Academic Year 2020 – 2021
Department of Mathematics
BSc degree in Computer Science
Supervisor: Federico Turrin

Low interaction ICS honeypot: implementation and testing

Francesco Trolese

February 21, 2021

Abstract

Industrial Control Systems have the function of managing and controlling the so-called critical infrastructures, such as nuclear plants, aqueducts and power grids. Their purpose is to link the cyber and the physical worlds. Unfortunately, the spread of ICS and of Programmable Logic Controllers (PLC) have made them the target of elaborated cyberattacks aimed at compromising the infrastructure that they manage. These attacks, besides creating financial losses, may cause also a huge danger to people's safety (e.g. the compromising of a nuclear plant can provoke a leak of radioactive material).
In this context, honeypots are a great tool to collect attacker data, such as malware code and system vulnerabilities exploited in order to detect in advance zero-day exploits and examine the behavior of the adversaries.
This report presents the implementation of a low interaction virtual honeypot, deployed using Honeyd. Furthermore it describes the set of tests performed to study its strengths and its limits.

Contents

1 Background

1.1 Industrial Control System

Nowadays Industrial Control Systems (ICS) [7] have become a fundamental part of lots of industrial facilities. Power plants, aqueducts, telecommunication infrastructure and factories are only a few examples of critical facilities where ICS are used. ICS is a general term that includes several types of control systems, including Programmable Logic Controllers (PLC), Supervisory Control and Data Acquisition (SCADA) systems [7] and Remote Terminal Units (RTUs) often found in the industrial sectors and critical infrastructures. The main problem with these devices is that they are often in charge of monitoring and regulating operations and processes in the so-called "critical infrastructures" so the failure of these systems could cause enormous damages. This means that an attack against one of these utilities would not only be a threat to the company or to the institution that manages it, like a traditional cyber attack, but may endanger the physical safety of thousands of people. An attack towards the ICS of an electric grid could provoke, for example, the blackout of a hospital, of a city, or even of an entire region.
Unfortunately, these types of attacks are not just fiction. Stuxnet[4] malware, discovered in 2010, was the first cyber-warfare weapon ever documented. It was a highly complex attack that used four zero-day exploits to spread and reach its targets (the PLC models 315 and 417 by Siemens) modifying their ladder logic code[3] (ladder logic is the programming language used to write programs used by PLCs). The final (partially gained) goal of this malware was to sabotage the Iranian nuclear facilities.
Stuxnet led the way to various ICS attacks. Some of the most important ones are Havex[4] (also known as Dragonfly) which threatened the energy sector in the USA and Europe in 2013 and Crashoverride[4], which succeeded in causing a power outage in Ukraine in December 2016.

1.2 Honeypots

In order to identify these attacks and defeat them at an early stage of their spreading it has been introduced the use of honeypots in ICS. Honeypots[6] are closely monitored network decoys that purposefully expose a set of vulnerabilities and services that can be analyzed and exploited by an attacker. They serve several purposes: they can distract adversaries from more valuable machines on a network, they can provide early warning about new attack and exploitation trends and they allow in-depth examination of adversaries during and after the exploitation. According to the level of interaction that is provided to an attacker, honeypots are classified in two categories: high interaction honeypots, which can simulate all aspects of an operating system and can offer the exact same level of interaction of a real system, and low interaction honeypots that simulate only some parts of the system, often the network stack, and offer to the attacker a low number of functionalities. It exists another differentiation between physical and virtual honeypots.[6]
A physical honeypot exists as a machine with a corresponding IP address on the network whereas a virtual honeypot is hosted on another machine that responds to network traffic directed to the virtual honeypot. Using virtual honeypots, it is possible to populate a network with hosts running a variety of different operating systems without actually providing a physical host for each one of them. However, to convince adversaries that a virtual honeypot is running a certain operating system, it is necessary to simulate the TCP/IP stack of the target operating system.
From these definitions, it might seem that a low interaction honeypot is not a valid solution but actually this isn't true: although a high interaction honeypot can be more hardly recognized by an attacker than a low interaction one, providing a larger piece of valuable information, this doesn't come without a price. In fact, a high interaction honeypot is actually implemented as a real system, meaning that it could be exploited by the attacker to break into the network where the honeypot is located.

2 Implementation

The honeypot used to carry out the tests has been implemented using Honeyd and farpd. The python scripts and the source code of the Siemens website have been developed by the open source SCADA HoneyNet project[5].

2.1 Honeyd

Honeyd is a daemon that simulates the TCP/IP stack of operating systems to create virtual honeypots[6]. It listens to network requests addressed to the virtual honeypots that it manages and responds according to the services that run on the virtual host. The response packets sent by Honeyd are modified by its personality engine to match the behavior of the configured operating system personality. The network stack behavior of a Honeyd virtual honeypot is defined by its personality which is modelled using the Nmap fingerprint list as a reference. The personality used by the honeypot is that of a Siemens Simatic S7-300 PLC. The parameters which define the machine to simulate, such as MAC and IP address, operating system name, network information, open ports and services emulated in them, are provided to Honeyd by a configuration file, passed as a parameter in the execution of the command Honeyd. The source code of the python scripts and of the Honeyd configuration file and a short implementation guide can be found at github.com/ftrole/honeySiemens.
The command used to deploy the honeypot is:

```
sudo honeyd -d -p nmap-os-db -i enp4s0f0 -l Honeyd.log -f Honeyd.conf IP
                          -disable-webserver
```

where:
— `-d`: runs Honeyd without demonizing it and with log messages in the terminal;
— `-p file`: provides the file with the os fingerprints;
— `-i interface`: interface is the host network interface occupied by the honeypot;
— `-l logfile`: specifies the file into which are saved the log messages;
— `-f configfile`: points at the text file with the configuration that Honeyd will run;
— `IP`: is the IP address of the virtual host or the virtual network defined in the configuration file;
— `-disable-webserver`: disables the Honeyd default webserver: in this case http connection is handled by a python script triggered by a TCP connection to port 80 of the honeypot.

2.2 Farpd

Farpd[1] is a daemon that replies to any ARP request for an IP address matching a specified destination address but only if there is not another host claiming it. This allows the physical host where the virtual honeypot is deployed to claim the IP address of the honeypot, making possible for Honeyd to reply to network packets that have the destination IP address of one of the machines that it simulates. The command used to run Farpd to correctly receive the traffic addressed to the implemented honeypot is:

```
sudo farpd -d -i enp4s0f0 <IP>
```

where:
— `-d`: executes Farpd without demonizing and shows the log in real-time;
— `-i interface`: represents the host network interface occupied by the honeypot;
— `IP`: is the IP address of the honeypot or the subnet mask of the simulated network.

3 Tests and results

In this section are reported the results of the tests run on a virtual honeypot deployed as described in the previous sections in a machine with the following characteristics:

- operating system: Ubuntu 18.04;
- Honeyd v1.6d;
- Python 2.7.

3.1 Network topology

The machine from where the tests are performed is located in the same network as the honeypot host and runs Windows 10 os.

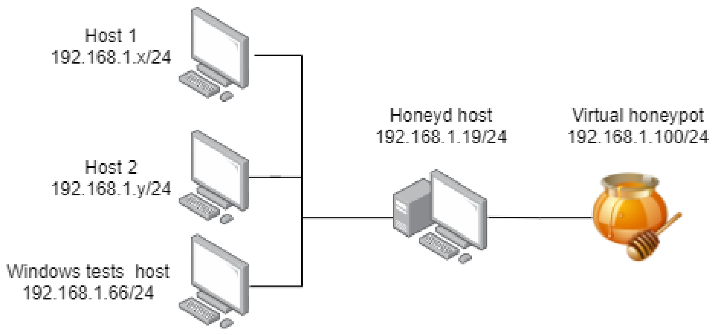


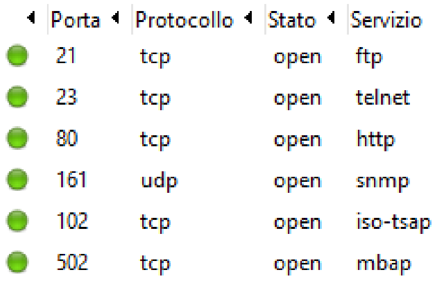
Figure 1 – Diagram representing the topology of the network where tests have been carried out

3.2 Nmap

In order to scan ports and services it has been used Nmap [5], an open source network scanning tool able to detect which operating system and services are running on a particular device.
The command used to perform a complete test on the honeypot host is:

```
nmap -sS -sU -p 1-1024 -T4 -A -v IP
```

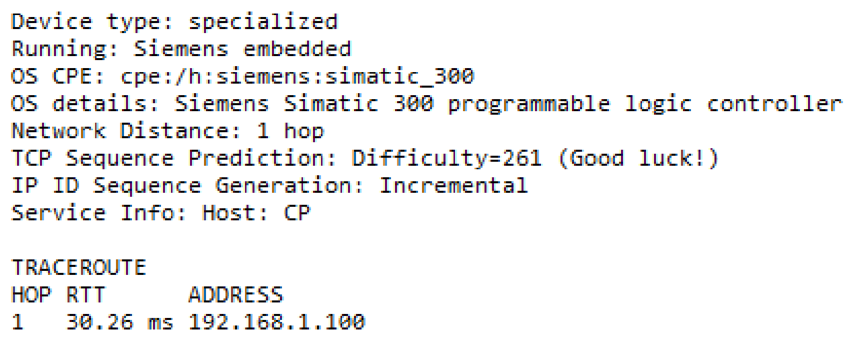
where IP is the address of the honeypot. The result obtained with the scan are reported below.



	Porta	Protocollo	Stato	Servizio
🟢	21	tcp	open	ftp
🟢	23	tcp	open	telnet
🟢	80	tcp	open	http
🟢	161	udp	open	snmp
🟢	102	tcp	open	iso-tsap
🟢	502	tcp	open	mbap

Figure 2 – Zenmap (a Nmap gui) port scanning results

As expected, the 6 open ports detected are the same as defined in the Honeyd configuration file and the operating system matches the identity of a siemens S7-300 PLC.



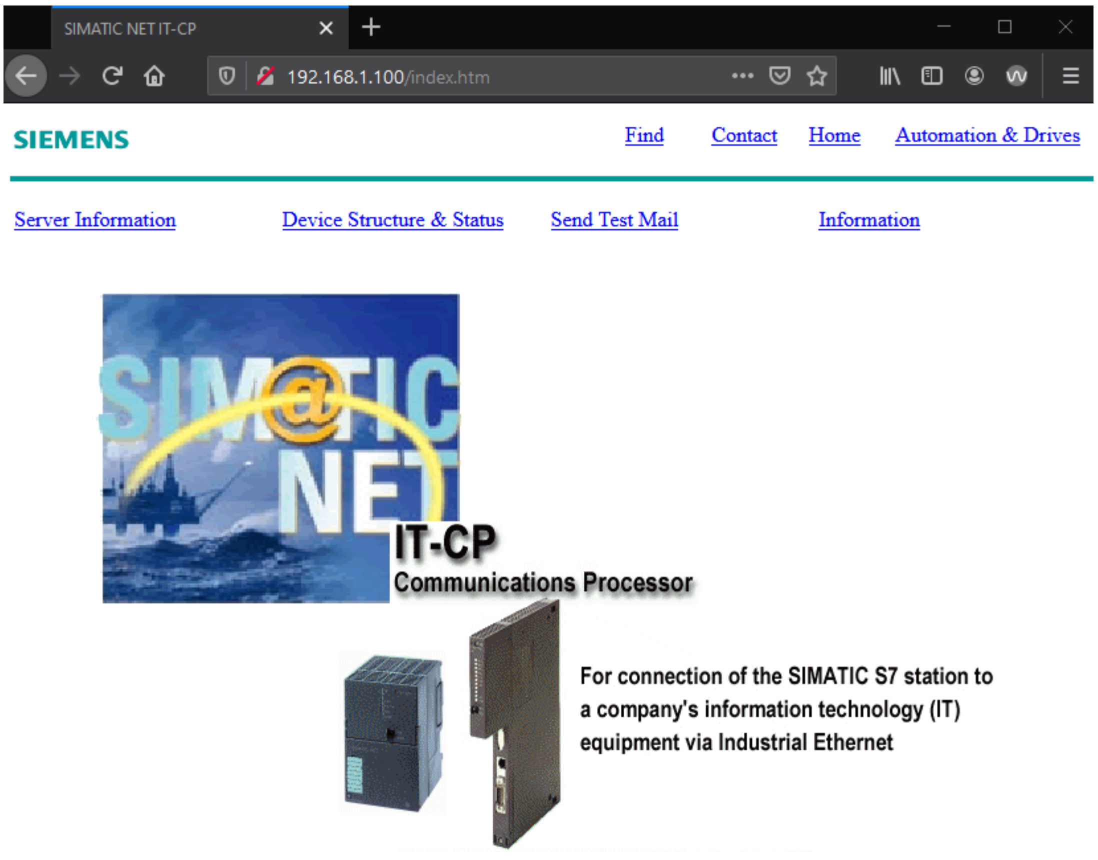
```
Device type: specialized
Running: Siemens embedded
OS CPE: cpe:/h:siemens:simatic_300
OS details: Siemens Simatic 300 programmable logic controller
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: Host: CP

TRACEROUTE
HOP RTT      ADDRESS
1   30.26 ms 192.168.1.100
```

Figure 3 – Nmap OS detection results

3.3 HTTP

To test the HTTP connection it is sufficient to make a request from the browser of a machine in the same network of the honeypot. The result obtained is a website really similar to the one that a Siemens PLC would show which provide some data about the PLC and allow an user to interact with it.

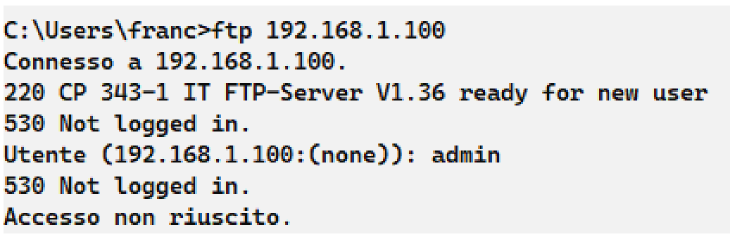


For connection of the SIMATIC S7 station to a company's information technology (IT) equipment via Industrial Ethernet

Figure 4 – Screenshot of the homepage emulated by the honeypot

3.4 FTP

When the attacker try to start a FTP connection with the honeypot, he will see a login message which asks for a password. The message is identical to that of a Siemens PLC but the result of the login in this case will be always a failure.

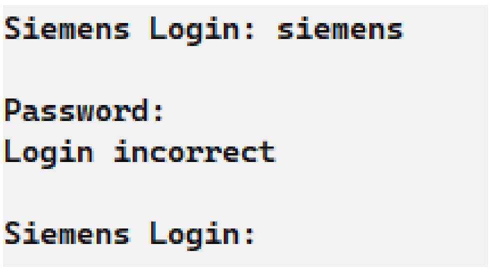


```
C:\Users\franc>ftp 192.168.1.100
Connesso a 192.168.1.100.
220 CP 343-1 IT FTP-Server V1.36 ready for new user
530 Not logged in.
Utente (192.168.1.100:(none)): admin
530 Not logged in.
Accesso non riuscito.
```

Figure 5 – Result of a FTP connection to the honeypot

3.5 Telnet

The attempt of connecting with telnet to the honeypot will display a login message prompting the user to insert a password. However, as with the FTP service, it is not possible to login.



```
Siemens Login: siemens

Password:
Login incorrect

Siemens Login:
```

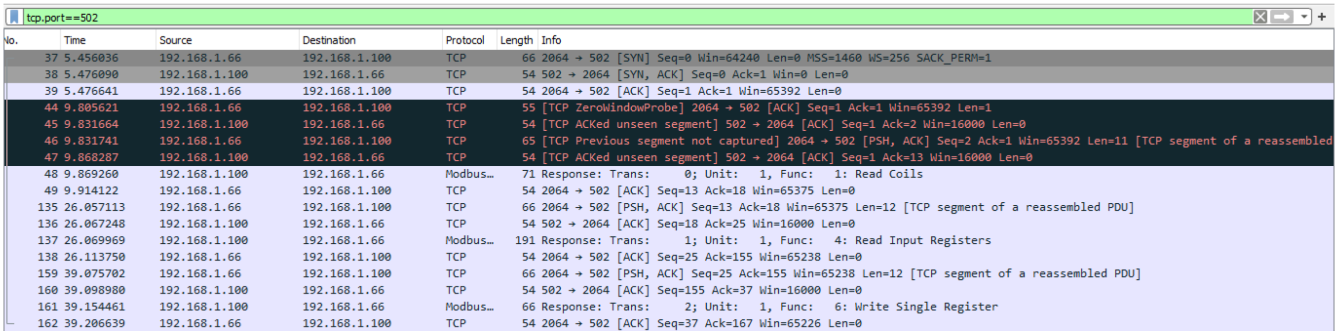
Figure 6 – Result of a telnet connection to the honeypot

3.6 Modbus

In order to test the behavior of the honeypot with Modbus TCP communication it has been used ModbusTool, a software that allows to perform reading and writing actions.
The honeypot simulates in the correct way the operations performed using this protocols, behaving exactly as a Siemens PLC would do. The following image shows the network traffic captured by Wireshark of a Modbus connection including read/write operations.

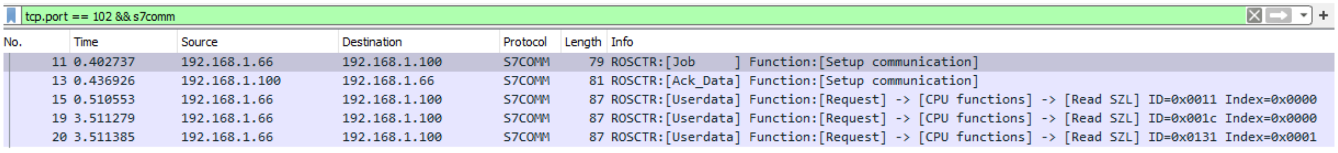
3.7 S7comm

S7comm is a Siemens proprietary protocol specific of the Siemens S7 PLC.
The S7comm communication testing has been carried out using a S7comm client provided by the open source Snap7 library[2]. The honeypot responds to the read/write requests exactly as it should. The following image shows the network traffic captured by Wireshark.


tcp.port==502

No.	Time	Source	Destination	Protocol	Length	Info
37	5.456036	192.168.1.66	192.168.1.100	TCP	66	2064 → 502 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
38	5.476090	192.168.1.100	192.168.1.66	TCP	54	502 → 2064 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
39	5.476641	192.168.1.66	192.168.1.100	TCP	54	2064 → 502 [ACK] Seq=1 Ack=1 Win=65392 Len=0
44	9.805621	192.168.1.66	192.168.1.100	TCP	55	[TCP ZeroWindowProbe] 2064 → 502 [ACK] Seq=1 Ack=1 Win=65392 Len=1
45	9.831664	192.168.1.100	192.168.1.66	TCP	54	[TCP ACKed unseen segment] 502 → 2064 [ACK] Seq=1 Ack=2 Win=16000 Len=0
46	9.831741	192.168.1.66	192.168.1.100	TCP	65	[TCP Previous segment not captured] 2064 → 502 [PSH, ACK] Seq=2 Ack=1 Win=65392 Len=11 [TCP segment of a reassembled
47	9.868287	192.168.1.100	192.168.1.66	TCP	54	[TCP ACKed unseen segment] 502 → 2064 [ACK] Seq=1 Ack=13 Win=16000 Len=0
48	9.869260	192.168.1.100	192.168.1.66	Modbus…	71	Response: Trans: 0; Unit: 1, Func: 1: Read Coils
49	9.914122	192.168.1.66	192.168.1.100	TCP	54	2064 → 502 [ACK] Seq=13 Ack=18 Win=65375 Len=0
135	26.057113	192.168.1.66	192.168.1.100	TCP	66	2064 → 502 [PSH, ACK] Seq=13 Ack=18 Win=65375 Len=12 [TCP segment of a reassembled PDU]
136	26.067248	192.168.1.100	192.168.1.66	TCP	54	502 → 2064 [ACK] Seq=18 Ack=25 Win=16000 Len=0
137	26.069969	192.168.1.100	192.168.1.66	Modbus…	191	Response: Trans: 1; Unit: 1, Func: 4: Read Input Registers
138	26.113750	192.168.1.66	192.168.1.100	TCP	54	2064 → 502 [ACK] Seq=25 Ack=155 Win=65238 Len=0
159	39.075702	192.168.1.66	192.168.1.100	TCP	66	2064 → 502 [PSH, ACK] Seq=25 Ack=155 Win=65238 Len=12 [TCP segment of a reassembled PDU]
160	39.098980	192.168.1.100	192.168.1.66	TCP	54	502 → 2064 [ACK] Seq=155 Ack=37 Win=16000 Len=0
161	39.154461	192.168.1.100	192.168.1.66	Modbus…	66	Response: Trans: 2; Unit: 1, Func: 6: Write Single Register
162	39.206639	192.168.1.66	192.168.1.100	TCP	54	2064 → 502 [ACK] Seq=37 Ack=167 Win=65226 Len=0

Figure 7 – Wireshark analysis of the modbus traffic (filtered with tcp.port==502)


tcp.port == 102 && s7comm

No.	Time	Source	Destination	Protocol	Length	Info
11	0.402737	192.168.1.66	192.168.1.100	S7COMM	79	ROSCTR:[Job] Function:[Setup communication]
13	0.436926	192.168.1.100	192.168.1.66	S7COMM	81	ROSCTR:[Ack_Data] Function:[Setup communication]
15	0.510553	192.168.1.66	192.168.1.100	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
19	3.511279	192.168.1.66	192.168.1.100	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
20	3.511385	192.168.1.66	192.168.1.100	S7COMM	87	ROSCTR:[Userdata] Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0131 Index=0x0001

Figure 8 – Wireshark analysis of the modbus traffic (filtered with tcp.port == 102)

3.8 SNMP

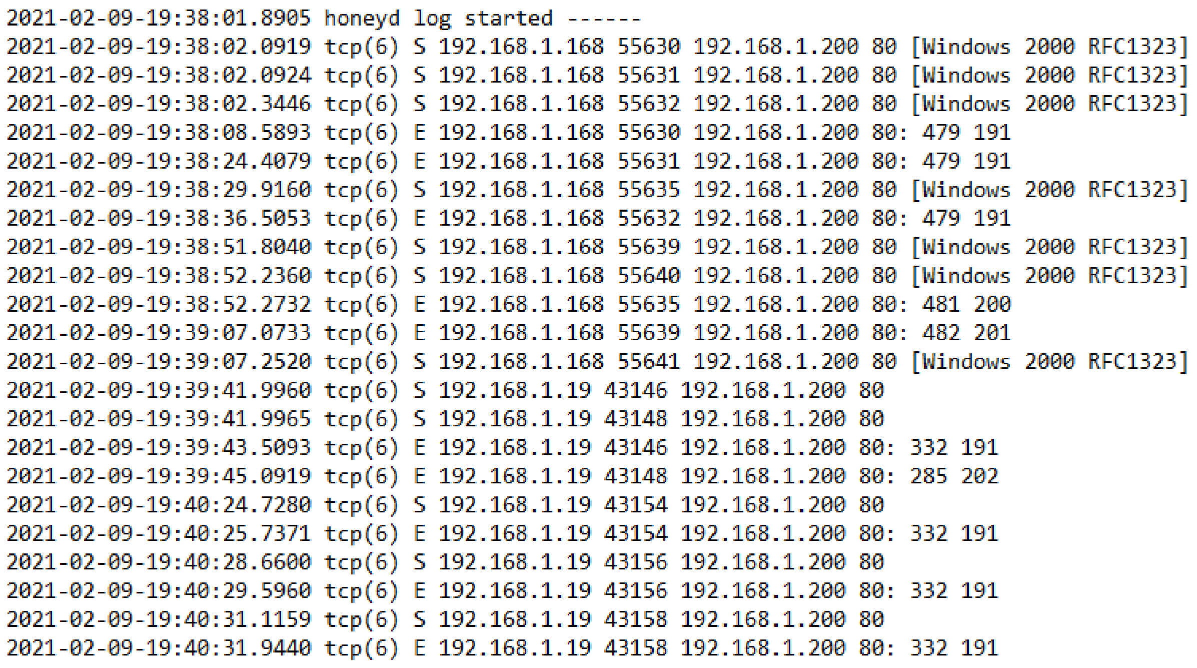
A confirmation on the right identity of the honeypot is given also by performing a snmpwalk against its IP address. The behavior registered matches the one expected by a Siemens PLC.

```
iso.3.6.1.2.1.1.1.0 = STRING: "Siemens, SIMATIC, S7-300"
iso.3.6.1.2.1.1.2.0 = OID: ccitt.0
iso.3.6.1.2.1.1.3.0 = Timeticks: (181165360) 20 days, 23:14:13.60
iso.3.6.1.2.1.1.4.0 = ""
iso.3.6.1.2.1.1.5.0 = ""
iso.3.6.1.2.1.1.6.0 = ""
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.2.1.0 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.1.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "Siemens, SIMATIC NET, CP343-1 IT, 6GK7 343-1GX20-0XE0, HW: Version 1, FW: Version V1.1.4, Fast Ethernet
 Port 1, Rack 0, Slot 4, 100 Mbit, full duplex, autonegotiation"
iso.3.6.1.2.1.2.2.1.3.1 = INTEGER: 6
iso.3.6.1.2.1.2.2.1.4.1 = INTEGER: 1500
iso.3.6.1.2.1.2.2.1.5.1 = Gauge32: 100000000
iso.3.6.1.2.1.2.2.1.6.1 = STRING: "8:0:6:72:7c:8"
iso.3.6.1.2.1.2.2.1.7.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.8.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.9.1 = Timeticks: (2448580) 6:48:05.80
iso.3.6.1.2.1.2.2.1.10.1 = Counter32: 83436854
iso.3.6.1.2.1.2.2.1.11.1 = Counter32: 445126
iso.3.6.1.2.1.2.2.1.12.1 = Counter32: 822805
iso.3.6.1.2.1.2.2.1.13.1 = Counter32: 0
iso.3.6.1.2.1.2.2.1.14.1 = Counter32: 0
iso.3.6.1.2.1.2.2.1.15.1 = Counter32: 0
iso.3.6.1.2.1.2.2.1.16.1 = Counter32: 25980254
iso.3.6.1.2.1.2.2.1.17.1 = Counter32: 303652
iso.3.6.1.2.1.2.2.1.18.1 = Counter32: 111
iso.3.6.1.2.1.2.2.1.19.1 = Counter32: 0
```

Figure 9 – Result of snmpwalk against the honeypot

3.9 Honeyd log

In the tests reported above it has always been assumed the attacker's point of view. On the other hand, the honeypot administrator, whenever Honeyd receives a request of connection, can see a piece of useful information about the attacker. This information is displayed in the terminal but is also saved in a log file and, if desired, stored in a SQL database. These useful features of Honeyd allow to carefully analyze the network traffic in order to extract as more knowledge as possible about the behavior of the attacker.



```
2021-02-09-19:38:01.8905 honeyd log started ------
2021-02-09-19:38:02.0919 tcp(6) S 192.168.1.168 55630 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:02.0924 tcp(6) S 192.168.1.168 55631 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:02.3446 tcp(6) S 192.168.1.168 55632 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:08.5893 tcp(6) E 192.168.1.168 55630 192.168.1.200 80: 479 191
2021-02-09-19:38:24.4079 tcp(6) E 192.168.1.168 55631 192.168.1.200 80: 479 191
2021-02-09-19:38:29.9160 tcp(6) S 192.168.1.168 55635 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:36.5053 tcp(6) E 192.168.1.168 55632 192.168.1.200 80: 479 191
2021-02-09-19:38:51.8040 tcp(6) S 192.168.1.168 55639 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:52.2360 tcp(6) S 192.168.1.168 55640 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:38:52.2732 tcp(6) E 192.168.1.168 55635 192.168.1.200 80: 481 200
2021-02-09-19:39:07.0733 tcp(6) E 192.168.1.168 55639 192.168.1.200 80: 482 201
2021-02-09-19:39:07.2520 tcp(6) S 192.168.1.168 55641 192.168.1.200 80 [Windows 2000 RFC1323]
2021-02-09-19:39:41.9960 tcp(6) S 192.168.1.19 43146 192.168.1.200 80
2021-02-09-19:39:41.9965 tcp(6) S 192.168.1.19 43148 192.168.1.200 80
2021-02-09-19:39:43.5093 tcp(6) E 192.168.1.19 43146 192.168.1.200 80: 332 191
2021-02-09-19:39:45.0919 tcp(6) E 192.168.1.19 43148 192.168.1.200 80: 285 202
2021-02-09-19:40:24.7280 tcp(6) S 192.168.1.19 43154 192.168.1.200 80
2021-02-09-19:40:25.7371 tcp(6) E 192.168.1.19 43154 192.168.1.200 80: 332 191
2021-02-09-19:40:28.6600 tcp(6) S 192.168.1.19 43156 192.168.1.200 80
2021-02-09-19:40:29.5960 tcp(6) E 192.168.1.19 43156 192.168.1.200 80: 332 191
2021-02-09-19:40:31.1159 tcp(6) S 192.168.1.19 43158 192.168.1.200 80
2021-02-09-19:40:31.9440 tcp(6) E 192.168.1.19 43158 192.168.1.200 80: 332 191
```

Figure 10 – Example of Honeyd log files containing the log of a SNMP connection

4 Conclusions

The honeypot introduced by this project provides a discrete variety of services that could trick the attacker and make him guess he is interacting with a real PLC. However, it should be considered that this is the implementation of a low interaction honeypot. The simulated services provide a limited number of features to interact with and the honeypot lacks of production ICS environment generated data. These shortcomings can be determining factors that make an expert attacker realizes he is not interacting with a real PLC, preventing him to share valuable information.
These aspects could be the starting points for future researches, aimed at making the honeypot more realistic from the point of view of a hypothetical attacker.

References

[1] Farpd, an arp reply daemon. Technical report.

[2] Step7 open source ethernet communication. Technical report.

[3] Erickson and K. T. Programmable logic controllers. *IEEE potentials*, (15.1):14–17, 1996.

[4] K. E. Hemsley, E. Fisher, and et al. History of industrial control system cyber incidents. Technical report, 2018.

[5] G. F. Lyon. Nmap network scanning: The official nmap project guide to network discovery and security scanning. *Insecure*, 2009.

[6] P. Niels. Honeyd: A virtual honeypot daemon (extended abstract).

[7] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn. Guide to industrial control systems (ics) security. *Nist special publication*, (800-82), 2014.