Testbed Reconnaissance/Scouting/Vulnerability Analysis

Francisco Catarino Mendes
Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
uc2019222823@student.uc.pt

Leonardo Oliveira Pereira
Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
uc2020239125@student.uc.pt

***Abstract*—This paper intends to give a reconnaissance, scouting, and vulnerability analysis procedure about a given testbed. The testbed will first be introduced, so one can know what is the technology being dealt with, followed by the task plan of what will be done and the toolset and techniques that are going to be used to achieve the goals. Finally, a discussion of the findings made in the reconnaissance, scouting, and vulnerability analysis processes will take place, particularly regarding the devices and functions found, the process-level information, the protocols, the existing communication flows, and the actual detected vulnerabilities.**

***Index Terms*—Testbed, Vulnerabilities, Scouting, Reconnaissance, Modbus, SCADA**

I. Introduction

This work has the objective of simulating potential attacks in the industrial world, particularly against the device of the testbed scenario targeted, which is a Modicon M340 - Mid-range PLC for industrial processes and infrastructure, from Schneider Electric [1].

Figure 1 shows the topology of the testbed, while Figure 2 shows the actual SCADA system in the HMI:

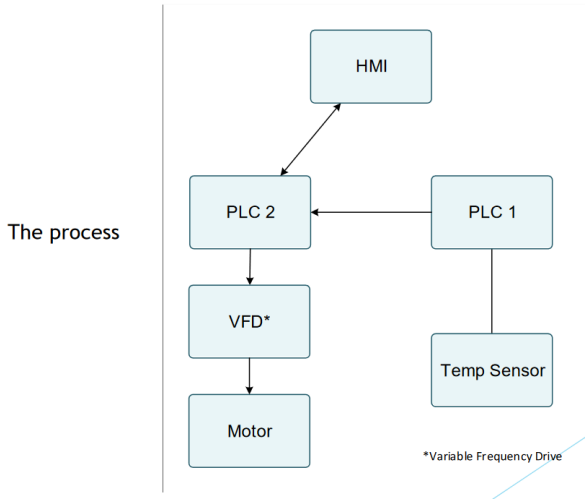

Figure 1: Topology of the testbed

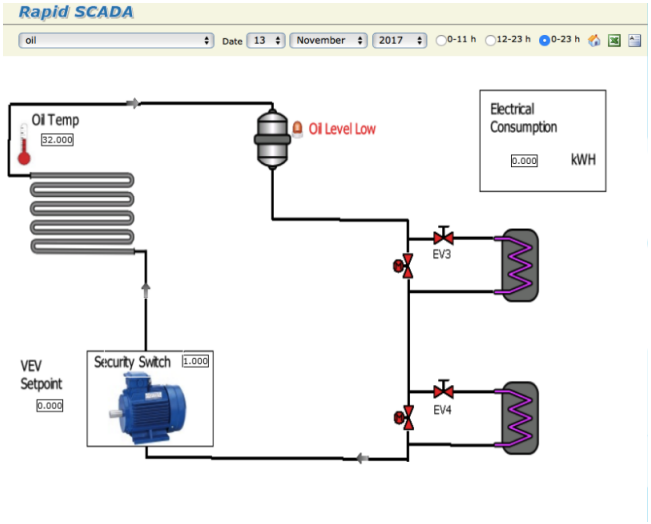

Figure 2: How the SCADA process is shown by the HMI

In short:

1) **HMI** - human-machine interface, used to display the system status, process variables (like temperatures or speeds), and alarms. It can also be used to input commands or setpoints to the PLCs;
2) **PLCs** - one tasked with gathering data from sensors, another responsible for higher-level decision-making or coordinating multiple control processes;
3) **Temp sensor** - measures temperature and informs one of the PLC's;
4) **VFD** - used to control the speed and torque of electric motors, by varying the frequency and voltage supplied to the motor;
5) **Motor** - the part of the system doing physical work.

As for the process flow, the temperature is monitored by the Temp sensor, and the data is sent to PLC 1, which processes this data and communicates with PLC 2 to adjust the control strategy as needed. Then, PLC 2 uses the input from PLC 1 and the HMI to control the VFD, which in turn adjusts the Motor's operation to meet the process requirements.

The task plan section of this assignment explains further what will be the strategy used to achieve the objectives proposed.

II. Definitions

Here, the most relevant terms of this assignment are explained, in order to make available the highest possibility of understanding what is being talked about throughout the paper.

PLC - short for Programmable Logic Controller, it is a type of computer used in industrial and commercial control applications. These devices can automate a specific process, machine function, or even an entire production line. A PLC gets its data from connected sensors or input devices, goes through it, and triggers results based on pre-established parameters. Depending on the inputs and outputs, a PLC can monitor and store information such as machine productivity or operating temperature, automatically start and stop processes, generate alarms if something is malfunctioning, and others [2].

SCADA - short for Supervisory control and data acquisition, it is a system of software and hardware elements (PLCs for example) that control and monitor industrial processes by directly interacting with plant-floor machinery and viewing real-time data. These systems are really important for industrial organizations since they help to maintain efficiency, analyze data so that the best decisions are made regarding change, and communicate system issues to help mitigate downtime issues [3].

IACS - short for Industrial Automation and Control Systems, these are electromechanical and mechanical devices that perform several control, monitoring, and actuation processes on many logic devices and process-type systems. They can also track and control many processes through sensors on machines, smart devices, and software and hardware that turn sensor information into different control outputs. Finally, they improve the efficiency, quality, safety and reliability of industrial operations by automating tasks that would otherwise require manual work [4].

Modbus Protocol - an industrial protocol that was developed in 1979 to make communication possible between automation devices. Originally designed for serial cables, has been meanwhile ported to TCP/IP. It uses a master/slave relationship, where usually the "SCADA server" (or the HMI) takes the role of Master, and the PLCs take the role of slaves. The master must regularly poll the slaves to get information. However, when talking about security, this protocol is not the best, as it has cleartext communications and no authentication.

Standard Modbus operations allow the reading and writing data from or to a device, and one can see its main function codes in Figure 3:

Function name	FC
Read coils	1
Write single coil	5
Read holding registers	3
Write single register	6
Write multiple registers	16
R/W multiple registers	23

Figure 3: Modbus Function Codes

Modbus-accessible data is normally stored in one of four data banks or address ranges: coils, discrete inputs, holding registers, and input registers.

III. Discussion

In this section of the assignment, the main operations occur. To begin with, the task plan is exposed, followed by the information about the tools and techniques going to be used to try and infiltrate the testbed. Then, the exploitation process is conducted, described, and analyzed, composed of the scouting phase and the attack phase.

A. Task Plan

This is where the strategy is explained in terms of how the evidence-collection procedure is planned to be undertaken.

There are multiple avenues one can take to perform the exploitation. In the reconnaissance phase, for example, a Network reconnaissance - FIN scan can be done, which is more stealthy, but prone to unreliability, as shown in Figure 4:

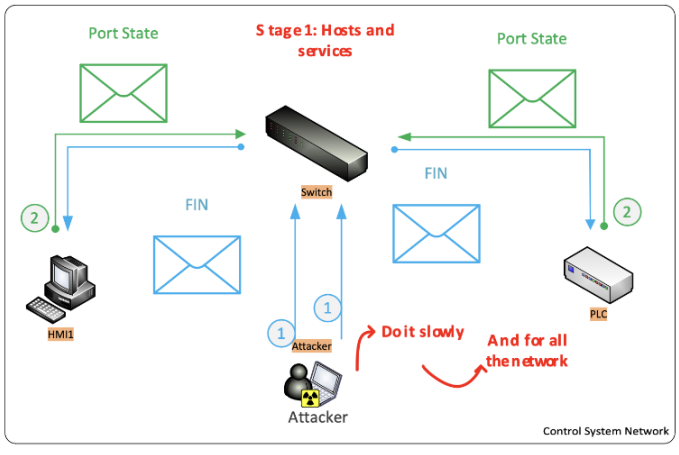

Figure 4: Scouting Example: Network reconnaissance - FIN scan

The defined objectives concerning scouting are to gather information on devices and function codes, process-level information, the protocols being used, and the existing communication flows. A scouting procedure shares a lot of similarities with the preliminary steps for a pen testing session, like planning before executing, gathering as much intelligence as possible from out-of-band sources, and, of course, striving to be invisible.

In order to obtain that kind of data, continuing on the ”being invisible” way of thinking, once that is in place, range recognition will be performed, using the Nmap scan stages, after which target profiling is executed. At the end of all this, some findings should be obtained and therefore analyzed. This is the summary of what will happen later in the paper, in the Scouting section.

Once the search is over, attacks should be manufactured with the data gathered at that point. Multiple attacks can be made on SCADA Systems, mainly Denial-of-Service (DoS) attacks (meant to shut down a machine or network, making it inaccessible to its intended users) by way of Man-in-the-Middle (MitM) attacks. Figures 5, 6, and 7 provide some examples of attacks:

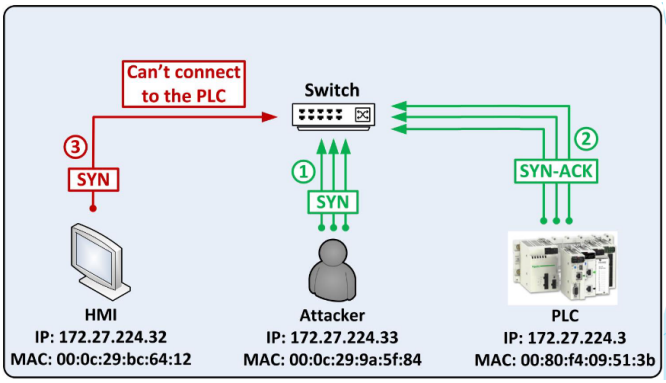

Figure 5: Flooding attack example

As for the attacks chosen to take place in this scenario, two will be performed, which are a MitM for eavesdropping (sniffing), and a MitM for actual Modbus Manipulation, to manipulate inflight data, both by way of ARP poisoning. Again, this is a brief explanation of what will take place in the MitM Attacks section later on.

B. Toolset and Techniques

The list of tools and techniques going to be used for the performance of the tasks just described goes as follows:

1) **VMware ESXi 7.0 -** a type-1 hypervisor that enables virtualization by allowing multiple virtual machines to run on a single physical host. It is a bare-metal hypervisor, which means it runs directly on the server hardware without requiring an underlying operating system. This allows for greater efficiency and flexibility in managing and deploying virtual machines [5].
2) **Kali Linux -** a purpose-specific Linux distribution for pen testing and hacking. Provides common tools, configurations, and automation which allows the user to focus on the task that needs to be completed, not the surrounding activity. It also contains industry specific modifications as well as several hundred tools targeted towards various Information Security tasks [6].
3) **Nmap -** a free and open-source utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and others. It was designed to rapidly scan large networks but works fine against single hosts [7].

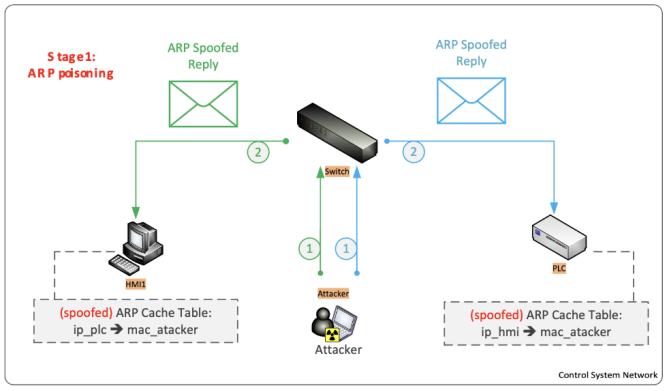

Figure 6: MiTM - ARP poisoning example

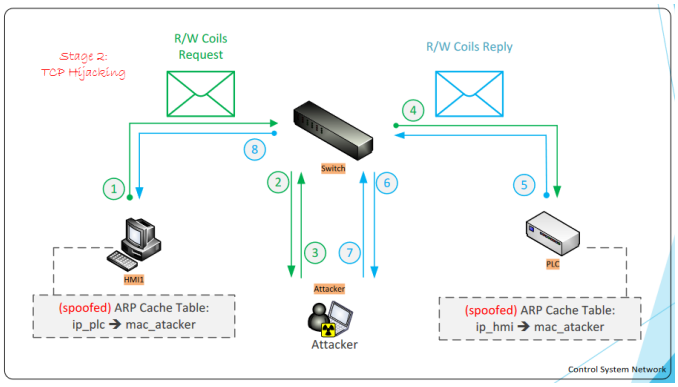

Figure 7: MiTM - TCP hijacking example

Nmap presents the following scan stages:

a) *Script pre-scanning* - the Nmap Scripting Engine (NSE) uses a collection of special-purpose scripts to gain more information about remote systems;
b) *Target enumeration* - Nmap researches the host specifiers provided by the user, which may be a combination of host DNS names, IP addresses, CIDR network notations, and more. Then, Nmap resolves these specifiers into a list of IPv4 or IPv6 addresses for scanning;
c) *Host discovery (ping scanning)* - discovering which targets on the network are online and thus worth deeper investigation;
d) *Reverse-DNS resolution* - Nmap looks up the reverse-DNS names of all hosts found online by the ping scan;
e) *Port scanning* - probes are sent, and the responses (or non-responses) to those probes are used to classify remote ports into states such as open, closed, or filtered;
f) *Version detection* - Nmap sends a variety of probes to the open ports and matches any responses against a database of known service signatures;
g) *OS detection* - different operating systems implement network standards in subtly different ways. Requested with the -O option;
h) *Traceroute* - Nmap can find the network routes to many hosts in parallel, using the best available probe packets as determined by Nmap's previous discovery phases. Enabled by the –traceroute option;
i) *Script scanning* - most NSE scripts run during this main script scanning phase;
j) *Output* - Nmap collects all the information it has gathered and writes it to the screen or to a file;
k) *Script post-scanning* - after Nmap has completed its scanning and normal output, scripts in this phase can process results and deliver final reports and statistics.

4) **Ettercap -** a multipurpose sniffer and content filter for MiTM attacks. It features sniffing of live connections, content filtering on the fly, and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis [8].
5) **Wireshark -** a network packet analyzer, presents captured packet data in as much detail as possible. It is a measuring device for examining what's happening inside a network cable [9].
6) **smod Modbus pen testing framework -** a modular framework with every kind of diagnostic and offensive feature one could need in order to pentest the Modbus protocol. It is a full Modbus protocol implementation using Python and Scapy [10].

C. Process and Results

In this section, it is explained how the overall attack process was carried forward, divided into the scouting and the MitM attacks phases, including along the way all the necessary results, analyses, or explanations.

1) Scouting

: Before starting to explain how this operation went, first, one must specify what is the meaning of the interfaces of the Virtual Machine being used. **Eth0** is the IP Network connected to the internet, **10.254.0.165**, the entry point in the VM. **Eth1** and **Eth2** are connected to the SCADA field network, and initially, they have no predefined IP, so stealthy scouting of the network is possible.

First off, ARP replies on non-local interfaces should be disabled, like illustrated in Figure 8:

```
net.ipv4.conf.eth1.arp_ignore = 1
net.ipv4.conf.eth1.arp_announce = 2
net.ipv4.conf.eth1.arp_notify = 0
```

Figure 8: Disabling ARP replies on non-local interfaces

With arp_notify set to 0, notifications of address and device changes are deactivated. With arp_ignore set to 1, it replies only if the target IP address is a local address configured on the incoming interface. With arp_annouce set to 2, the source IP address of the IP packet is ignored and an attempt is made to select a native address that can communicate with the destination IP address.

Starting with the first step now, range recognition, the objective is to enumerate the devices present in our address range. Figures 9 and 10 display the results of a command that executes an ARP scan on the **eth1** interface, using a spoofed IP address, by issuing ARP queries for an IP range. Throttling is ensured by means of the –interval option (in milliseconds):

```
┌──(root㉿kali0EI04)-[~]
└─# arp-scan --interface=eth1 172.27.224.1-172.27.224.253 --arpspa 172.27.224.44 -v --interval=250
Interface: eth1, type: EN10MB, MAC: 00:0c:29:0f:da:d2, IPv4: 172.27.224.44
Starting arp-scan 1.10.0 with 253 hosts (https://github.com/royhills/arp-scan)
172.27.224.10   00:0c:29:0e:cd:7d       VMware, Inc.
172.27.224.40   00:0c:29:56:ce:6b       VMware, Inc.
172.27.224.40   00:0c:29:56:ce:75       VMware, Inc. (DUP: 2)
172.27.224.41   00:0c:29:fa:e7:74       VMware, Inc.
172.27.224.41   00:0c:29:fa:e7:7e       VMware, Inc. (DUP: 2)
172.27.224.42   00:0c:29:5f:c7:40       VMware, Inc.
172.27.224.42   00:0c:29:5f:c7:4a       VMware, Inc. (DUP: 2)
172.27.224.43   00:0c:29:60:25:c3       VMware, Inc.
172.27.224.43   00:0c:29:60:25:cd       VMware, Inc. (DUP: 2)
172.27.224.45   00:0c:29:d7:9c:08       VMware, Inc.
172.27.224.45   00:0c:29:d7:9c:12       VMware, Inc. (DUP: 2)
172.27.224.46   00:0c:29:5c:da:d6       VMware, Inc.
172.27.224.46   00:0c:29:5c:da:e0       VMware, Inc. (DUP: 2)
172.27.224.47   00:0c:29:73:76:da       VMware, Inc.
172.27.224.47   00:0c:29:73:76:e4       VMware, Inc. (DUP: 2)
172.27.224.48   00:0c:29:16:df:14       VMware, Inc.
172.27.224.48   00:0c:29:16:df:1e       VMware, Inc. (DUP: 2)
172.27.224.50   00:0c:29:e0:d1:76       VMware, Inc.
172.27.224.50   00:0c:29:08:a5:07       VMware, Inc. (DUP: 2)
172.27.224.50   00:0c:29:e0:d1:80       VMware, Inc. (DUP: 3)
172.27.224.50   00:0c:29:08:a5:11       VMware, Inc. (DUP: 4)
172.27.224.51   00:0c:29:65:c1:9b       VMware, Inc.
172.27.224.51   00:0c:29:65:c1:a5       VMware, Inc. (DUP: 2)
172.27.224.80   00:0c:29:83:0d:c0       VMware, Inc.
172.27.224.80   00:0c:29:83:0d:ca       VMware, Inc. (DUP: 2)
172.27.224.200  00:ad:24:c1:24:73       D-Link International
172.27.224.245  08:00:06:12:c0:de       SIEMENS AG
```

Figure 9: ARP scan on the **eth1** interface (1)

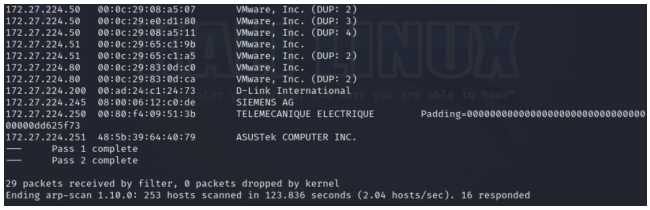

Figure 10: ARP scan on the **eth1** interface (2)

From these results, one can conclude that most IP addresses correspond to the virtual machines of our colleagues, and ASUSTek Computer INC. is just an ASUS computer, but some findings are interesting, like the first IP address that has no manufacturer correspondence, and then the D-Link International, Siemens AG, and TELEMECANIQUE ELECTRIQUE devices. For those additional scans are needed.

Therefore, Nmap is put into action. To confirm the results of the previous arp-scan, the following command was run, which executes a network range scan on the **eth1** interface, by issuing ICMP queries (-sn), with throttling ensured by means of the –scan-delay option (in seconds) and DNS queries disabled, shown in Figures 11 and 12, with the results also visible:

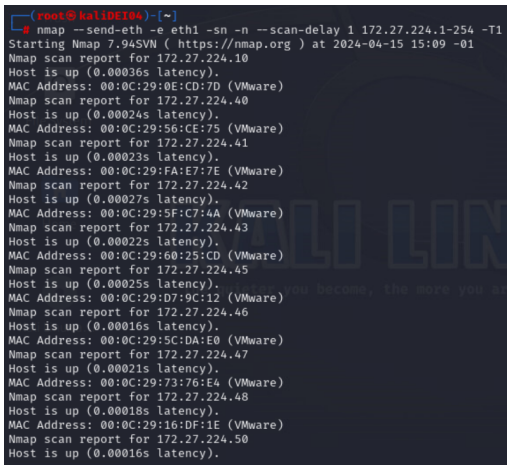

Figure 11: Nmap scan on the **eth1** interface (1)

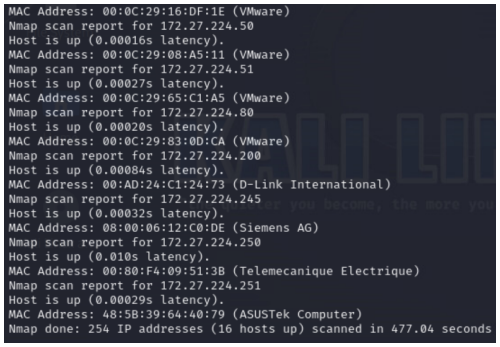

Figure 12: Nmap scan on the **eth1** interface (2)

Once again, several devices related to VMs appear, but those different ones are also still there. To analyze each one of them with due care, first, a TCP SYN scan is performed, which is a stealth scan used to determine if ports on a target system are open, closed, or filtered. Starting with the first one, **172.27.224.10**, the following results are obtained, in Figure 13:

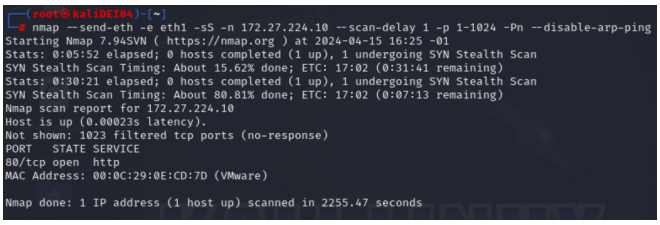

Figure 13: TCP SYN scan on **172.27.224.10**

It is discovered that port 80 is open, which corresponds to the http protocol, giving suspicions that this IP might correspond to the HMI of the testbed. So, to confirm, the IP address was accessed, which returned this page, in Figure 14:

Figure 14: Accessing **172.27.224.10**

One can see that it returned the IIS7 service, a Microsoft web server. A final step is required, which is to try and access a common URL that translates to SCADA and IACS systems. The findings are displayed in Figure 15:

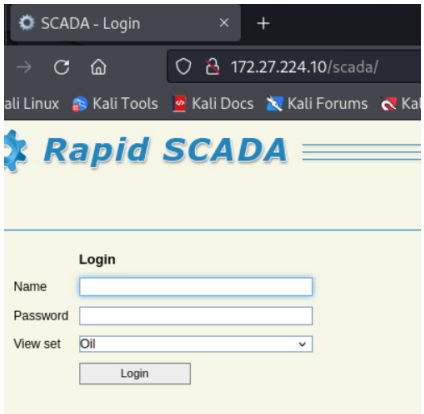

Figure 15: Accessing **172.27.224.10/scada**

The login page of the system Rapid SCADA was reached. This information successfully confirms that this IP address, indeed, matches the HMI of the testbed.

Moving on to the D-Link International device, a TCP SYN scan was also performed, as shown in Figure 16:

```
┌──(root㉿kaliDEI04)-[~]
└─# nmap --send-eth -e eth1 -sS -n 172.27.224.200 --scan-delay 1 -p 1-1024 -Pn --disable-arp-ping
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-15 17:07 -01
Stats: 0:11:50 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 69.24% done; ETC: 17:24 (0:05:15 remaining)
Nmap scan report for 172.27.224.200
Host is up (0.00076s latency).
Not shown: 1023 closed tcp ports (reset)
PORT   STATE SERVICE
23/tcp open  telnet
MAC Address: 00:AD:24:C1:24:73 (D-Link International)

Nmap done: 1 IP address (1 host up) scanned in 1025.16 seconds
```

Figure 16: TCP SYN scan on **172.27.224.200**

From the data acquired, it is probably accurate to say that this device corresponds to a network device like a router or switch, not a PLC. Nevertheless, one service was found open on port 23, which is the standard port for Telnet, a protocol used for accessing remote devices or servers. Telnet is generally considered insecure because it transmits data, including passwords, in plaintext. Therefore, a door for potential vulnerabilities is also open here.

As for the Siemens AG device, the same TCP SYN scan was run in order to investigate it. The results are in Figure 17:

```
┌──(root㉿kaliDEI04)-[~]
└─# nmap --send-eth -e eth1 -sS -n 172.27.224.245 --scan-delay 1 -p 1-1024 -Pn --disable-arp-ping
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-15 15:46 -01
Nmap scan report for 172.27.224.245
Host is up (0.00019s latency).
Not shown: 1021 closed tcp ports (reset)
PORT    STATE SERVICE
80/tcp  open  http
102/tcp open  iso-tsap
502/tcp open  mbap
MAC Address: 08:00:06:12:C0:DE (Siemens AG)

Nmap done: 1 IP address (1 host up) scanned in 1025.23 seconds
```

Figure 17: TCP SYN scan on **172.27.224.245**

Given the identified open ports, particularly ISO-TSAP (102) and Modbus (502), it can be said that this device is likely part of the operational technology (OT) infrastructure of the testbed, which could potentially be a PLC. To confirm these thoughts, a custom Nmap script is going to be used, that identifies and enumerates Siemens SIMATIC S7 PLCs, illustrated in Figure 18:

```
┌──(root㉿kaliDEI04)-[~]
└─# nmap -p 102 --script s7-enumerate -sV 172.27.224.245
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-19 13:37 -01
NSE: DEPRECATION WARNING: bin.lua is deprecated. Please use Lua 5.3 string.pack
Nmap scan report for 172.27.224.245
Host is up (0.00032s latency).

PORT    STATE SERVICE    VERSION
102/tcp open  iso-tsap?
MAC Address: 08:00:06:12:C0:DE (Siemens AG)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 217.67 seconds
```

Figure 18: Custom Nmap script run on **172.27.224.245**

As the expected PLC enumeration details are not visible, it's not possible to confirm from this output whether the device is a Siemens S7 PLC. Due to that, no attacks are going to be planned against this device, as its nature remains uncertain.

Now for the last device, TELEMECANIQUE ELECTRIQUE, the same process will be executed. Figure 19 has the TCP SYN scan results:

```
┌──(root㉿kaliDEI04)-[~]
└─# nmap --send-eth -e eth1 -sS -n 172.27.224.250 --scan-delay 1 -p 1-1024 -Pn --disable-arp-ping
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-15 15:21 -01
Nmap scan report for 172.27.224.250
Host is up (0.0056s latency).
Not shown: 1021 closed tcp ports (reset)
PORT    STATE SERVICE
21/tcp  open  ftp
80/tcp  open  http
502/tcp open  mbap
MAC Address: 00:80:F4:09:51:3B (Telemecanique Electrique)

Nmap done: 1 IP address (1 host up) scanned in 1025.23 seconds
```

Figure 19: TCP SYN scan on **172.27.224.250**

Crucially, port 502 (Modbus) is detected. These open ports suggest that the TELEMECANIQUE ELECTRIQUE device is set up to participate in network communication for file transfer, web interface access, and industrial control communications, which possibly means it might be a PLC. Therefore, once again a custom Nmap script is going to be run, this time one that identifies and enumerates Schneider Electric Modicon PLCs. Figure 20 exhibits this process:

```
┌──(root㉿kaliDEI04)-[~]
└─# nmap -p 502 --script modicon-info.nse -Pn 172.27.224.250
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-15 15:42 -01
NSE: DEPRECATION WARNING: bin.lua is deprecated. Please use Lua 5.3 string.pack
Nmap scan report for 172.27.224.250
Host is up (0.0026s latency).

PORT    STATE SERVICE
502/tcp open  Modbus
| modicon-info:
|   Vendor Name: Schneider Electric
|   Network Module: BMX P34 20302
|   CPU Module: BMX P34 20302
|   Firmware: v2.4
|   Memory Card: BMXRMS008MP
|   Project Information: Project - V6.0   CDIS-PC C:\Users\CDIS\Desktop\cdis.STU
|   Project Revision: 0.0.30
|_  Project Last Modified: 4/5/2019 16:46:48
MAC Address: 00:80:F4:09:51:3B (Telemecanique Electrique)

Nmap done: 1 IP address (1 host up) scanned in 0.48 seconds
```

Figure 20: Custom Nmap script run on **172.27.224.250**

From these findings, it is accurate to confirm that this device is, indeed, a PLC, that goes by the name of Schneider Electric PLC. Also, when accessing **172.27.224.250** directly, the device's image shows up, as displayed in Figure 21:

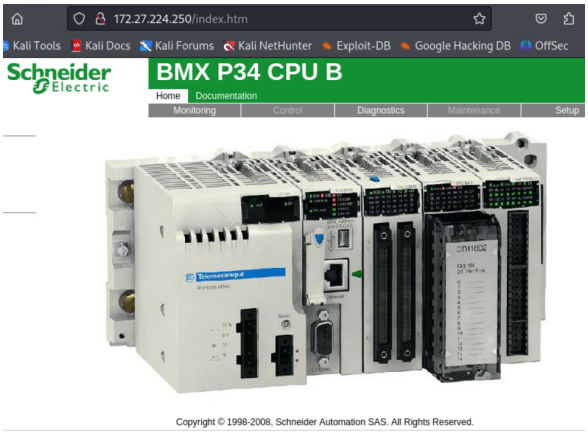

Figure 21: Accessing **172.27.224.250**

To finalize the reconnaissance procedure, the smod Modbus pen testing framework is going to be utilized to gather some more data. The objective is to obtain valid Function Codes, and valid UIDs (Unit Identifiers, used by the master to address and communicate with specific slaves). Other functionalities could be used, like reading coils for example.

Concerning the Schneider Electrique PLC, Figure 22 shows the Function codes discovered for UID = 1, while Figure 23 illustrates the UID range accepted (1-10). Of course, the parameter RHOSTS has to be set to the IP address of the Schneider Electrique PLC.

```
SMOD modbus(modbus) >use modbus/scanner/getfunc
SMOD modbus(getfunc) >show options
 Name      Current Setting  Required  Description
 ----      ---------------  --------  -----------
 Output    True             False     The stdout save in output directory
 RHOSTS                     True      The target address range or CIDR identifier
 RPORT     502              False     The port number for modbus protocol
 Threads   1                False     The number of concurrent threads
 UID       None             True      Modbus Slave UID.
SMOD modbus(getfunc) >set RHOSTS 172.27.224.250
SMOD modbus(getfunc) >set UID 1
SMOD modbus(getfunc) >exploit
[+] Module Get Function Start
[+] Looking for supported function codes on 172.27.224.250
[+] Function Code 1(Read Coils) is supported.
[+] Function Code 2(Read Discrete Inputs) is supported.
[+] Function Code 3(Read Multiple Holding Registers) is supported.
[+] Function Code 4(Read Input Registers) is supported.
[+] Function Code 5(Write Single Coil) is supported.
[+] Function Code 6(Write Single Holding Register) is supported.
[+] Function Code 8(Diagnostic) is supported.
[+] Function Code 15(Write Multiple Coils) is supported.
[+] Function Code 16(Write Multiple Holding Registers) is supported.
[+] Function Code 22(Mask Write Register) is supported.
[+] Function Code 23(Read/Write Multiple Registers) is supported.
[+] Function Code 43(Read Device Identification) is supported.
[+] Function Code 90 is supported.
SMOD modbus(getfunc) >
```

Figure 22: Checking valid FCs for Schneider PLC

```
SMOD modbus(getfunc) >use modbus/scanner/uid
SMOD modbus(uid) >set RHOSTS 172.27.224.250
SMOD modbus(uid) >show options
 Name      Current Setting  Required  Description
 ----      ---------------  --------  -----------
 Function  1                False     Function code, Defualt:Read Coils.
 Output    True             False     The stdout save in output directory
 RHOSTS    172.27.224.250   True      The target address range or CIDR identifier
 RPORT     502              False     The port number for modbus protocol
 Threads   1                False     The number of concurrent threads
SMOD modbus(uid) >exploit
[+] Module Brute Force UID Start
[+] Start Brute Force UID on : 172.27.224.250
[+] UID on 172.27.224.250 is : 10
SMOD modbus(uid) >
```

Figure 23: Checking valid UID range for Schneider PLC

Here concludes the reconnaissance and scouting phases of this assignment. Next up, with the information obtained, some proper attacks are going to be planned and executed.

2) Vulnerabilities and MitM Attacks

: Before starting the attack procedure, some brief explanations of what are MitM attacks and ARP poisoning:

Man-in-the-Middle attacks (MitM) - a type of cyber attack in which the attacker secretly intercepts and relays messages between two parties who believe they are communicating directly with each other, with the possibility of masquerading and giving incorrect data purposefully [11].

ARP poisoning - a type of attack carried out over a Local Area Network (LAN) that involves sending malicious ARP packets to a default gateway on a LAN to change the pairings in its IP to the MAC address table. Besides MitM attacks, ARP Poisoning can be used to cause a DoS condition over a LAN by simply intercepting or dropping and not forwarding the target's packets [12].

Regarding vulnerabilities that have a chance to exist, from the data gathered in scouting the HMI is potentially open to different types of injections in its login page, which could lead to full unintended access to the SCADA system. As for the telnet protocol of the D-Link International device, as it was already said, it is a protocol of an unsafe nature. Despite these being possible vulnerabilities, they ended up not being verified in this assignment.

Moving on to the MitM attacks, to begin with, ettercap needs to be launched, using the GTK interface. There, after selecting the interface **eth1** and enabling Sniffing at startup, the targets are selected, which are **172.27.224.10** and **172.27.224.250**. The ARP poisoning option is selected, and the sniff remote connections option is also enabled. Finally, the attack can be deployed. Figures 24, 25, and 26 present most of this procedure:

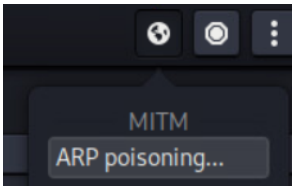

Figure 24: Selecting ARP Poisoning

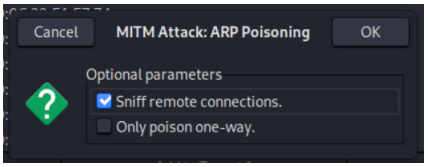

Figure 25: Enabling sniff remote connections

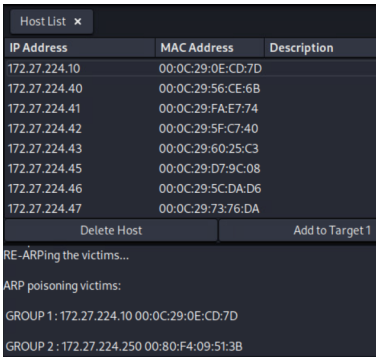

Figure 26: List of IP Addresses and ARP Poisoning targets

In order to sniff ongoing traffic, one just opens a second terminal and executes the **sudo tshark –i eth1** command. The findings are exposed in Figure 27:

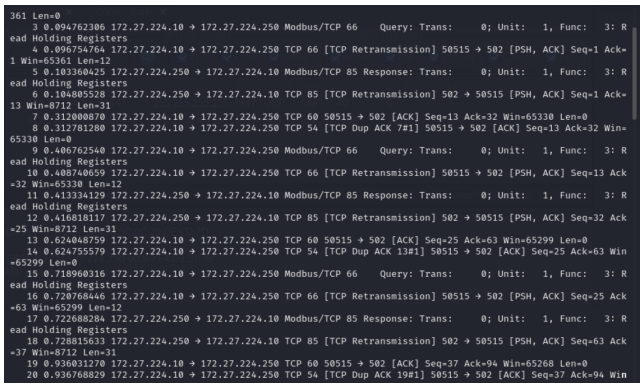

Figure 27: List of IP Addresses and ARP Poisoning targets

To analyze this information, Wireshark is very helpful, as it filters Modbus/TCP traffic flows, registers Modbus FC 3 responses, and also comes with a holding register with temperature information, which is the one that will be targeted for corruption and masquerading. Figures 28, 29, and 30 show all of this data, respectively:

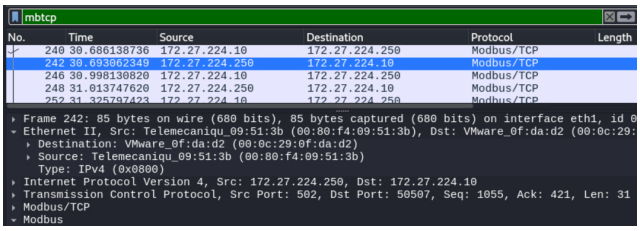

Figure 28: Modbus/TCP traffic flows in Wireshark

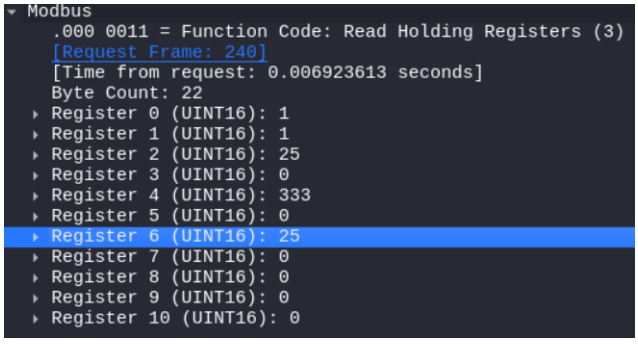

Figure 29: Modbus FC 3 responses (1)

The first part of the MitM attack is over. Now it is time for some ettercap scripting. The filter created to manipulate inflight data appears in Figure 30, while Figure 31 illustrates the loading of said filter:

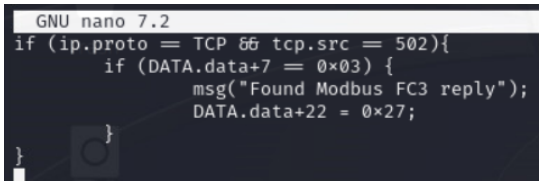

Figure 30: Ettercap filter used

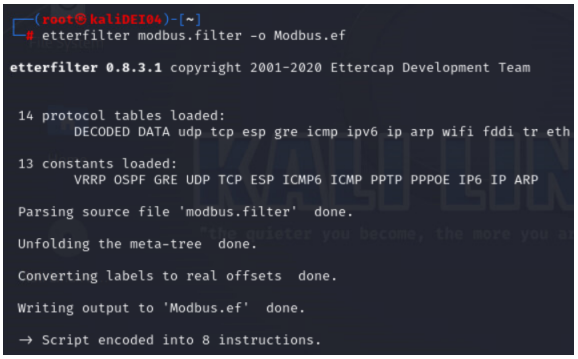

Figure 31: Loading Ettercap filter

After loading the filter in the GTK interface, the MitM is executed. The HMI is now blinded by a fake temperature value of 39º. This way, one can do whatever he pleases without anyone noticing it via the HMI. To confirm that it is indeed working, Figure 32 shows the 16-bit word for the 7th holding register (counting from 0) corresponding to the new temperature value, while Figure 33 displays the replies made due to the code in the filter:

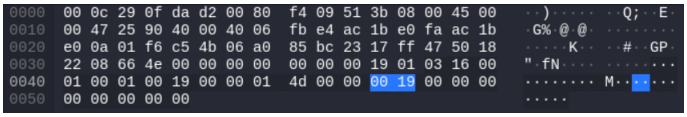

Figure 32: 16-bit word for the 7th holding register

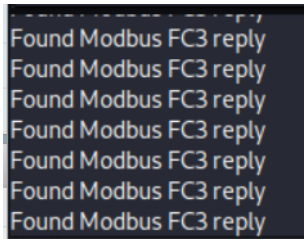

Figure 33: Modbus FC 3 replies

This marks the end of the assignment.

IV. Conclusion

All in all, this SCADA system of our testbed can be said to be vulnerable. It has multiple vulnerabilities, critically in the PLC discovered, and might even have some more that were not fully tested. Furthermore, it is considered to be very easy to do scouting and reconnaissance work around this testbed, which is also a big problem. The analysis and the attacks in full, considering all the phases, were considered a success.

Future work will be made on this subject, specifically in Assignment 2, where we will try to solve most of the vulnerabilities and threats found in the testbed that were discussed here. The objective will be to present defense, mitigation and remediation strategies, as well as demonstrate them.

References

[1] Schneider Electric: ”Modicon M340”, Available at https://www.se.com/uk/en/product-range/1468-modicon-m340/.

[2] Unitronics: ”What is the definition of ”PLC”?”, available at https://www.unitronicsplc.com/what-is-plc-programmable-logic-controller/.

[3] Inductive Automation: ”SCADA: Supervisory Control and Data Acquisition”, available at https://inductiveautomation.com/resources/article/what-is-scada.

[4] MWES: ”Industrial Automation & Control Systems”, available at https://www.mwes.com/types-of-industrial-control-systems/industrial-automation-and-controls/.

[5] Medium: ”VMware ESXi 7”, available at https://medium.com/@btech-engineering/vmware-esxi-7-38b28ab105e.

[6] Kali: ”What is Kali Linux?”, available at https://www.kali.org/docs/introduction/what-is-kali-linux/.

[7] Nmap, available at https://nmap.org/.

[8] Ettercap, available at https://www.ettercap-project.org/.

[9] Wireshark, available at https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html.

[10] smod Modbus pen testing framework, available at https://github.com/oliverhavrila/smod/.

[11] Yasar, K.: ”man-in-the-middle attack (MitM)”, available at https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM.

[12] Radware: ”ARP Poisoning”, available at https://www.radware.com/security/ddos-knowledge-center/ddospedia/arp-poisoning/.