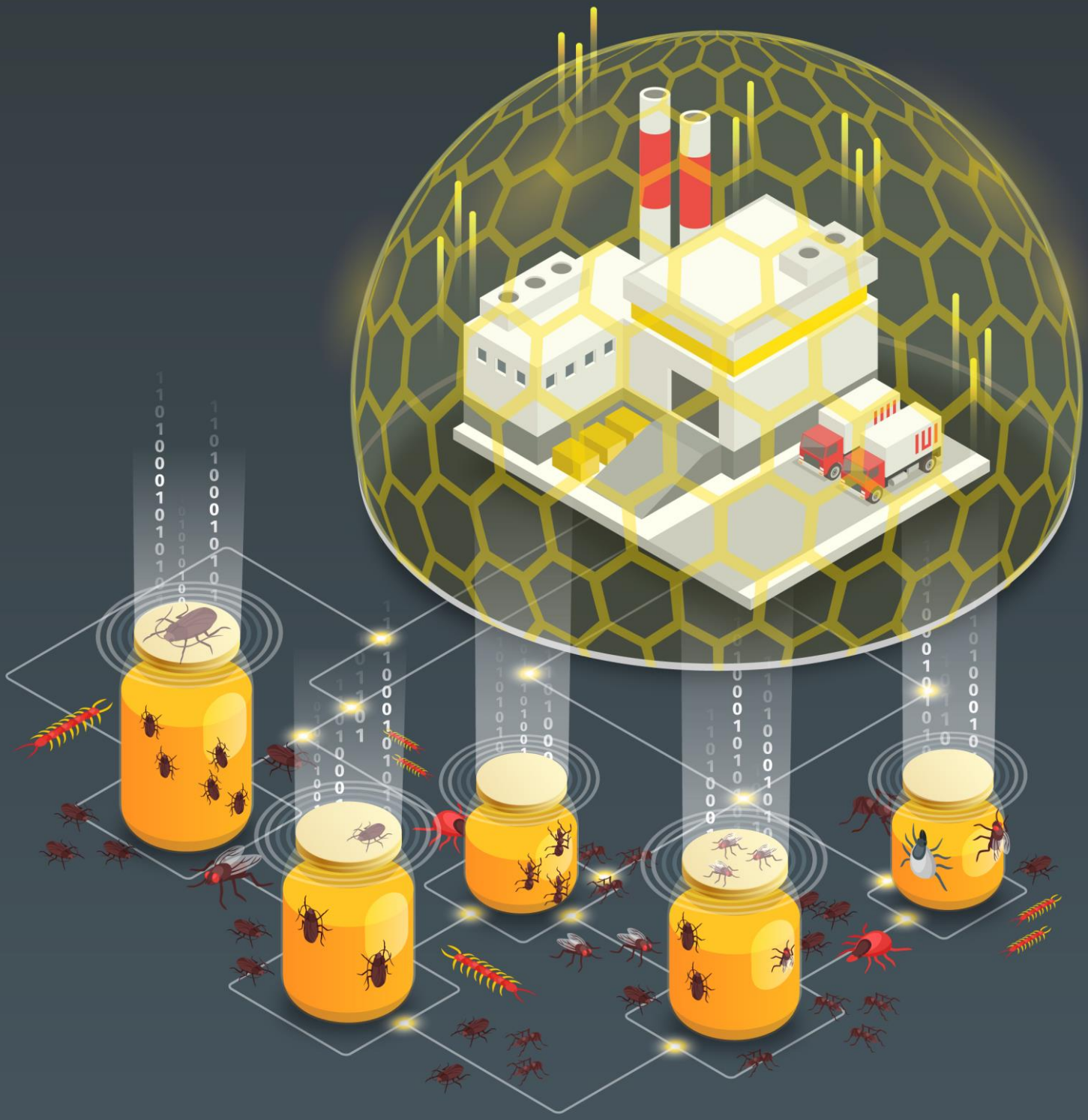


Industrial honeypot implementation guide

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y EMPRESA

incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE

incibe cert_

October 2019

INCIBE-CERT_INDUSTRIAL_HONEYPOT_IMPLEMENTATION_GUIDE_2019_v1

This publication belongs to INCIBE (Spanish National Cybersecurity Institute) and is subject to a Creative Commons Attribution-Non-commercial 3.0 Spain licence. As such, the copying, distribution, and public communication of this guide is permitted under the following conditions:

- Attribution. The content of this report may be fully or partially reproduced by third parties, provided that they cite its origin and make express reference to INCIBE or INCIBE-CERT and its website: https://www.incibe.es/. This attribution shall, under no circumstance, indicate that INCIBE supports this third party or supports the use that it makes of its study.
- Non-commercial Use. The original material and the studies deriving therefrom may be distributed, copied, and exhibited, provided that their use is not for commercial purposes.

When re-using or distributing the study, the terms of the licence of this study must be made clear. Some of these terms may be waived if permission is obtained from INCIBE-CERT as the copyright owner. Full licence text: https://creativecommons.org/licenses/by-nc-sa/3.0/es/.

Contents

1. About this guide 6
2. Organisation of the document 7
3. Introduction 8
4. Honeynet 9
4.1. Description of a honeynet 9
4.2. General honeynet architecture 9
4.3. Components 10
4.3.1. Honeywall 10
4.3.2. Honeypot 11
5. Classification of honeypots 12
5.1. Type of interaction 12
5.1.1. High interaction 12
5.1.2. Low interaction 13
5.2. Type of equipment 14
5.2.1. Physical 14
5.2.2. Virtual 15
5.3. Type of behaviour 15
5.3.1. Precise detection methods 15
5.3.2. Intrusion protection 16
5.3.3. Disable actions 16
5.3.4. Deceleration or defence against automated attacks 16
5.3.5. None 16
5.4. Type of role 16
5.4.1. Client 16
5.4.2. Server 17
6. Tools 18
6.1. HoneyMonkey 18
6.2. BeEF 19
6.3. HoneyBadger 20
6.4. Spidertrap 20
6.5. Weblabyrinth 21
6.6. Network Obfuscation and Virtualized Anti-Reconnaissance 21
7. Projects 22
7.1. The Honeynet Project 22
7.2. Honeynet SCADA by DigitalBond 22
7.3. Conpot and GasPot 23
7.4. Honeyd 24

8. Deployment of an industrial honeypot 25
8.1. Tools used 25
8.2. Honeypot design 25
8.3. System hardening of the host OS 26
8.4. Honeyd Installation 26
8.4.1. Honeyd download 26
8.4.2. Installation of dependencies 26
8.4.3. Honeyd compilation and installation 27
8.5. Honeyd configuration 27
8.6. Honeypot maintenance 32
8.7. Results 32
8.7.1. Tools used 32
8.7.2. Checking services 32
8.7.3. Log analysis 40
9. Conclusions 42
Appendix: Cheat sheet 43
10. References 44

LIST OF FIGURES

Figure 1 Example of a honeynet architecture 10
Figure 2 Classification of honeypots 12
Figure 3 High interaction honeypot 13
Figure 4 Low interaction honeypot 13
Figure 5 Client honeypot 17
Figure 6 Server honeypot 17
Figure 7 HoneyMonkey phases of implementation 19
Figure 8 BeEF logo 19
Figure 9 HoneyBadger logo 20
Figure 10 NOVA logo 21
Figure 11 The Honeynet Project logo 22
Figure 12 Architecture proposal for an industrial honeynet 23
Figure 13 Conpot and GasPot logos 24
Figure 14 Git installation and Honeyd download commands 26
Figure 15 Command for installing dependencies 27
Figure 16 Commands for the installation of Honeyd 27
Figure 17 Creation of the folder 27
Figure 18 State of the folder at this point 28
Figure 19 Edited line in the file “honeyd-http-siemens.py” 29
Figure 20 Edited line in the file “honeyd-telnet-siemens.py” 29
Figure 21 Complete name of the OS obtained from nmap-os-db 29
Figure 22 Name of the OS added in the last line of nmap.assoc 29
Figure 23 Previously created folder, with the configuration file 30
Figure 24 Contents of the configuration file honeyd.conf 30
Figure 25 Command and parameters for running the Honeyd virtual machine 31
Figure 26 Result of running the honeyd command 31
Figure 27 Results of port scanning in the honeypot 33

Figure 28 Results of scanning the honeypot OS 34
Figure 29 Web page emulated in the honeypot 35
Figure 30 Result of the FTP connection to the honeypot 35
Figure 31 Status ModbusTool in Idle 36
Figure 32 Status ModbusTool in Connected 36
Figure 33 Dropdown of ModbusTool functions 37
Figure 34 Exchange of record reading packages between ModbusTool and the honeypot 37
Figure 35 Exchange of record writing packages between ModbusTool and the honeypot 37
Figure 36 Screenshot of the client snap7 38
Figure 37 Exchange of connection packages between the client Snap7 and the honeypot 39
Figure 38 Result of a Telnet attempt to access the honeypot 39
Figure 39 Result of running the snmpwalk against the honeypot 40
Figure 40 ICMP request log format 40
Figure 41 FTP request log format 41
Figure 42 Telnet connections 41
Figure 43 SNMP request log format 41
Figure 44 HTTP requests to honeypot 41
Figure 45 Modbus connections 41
Figure 46 S7comm connection 41

LIST OF TABLES

Table 1 Comparison of low and high interaction honeypots 14
Table 2 Comparison of physical and virtual honeypots 15
Table 3 List of used tools 25
Table 4 List of tools used in the testing phase 32

1. About this guide

This guide defines the concept of a honeypot, the recommended requirements for its correct implementation, its classification based on different criteria (iteration, equipment, behaviour and role) and its progress to the present day, paying special attention to honeynets and the way in which they are typically implemented.

The necessary steps for building an industrial honeypot from scratch are included, with illustrations and examples, specifying its possible uses and its most important characteristics.

2. Organisation of the document

This document consists of an 3.Introduction to honeynets, providing further detail on honeypots, through 5 main sections, divided into different categories, addressing points such as: what they are, the way in which they are classified, changes that have taken place since their beginnings, the most well-known projects and the step-by-step creation of an industrial honeypot.

The first section, 4.-Honeynet, is focused on defining the parts that make up this network, which consists of a honeywall, acting as a firewall, and honeypots, that act as decoys in order to gather information from potential attackers.

After explaining the concepts related to the study, it is time for 5.-Classification of honeypots, which reflects the different possible types depending on the chosen approach. According to the type of interaction (low or high), the type of equipment we will use (physical machinery, virtual machinery or both), the use or behaviour we want to give our honeypot (to either detect, protect against intruders, disable actions of the attacker or harming the attacker through a defence that slows them down) and the type of role the honeypot will play (server, which will be prepared to receive attacks, or client, which will send requests against networks or malicious devices).

Section 6.-Tools, is focused on how security and the ways we gather information on attackers have changed over the years, in order to improve the defence of industrial control systems.

Then, there is an overview of the different stand-out 7.-Projects on industrial honeypots, noting when they were created, their main objectives and why they were created.

The last section, 8.-Deployment of an industrial honeypot, describes, like an installation manual, the steps to create an industrial honeypot from scratch, taking into account the tools that will be used, the design and a good system hardening of the base operating system, as well as a good configuration and its subsequent maintenance.

To conclude the guide, chapter 9.-Conclusions contains the important points learnt throughout the guide.

3. Introduction

The search for how to improve the cybersecurity of our systems is a continuous process. The main problem is determining how an attacker may act or the methods they will use to make sure their actions are successful.

One of the tools used with this proposal is the so-called honeynet, a network designed with different tools and machines (Windows, Linux, Solaris, etc.) that are exclusively dedicated to being the target of attackers, in order to subsequently be able to detect them, obtain information on them and know how they have carried out the different attacks against it, for purposes of developing protective measures against these cyberattacks.

How to design and deploy a honeynet is not as trivial a task as we would like it to be, and given that it consists of a network of connected honeypots, we will focus on defining one of these nodes, which is a known and widespread term. These honeypots are tools or systems that are designed in order to deceive the attacker, hence the name “honeypot”, with respect its aim of attracting attackers. The use of this type of tool can result in both companies and researchers obtaining valuable information about attackers, as well as providing protection against possible intrusion attempts, to industrial control systems, since they act as prevention, detection and response.

4. Honeynet

Day after day, cyberattackers are learning and using new techniques in order to try to compromise systems, which means a worldwide increase in cyberattacks.

A honeypot is a special type of network that has been designed and prepared as a means of defence, in order to be able to be attacked and gather a large amount of information about the methods and techniques used by cyberattackers. The use of this type of network began in 1999 with "The Honeynet Project"[1]*, when its founder Lance Spitzner unveiled the honeypot concept.*

4.1. Description of a honeynet

With a honeynet, the goal is to achieve an as realistic as possible simulation of what a real network would be, including production systems, servers, services, etc. They are designed to be able to be compromised by attackers and structured to extract, in the best possible way, the largest amount of information, with regard to learning about the techniques and tools used by cyberattackers.

A honeynet's degree of success lies in being able to monitor all of the attacker's movements and actions within the network. All traces left by cyberattackers, due to their actions and the use of tools, are analysed and monitored in order to be able to know what tactics are used and what is the final objective to be achieved. Furthermore, an important factor is the attackers' skill because, if the attacker has experience, they may be able to detect that it is a honeynet and stop the attack. In addition, there are tools on the Internet that can identify networks that use honeypots, such as Honeypot Or Not from Shodan[2].

The honeynet's most significant challenge lies in identifying and locating the activities carried out by cyberattackers within the network, which is done through a tool for capturing traffic (Wireshark, tcpdump, etc.), an analysis and study of the packages circulating through the network, trying to identify the attacker's traffic, in order to monitor it and try to find out their techniques, tools and objectives.

4.2. General honeynet architecture

In general, there are no specific architecture models when it comes to setting up a honeynet, in other words, there is no standard, so the vast majority of them have different architectures. The tools for control and the elements for an industrial simulation, such as PLC, HMI, SCADA, or the activity log and the types of analysis used for the actions of intruders vary. At the end of the day, a honeynet is not a product; software is not installed to later make it operational, it is a highly controlled network for containing and analysing attackers in real time.

The main components that make up a honeynet are:

- A gateway, called a honeywall, through which all traffic, both coming in and going out, passes and through which the attacker is obligated to pass through.

[1] https://www.honeynet.org/

[2] https://honeyscore.shodan.io/

- The honeypots are equipment meant to simulate end equipment and, therefore, candidates for being attacked. If it involves simulating an industrial environment, with automation components, it includes components such as PLC, SCADA, HMI, RTU, etc., and elements to make the attacker believe they are within the network of an industry and not within a replicated environment. These are always prepared to simulate an environment as realistic as possible, trying to deceive the attacker in order to make them think that they are accessing real equipment in production, insomuch as, if at any time they had any suspicion, they would be able to complete the attack without leaving any information.

4.3. Components

The two main components of a honeynet are described below:

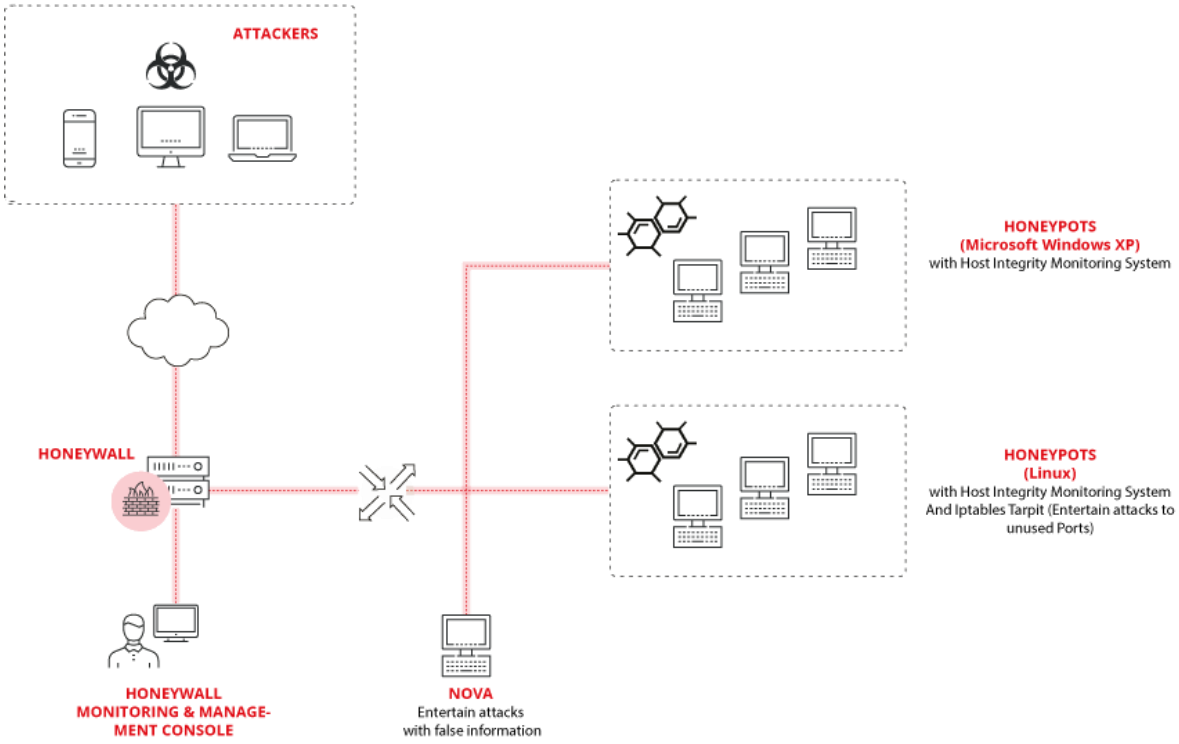


Figure 1 Example of a honeynet architecture

4.3.1. Honeywall

A honeywall is a machine that is exclusively prepared to act as a firewall, in other words, to filter and monitor the traffic generated in a honeynet. Therefore, ideally, audit tools, network analysers and IDS are used or combined in it.

The honeynet must capture the greatest amount of useful data and information so that, subsequently, new types of attacks, strategies and tools used by intruders may be analysed and extracted. All of the above must be done without the cyberattacker noticing that they are being monitored, so the entire data capture and analysis process must be undertaken in the most transparent and careful way possible.

With regard to the honeywall's design, it would ideally have several elements, called sensors, which collect data in a number of places, both inside and outside the honeynet, that is, to layer. One of the main components is the firewall, which is placed at the entry in order to be able to analyse and capture all the data traffic coming in and out. Furthermore, it will serve to alert and warn whenever an attack is being carried out. Another element that can be implemented is an intrusion detection system (IDS), which is used to analyse network traffic, comparing it with known attack signatures or according to suspicious behaviour patterns of an attacker, such as port scanning or sending and receiving malformed packages, among others.

It is ideal to implement both tools, turning the honeywall into a complete tool that combines intelligence and blocking abilities, this being the point where all packages must go to enter the honeynet's internal network.

4.3.2. Honeypot

A honeypot is a simulated system that serves as a security tool, located within a network and designed to receive attacks. Its main objective is to obtain valuable data on cyberattackers and their methods, tools used, intrusion techniques and modus operandi.

When building a honeypot, we must decide between the two existing options: physical or virtual equipment. The differences between the two are that a physical computer is fully functional and will make it more difficult for the attacker to realise that he is inside a honeypot. On the other hand, the simulated honeypot is not fully functional, it has a service that allows you to simultaneously imitate thousands of operating systems and their characteristics. The existing tools for creating virtualised honeypots are able to simulate the operating system at the TCP/IP stack level, so they allow you to deceive network and port scanning tools such as nmap and xprobe; as it is a virtualisation subsystem, it allows you to have real services such as http, ftp or telnet, with the aim of making the attacker believe that he is in a real environment.

For SCI environments, different aspects must be taken into account in order to set up an industrial network as complete as possible, especially regarding PLC devices or SCADA systems, as we will need to decide which ones are used and define whether real or virtual ones should be used. The act of completely simulating the industrial network involves including each of the corresponding elements in the network, having control and automation elements to be able to deploy a honeynet in conditions and obtain the most and best information possible.

5. Classification of honeypots

Honeypots may be classified according to diverse characteristics or features. Therefore, prior to building one, we must set our objectives in order to choose the type of honeypot that is best adapted to such needs according to its function. Subsequently, we will discuss a classification of honeypots according to the type of interaction, equipment, behaviour and role.

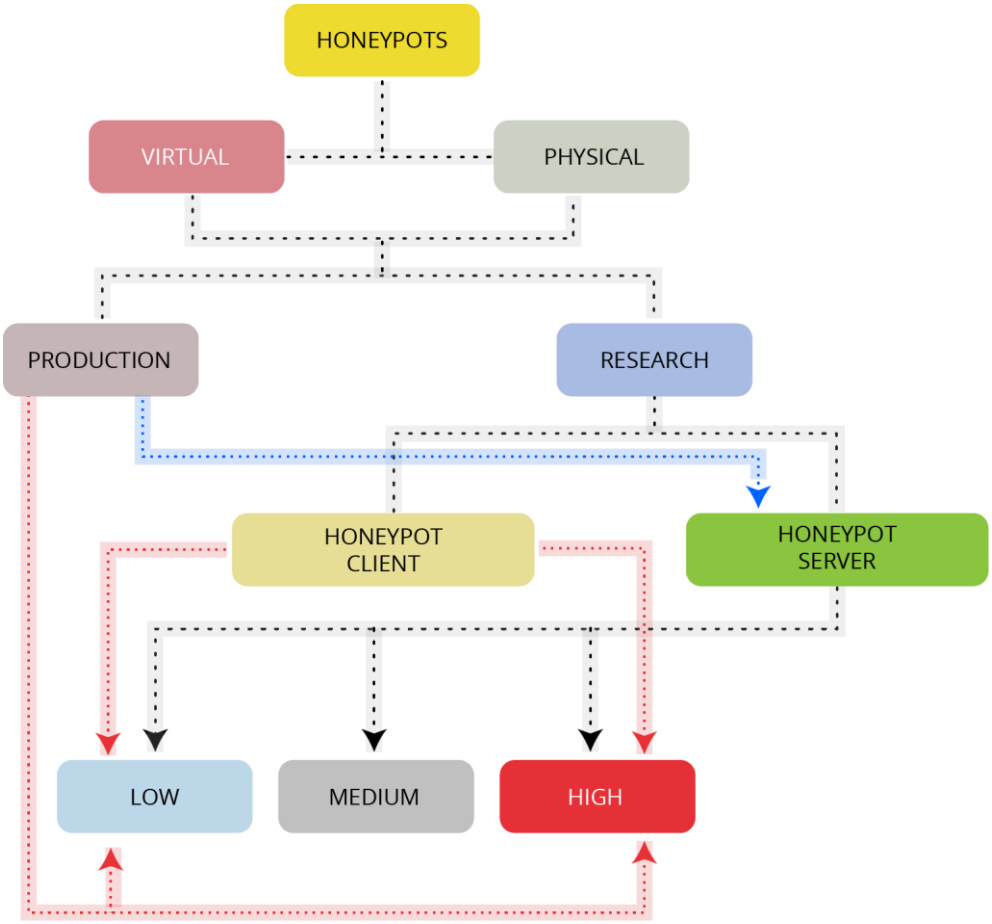


Figure 2 Classification of honeypots

5.1. Type of interaction

According to the degree of complexity that the honeypots have when it comes to the assembly and the interaction, they let the attacker have, they may be classified as high or low interaction.

5.1.1. High interaction

Normally, high interaction honeypots are fully operational systems with applications installed. With a view to information extraction, this type of system will provide more information than a low interaction system on the actions, techniques and tools users use within the honeypot.

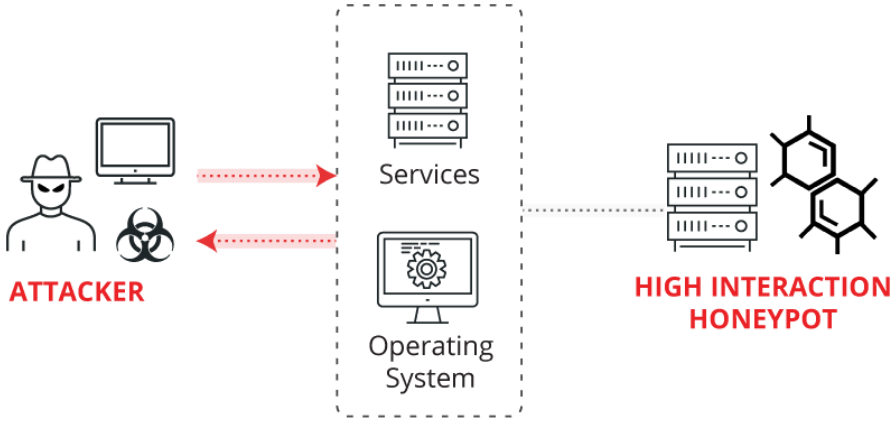


Figure 3 High interaction honeypot

Care should be taken with the configuration of this type of honeypot since, in the case of not having it properly configured and controlled, attackers could use it as an access point to the rest of the network systems. Therefore, it should always be isolated so that no systems are compromised.

5.1.2. Low interaction

Low interaction systems are typically user-friendly systems. They have a fast operationalisation, given that once it is installed and configured, the only thing to do is start it up. However, the applications and services they simulate are not fully functional, given that they have partial, often repetitive implementations, making them easily identifiable by expert attackers, so they tend to be more limited when it comes to information gathering.

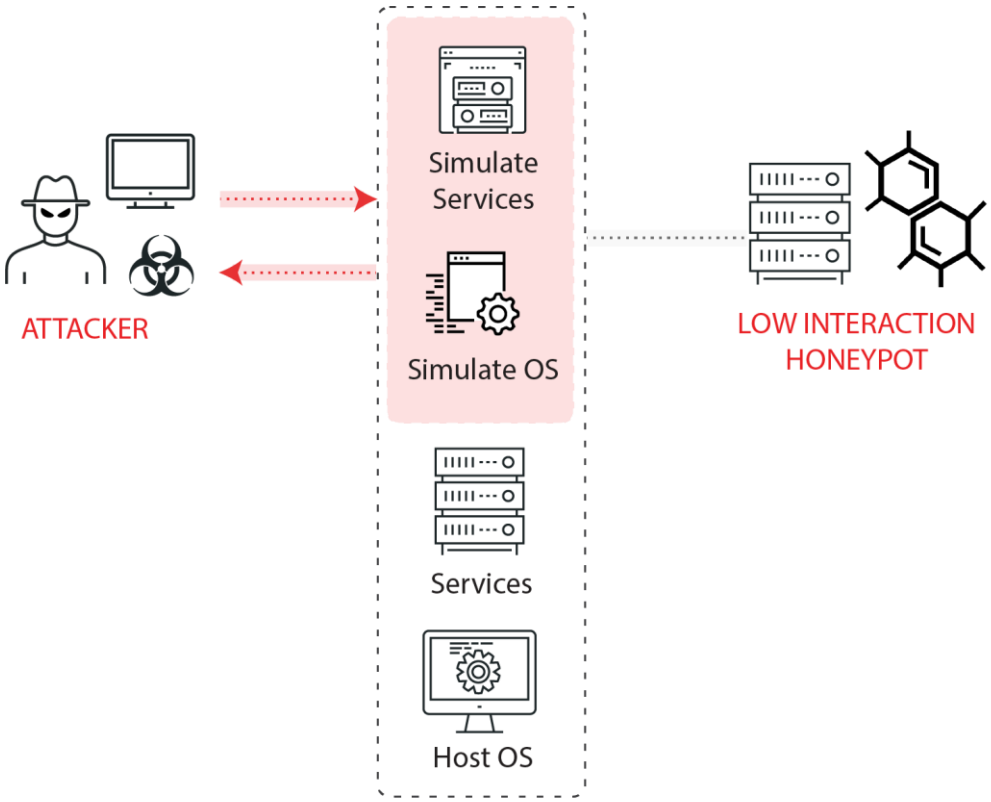


Figure 4 Low interaction honeypot

	High interaction	Low interaction
Simulation	They use real services, applications or devices. Their identification is often complex.	They simulate services or operating systems. They have a good chance of being easily detected as a trap.
Objectives	To discover new attacks or previously undetected abnormal behaviours.	To discover already known automated or vulnerability assessment tools in services.
Information	They capture a large amount of very valuable information as they occasionally contain records of unknown attacks. Their implementation is perfect for in-depth analysis and research.	The amount of collected resources is limited. Not the best option if you want to carry out an in-depth analysis of the system.
Risks	They are exposed to a greater risk against attackers, since they can have the entire network at their disposal if the honeypot is not properly isolated.	They are exposed to a lower risk against attackers because typically everything is virtualised.

Table 1 Comparison of low and high interaction honeypots

5.2. Type of equipment

Depending on the devices to be used for building a honeypot, the network may be made up of virtual or physical systems. Therefore, in order to build either a high or low interaction honeypot, it can be done by using or combining different types of equipment.

5.2.1. Physical

A physical honeypot involves a real computer ready to be attacked from the outside. As it is a physical computer, its functionality is the standard offered by computers, without any restrictions. This makes it more difficult for the attacker to identify the honeypot, but limits the number of attacks because they are unable to select the desired available services.

The price of a physical honeypot should be taken into account, given the need to have hardware, not only software. Furthermore, it involves much more maintenance.

The vast majority of the time, high interaction systems are physical systems.

5.2.2. Virtual

A virtual honeypot is basically a real system but run on a virtual machine. The virtualisation offers the benefit that simulating different types of devices on the same physical equipment is possible. The number and type of services offered will depend on the implementation undertaken by the virtual honeypot developer, as well as the type of interaction.

The maintenance of a virtual honeypot will depend on the number of services offered and the interaction exhibited. If full operational services are available, with a high interaction honeypot, maintenance will be similar to that of a physical computer.

As a general rule, virtualised honeypots are usually low interaction.

	Virtual Honeypot	Physical Honeypot
Advantages	Easily deploy a large number of honeypots. Scalability and easy maintenance. Inexpensive. Quick and simple implementation. Ideal for low interaction honeypots.	Real machine on the network. More realistic, more difficult to identify as a honeypot Ideal for high interaction honeypots.
Disadvantages	More easily detected by attackers. Difficulty simulating complex or broad systems. Collect less information.	Expensive. Not practical for places with numerous IP addresses.

Table 2 Comparison of physical and virtual honeypots

5.3. Type of behaviour

Honeypots usually only implement one type of behaviour, because while they could present different ones, this could make them more easily detectable.

Depending on the mechanisms implemented in the honeypot to behave against different threats, they may be classified and help prevent attacks in different ways.

5.3.1. Precise detection methods

The advantage of honeypots for precise detection is that there are fewer false positives, making it possible to capture essential information on new techniques or tools for exploiting vulnerabilities and being able to work with communications that are encrypted and under IPv6 networks.

5.3.2. Intrusion protection

The idea of this countermeasure is to try and confuse attackers, with the goal of wasting time and taking advantage of this in order to detect the attacker's activities, to take the necessary countermeasures to block it so that the attacker never reaches the target.

5.3.3. Disable actions

During this process, the attacker's actions are allowed, but are ultimately disabled so that they cannot take advantage of such vulnerabilities. The attacker is able to reach their target, but the requests are manipulated so that they fail.

5.3.4. Deceleration or defence against automated attacks

With this type of honeypot, the attacker is slowed down in all of their malicious actions. If an attacker encounters a vulnerability, their attack techniques will be slowed-down by the solution. This is achieved through modifications or alterations in the network packages, modifying parameters such as "Window Size" by setting it to zero or leaving the attacker on hold.

This type of technique is perfect for preventing the speed of worm spreading that may have affected the honeypot or the network.

5.3.5. None

In this case, the honeypot does not undertake any action or countermeasure related to the attacker's actions, therefore there is no limitation of the scope or damages that may be caused.

5.4. Type of role

Currently, honeypots can be distinguished according to the type of role they play. The roles that may be played are server, that is, the honeypot receives the attacks, or client, in which it makes requests against servers or malicious applications.

5.4.1. Client

A client honeypot serves to imitate a software that uses the services on a server. One of the most classic examples is having a browser that visits different web pages with the objective of having those attacked by taking advantage of some vulnerability. It is based on going to websites and gathering information on the attacks and security risks.



Figure 5 Client honeypot

5.4.2. Server

The operation of server honeypot consists of luring attackers to a secure or isolated environment in order to be able to conduct research studies or remove possible attackers from the real network.

It is based on the simulation of an environment as realistic and credible as possible, so it will be more difficult for attackers to differentiate this type of network from a real one. Typically, to attract their attention, applications, services or industrial automation devices are simulated, trying to capture the cyberattacker's attention.

When an attacker falls into the trap, all the actions, tools and techniques they use to achieve their objectives are recorded, allowing administrators or researchers to obtain new information, which allows for better protection and knowledge against future attackers.



Figure 6 Server honeypot

6. Tools

Honeypots have evolved from their creation until now, and they will continue to do so going forward, using new tools and techniques. Although their main mission has not changed, the type of information sought and the way to collect it has.

Currently, it is possible to use a wide range of tools, including HoneyMonkey, BeEF and HoneyBadger, aimed at extracting information from attackers, or Spider trap, Weblabyrinth and Nova for slowing down the attacker through deception and traps.

6.1. HoneyMonkey

A HoneyMonkey is a special kind of tool created by Microsoft Research that, using a computer network or virtual machines, is able to receive and analyse attacks by visiting suspicious websites. Its main mission is to detect new attack types or patterns and formulas for infection that take advantage of the browsers' vulnerabilities.

Some of the main characteristics of this projects are the following:

- It is made up of an "exploration" module and another for data collection.
  - Active exploration consists of automatically exploring, by means of agents, a list of web pages, gathering information on possible attacks.
- Operates like an automatic browsing system.
  - It visits all types of web pages so that one of them tries to take advantage of vulnerabilities in the browser.
  - It is a type of client honeypot.
- Browsers may be configured.
  - To stay up to date with the latest updates or run without any specific update, so that a website exploits or takes advantage of any vulnerability.
- HoneyMonkey operation is based in snapshots.
  - A snapshot of the records, runtimes and memory is taken before visiting the pages.
  - After visiting the pages with malware, another snapshot is taken and both are compared in order to see the effects it has created and which vulnerability it has exploited.

HoneyMonkey runs on Windows XP with various levels of updates or patches, some of them with only patches, others with some vulnerabilities and finally those without any patch or update.

Once the machines and browsers are configured, the collection and analysis, made up of the following phases, must be carried out:

- Crawling phase. A list is made ahead of time with the web pages classified as potentially dangerous, in other words, those websites from which they will try to take advantage of vulnerabilities in our browser to exploit them. The system increases the size of the list by adding the external links found on each website, because it is likely that these are also malicious.

- Analysis of exploits. HoneyMonkey uses the black box system for detecting vulnerabilities. Each virtual machine runs Internet Explorer in order to visit each page on the list. During the process, all the data from the record, read and write operations are noted.
- Detection of malware. All changes made to files outside the browser's temporary folder will have exploited some browser vulnerability because they are not accepted, nor do they allow pop-ups or software installation. The files are analysed by a malware detection programme and manually reviewed. Finally, the machine is restarted to return to its initial state and continue with the rest of the pages.

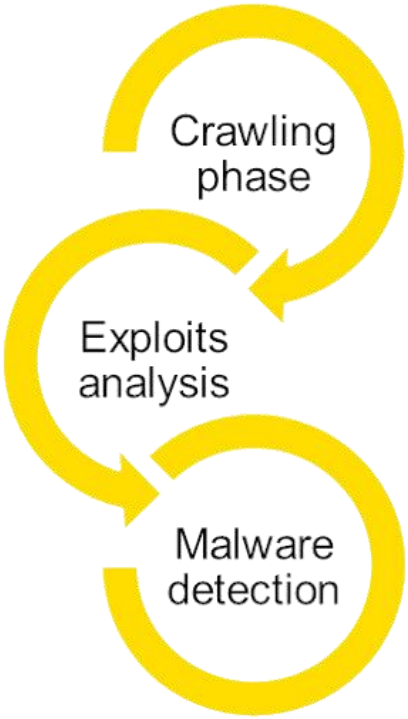


Figure 7 HoneyMonkey phases of implementation

6.2. BeEF

BeEF (Browser Exploitation Framework) is a tool for penetration testing that focuses on the web browser, its operation and data collection, and is based on running scripts on the attacker's computer. During the attack, it is embedded within a secure web page and gathers useful security information for analysts. Some of the problems encountered are due to the difficulty in deployment and development, and in the attackers' experience when it comes to discovering honeypots.



Figure 8 BeEF logo

While BeEF is active and operating it extracts information from the browser. The information it can obtain is:

- Name and version of the browser.
- The plug-ins used (Java, ActiveX, VBS, Flash, etc.).
- Window size.
- The User Agent (UA).

In addition to this data, thanks to the use of a series of plugins, more details about the attacker's machine can be extracted. Included in this information is:

- Details of the Java Virtual Machine (JVM).
- Quantity of memory.
- Details of the operating system.
- Number of processors.
- Network card type (NIC).
- IP Addresses.
- Use of TOR.
- Use of social media.
- Geolocation.

6.3. HoneyBadger

HoneyBadger is a framework used for aiding in the detection and geolocation of attackers. Its use, combined with the Molehunt tool, allows the user to create documents that, when opened, report thereon, helping to identify and geolocate the attacker.



Figure 9 HoneyBadger logo

Its operation is based in offering attackers the administrative functions that they wish to control. According to its creators, it can work in the form of ActiveX or Java applets, making the attacker believe, once executed, that he has managed to breach the website.

HoneyBadger uses TCP flow analysis to detect and identify attacks, combining a variety of TCP injection attacks in order to ensure that it is a real attack, thus avoiding false positives.

The result is obtained in the geolocation of the attacker with an approximately 20 metre margin of error. Its operation is similar to the technology used by smartphones for geolocation, by means of triangulation-based positioning.

6.4. Spidertrap

A Spidertrap is a honeypot aimed at decelerating and hindering the task of the crawlers used for indexing web pages.

Its operation is based on the creation of nested loops or web pages so that the crawler service is caught and cannot continue with its task, and may even fail if it is not correctly built.

6.5. Weblabyrinth

Just like the previous tool, this is also targeted at a web environment and not at creating a complete and operational physical honeypot.

Weblabyrinth is designed for creating, as its name suggests, a labyrinth of web pages in order to try and confuse crawlers. The majority of the pages are false, so that they delay the tasks of possible malicious crawlers in their search for information about the web.

6.6. Network Obfuscation and Virtualized Anti-Reconnaissance

This tool, commonly known by the acronym NOVA, makes it possible to create a honeynet by using an improved version of honeyd in order to create the honeypot. Its operation is based in basic rules of source IP, destination IP, port, package size, etc.

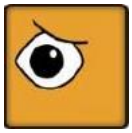


Figure 10 NOVA logo

It has machine learning that permits the activation of alerts when it detects suspicious activity or abnormal attempts to access. To slow down attacks it provides false data to attackers, protecting the internal systems.

By means of a very user-friendly web panel it enables the configuration and review of information from the generated honeypots.

7. Projects

Over the years, projects have been emerging regarding industrial honeypots for the protection and analysis of attackers. Some of the most stand-out and popular ones are The Honeynet Project, Conpot/GasPot and Honeyd, which we will discuss below.

7.1. The Honeynet Project

The project, created in 1999, belongs to an organisation dedicated to researching the most recent attacks, designing open source tools to improve Internet security and learning how attackers or cybercriminals act.



Figure 11 The Honeynet Project logo

Its main objectives and targets are:

- Provide tools and techniques used by "The Honeynet Project" so that other organisations may benefit from its use. Some of the tools are Cuckoo, Capture-HPC, Glastopf, HoneyC, Honeyd and Honeywall.
- Raise awareness about all the existing dangers and threats on the Internet.
- Carry out data analysis research, develop unique security tools and collect and analyse information on attackers and the malicious software they use.

7.2. Honeynet SCADA by DigitalBond

Honeynet SCADA is a project whose main objective is to build a feasible software for simulating a variety of industrial devices, such as PLC, SCADA or DCS architectures.

Through a Linux system multiple devices and industrial networks can be simulated. The Honeyd project is also used for its implementation, making it possible to create different virtual machines and simulate their services as an operating system.

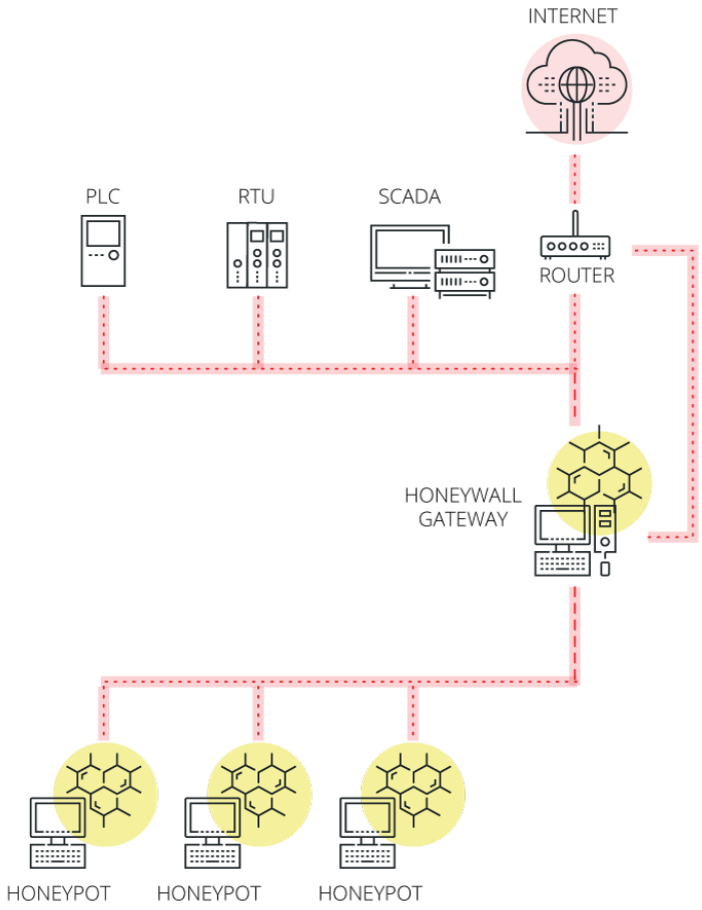


Figure 12 Architecture proposal for an industrial honeynet

The Honeynet SCADA project is made up of some of the following components:

- Monitoring
  - Use of a 3rd generation Honeywall
  - Quickdraw IDS Signatures
- Objective
  - Creation of virtual PLC or use of a physical one
- Available services
  - HTTP: Portion of a PLC Schneider web-based management
  - FTP: Portion of FTP service for administrative purposes
  - Telnet: Low-interaction Telnet Service
  - Modbus TCP: Realistic points programmed in the honeynet. Its values change somewhat randomly
  - SNMP: Implemented SNMP service, uses information from the PLC
  - VxWorks Debugger: Simulates the service listening to the UDP/17185 port

7.3. Conpot and GasPot

Conpot is a honeypot that was designed based on low interaction industrial control systems. Among its advantages is that it is easy to deploy, modify and expand. Its easy use and the ability to simulate a wide range of protocols and systems makes it able to implement almost

any system. The project is currently included in the “The Honeynet Project”. By default, it simulates a Siemens SIMATIC S7-200 PLC and Modbus, SNMP and HTTP protocols.

On the other hand, GasPot has been designed to simulate a liquid tank’s meter, corresponding to the Veeder Root Guardian AST model, a device designed for the control and compliance of inventory in storage tanks. Its main functions include monitoring pump levels, pumping systems and tank inventory.

Oil industry companies are those that mostly use this type of meter, since they serve to measure the tanks’ fuel levels. It is an open source project that can be implemented and installed by any user, by downloading it from its repository on GitHub.[3]

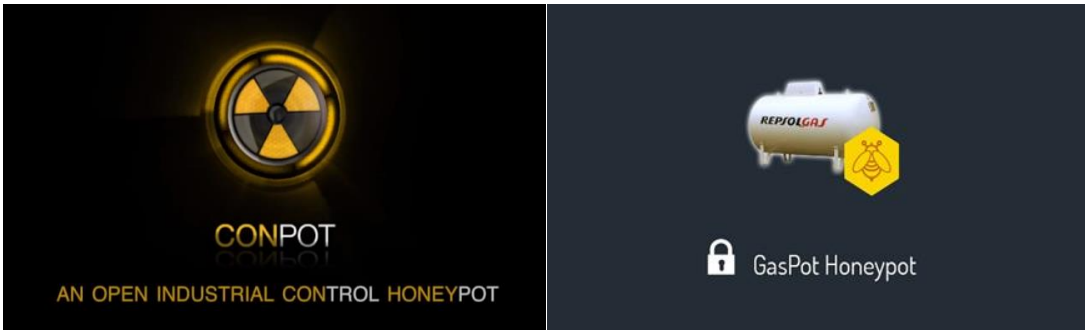


Figure 13 Conpot and GasPot logos

7.4. Honeyd

Honeyd involves software created by Niels Provos that enables the creation of multiple honeypots and their virtual execution on the network. These machines may be configured to simulate different types of components, operating systems or services in order to extract useful information about attackers.

The two main objectives of this software are to distract attackers and its function as a honeypot. The goal of the distraction is to keep the attackers focused on it, delaying and slowing down their actions. The use of honeypot is geared more towards studying and researching the forms of attack used.

In section 8. Deployment of an industrial honeypot, shows how to build a honeypot using this software.

[3] https://github.com/sjhilt/GasPot

8. Deployment of an industrial honeypot

This section contains precise, step-by-step instructions on how to deploy an industrial honeypot. For developing this honeypot, the Honeyd tool will be used, which enables the creation of a virtual machine, with the most convenient network services and features, and the scripts developed by SCADA HoneyNet Project, since they include services to emulate a Siemens or Schneider Electric PLC.

The Siemens Simatic S7-300 device has been chosen for this guide, as it is one of the most widely used in the industrial sector.

8.1. Tools used

Below, the tools used for configuring the honeypot are listed and described.

Software tool	Specifications
Base operating system	Ubuntu Desktop 19.04 - 64 bits[4]
Honeyd	1.6d[5]
Python	2.7.16
Bash	5.0.3
Perl	5
Scripts SCADA HoneyNet Project[6]	honeyd-ftp-siemens.py honeyd-http-siemens.py honeyd-telnet-siemens.py honeyd-s7.py honeyd-snmp-siemens.py honeyd-modbus.py

Table 3 List of used tools

8.2. Honeypot design

This guide has chosen to install a 64-bit Ubuntu Desktop 19.04 base operating system, in which the honeyd tool will be installed to subsequently be able to deploy the SCADA HoneyNet Project scripts.

The scripts that will be used for simulating the services in the honeypot are the following:

- **honeyd-ftp-siemens.py:** FTP in the port TCP 21. Will simulate the FTP server of the CP 343-1 IT model. It is not possible to log-in in the simulation.

[4] https://ubuntu.com/download/desktop

[5] https://github.com/DataSoft/Honeyd

[6] https://sourceforge.net/projects/scadahoneynet/

- **honeyd-telnet-siemens.py:** Telnet in the port TCP 23. It is also not possible to log-in in the simulation.
- **honeyd-s7.py:** S7 in the port 102. It will have a basic simulation of the S7 internal server (Siemens control protocol) of the CP 343-1 IT.
- **honeyd-http-siemens.py:** HTTP in the port TCP 80. It will have a basic simulation of the CP 343-1 IT's frontend.
- **honeyd-snmp-siemens.py:** SNMP in the port UDP 161.
- **honeyd-modbus.py:** MODBUS in the port 502. It will have a basic simulation of a Modbus server, with a response for numerous package and request codes.

8.3. System hardening of the host OS

Because this machine will be exposed to potential attacks, the base operating system should be hardened so that it is not vulnerable, and all attention is turned to the services simulated by Honeyd and not those of the Linux itself. Therefore, it starts from an already hardened system with the following adopted security measures:

- Firewall configuration by means of iptables.
- Changing and strengthening the user's password.
- Disabling the root user.
- Disabling unnecessary services.
- System updates.
- Strengthening access by means of SSH.

8.4. Honeyd Installation

Honeyd installation will be done by means of downloading the sources, installing the units and compiling the programme.

8.4.1. Honeyd download

The latest available version of Honeyd is found in the GitHub repository[7]. The easiest way to download it is as follows:

```
sudo apt-get install git
git clone https://github.com/DataSoft/Honeyd
```

Figure 14 Git installation and Honeyd download commands

8.4.2. Installation of dependencies

Honeyd depends on the following libraries:

- **libevent:** API that provides a mechanism for running a callback function when a specific event occurs in a file descriptor or after a timeout has been reached.
- **libdumbnet:** provides a simplified and portable interface for numerous low-level network routines.
- **libpcap:** involves a package capture library by command line.
- **libpcre:** C library of regular expressions inspired by Perl.
- **libedit:** software that enables the edition of files in command lines.

[7] https://github.com/DataSoft/Honeyd

- **bison and flex:** generators of general-purpose syntactic analysers.
- **zlib:** software library used for data compression.
- **python:** programming language needed for running the SCADA HoneyNet project scripts.

The command to run in order to complete the installation of the same will be the following:

```
sudo apt-get install libevent-dev libdumbnet-dev libpcap-dev libpcre3-dev
libedit-dev bison flex libtool automake zlib1g-dev python
```

Figure 15 Command for installing dependencies

8.4.3. Honeyd compilation and installation

Once the previous steps are completed, the programme will be then compiled and installed. Inside the previously downloaded **Honeyd** directory:

```
./autogen.sh
./configure
make
sudo make install
```

Figure 16 Commands for the installation of Honeyd

In order to have the results and configurations in one place, a folder should be created that includes the SCADA Honeynet Project scripts that will subsequently be downloaded, and the honeyd configuration and log file honeyd.conf and honeyd.log.

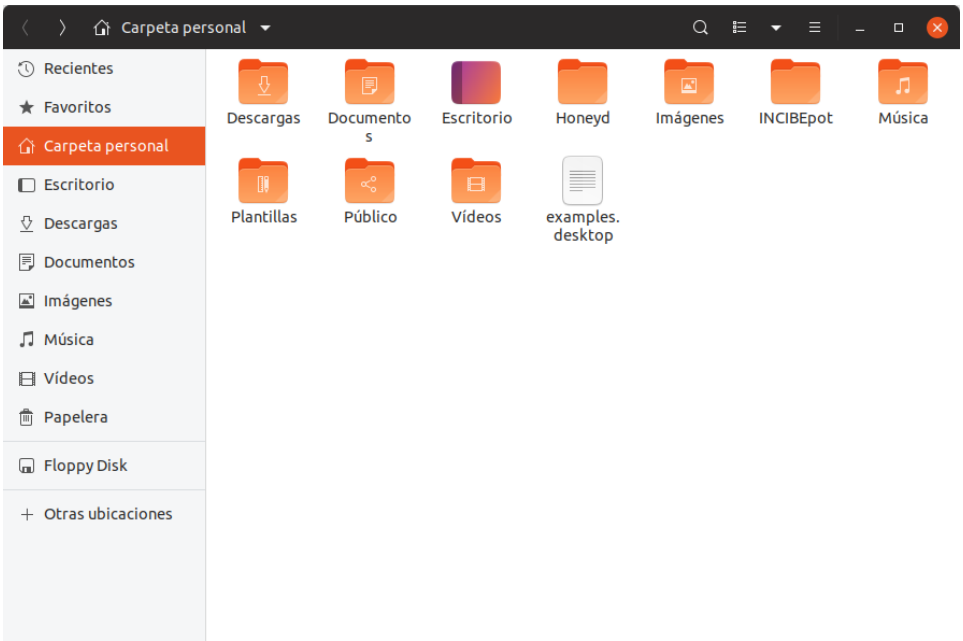


Figure 17 Creation of the folder

8.5. Honeyd configuration

For a proper configuration of the Honeyd, it will be necessary to comprehend the files located in **/usr/share/honeyd**, including:

- **config.sample:** file that contains an example of the Honeyd configuration.
- **nmap.assoc:** list of fingerprints that may be detected by **nmap**.
- **nmap-os-db:** fingerprints database from **nmap**.
- **nmap-mac-prefixes:** list of MAC identifiers from each manufacturer. They will be used to provide more authenticity to scans by MAC. In this case, they have used those from the Siemens automation division. In order to emulate another device, another prefix may be selected.

The objective of such files is to give a response as realistic as possible from the simulated OS when scans are being done by means of nmap or another similar tool.

On the other hand, the simulation of services is done by means of running scripts as a response to the receipt of a request from a Honeyd open port. This tool has a wide range of scripts that are found in the directory **/usr/share/honeyd/scripts.** Some of these are in **bash** or **perl** format, therefore it would be necessary to have such interpreters installed in order to run them. With this, a wide variety of IT machines, such as Linux, Windows or another embedded version may be simulated.

However, Honeyd does not have scripts that simulate the services of the SCI itself, therefore scripts have been used that are programmed in Python from the SCADA Honeynet Project[8]*. There are scripts for services from two PLC from different brands such as, Siemens and Schneider Electric, for example.*

In order to make the work easier, it is recommended to move the entire folder **/cernscadahoneynet/files/scripts,** located at the root of the SCADA HoneyNet Project, to the folder that was previously created.

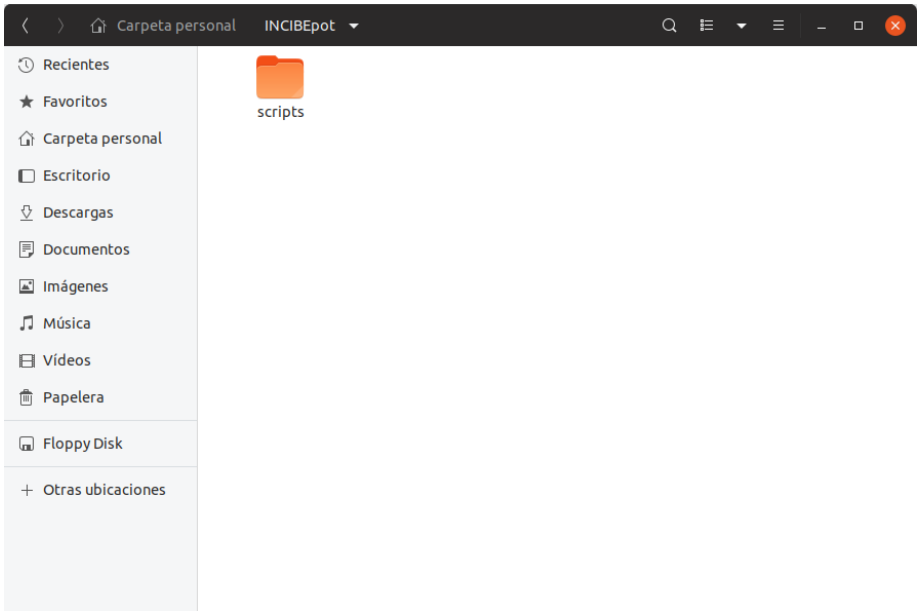


Figure 18 State of the folder at this point

Prior to starting up the honeypot, two modifications must be made:

[8] http://scadahoneynet.sourceforge.net/

- Open the file **honeyd-http-siemens.py** and edit the **webroot** variable, including the absolute path where the folder **web-siemens** is found, next to the edited file.

```
webroot = "/home/incibepot/INCIBEpot/scripts/web-siemens"
```

Figure 19 Edited line in the file "honeyd-http-siemens.py"

In the case of wanting to simulate the Schneider website, changing the "web-siemens" directory to "web-schneider" is enough. To show any other web page, the web.sh script could be used, located in the folder **/scripts/backdoor** within the honeyd installation.

- Since there is no script that simulates Telnet in a Siemens PLC, duplicate the file **honeyd-telnet-schneider.py** and rename the copy **honeyd-telnet-siemens.py**. It is necessary to modify the file so that it indicates that it is a Siemens, not Schneider Electric, computer.

```
logintext = "\n\rSiemens Login: "
```

Figure 20 Edited line in the file "honeyd-telnet-siemens.py"

- If it is not already included, extract the full name of the OS to be simulated from the **nmap-os-db** file (the full name immediately after the desired product **Fingerprint**, in our case SIMATIC 300 PLC), which is located in the path **/usr/share/honeyd/**, and add it, editing it as root, at the end of the **nmap.assoc** list, followed by a semicolon (it is in the same directory as the previous file). If it is already on the list, just make sure that line is uncommented. In this case, the OS to simulate will be **Siemens Simatic 300 programmable logic controller.**

```
# SIMATIC 300 PLC
Fingerprint Siemens Simatic 300 programmable logic controller
Class Siemens | embedded || specialized
```

Figure 21 Complete name of the OS obtained from nmap-os-db

```
#ZyXEL ZyWALL 2 Plus firewall;
#ZyXEL ZyWALL 5 firewall;
Siemens Simatic 300 programmable logic controller;
```

Figure 22 Name of the OS added in the last line of nmap.assoc

Subsequently, the configuration file of the honeypot must be created, which will include all the parameters of the machine to be simulated, such as open ports, services that are emulated in them, network information, such as MAC or IP address, operating system name, among others. In this case, it will be called **honeyd.conf** and will be located within the previously created folder.

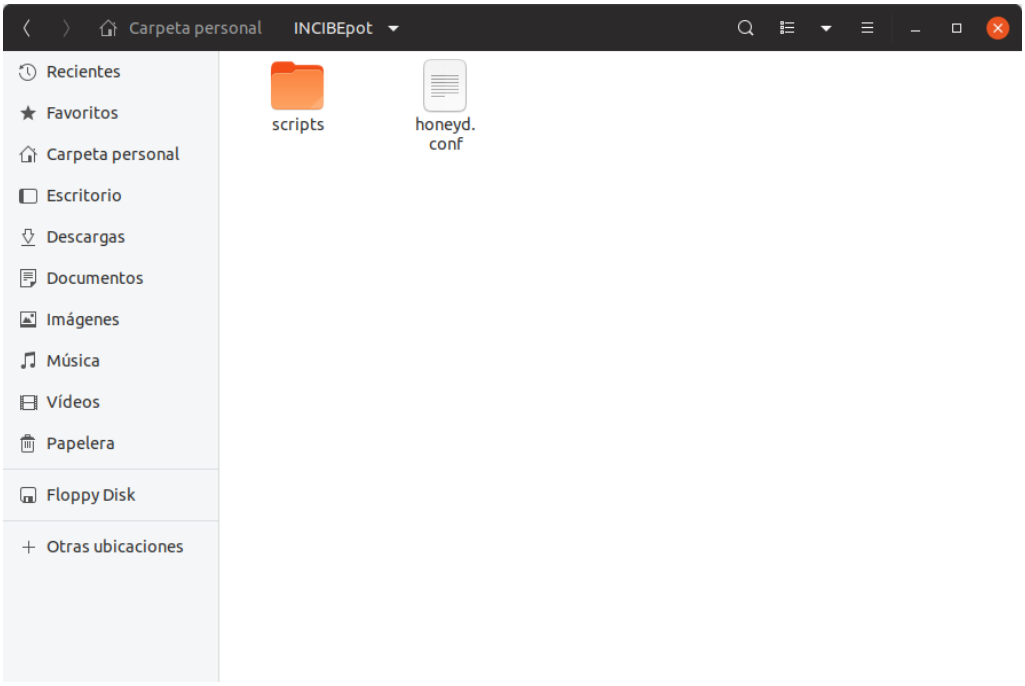


Figure 23 Previously created folder, with the configuration file

The following content should be entered into the file:

```
create siemens
set siemens ethernet "00:1f:f8:cc:d0:23" # Siemens Automation MAC ID
set siemens default tcp action closed
set siemens default udp action reset
set siemens personality "Siemens Simatic 300 programmable logic
controller"
add siemens tcp port 21 "python /home/incibepot/INCIBEpot/scripts/honeyd-
ftp-siemens.py"
add siemens tcp port 23 "python /home/incibepot/INCIBEpot/scripts/honeyd-
telnet-siemens.py"
add siemens tcp port 80 "python /home/incibepot/INCIBEpot/scripts/honeyd-
http-siemens.py"
add siemens tcp port 102 "python
/home/incibepot/INCIBEpot/scripts/honeyd-s7.py"
add siemens udp port 161 "python
/home/incibepot/INCIBEpot/scripts/honeyd-snmp-siemens.py"
add siemens tcp port 502 "python
/home/incibepot/INCIBEpot/scripts/honeyd-modbus.py"
set siemens uptime 4532786 # 52 days on.
bind <IP_honeypot> siemens # In our use case, the IP is 192.168.252.130
```

Figure 24 Contents of the configuration file honeyd.conf

Where each command and parameter entered means:

- **create <name>:** defines a template for a simulated OS.
- **set <template> ethernet "MAC":** defines the MAC of the honeypot. The list of MACs corresponding to each manufacturer can be found in the aforementioned **nmap-mac-prefixes** file.

- **set <template> personality "<name>":** defines the name of the OS to be emulated, which will be checked against the **nmap** and **nmap.assoc** database to respond to the **nmap** or **xprobe2** OS scan. It should be identical to that included in the **nmap.assoc file** in previous steps.
- **set default <portType> action <actionName>:** indicates to the Honeyd how to treat the undefined connections in the specific ports.
- **add <template> <portType> port <#> "<script>":** the opening and assignment of specific ports is carried out, specifying the path to a script that will be run by listening to that port, to emulate the service in question.
- **set <template> uptime <timestamp>:** artificially establishes the time that has passed since the device was turned on. In some operating systems it can be extracted from the TCP **timestamp**, so, by this method, we can find out how long it has been on.
- **bind <ipaddress> <template>:** assigns an IP address to a simulated OS. Such IP has to be on the same network as the host interface that will listen to the *honeyd.* For the preparation of this guide, the IP 192.168.252.130 was used.

Once we have moved the scripts and completed the configuration file, the honeypot will be deployed. To do so, the following command will be run at the root of the folder created at the beginning, for example:

```
sudo honeyd -d -p nmap-os-db -i ens38 -l honeyd.log -f honeyd.conf IP -u
0 -g 0 --disable-webserver
```

Figure 25 Command and parameters for running the Honeyd virtual machine

Where:

- **-d:** run Honeyd without demonising and with debug messages.
- **-p file:** allows Honeyd to read **nmap** fingerprints contained in that file.
- **-i interface:** specifies the host network interface that will occupy the honeypot.
- **-l logfile:** specifies the log file. In this way, there will be two logs: one of **iptables** and one of Honeyd.
- **-f config:** specifies the specific configuration file that Honeyd will run.
- **IP:** IP address of the emulated virtual machine or subnet which contains several emulated machines (in the case of more than one). This will be the network the Honeyd will emulate. Must be the same network as the host interface specified in the command with the -i argument.
- **-u:** establishes the UID with which the Honeyd is operating.
- **-g:** establishes the GID with which the Honeyd is operating.
- **--disable-webserver:** disables the web server that honeyd has configured by default.

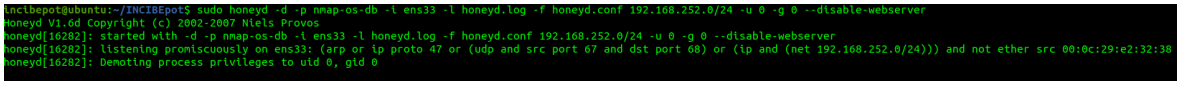


Figure 26 Result of running the honeyd command

A cheat sheet has been prepared with all of the commands summarised on page 43.

8.6. Honeypot maintenance

A honeypot deployed by means of honeyd involves virtualising a host operating within a machine. Therefore, its maintenance tasks will be similar to those that could be applied to any device with available raised services. Included in these tasks are:

- Verify the operation of the honeypot's services.
- Check that the host machine is active and operating correctly.
- Verify the connectivity of the honeypot with the rest of the network.
- Ensure that the security measures applied to the host are correct.
- Review the logs generated by both honeyd and the firewall of the host on a regular basis.
- Honeyd may be automatically started up by means of a script, which is useful in the event of a failure in the power grid or if the host machine shuts down or reboots.

8.7. Results

In this section, the correct operation of the honeypot will be checked by using a set of tools for the purposes of network analysis and devices, as well as the interpretation of the logs generated by the honeypot.

8.7.1. Tools used

In order to carry out the tests on the machine emulated with Honeyd, several tools will be used to analyse the simulated services and the fidelity of their responses to different requests in their task of pretending to be, in this case, a Siemens PLC. Below, the tools used are listed:

Software tool	Specifications
Zenmap - Linux[9]	7.70
ModbusTool - Windows[10]	1.0
Wireshark - Windows[11]	3.0.5
Snap7[12]	1.4.0

Table 4 List of tools used in the testing phase

8.7.2. Checking services

- **Scanning with nmap**

The programme used for port and services scanning is Zenmap, a version of nmap with a graphic interface. The scan result shows the open ports specified in the honeyd configuration file, as well as information regarding the operating system and the fingerprint obtained.

[9]https://nmap.org/zenmap/

[10]http://www.modbusmaster.com/

[11]https://www.wireshark.org/#download

[12]http://snap7.sourceforge.net/

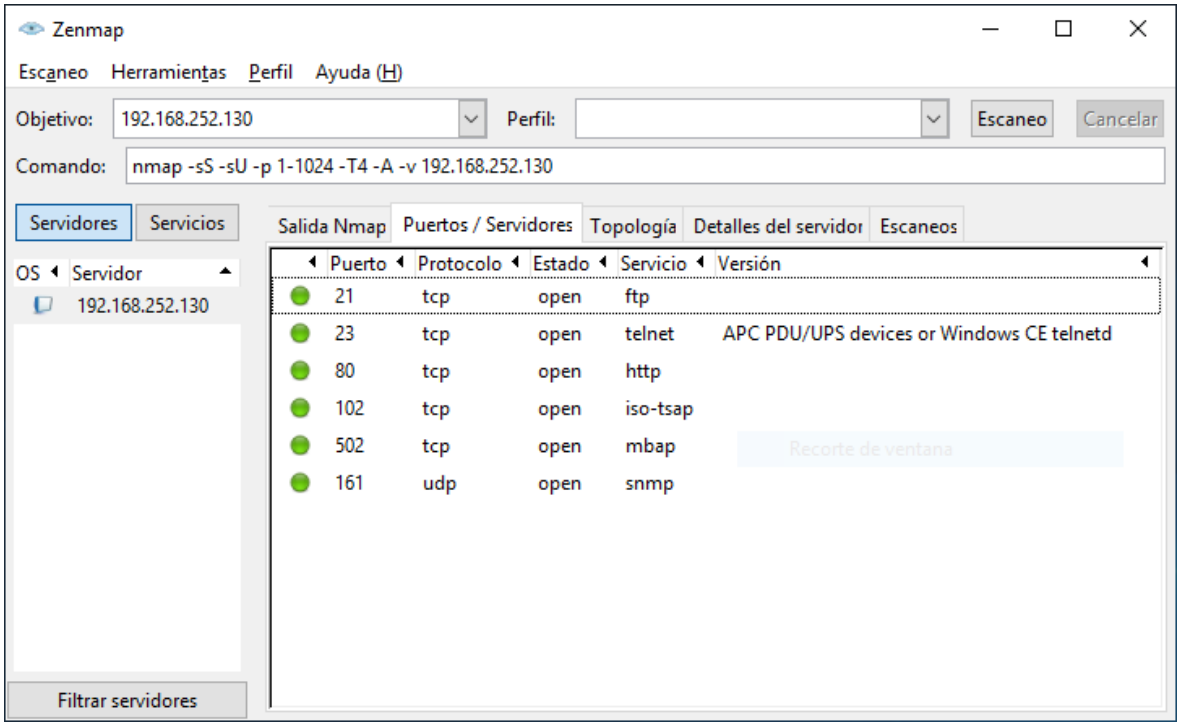


Figure 27 Results of port scanning in the honeypot

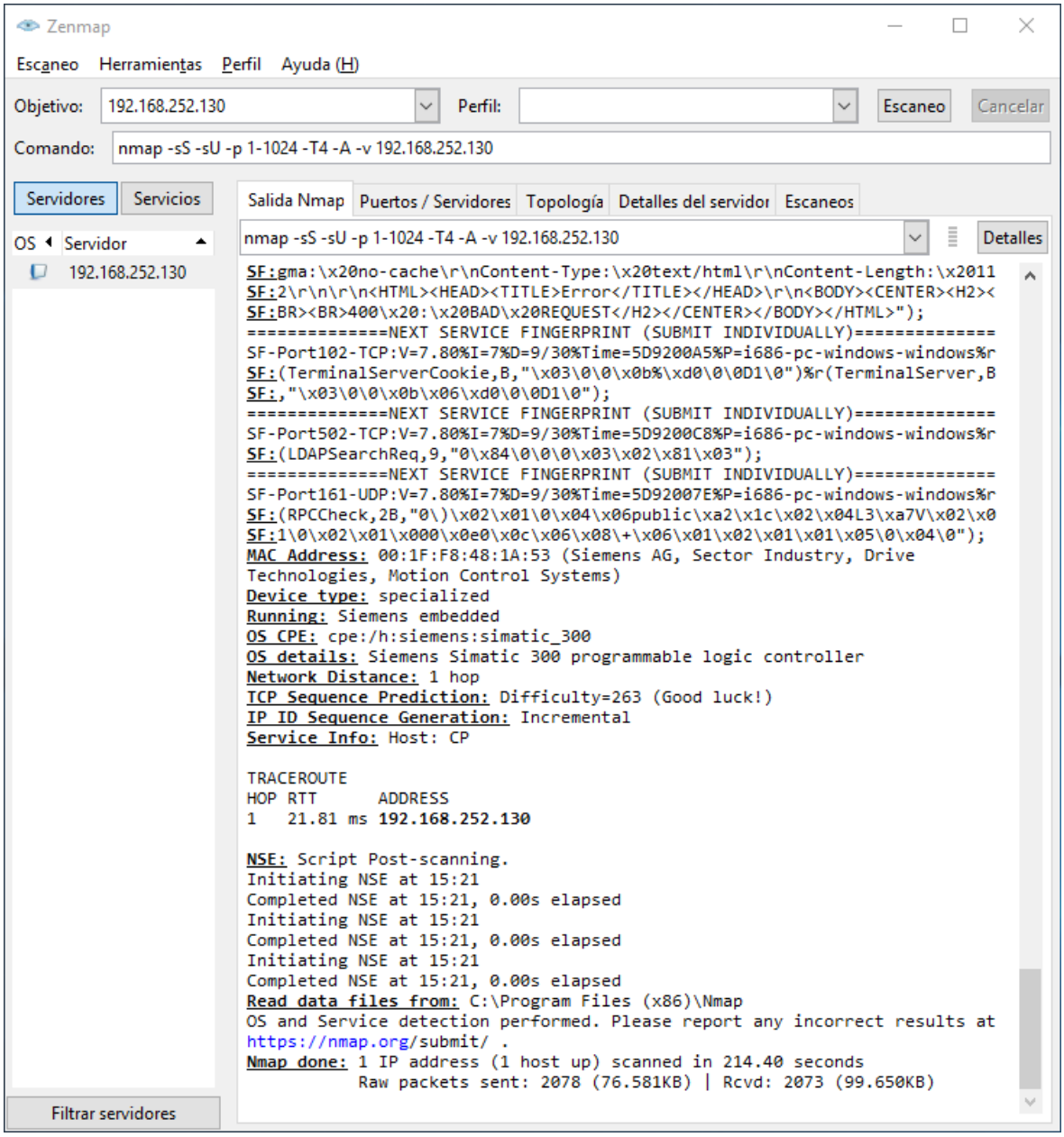


Figure 28 Results of scanning the honeypot OS

- **HTTP**

An HTTP connection was made from a browser on a machine within the network of the same in order to check the operation of the web service. A similar web page to one a Siemens Simatic PLC would display is shown, with which it can interact to a certain level, providing some data of the simulated device.



Figure 29 Web page emulated in the honeypot

- **FTP**

When making an FTP connection against the honeypot, depending on the user's input, the service is expected to display the login message along with its different responses. In this case, the corresponding error codes are obtained, just as a real FTP service would do. It should be noted that it is not possible to login in the simulated service.

```
incibepot@ubuntu:~$ ftp 192.168.252.130
Connected to 192.168.252.130.
220 CP 343-1 IT FTP-Server V1.36 ready for new user
Name (192.168.252.130:incibepot): admin
530 Not logged in.
Login failed.
Remote system type is CP.
ftp> exit
221 Closing control connection; Thank you for using our FTP server.
incibepot@ubuntu:~$
```

Figure 30 Result of the FTP connection to the honeypot

- **Modbus**

To carry out this test ModbusTool for Windows was used, with which Modbus record reading and writing packages can be sent. The honeypot correctly responds to the Modbus TCP messages, just as a PLC with Modbus TCP communication would do.

The following steps will be followed in order to generate Modbus messages:

- Enter the IP address of the Honeypot in the ModbusTool Connection section.

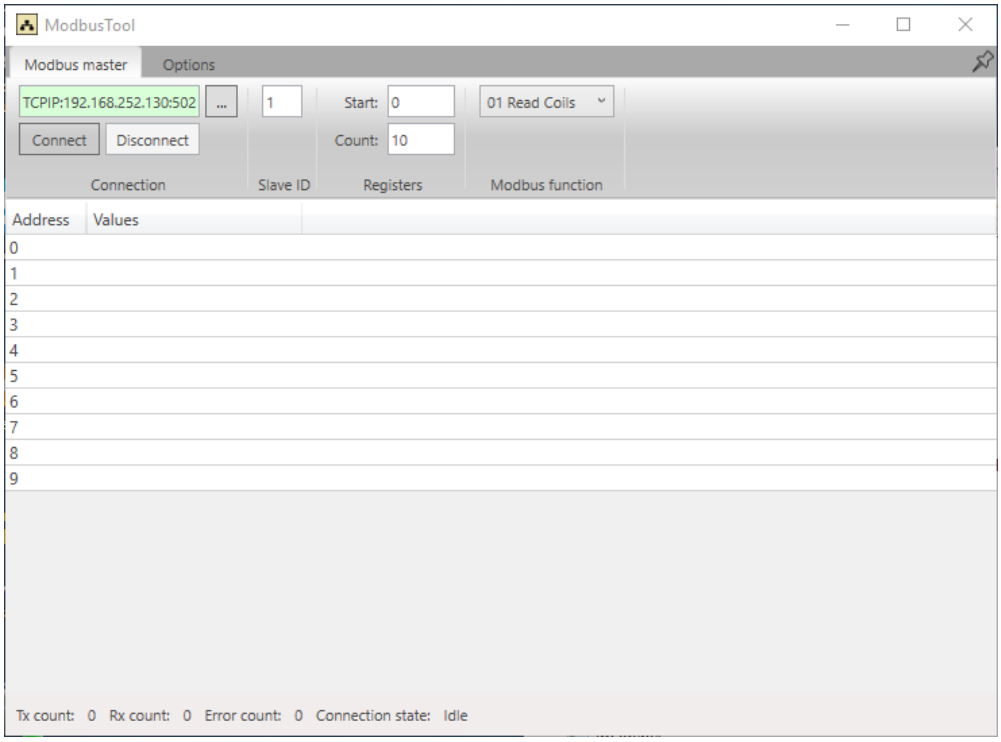


Figure 31 Status ModbusTool in Idle

- Click Connect. The status should change to Connected and that value should be able to seen in the records.

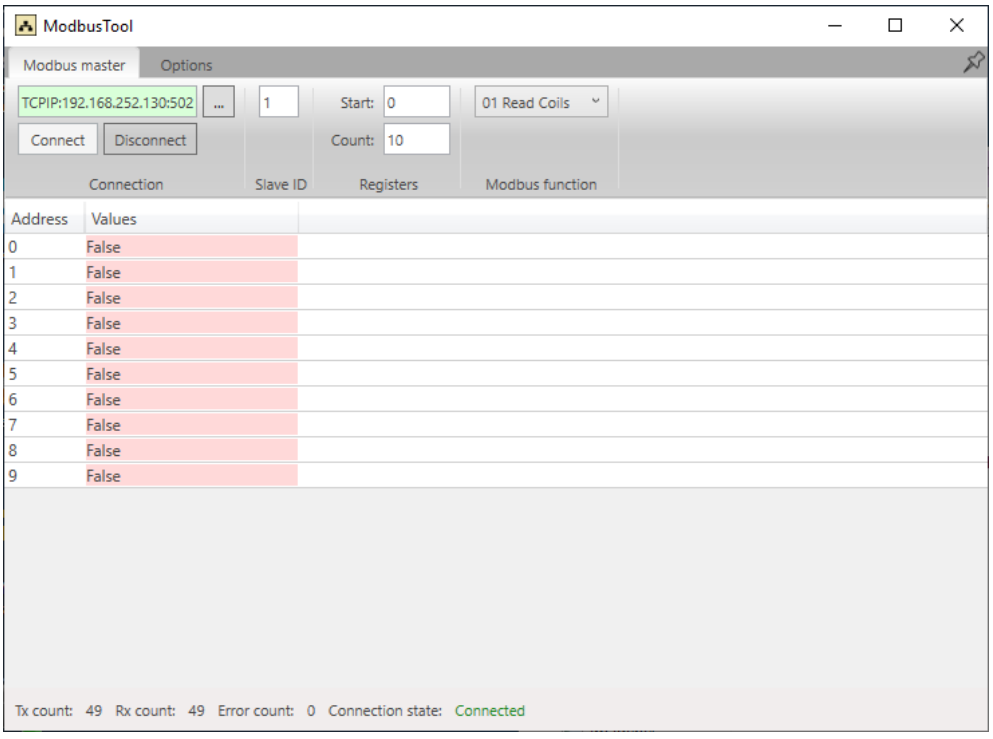


Figure 32 Status ModbusTool in Connected

- The Modbus function that the programme will complete by default is “01 Read Coils”. If you want to change to another one, select the one you want in the Modbus Function tab dropdown.

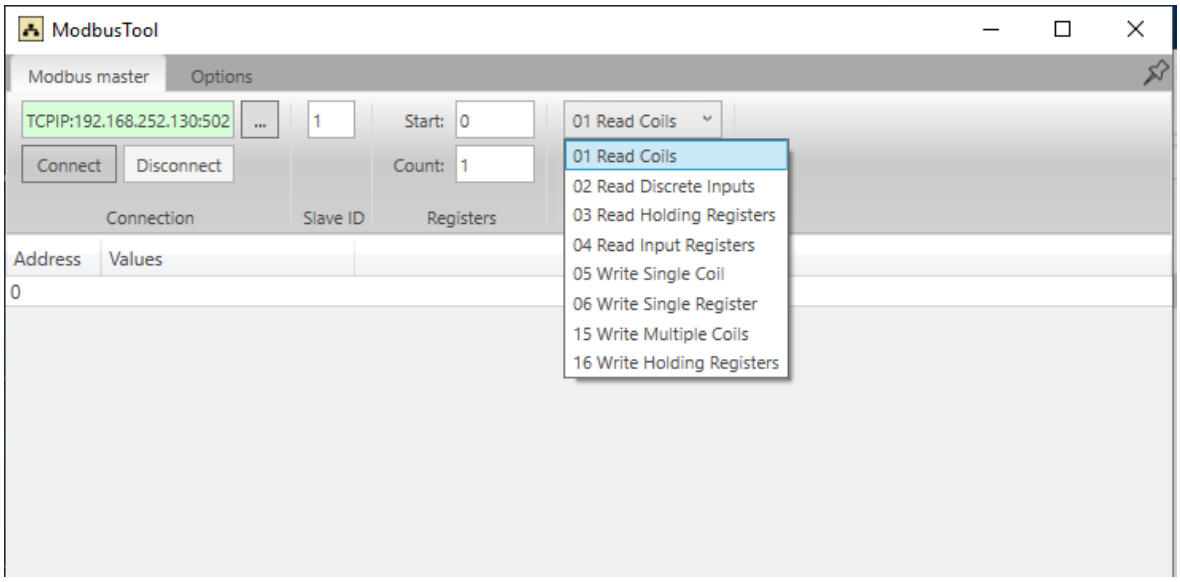


Figure 33 Dropdown of ModbusTool functions

Below are screenshots of the packages sent by the programme, which may be located with the following Wireshark filter: *tcp.port==502*.

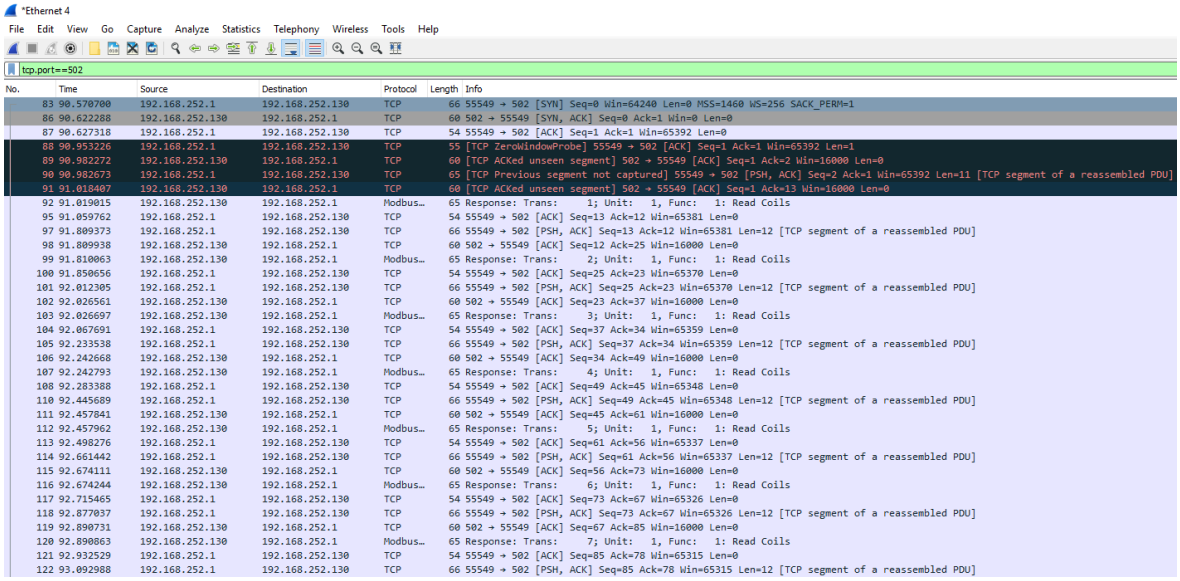


Figure 34 Exchange of record reading packages between ModbusTool and the honeypot

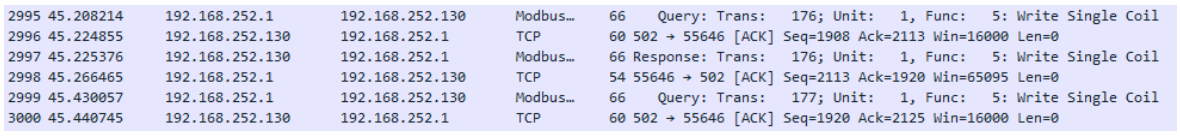


Figure 35 Exchange of record writing packages between ModbusTool and the honeypot

- **S7**

It is also possible to send S7comm packages to the honeypot. For this, the example client from the Snap7 library, **clientdemo.exe**, was used, located in the directory **\snap7-full-1.4.0\rich-demos\x86_64-win64\bin.** The honeypot responds to requests just as a Siemens S7 would.

For generating S7 messages, the following steps are followed:

- Enter the IP address of the Honeypot in the aforementioned programme.
- Click on **Connect.**

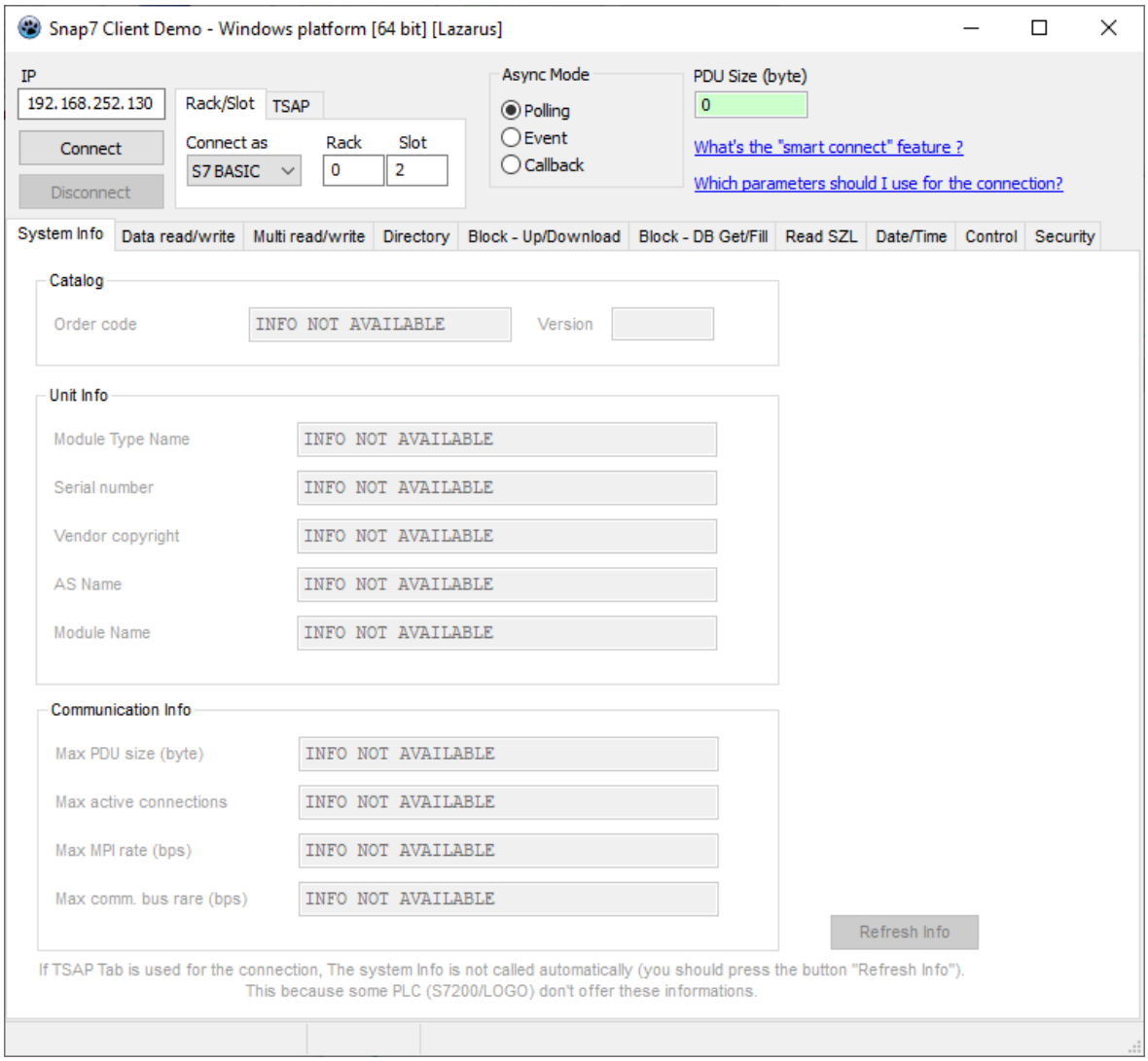


Figure 36 Screenshot of the client snap7

This exchange may be checked using the *tcp.port == 102 && s7comm* filter in Wireshark:

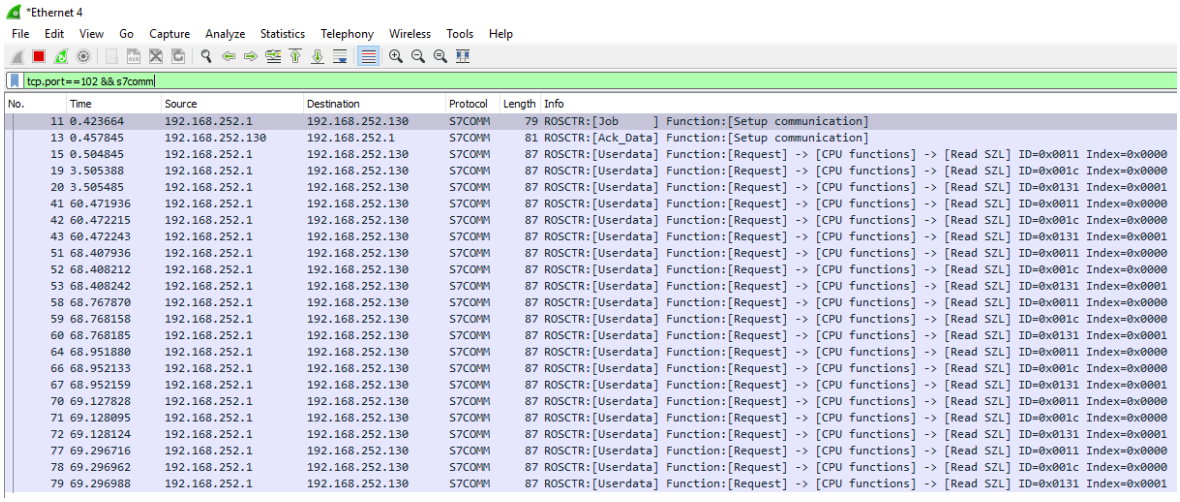


Figure 37 Exchange of connection packages between the client Snap7 and the honeypot

- **Telnet**

The result of making a telnet connection with the honeypot is expected, obtaining the user and password entry messages. Just as with the FTP service, it is not possible to login.

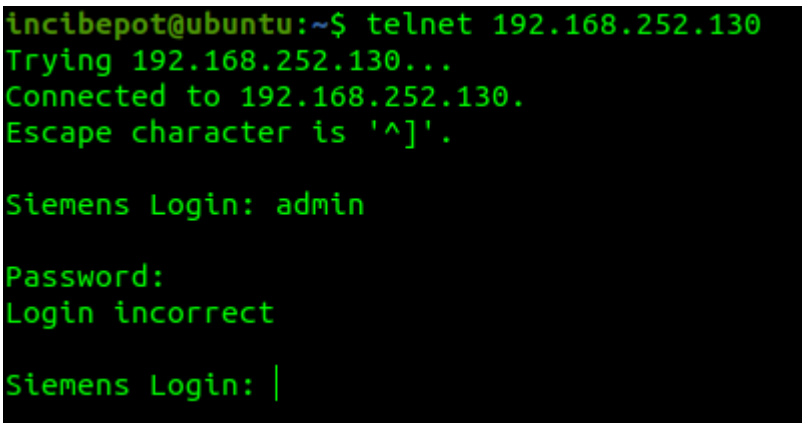


Figure 38 Result of a Telnet attempt to access the honeypot

- **SNMP**

It is also possible to complete SNMP polling in the honeypot. The screenshot below shows the result of carrying out a snmpwalk against its IP, confirming the expected behaviour of the service, obtaining detailed information on the simulated device's hardware and software.

```
incibepot@ubuntu:~$ snmpwalk -t 10 -v 1 -c public 192.168.252.130
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, autonegot
iation"
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 39 Result of running the snmpwalk against the honeypot

8.7.3. Log analysis

Below, the contents generated by Honeyd in the log will be shown, as attempted connections to it are made. In each entry the following information is recorded:

- Date, hour, minute and second at which the request was made.
- Protocol of the registered package.
- Type of connection: “E” if it is a closing, “S” if it is a connection initiation and “-” if it does not belong to any connection.
- Source IP and port
- Destination IP and port
- In some TCP packages are not part of a connection, Honeyd includes the size of the packages and the TCP flags.
- Occasionally, there is a last column with the operating system of the source machine, obtained by means of passive fingerprinting.

```
2019-09-30-04:38:16.5826 icmp(1) - 192.168.252.1 192.168.252.130: 8(0): 60
2019-09-30-04:38:17.5549 icmp(1) - 192.168.252.1 192.168.252.130: 8(0): 60
2019-09-30-04:38:18.5624 icmp(1) - 192.168.252.1 192.168.252.130: 8(0): 60
2019-09-30-04:38:19.5341 icmp(1) - 192.168.252.1 192.168.252.130: 8(0): 60
```

Figure 40 ICMP request log format

```
2019-09-30-06:37:56.6694 tcp(6) S 192.168.252.129 33610 192.168.252.130 21
2019-09-30-06:37:57.6413 tcp(6) E 192.168.252.129 33610 192.168.252.130 21: 0 0
2019-09-30-06:37:59.6580 tcp(6) S 192.168.252.129 33610 192.168.252.130 21
2019-09-30-06:38:35.3690 tcp(6) E 192.168.252.129 33610 192.168.252.130 21: 24 283
2019-09-30-06:39:03.3773 tcp(6) - 192.168.252.129 33610 192.168.252.130 21: 40 R
2019-09-30-06:39:31.8573 tcp(6) - 192.168.252.129 33610 192.168.252.130 21: 40 R
2019-09-30-06:39:59.8654 tcp(6) - 192.168.252.129 33610 192.168.252.130 21: 40 R
2019-09-30-06:40:19.2337 tcp(6) - 192.168.252.129 33610 192.168.252.130 21: 40 R
2019-09-30-06:40:27.8729 tcp(6) - 192.168.252.129 33610 192.168.252.130 21: 40 R
```

Figure 41 FTP request log format

```
2019-09-30-07:12:31.7646 tcp(6) S 192.168.252.129 57280 192.168.252.130 23
2019-09-30-07:14:10.8996 tcp(6) E 192.168.252.129 57280 192.168.252.130 23: 44 102
```

Figure 42 Telnet connections

```
2019-09-30-07:14:53.6762 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.7000 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 40 67
2019-09-30-07:14:53.7119 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.7257 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 44
2019-09-30-07:14:53.7479 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.7589 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 47
2019-09-30-07:14:53.7846 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.7939 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 43
2019-09-30-07:14:53.8200 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.8297 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 43
2019-09-30-07:14:53.8560 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.8754 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 43
2019-09-30-07:14:53.8927 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.9364 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 44
2019-09-30-07:14:53.9648 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:53.9749 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 44
2019-09-30-07:14:54.0001 udp(17) S 192.168.252.129 58689 192.168.252.130 161
2019-09-30-07:14:54.0108 udp(17) E 192.168.252.129 58689 192.168.252.130 161: 43 46
```

Figure 43 SNMP request log format

```
2019-09-30-06:17:48.3266 tcp(6) S 192.168.252.1 55162 192.168.252.130 80 [Windows 2000 RFC1323]
2019-09-30-06:17:48.3626 tcp(6) E 192.168.252.1 55157 192.168.252.130 80: 12 0
2019-09-30-06:17:48.7947 tcp(6) E 192.168.252.1 55162 192.168.252.130 80: 53 226
2019-09-30-06:17:48.7948 tcp(6) S 192.168.252.1 55163 192.168.252.130 80 [Windows 2000 RFC1323]
2019-09-30-06:17:49.2626 tcp(6) S 192.168.252.1 55164 192.168.252.130 80 [Windows 2000 RFC1323]
```

Figure 44 HTTP requests to honeypot

```
2019-09-30-06:19:17.5390 tcp(6) S 192.168.252.1 55233 192.168.252.130 502 [Windows 2000 RFC1323]
2019-09-30-06:19:17.5743 tcp(6) E 192.168.252.1 55230 192.168.252.130 502: 223 0
```

Figure 45 Modbus connections

```
2019-09-30-06:19:57.5350 tcp(6) S 192.168.252.1 55250 192.168.252.130 102 [Windows 2000 RFC1323]
2019-09-30-06:19:57.5711 tcp(6) E 192.168.252.1 55249 192.168.252.130 102: 52 0
2019-09-30-06:20:02.5383 tcp(6) S 192.168.252.1 55251 192.168.252.130 102 [Windows 2000 RFC1323]
2019-09-30-06:20:02.5745 tcp(6) E 192.168.252.1 55250 192.168.252.130 102: 18 0
2019-09-30-06:20:07.5795 tcp(6) E 192.168.252.1 55251 192.168.252.130 102: 48 0
```

Figure 46 S7comm connection

Just as it does with other devices within a network, the sending of this log by syslog could be configured so that it may be treated by a centralised log system.

9. Conclusions

As reflected in the statistics[13], the number of cyberattacks against industrial environments, just like the rest, is increasing every day. Each of these attacks may have a great impact worldwide, putting industrial environments in the attackers' sights. In order to prepare a more effective response against these possible attacks, numerous cybersecurity reference agencies have initiatives focused on studying the latest cyberattack techniques used on this type of network.

One way to protect ourselves is by learning and understanding how attackers act, a task that, thanks to the implementation and use of honeypots, may prove easier and provides multiple benefits, including learning new attack techniques and distracting them from their real objective. However, we must also take into account and control their disadvantages, such as exposing our network devices to the outside or providing attackers access to certain types of services and devices.

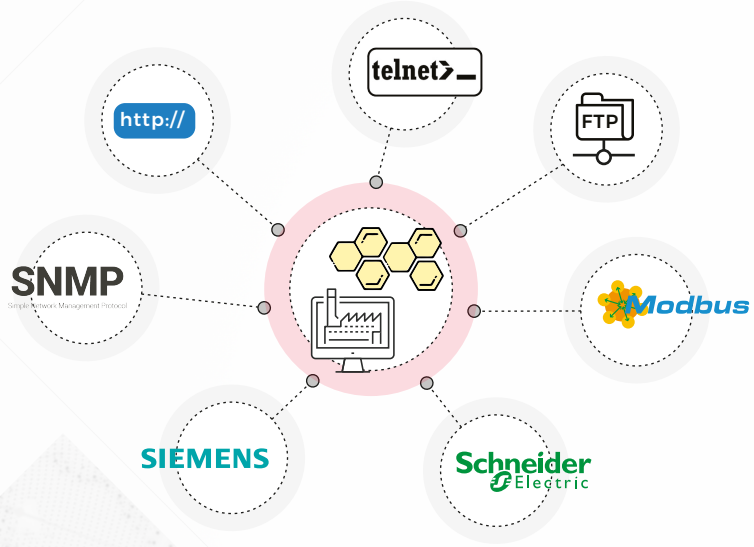
The tool used in this study, Honeyd, allows users to deploy honeypots in order to identify the possible threats facing an industrial network, simulating different elements of its architecture, such as PLC, different operating systems, services, etc., thus gathering information on the alleged attackers in order to improve their level of cybersecurity.

[13] https://vestertraining.com/sectores-industriales-recibieron-mas-ciberataques-2018/

incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE

incibe-cert_

Deployment of an industrial honeypot



HONEY INSTALLATION

GIT INSTALLATION
```
sudo apt-get install git
```

HONEYD DOWNLOAD
```
git clone https://github.com/DataSoft/Honeyd
```

DEPENDENCIES INSTALLATION
```
sudo apt-get install libevent-dev libdumbnet-dev libpcap-dev libpcre3-dev
libedit-dev bison flex libtool automake zlib1g-dev python net-tools
```

HONEYD COMPILATION AND INSTALLATION
```
cd Honeyd/
./autogen.sh
./configure
make
```

CREATION OF DIRECTORY FOR CONFIGURATION FILES
```
cd ..
mkdir <path_name>
```

HONEYPOT CONFIGURATION

SCADA HONEYNET PROJECT DOWNLOAD
```
http://www.sf.net/projects/scadahoneynet
```

MOVE SCRIPTS DIRECTORY TO HONEYPOT PATH
```
cd <download_path>
tar -xvzf <scadahoneynet_file.tar>
cp -a ./cernsacadahoneynet/files/scripts <path_name>/scripts
```

This last path will be labeled as <scripts_path> for the following steps.

SCRIPT WEB MODIFICATION

Edit <scripts_path>/honeyd-http-siemens.py

```
webroot = "/var/cshoneynet/scripts/web-siemens" -> webroot = "<scripts_path>/web-siemens"
```

RENAME AND MODIFY TELNET FILE

Inside <scripts_path>

```
cp honeyd-telnet-schneider.py honeyd-telnet-siemens.py
```

Modify honeyd-telnet-siemens.py file:

```
logintext = "\n\rVxWorks login: " -> logintext = "\n\rSiemens Login: "
```

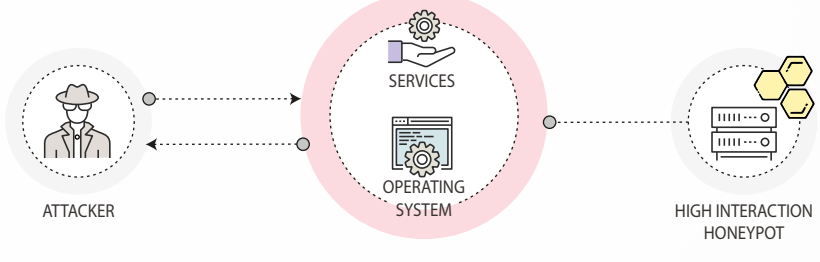
MODIFY NMAP.ASSOC FILE
```
cat /usr/share/honeyd/nmap-os-db | grep "Siemens\ Simatic\ 300"
```

Add the result (without Fingerprint) to the end of the /usr/share/honeyd/nmap.assoc list. In case it is already there, make sure that it does not remain as a comment.

CONFIGURATION FILE

Create a configuration file (<filename.conf>) inside the directory <path_name> and including the following configuration lines:

```
create siemens
set siemens ethernet "00:1f:f8:cc:d0:23"
set siemens default tcp action closed
set siemens default udp action reset
set siemens personality "Siemens Simatic 300 programmable logic controller"
add siemens tcp port 21 "python <scripts_path>/honeyd-ftp-siemens.py"
add siemens tcp port 23 "python <scripts_path>/honeyd-telnet-siemens.py"
add siemens tcp port 80 "python <scripts_path>/honeyd-http-siemens.py"
add siemens tcp port 102 "python <scripts_path>/honeyd-s7.py"
add siemens udp port 161 " python <scripts_path>/honeyd-snmp-siemens.py"
add siemens tcp port 502 " python <scripts_path>/honeyd-modbus.py"
set siemens uptime <timestamp in seconds>
bind <ip_address> siemens
```

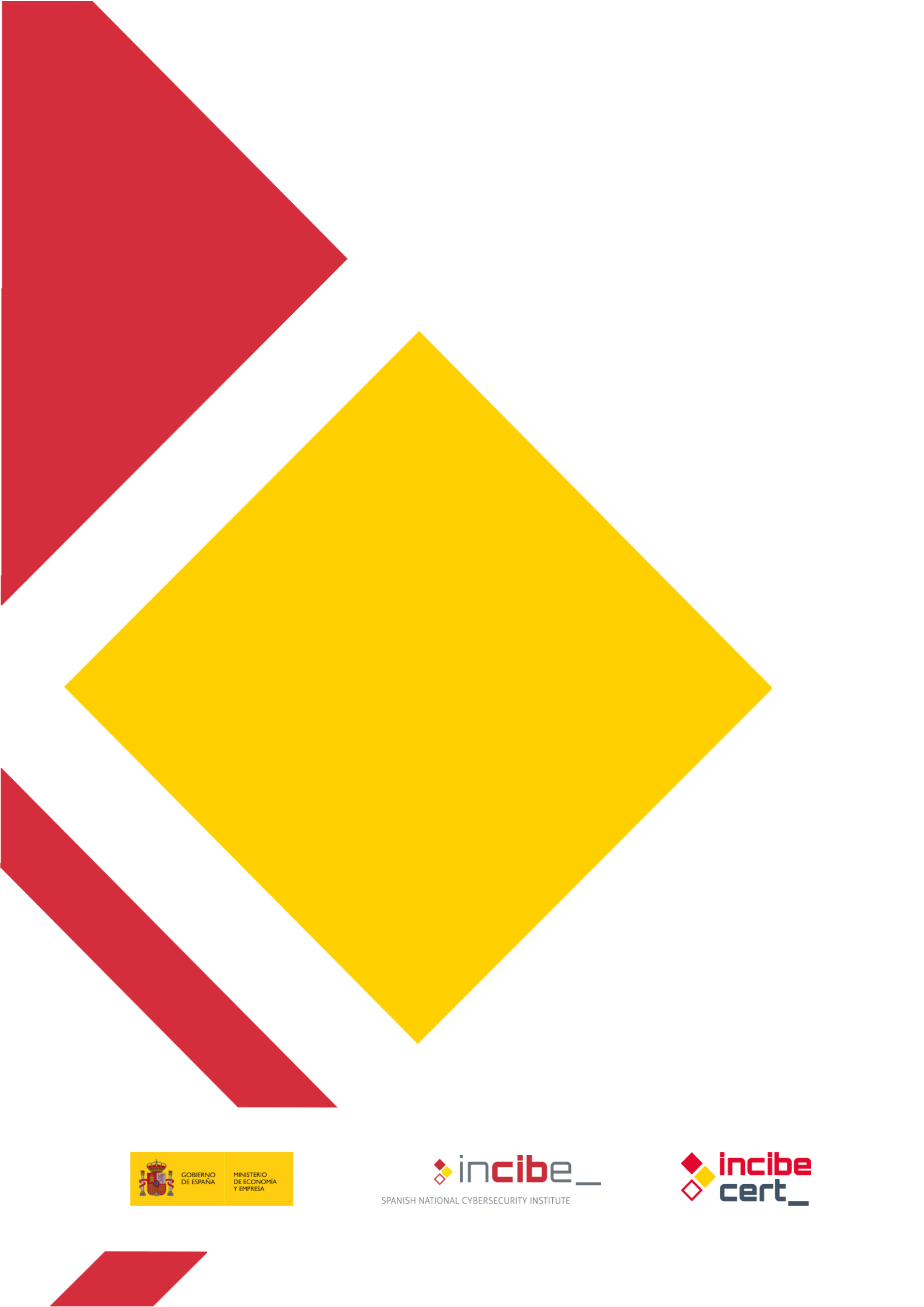


RUNNING HONEYD

```
sudo honeyd -d -p nmap-os-db -i <interface> -l <log_name.log> -f <filename.conf>
<IP_address_or_subnet> -u 0 -g 0 --disable-webserver
```

10. References

Reference	Title, author, date and link
[Ref.- 1]	"Conoce a tu enemigo: Definiendo honeynets virtuales". Honeynet Project. 4 de septiembre de 2012. URL: http://his.sourceforge.net/honeynet/papers/virtual/
[Ref.- 2]	"Honeypots". 9 de marzo de 2012. URL: http://tecnoloxiaxa.blogspot.com/2012/03/honeypots.html
[Ref.- 3]	"Honeyd". Wikipedia. 10 de julio de 2019. URL: https://en.wikipedia.org/wiki/Honeyd
[Ref.- 4]	"Honeynet". Wikipedia. 19 de julio de 2019. URL: https://es.wikipedia.org/wiki/Honeynet
[Ref.- 5]	"Honeypot". Wikipedia. 19 de julio de 2019. URL: https://es.wikipedia.org/wiki/Honeypot
[Ref.- 6]	"BeEF Project". Brendan Coles. 11 de marzo de 2018. URL: https://github.com/beefproject/beef/wiki/
[Ref.- 7]	"Real vs. virtual honeypots". Brien Posey. 11 de marzo de 2005. URL: https://searchenterprisedesktop.techtarget.com/tip/Real-vs-virtual-honeypots
[Ref.- 8]	"Open Source Tools for Active Defense Security". Ed Moyle. 5 de junio de 2018. URL: https://securityintelligence.com/open-source-tools-for-active-defense-security/
[Ref.- 9]	"Honeynets, una desconocida en la seguridad informática". Enseñanza CCOO Andalucía. 5 de noviembre de 2009. URL: https://www.feandalucia.ccoo.es/docu/p5sd6337.pdf
[Ref.- 10]	"The Honeynet Project". The Honeynet Project. URL: https://www.honeynet.org/about
[Ref.- 11]	"Honeypot, una herramienta para conocer al enemigo". INCIBE-CERT. INCIBE (Instituto Nacional de Ciberseguridad de España). 14 de junio de 2018. URL: https://www.incibe-cert.es/blog/honeypot-herramienta-conocer-al-enemigo
[Ref.- 12]	"Honeypots industriales". INCIBE-CERT. INCIBE (Instituto Nacional de Ciberseguridad de España). 23 de marzo de 2017. URL: https://www.incibe-cert.es/blog/honeypots-industriales
[Ref.- 13]	"¿Qué es un honeypot?". IONOS Digital Guide. 8 de agosto de 2017. URL: https://www.ionos.es/digitalguide/servidores/seguridad/honeypot-seguridad-informatica-para-detectar-amenazas/
[Ref.- 14]	"How does a Honeypot work?". Manoj Murali. 31 de enero de 2015. URL: https://www.quora.com/How-does-a-Honeypot-work


GOBIERNO
DE ESPAÑA
MINISTERIO
DE ECONOMÍA
Y EMPRESA
incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE
incibe
cert_