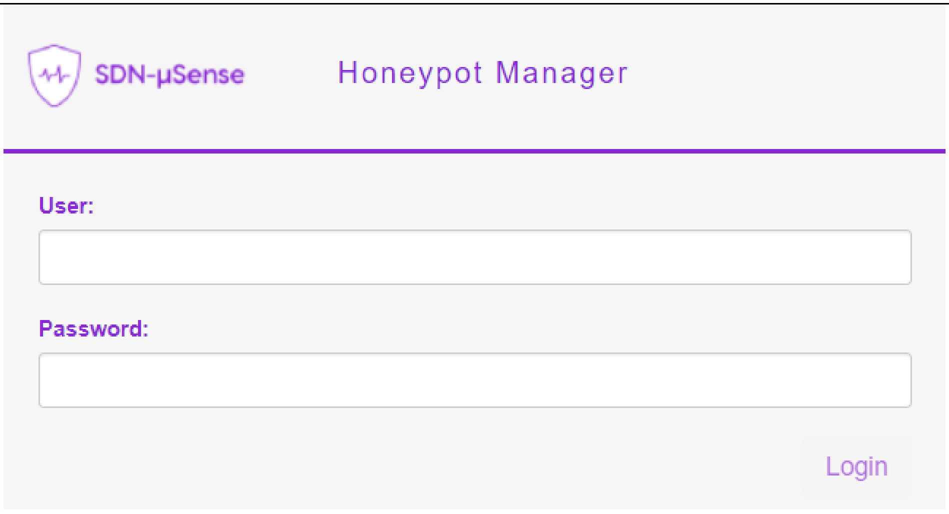
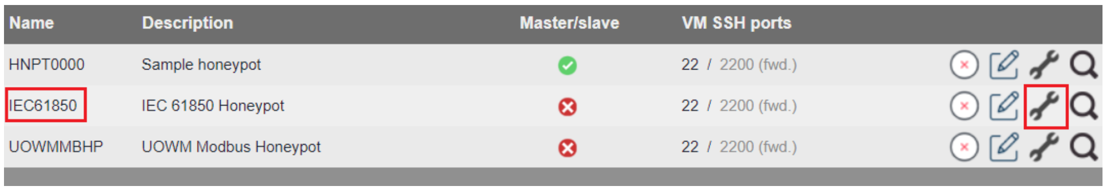
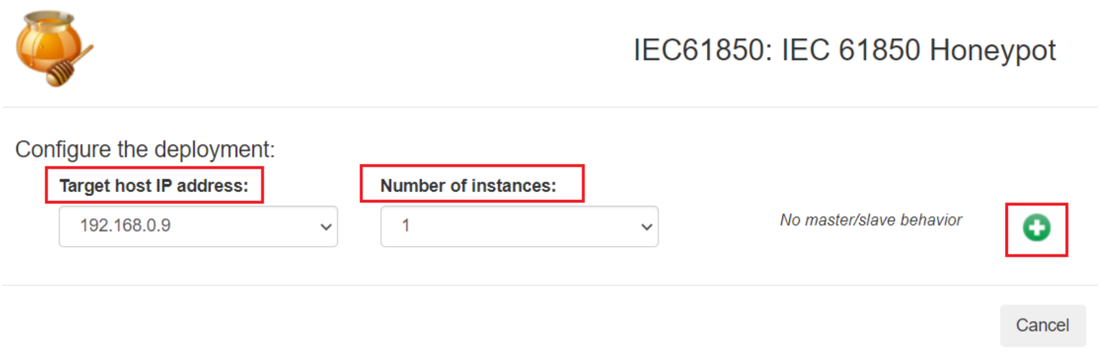
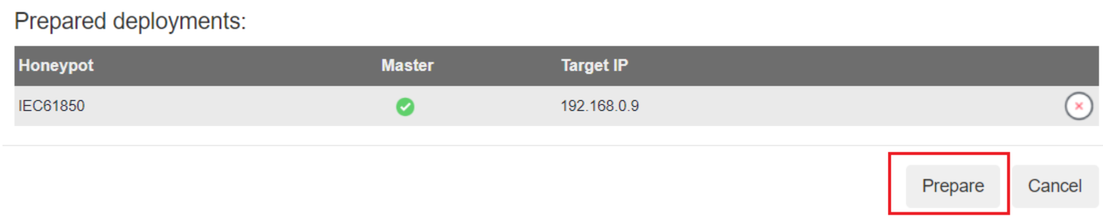
IEC61850: IEC 61850 Honeypot

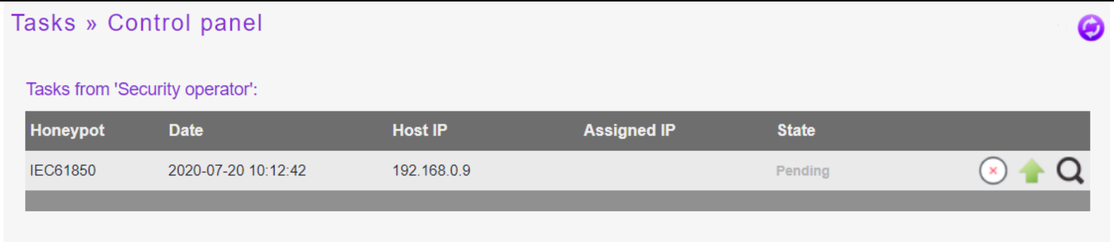
Master/slave behavior:	✗
OVA path:	C:/IEC61850HP.ova
Network interface:	enp0s8
Username:	hp_352761
Password:	ilu65a6GHJSA
Starting command:	/home/hp_352761/sdn-honeypot/start_honeypot.sh
Stopping command:	/home/hp_352761/sdn-honeypot/stop_honeypot.sh
VM SSH port:	22
VM SSH forwarded port:	2200
Listened port:	102
Active:	✗

- The port 22 has to be open in the target host.

Test steps

1	Login in the Honeypot Manager application.

	 SDN-μSense Honeypot Manager User: Password: Login
2	Click on the option “Honeypots → Manage honeypots”, and prepare the deployment clicking on the “tool” icon corresponding to the IEC61850 honeypot.  Name | Description | Master/slave | VM SSH ports HNPT0000 | Sample honeypot | ✓ | 22 / 2200 (fwd.) IEC61850 | IEC 61850 Honeypot | ✗ | 22 / 2200 (fwd.) UOWMMBHP | UOWM Modbus Honeypot | ✗ | 22 / 2200 (fwd.)
3	Select the target host, and the number of instances wanted to be deployed. Then, click on the “plus” icon.  IEC61850: IEC 61850 Honeypot Configure the deployment: Target host IP address: 192.168.0.9 Number of instances: 1 *No master/slave behavior* Cancel The deployment will be prepared. Finally, press the “Prepare” button.  Prepared deployments: Honeypot | Master | Target IP IEC61850 | ✓ | 192.168.0.9 Prepare Cancel
4	In “Tasks → Control panel”, a new task appears as PENDING (under the section “Tasks from ‘Security operator’”).

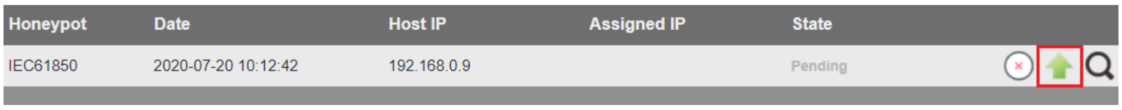


Tasks » Control panel

Tasks from 'Security operator':

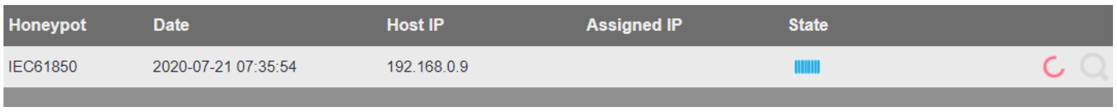
Honeypot	Date	Host IP	Assigned IP	State
IEC61850	2020-07-20 10:12:42	192.168.0.9		Pending

5 Click on the “Deploy” button.



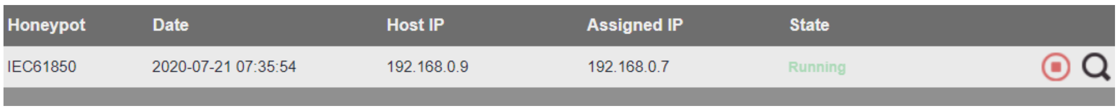
Honeypot	Date	Host IP	Assigned IP	State
IEC61850	2020-07-20 10:12:42	192.168.0.9		Pending

Wait for the honeypot to be deployed.



Honeypot	Date	Host IP	Assigned IP	State
IEC61850	2020-07-21 07:35:54	192.168.0.9		

When deployed, the VM containing the honeypot has its own assigned IP address.



Honeypot	Date	Host IP	Assigned IP	State
IEC61850	2020-07-21 07:35:54	192.168.0.9	192.168.0.7	Running

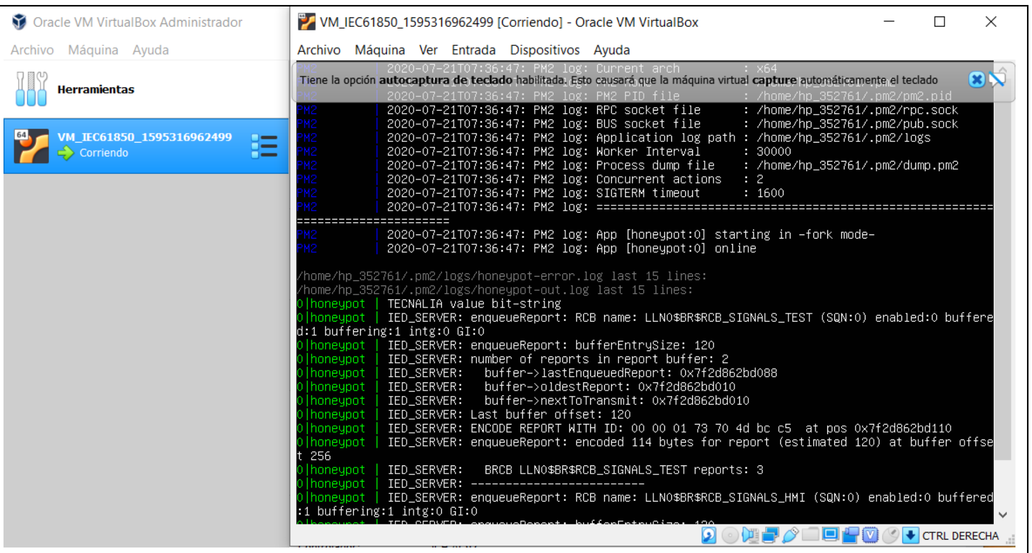
Input data Although the input data are built using the GUI, the REST function in charge of the deployment receives a Task in JSON format (as body):

```
{
    "uuid": "ac6b6d66-30e8-4bc2-95f9-1f33ad520e4b",
    "date": "2020-07-21 08:11:37",
    "name": "TSK_1595319097104",
    "origin": "M",
    "honeypot": "IEC61850",
    "master": true,
    "state": "Running",
    "targetIp": "192.168.0.9",
    "privateIp": null,
    "instanceId": null
}
```

And the previous state of the Task (as path variable):

```
  "previousState": "Pending"
```

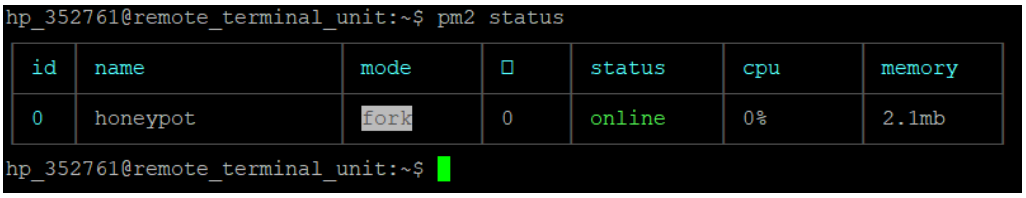
Result The honeypot has to be deployed properly.



```
PM2          | 2020-07-21T07:36:47: PM2 log: Current arch        : x64
Tiene la opción autocaptura de teclado habilitada. Esto causará que la máquina virtual capture automáticamente el teclado
PM2          | 2020-07-21T07:36:47: PM2 log: PM2 PID file        : /home/hp_352761/.pm2/pm2.pid
PM2          | 2020-07-21T07:36:47: PM2 log: RPC socket file     : /home/hp_352761/.pm2/rpc.sock
PM2          | 2020-07-21T07:36:47: PM2 log: BUS socket file     : /home/hp_352761/.pm2/pub.sock
PM2          | 2020-07-21T07:36:47: PM2 log: Application log path : /home/hp_352761/.pm2/logs
PM2          | 2020-07-21T07:36:47: PM2 log: Worker Interval     : 30000
PM2          | 2020-07-21T07:36:47: PM2 log: Process dump file   : /home/hp_352761/.pm2/dump.pm2
PM2          | 2020-07-21T07:36:47: PM2 log: Concurrent actions  : 2
PM2          | 2020-07-21T07:36:47: PM2 log: SIGTERM timeout     : 1600
PM2          | 2020-07-21T07:36:47: PM2 log: ==========================================================
======================
PM2          | 2020-07-21T07:36:47: PM2 log: App [honeypot:0] starting in -fork mode-
PM2          | 2020-07-21T07:36:47: PM2 log: App [honeypot:0] online

/home/hp_352761/.pm2/logs/honeypot-error.log last 15 lines:
/home/hp_352761/.pm2/logs/honeypot-out.log last 15 lines:
0|honeypot | TECNALIA value bit-string
0|honeypot | IED_SERVER: enqueueReport: RCB name: LLN0$BR$RCB_SIGNALS_TEST (SQN:0) enabled:0 buffere
d:1 buffering:1 intg:0 GI:0
0|honeypot | IED_SERVER: enqueueReport: bufferEntrySize: 120
0|honeypot | IED_SERVER: number of reports in report buffer: 2
0|honeypot | IED_SERVER:   buffer->lastEnqueuedReport: 0x7f2d862bd088
0|honeypot | IED_SERVER:   buffer->oldestReport: 0x7f2d862bd010
0|honeypot | IED_SERVER:   buffer->nextToTransmit: 0x7f2d862bd010
0|honeypot | IED_SERVER: Last buffer offset: 120
0|honeypot | IED_SERVER: ENCODE REPORT WITH ID: 00 00 01 73 70 4d bc c5  at pos 0x7f2d862bd110
0|honeypot | IED_SERVER: enqueueReport: encoded 114 bytes for report (estimated 120) at buffer offse
t 256
0|honeypot | IED_SERVER:   BRCB LLN0$BR$RCB_SIGNALS_TEST reports: 3
0|honeypot | IED_SERVER: -------------------------
0|honeypot | IED_SERVER: enqueueReport: RCB name: LLN0$BR$RCB_SIGNALS_HMI (SQN:0) enabled:0 buffered
:1 buffering:1 intg:0 GI:0
```

Oracle VM VirtualBox Administrador
Archivo Máquina Ayuda
Herramientas
VM_IEC61850_1595316962499
Corriendo
VM_IEC61850_1595316962499 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
CTRL DERECHA

Enter the VM and type the command “pm2 status“. The honeypot has to be “online“.



```
hp_352761@remote_terminal_unit:~$ pm2 status
┌─────┬──────────────┬──────────┬────┬──────────┬──────┬──────────┐
│ id  │ name         │ mode     │ ↺  │ status   │ cpu  │ memory   │
├─────┼──────────────┼──────────┼────┼──────────┼──────┼──────────┤
│ 0   │ honeypot     │ fork     │ 0  │ online   │ 0%   │ 2.1mb    │
└─────┴──────────────┴──────────┴────┴──────────┴──────┴──────────┘
hp_352761@remote_terminal_unit:~$
```

Test Case Result	Achieved

7 Conclusions

This deliverable aims at describing the architecture and detailed designed of the honeypots that emulate the industrial protocols and the Honeypot Manager developed as part of the SDN-microSENSE Risk Assessment Framework (S-RAF).

The industrial protocols honeypots are designed to serve as a decoy mechanism to possible attackers, facilitating early detection of threats and attack patterns, and the mitigation of potential risks. The honeypots developed in SDN-microSENSE cover the most relevant industrial protocols used in the EPES. The industrial protocols emulated are: i) IEC 61850, ii) iED60870-5-104 and iii) Modbus.

Complementary to these three different honeypots, SDN-microSENSE project offers the Honeypot Manager that has two functionalities: i) To provide means to develop in an automatic way those honeypots that the security operator or manager decide to deploy in the network and ii) to process the information about unknown anomalies (like zero-day attacks) and to indicate to the SDN controller the required information to re-arrange the honeypots network in order to collect information for unknown anomalies.

The deliverable also summarizes the innovations brought by Industrial protocols honeypots and Honeypot Manager with respect to the state of the art (section 3.2).

The development of the honeypots in SDN-microSENSE has taken into account the state of the art of the honeypots existing (section 2 and 3) together with the specification presented in the deliverable D2.3 of this project [SMS20-D23] (section 4).

The present document gathers the overall functional and architectural overview and the detailed description of the developed components' designs including the interfaces offered to other components in the SDN-microSENSE reference architecture. Furthermore, this document presents the guides for the deployment of these components.

The document also collects the results of the unit tests performed over the honeypots and the Honeypot Manager, but the communication with the other components not developed in this task will be done later during the integration activities (WP7) and will be documented properly in the deliverable D7.5

References

[60870] IEC 60870-5-104:2006+AMD1:2016 CSV - Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles.

[61850] IEC 61850:2020. Communication networks and systems for power utility automation

[61850-6] IEC 61850-6:2009 Communication networks and systems for power utility automation - Part 6: Configuration description language for communication in electrical substations related to IEDs. Available: https://webstore.iec.ch/publication/6013 [Accessed: 25- Jun- 2020].

[AAO13] Ayeni, O. A., Alese, B. K., & Omotosho, L. O. (2013). Design and implementation of a medium interaction honeypot. International Journal of Computer Applications, 975, 8887.

[ABH03] Abhilash Verma: Production Honeypots. An Organizations view“, GIAC October 2003

[ATE20] A European H2020 Project – ATENA ", *A European H2020 Project - ATENA*, 2020. [Online]. Available: https://www.atena-h2020.eu/. [Accessed: 25- Jun- 2020].

[BJM+14] D. Buza, F. Juhász, G. Miru, M. Félegyházi and T. Holczer, "CryPLH: Protecting Smart Energy Systems from Targeted Attacks with a PLC Honeypot", *Lecture Notes in Computer Science*, pp. 181-192, 2014. Available: 10.1007/978-3-319-10329-7_12 [Accessed 25 June 2020].

[CAB+17] Chovancová, E., Adám, N., Baláž, A., Pietriková, E., Feciľak, P., Šimoňák, S., & Chovanec, M. (2017). Securing distributed computer systems using an advanced sophisticated hybrid honeypot technology. Computing and Informatics, 36(1), 113-139.

[CLL+18] J. Cao, W. Li, J. Li and B. Li, "DiPot: A Distributed Industrial Honeypot System", *Lecture Notes in Computer Science*, pp. 300-309, 2018. Available: 10.1007/978-3-319-73830-7_30 [Accessed 25 June 2020].

[COH] Conpot: Honeypot de Sistemas de Control Industrial. Available at https://revista.seguridad.unam.mx/numero29/conpot-honeypot-de-sistemas-de-control-industrial [Accessed: 25- Jun- 2020].

[COL17] G. Collins, "Pymodbus Documentation", 2017.

[CON20] CONPOT ICS/SCADA Honeypot. Available at conpot.org. [Accessed: 25- Jun- 2020].

[CRW04] Clarke, G., Reynders, D., & Wright, E. (2004). Practical modern SCADA protocols: DNP3, 60870.5 and related systems. Newnes

[DAL19] C. Dalamagkas et al., "A Survey On Honeypots, Honeynets And Their Applications On Smart Grid", *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019. Available: 10.1109/netsoft.2019.8806693 [Accessed 19 June 2020].

[DSI+19] C. Dalamagkas, P. Sarigiannidis, D. Ioannidis, E. Iturbe, O. Nikolis, F. Ramos, E. Rios, A. Sarigiannidis and D. Tzovaras, "A Survey On Honeypots, Honeynets And Their Applications On Smart Grid", in 2019 IEEE Conference on Network Softwarization (NetSoft), 2019.

[FDF+18]	W. Fan, Z. Du, D. Fernández and V. A. Villagrá, "Enabling an Anatomic View to Investigate Honeypot Systems: A Survey," in IEEE Systems Journal, vol. 12, no. 4, pp. 3906-3919, Dec. 2018, doi: 10.1109/JSYST.2017.2762161.
[HCS08]	Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies forthe modbus protocols,"International Journal of Critical Infrastructure Protection, vol. 1, pp. 37–44, 12 2008.
[HDEF]	Honeypot definition. Available at: https://searchsecurity.techtarget.com/definition/honey-pot [Accessed 22 Jun. 2020].
[IECSTD]	Core IEC Standards Available: https://www.iec.ch/smartgrid/standards/ . [Accessed: 28- Jun- 2020].
[KG15]	K. Koltys and R. Gajewski, "Shape: A honeypot for electric power substation," Journal of Telecommunications and Information Technology, no. 4, p. 37, 2015.
[KGV+17]	Karthikeyan, R., Geetha, D. T., Vijayalakshmi, S., & Sumitha, R. (2017). Honeypots for Network Security. International journal for Research & Development in Technology, 7(2), 62-66.
[KMS14]	Kaur, T., Malhotra, V., & Singh, D. (2014). Comparison of network security tools-firewall, intrusion detection system and Honeypot. Int. J. Enhanced Res. Sci. Technol. Eng, 200204.
[LIB61850-20]	Open Source Libraries for IEC 61850. Available: https://libiec61850.com/libiec61850 [Accessed: 25- Jun- 2020].
[MAS19]	Ahamed Mashhoor, "Benefits of the honeypots", 2019. [online]. Available at: https://www.egscyber.com/library/benefits-of-honeypots/. [Accessed: 20/07/ 2020].
[MCG+17]	D. Mashima, B. Chen, P. Gunathilaka and E. L. Tjiong, "Towards a grid-wide, high-fidelity electrical substation honeynet," 2017 IEEE International Conference on Smart Grid Communications (SmartGridComm), Dresden, 2017, pp. 89-95, doi: 10.1109/SmartGridComm.2017.8340689.
[MLC19]	D. Mashima, Y. Li and B. Chen, "Who's Scanning Our Smart Grid? Empirical Study on Honeypot Data", *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019. Available: 10.1109/globecom38437.2019.9013835 [Accessed 19 June 2020].
[MODBUS]	"MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b", Modbus-IDA, 2020.
[MRG19]	Matoušek, P., Ryšavý, O., & Grégr, M. (2019, September). Increasing Visibility of IEC 104 Communication in the Smart Grid. In 6th International Symposium for ICS & SCADA Cyber Security Research 2019 6 (pp. 21-30).
[MZG12]	Mansoori, M., Zakaria, O., & Gani, A. (2012). Improving exposure of intrusion deception system through implementation of hybrid honeypot. The International Arab Journal of Information Technology, 9(5), 436-444.
[NW16]	M. Nawrocki, M. Wählisch et al "A survey on honeypot software and data analysis." *arXiv preprint arXiv:1608.06249*. [Accessed 22 June 2020].
[NWS+16]	Nawrocki, M., Wählisch, M., Schmidt, T.C., Keil, C., & Schönfelder, J. (2016). A Survey on Honeypot Software and Data Analysis. ArXiv, abs/1608.06249.
[ONF-15]	Open Networking Foundation. OpenFlow Switch Specification. Version 1.5.1 (Protocol version 0x06). March 26, 2015. ONF TS-025. [online] Available at:

https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf [Accessed 26 Feb. 2020].

[PB19] PESATORI, M., & BARLOCCO, M. (2019). Design and implementation of a high interaction honeypot for a distributed control system.

[PH07] N. Provos and T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, 2007.

[PRO04] Provoset al., “A virtual honeypot framework.” inUSENIXSecurity Symposium, vol. 173, 2004, [Accessed 22 June 2020].

[PSL+20] [UOWM1] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, “A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics,”IEEE Communications Surveys & Tutorials, 2020.

[PYY+19] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan and Y. Zheng, "Recent Progress on Generative Adversarial Networks (GANs): A Survey", IEEE Access, vol. 7, pp. 36322-36333, 2019. Available: 10.1109/access.2019.2905015 [Accessed 29 June 2020].

[RIS13] L. Rist, “Introducing Conpot,” 2013. [Online]. Available: https://www.honeynet.org/node/1047

[RNK+20] N. Rowe, T. Nguyen, M. Kendrick, Z. Rucker, D. Hyun and J. Brown, "Creating Convincing Industrial-Control-System Honeypots", *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020. Available: 10.24251/hicss.2020.228 [Accessed 19 June 2020].

[RRM+18] R. R. S, R. R, M. Moharir, and S. G, “Scapy- a powerful interactivepacket manipulation program,” in 2018 International Conference onNetworking, Embedded and Wireless Systems (ICNEWS), 2018, pp. 1–5.

[RSL+20] P. Radoglou-Grammatikis, I. Siniosoglou, T. Liatifis, A. Korouniadis, K. Rompolos and P. Sarigiannidis, “Implementation and Detection of Modbus Cyberattacks: A Case Study,” 10th International Conference on Modern Circuits and Systems Technologies (MOCAST), 2020, to appear.

[SER18] "SerIoT – Secure and Safe Internet of Things", *Seriot-project.eu*, 2018. [Online]. Available: https://seriot-project.eu/. [Accessed: 25- Jun- 2020].

[SIS16] SISSDEN", *Sissden.eu*, 2020. [Online]. Available: https://sissden.eu/. [Accessed: 25-Jun- 2020]

[SMS20-D22] SDN-microSENSE Deliverable D2.2: . User & Stakeholder, Security and Privacy Requirements. Feb 2020.

[SMS20-D23] SDN-microSENSE Deliverable D2.3 Platform Specifications and Architecture

[SMS20-D24] SDN-microSENSE Deliverable D2.4. Pilot, Demonstration & Evaluation Strategy

[SMS20-D51] SDN-microSENSE Deliverable D5.1XL-SIEM System.

[SOY15] A. Serbanescu, S. Obermeier and D. Yu, "ICS Threat Analysis Using a Large-Scale Honeynet", 2015. Available: 10.14236/ewic/ics2015.3 [Accessed 19 June 2020].

[SPE19]	"SPEAR Project", *Spear2020.eu*, 2019. [Online]. Available: https://www.spear2020.eu/. [Accessed: 19- Jun- 2020].
[SPE20-D43]	SPEAR project Deliverable D4.3 – AMI Honeypots and Game-theory based Honeypot Manager
[SPI-03]	L. Spitzner, "Honeypots: catching the insider threat," in 19th Annual Computer Security Applications Conference, 2003. Proceedings., no. Acsac. IEEE, 2003, pp. 170–179. [Online]. Available: http://ieeexplore.ieee.org/document/1254322/
[SR01]	The Value of Honeypots, Part One: Definitions and Values of Honeypots by Lance Spitzner with extensive help from Marty Roesch last updated October 10, 2001
[TIT15]	"Benefits of Honeypots – There's More to Honeypots Than Wasting Hackers' Time", April 2015, [Online]. Available: https://www.webtitan.com/blog/honeypots-how-far-can-you-go-in-wasting-a-hackers-time/. [Accessed: 01/07/2020].
[TR-IEC104]	"Description and analysis of IEC 104 Protocol" Technical Report no. FIT-TR-2017-12. Faculty of Information TechnologyBrno University of TechnologyBrno, Czech Republic. Available at https://www.fit.vut.cz/research/publication-file/11570/TR-IEC104.pdf
[VKS15]	A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A modbus/tcpfuzzer for testing internetworked industrial systems," in 2015 IEEE 20thConference on Emerging Technologies & Factory Automation (ETFA).IEEE, 2015, pp. 1–6.
[WIC06]	Wicherski, G. (2006). Medium interaction honeypots. German Honeynet Project.
[WW19]	Wang, H., & Wu, B. (2019, March). SDN-based hybrid honeypot for attack capture. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (pp. 1602-1606). IEEE.
[YAK18]	Project YAKSHA – Cybersecurity Awareness and Knowledge Systemic High-level Application", *Project-yaksha.eu*, 2018. [Online]. Available: https://project-yaksha.eu/. [Accessed: 25- Jun- 2020].
[YU19]	C. Yinka-Banjo and O. Ugot, "A review of generative adversarial networks and its application in cybersecurity", Artificial Intelligence Review, vol. 53, no. 3, pp. 1721-1736, 2019. Available: 10.1007/s10462-019-09717-4.