Technical University of Crete

Diploma Thesis

EXPLORING HONEYPOT FINGERPRINTABILITY FOR STEALTHY ATTACK DETECTION

Author:
Despoina Ntolka

Thesis Committee:
Prof. Ioannidis Sotirios
Prof. Apostolos Dollas
Prof. Vasilis Samoladas

A thesis submitted in fulfillment of the requirements
for the diploma of Electrical and Computer Engineer

in the

School of Electrical and Computer Engineering
Microprocessor and Hardware Laboratory

December 19, 2023

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

EXPLORING HONEYPOT FINGERPRINTABILITY FOR STEALTHY ATTACK DETECTION

by Despoina Ntolka

As the field of cyber security continues to evolve due to escalating security attacks, honeypots are becoming increasingly important. Honeypots are strategic deception systems that emulate real services in order to attract cyber attackers. Apart from capturing malicious users, they also study their behavior and reveal their tactics. As a result, cyber experts are informed about the latest attack methodologies and strategies employed by adversaries. However, with every innovative defense strategy comes challenges. One major concern is the ability of attackers to identify honeypots compared to real systems by detecting differences in their behavior and responses. This research focuses on the issue of fingerprintability, and particularly examines weaknesses in low- and medium-interaction honeypots. The reason we choose low- and medium- interaction honeypots is because they are more vulnerable to be compromised than high-interaction ones due to their limited simulation depth. We, initially, install these honeypots in controlled environments and carefully analyze their behavior considering factors such as ports, banners, and headers. Then, we compare them to machines across the World Wide Web with the goal to detect relevant honeypots. The results of these comparisons are concerning as they reveal existing vulnerabilities among honeypots. As we move further, these findings become more crucial. With this research, we try to help cybersecurity efforts that aim to fix these weaknesses and ensure that honeypots remain reliable defensive tools, even as attackers improve their device discovery skills.

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

EXPLORING HONEYPOT FINGERPRINTABILITY FOR STEALTHY ATTACK DETECTION

by Despoina Ntolka

Καθώς ο τομέας της κυβερνοασφάλειας συνεχίζει να εξελίσσεται, τα honeypots γίνονται ολοένα και πιο σημαντικά. Τα honeypots είναι συστήματα παγίδες τα οποία μιμούνται πραγματικές υπηρεσίες με σκοπό να δελεάζουν κυβερνοεπιτιθέμενους. Εκτός από την προσέλκυση κακόβουλων χρηστών, μελετούν επίσης τη συμπεριφορά τους και αποκαλύπτουν τις τακτικές τους. Ως αποτέλεσμα, οι ειδικοί στον κυβερνοχώρο είναι ενημερωμένοι για τις πιο πρόσφατες μεθοδολογίες και στρατηγικές επιθέσεων που χρησιμοποιούν οι επιτιθέμενοι. Ωστόσο, με κάθε καινοτόμο στρατηγική άμυνας προκύπτουν προκλήσεις. Ένα κύριο ζήτημα είναι η ικανότητα των επιτιθέμενων να αναγνωρίσουν τα honeypots σε σύγκριση με τα πραγματικά συστήματα ανιχνεύοντας διαφορές στη συμπεριφορά και τις απαντήσεις τους. Αυτή η έρευνα εστιάζει στο θέμα της δακτυλικής αποτύπωσης, και συγκεκριμένα εξετάζει αδυναμίες στα honeypots χαμηλής- και μεσαίας- αλληλεπίδρασης. Ο λόγος που επιλέγουμε honeypots χαμηλής- και μεσαίας- αλληλεπίδρασης είναι γιατί είναι πιο επιρρεπή στο να αναγνωριστούν σε σχέση με τα honeypots υψηλής-αλληλεπίδρασης λόγω της περιορισμένου βάθους προσομοίωσης. Αρχικά, εγκαθιστούμε αυτά τα honeypots σε ελεγχόμενα περιβάλλοντα και αναλύουμε λεπτομερώς τη συμπεριφοράς τους λαμβάνοντας υπόψη παράγοντες όπως ports, banners, και headers. Στη συνέχεια, τα συγκρίνουμε με μηχανήματα σε όλο τον παγκόσμιο ιστό με σκοπό να ανακαλύψουμε παρόμοια honeypots . Τα αποτελέσματα αυτών των συγκρίσεων είναι ανησυχητικά, καθώς αποκαλύπτουν υπάρχουσες ευπάθειες μεταξύ των honeypots. Όσο προχωράμε παρακάτω, αυτά τα ευρήματα γίνονται αρκετά σημαντικά. Σκοπός μας είναι να βοηθήσουμε τις προσπάθειες της κυβερνοασφάλειας που στοχεύουν να διορθώσουν αυτές τις αδυναμίες και διασφαλίσουμε ότι τα honeypots παραμένουν αξιόπιστα αμυντικά εργαλεία, ακόμα και όταν οι εισβολείς βελτιώνουν τις δεξιότητές τους στην ταυτοποίηση συστημάτων.

Acknowledgements

I would like to thank those who have supported me throughout my academic years and in the completion of my thesis.

First of all, I would like to express my sincere gratitude to my advisor professor mr. Sotirios Ioannidis for initiating me to the world of security, supervising and motivating me.

I would like to thank my project advisors Panagiotis Ilia, Spyros Antonatos and Eva Papadogiannaki for their cooperation, support and our productive discussions.

I am, also, thankful to the Technical University of Crete (TUC) for the knowledge and support all these years.

Finally, I am deeply grateful to my family and friends for their inspiration and assistance.

Contents

List of Figures

Listings

List of Tables

Dedicated to my family and friends...

Chapter 1

Introduction

1.1 Motivation

As technology continues to play a major role in many aspects of our lives such as communication, entertainment and governance, the security of these systems becomes critical. Any intrusion or cyber attack on these systems can result in massive damages. This is why cyber security is crucial. Among the methods used to gather information about attacks and attackers, honeypots stand out. Honeypots do not just capture attackers, they also provide valuable insights into the techniques, strategies and methods attackers use.

However, the digital landscape is constantly evolving. There is a trend where hackers continuously enhance their skills for gain, good reputation or even terrorism. Many of them are experts at identifying and evading honeypots effectively. Once they detect the presence of a honeypot, its effectiveness and stealth are compromised. This situation highlights the need for cyber experts to detect their weaknesses and develop a new generation of honeypots.

Our research journey, overall, explores how attackers using their evolving tools can discern honeypots from real machines. We uncover some concerning findings during our investigation. Throughout our research, we identify weaknesses that need attention in order to prevent the detection of honeypots. We conduct an analysis, focusing on aspects like ports, banners, and headers, to understand the patterns exhibited by these honeypots. After examining ports, banners, and headers, we successfully identify machines that are being used as honeypots. It becomes evident that distinguishing between honeypots and real systems can be succeeded with the use of particular combinations of these characteristics.

These discoveries support the larger cyber security community, in addition to serving academic goals.

1.2 Scientific Contributions

The scientific contributions from our research are mentioned below:

- We thoroughly examine and categorize the features of low and medium interaction honeypots, including ports, headers, and banners. Specifically, we analyze ipphoney, citrix, cowrie, dionaea, elasticpot, elasticsearch, and honeytrap.
- We develop a fingerprinting method to identify honeypots and their type by examining their ports, banners and headers.
- We validate our assumptions we make at the start of our research, regarding honeypots non-careful design, deployment & maintenance.

1.3 Underlying Assumptions

We state the following assumptions:

- **Banners and Headers:** The banners and headers of the services can sometimes indicate that they are running in a honeypot.
- **Service Inconsistencies:** Genuine systems and honeypots may have differences in the services they offer.
- **Infrequent Updates:** Honeypots may not be updated as frequent as real machines. This can result in outdated services or in purpose vulnerabilities to attract attackers.
- **Many Honeypots on one Machine:** In a machine can run different honeypots at the same time.

1.4 Thesis Outline

The rest of the paper is organized as follows. Chapter 2 includes the basic theoretical background, we must be familiar in order to understand the research. In Chapter 3, we present related work that has conducted in the security area. Chapter 4 introduces to the methodology of the research, and in Chapter 5 we

refer to the results of this methodology. In Chapter 6, are referred the challenges and limitations we face during our study. Finally, Chapter 7 summarizes the research and proposes some relevant ideas for future work.

Chapter 2

Theoretical Background

In this chapter, we briefly look at the current state of honeypots. Also, we analyze the security features we use, as well the techniques we employ in order to gather network information.

2.1 Current State of Honeypots

A real war is raging in the cyberspace. Every entity on the Internet is building sophisticated defenses against the outer world, and even a crack in these barriers can be exploited to cause enormous damage [1]. For more than twenty years, honeypots have been a standard tool to give businesses a more comprehensive strategy [2]. The honeypot is one of the most popular mechanisms to gather information about attacks and attackers and is deliberately placed on the organisational network to be probed and attacked. A honeypot deceives and attracts an attacker, who attempts to gain unauthorized access to the network [3]. Due to the pressing necessity of understanding the strategies employed by cyber attackers, honeypots have advanced from simple baits to complex tools.

Based on where they are positioned, their effectiveness is dependent. Honeypots can be strategically placed both inside and outside the firewall. The external honeypot serves as a system that alerts incoming activity. In contrast, internal honeypot assists in identifying violations that have successfully evaded the firewall.

Besides their position, there are many types of honeypots with different purposes. One of the most popular ways to classify honeypots is based on the level of their interaction with adversary; low- , medium-, high- interaction. The low-interaction honeypots, usually, emulate only a single or very limited amount of services but their deployment and maintenance are relatively easy. Similar, are

the medium-interaction honeypots but provide a wider range of services. On the other hand, high-interaction honeypots provide the attacker with an operating system to interact. They are the most advanced type of honeypots, meaning we can gather the most information from them. This comes, however, with the highest demand for deployment, maintenance and also the highest risk.

There are two types of honeypots based on their purpose; research and production. Research honeypots are utilized to investigate attacks, uncovering the motivations behind them and the specific actions carried out. This, usually, requires high-interaction honeypots. In contrast, production honeypots work more like sensors, alerting organizations to possible compromises. These interactions are brief and concentrate more on identification, and for this reason are preferred low- and medium- interaction honeypots.

Honeypot implementation can be done using either software, hardware or hybrid, a combination of both. Hardware based honeypots imitate aspects, ranging from regular computers to specialized Supervisory Control and Data Acquisition (SCADA) systems. Other software structures employ virtualization techniques to enhance their authenticity.

Nowadays, it is crucial to prioritize scalability due to the increasing appearance of botnets and the growing number of cyberattacks aimed at organizations. Therefore, there has been an emphasis on developing honeypots that manage multiple connections simultaneously. Additionally, some of these honeypots incorporate features as redeployment mechanisms and Peer-to-Peer (P2P) functionality [1, 2].

Some worth-mentioning honeypots are Dionaea, Elasticpot, and Conpot that are low-interaction and consequently easy to deploy. Other high-interaction solutions are SIPHON and Honey Accounts and provide attackers with an environment that looks real [1, 13].

Overall, honeypots, even deceptive, have a decent purpose. Based on their unique characteristics they fortify the shield of cybersecurity by attracting and eliminating cyberattackers.

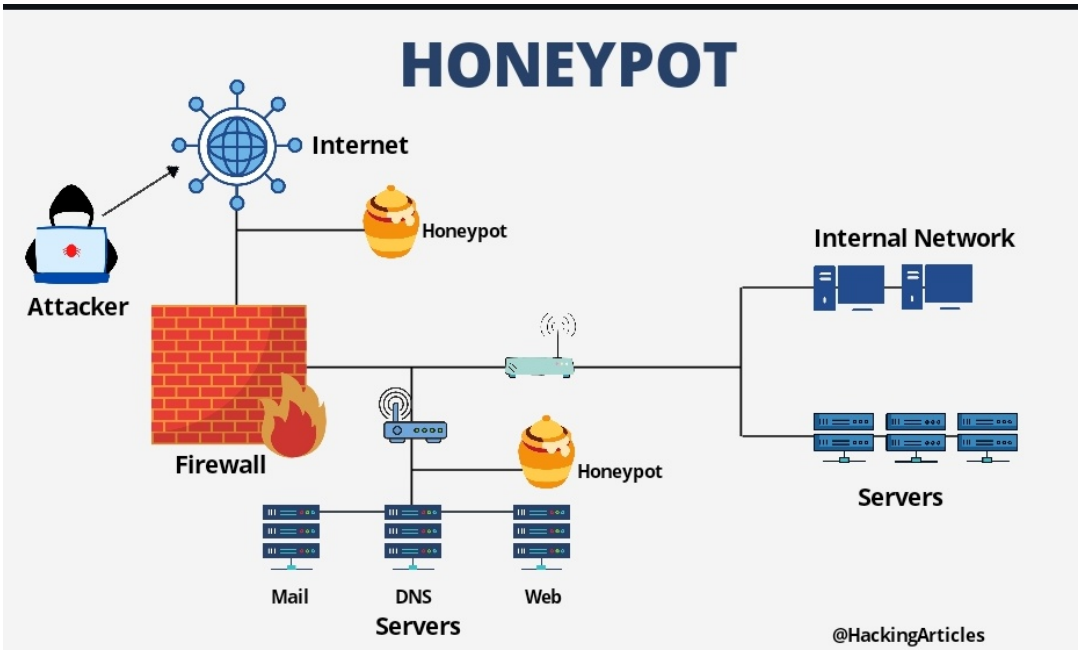

FIGURE 2.1: Honeypot

2.2 Internet Infrastructure and Security Resources

The bedrock of digital communication is the internet infrastructure and its security features, offering all users reliability and security. Between these features, the **Internet Assigned Numbers Authority (IANA)** is very important. Currently, IANA, which is a function of the Internet Corporation for Assigned Names and Numbers (ICANN), is in charge of root zone management for the Domain Name System (DNS), autonomous system number allocation and global IP address allocation. To be more specific, IANA is broadly responsible for the distribution of globally unique names and numbers, that are used in Internet protocols and are published as Request for Comments (RFC) documents [14].

Shodan is a tool that should not be overlooked. As the search engine for the Internet of Things, Shodan works different in contrast to traditional search engines that primarily index web content. Shodan is a search engine that lets users search for various types of servers (webcams, routers, servers) connected to the internet, using a variety of filters. Shodan has been created with good intentions for use by security professionals and researchers for data collection. However, the search results can also push hackers to seek this information and use it for more malicious purposes [15].

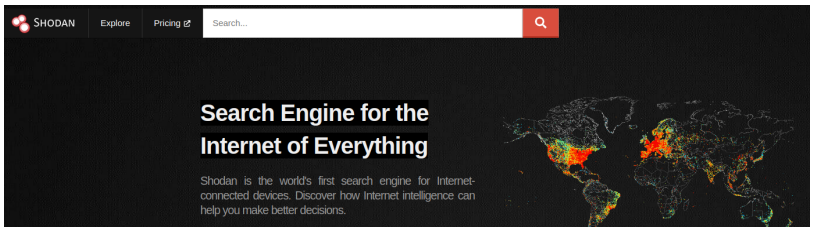

FIGURE 2.2: Shodan Search Engine

2.3 Networking Tools

Networking tools are very important for computer network management, monitoring and security. The administrators, using them, can spot connectivity problems, identify devices and examine network architectures.

One networking tool that stands out is **Nmap**. Nmap, also known as Network Mapper, is a tool used for scanning networks. Since it was created, Nmap is a first choice among network administrators. It offers features to analyze computer networks, such as discovering hosts and detecting services and operating systems. Like any tool, Nmap could potentially be used by malicious users. That was made, nevertheless, so that system administrators could use it to look for unauthorized servers or machines that do not follow security standards [16].

Netstat is a valuable addition to the networking toolkit. It stands for network statistics and provides features of Nmap. Netstat is a command line utility that gives a view of networks on operating systems like Unix, Linux, Windows and macOS. It, specifically, shows network connections for the Transmission Control Protocol, routing tables and a number of network interface and network protocol statistics [17].

2.4 Network Information Gathering

A very important concept, on which our research is based, is **Banners and Headers**. They are simply the metadata about a service. When a web server is connected it might receive a request, for a header or banner, containing details like the software being utilized and its version. This information is crucial for identifying the software's type and version that is currently being used.

So, it is critical to understand the **Network Information Gathering**, which includes methods for obtaining data about network systems.

Operating System (OS) Fingerprinting is the technique of examining data packets that come from a network in an effort to gather information for use in later attacks. Configuration attributes can also be gathered from remote devices using OS fingerprinting. When hackers discover a vulnerability in outdated networks or unpatched operating systems, these become big targets. Fingerprinting is divided in two categories; active and passive. Attackers send a packet to the targeted system in an active OS fingerprinting attempt, wait for the response, and then examine the contents of the TCP packet. In contrast, passive fingerprinting does not require any communication. It, simply, observes all traffic and analyzes packet properties. Both defenders and attackers utilize this method for their own purposes. Hackers use it to find weaknesses they can exploit, while network administrators employ it as a security measure [4, 18].

Banner grabbing is a form of OS fingerprinting. It is used to gain information about a computer system on a network and the services running on its open ports. Banner grabbing can be performed either actively or passively. When banner grabbing is actively, we request information using tools like Telnet or Netcat. In passive banner grabbing, we analyze intercepted traffic between two systems, but without connecting to the target system, often using tools like Wireshark or TCPDump. This can be used by administrators to list all of the services and systems available on their network. But a hacker, can use banner grabbing to locate network hosts, running operating systems and applications that have known vulnerabilities [19].

Chapter 3

Related Work

In this chapter, we analyze further the area of honeypots. Special attention is given to high-interaction honeypots. The discussion extends to network security focusing on fingerprinting techniques. Alongside this, we refer to the detection of honeypots and ways to prevent it. Finally, approaches for new generations of honeypots are suggested.

3.1 High-Interaction Honeypots

A high-interaction honeypot is actually configured to mirror a production system, and is designed to give an attacker full reign of an operating system in the event that they are lured into compromising it.

SIPHON is an example of high-interaction honeypots. Particularly, is a scalable high-interaction honeypot platform for IoT devices. With this design, Internet of Things devices are deployed in a single physical location and are linked to the global internet network via wormholes. As a result, a few devices are exposed over numerous geographically distributed IP addresses. Through the deployment and analysis of this system, it was observed an overwhelming amount of traffic to the devices, but comparatively not so many interactions with the device's web interface. Probably, this is due to little automation in attack tools to perform such attacks, and comparatively less amount of manual attacks compared to automated attacks [5].

During the last decade, many other high-interaction honeypots have been deployed on the Internet in order to analyze attackers' behavior. Interesting has an experiment, in which a high-interaction honeypot was used to monitor malicious activities on the Internet of a compromised machine. The results from this experimental analysis are consistent with security expert knowledge. They

reveal that as opposed to other ports, the attacks on port 22 (SSH) are often, only partially, automated by script kiddies. Also, strange is that honeypot fingerprinting appears low on attackers' priorities [6].

Another deployment of a high- interaction honeypot that lasted over one year, focuses on SSH service attacks. It appears that the attackers do not modify their dictionaries based on the victim's location. An important finding from the experiment was that the IP addresses, used for the invasions and dictionary attacks, were totally different. This seems to confirm that there are computers or communities devoted to particular kinds of attacks [7].

3.2 Fingerprinting Techniques in Networking

Digital fingerprints resemble real-world human fingerprints. Simply, a fingerprint is a collection of data that may be used to identify hardware, software, operating systems, and network protocols. In network fingerprinting (also known as footprinting), using that data, we correlate the datasets to identify software applications, databases, configurations, network services, operating system number and version, and more. An exploit technique against the target may include the use of this fingerprinting data [20].

Fingerprinting techniques have two types; active and passive. Passive operating system fingerprinting determines a host's OS only by monitoring network traffic. A multi-session model, that makes use of data features from TLS, TCP/IP and HTTP protocols, is a helpful technique for passive fingerprinting when multiple sessions can be seen in a time window. Based on this approach, can be identified operating system major and minor versions with accuracies of 99.4% and 97.5%, even in the face of data feature obscuring levels similar to those found on an enterprise network [4].

In addition to fingerprinting techniques, very efficient is a flexible and lightweight extension of the Linux netfilter packet filter framework, which is named natfilterd. It leverages specific characteristics of TCP timestamps, in order to identify hosts independently of their IP addresses. Besides the ability of the tool to count hosts behind a NAT gateway, it can also block TCP traffic from single hosts without blocking the gateway itself. After evaluations, the tool natfilterd achieves a runtime of O(log(n)) for matching packets against a database of n hosts. Therefore, is considered to be more reliable, and most important allows real-time traffic filtering [8].

Very interesting is the area of remote physical device fingerprinting. This approach, in contrast to traditional operating systems, does not need the cooperation of the device. It remotely takes advantage of minor differences in device hardware, and specifically uses clock skews. Additionally, the semi-passive and passive techniques can be used in situations where the fingerprinted device is protected by a NAT or firewall, as well when NTP or SNTP is used to maintain the device's system time [9].

These techniques overcome some difficulties caused by network configuration diversity and data obfuscation, providing insightful information about system properties.

3.3 Honeypot Detection and Evasion

A honeypot is one of the most popular mechanisms used to gather information about attacks and attackers. However, low-interaction honeypots are more vulnerable to fingerprinting attack, as they only emulate a few services. The detection of a honeypot can have severe consequences. For example, revealing the identity of the honeypot can end its usefulness forever, or worse an attacker can convert it into a bot to attack others [3]. The current generation of low- and medium- interaction honeypots uses off-the-shelf libraries to provide the transport layer. Based on a generic technique, that fingerprints low- and medium-interaction honeypots at Internet scale with just one packet and an ERR (Equal Error Rate) of 0.0183, it was found that the use of standard libraries results in detectable differences [10].

In an effort to catch these attacks, some methods have been developed. A fuzzy approach is one of them and it is used on low-interaction honeypots. It predicts the possibility of a fingerprinting attack on the honeypot by correlating the behaviors of an attack [3]. Another very important technique is related to the prediction of fingerprinting attacks on the honeypot system in real-time. This technique requires some actions by the attacker in order to analyze them, and therefore identify the fingerprinting attack. For the development of this technique, the honeypot tool KFSensor and the fingerprinting tools Nmap and Xprobe2 are used [11].

But the techniques that detect fingerprinting attacks are not enough alone, a new generation of honeypots with new architectures is required. One approach

for avoiding honeypot detection involves automated re-deployment of the honeypot, that in turn reduces the need for anti-detection honeypot configurations. Also, a significant detection risk for honeypots is the delays in processes that a honeypot mimics. In genuine systems such delays do not exist. Therefore, a reduction in the honeypot delay is necessary. Although expensive, the use of dedicated hardware for honeypots can help minimize their detection. The most important for avoiding honeypot detection is the introduction of dynamic approaches to honeypot development, as well as novel techniques in machine learning and artificial intelligence. The result is a more adaptable honeypot that is more difficult to detect [2]. Additionally, for this scope is developed an SDN-based intelligent honeynet; called HoneyMix. HoneyMix allows for fine-grained data control over honeynets and takes advantage of SDN's rich programmability to evade attacker detection mechanisms. For this purpose, HoneyMix makes several connections at the same time, using a variety of honeypots and selects the most appealing connection to persuade attackers to stay connected [12].

In conclusion, a honeypot system is critical, since it collects information on these attacks and their perpetrators. And, as time passes, more approaches for detecting fingerprinting attacks and stronger honeypots will be required to maintain the same effectiveness.

Chapter 4

Methodology

In this chapter, we provide an overview of our research approach. To begin with, we install the honeypots in controlled settings. After the installation, we conduct on each honeypot port scans to gather information about their port configuration. Our focus then shifts to examining the headers and banners of these honeypots, as they form the foundation of our investigation. Then, we utilize Shodan to query these headers and banners, so as to identify IP addresses associated with same characteristics. We undertake these IP addresses for a deep network analysis, because we want to discover their active ports and services. The IP addresses with many active ports, including ports related to our honeypots, are highly-possible to be honeypots, and require more investigation. To confirm this possibility, there is a final fingerprinting analysis. For every IP address, its services are compared with IANA records. From each IP address, whose services deviate from IANA, we evaluate its banners and headers with those from our established honeypots and search for matching characteristics. We cross verify our results to strengthen the reliability of our outcomes.

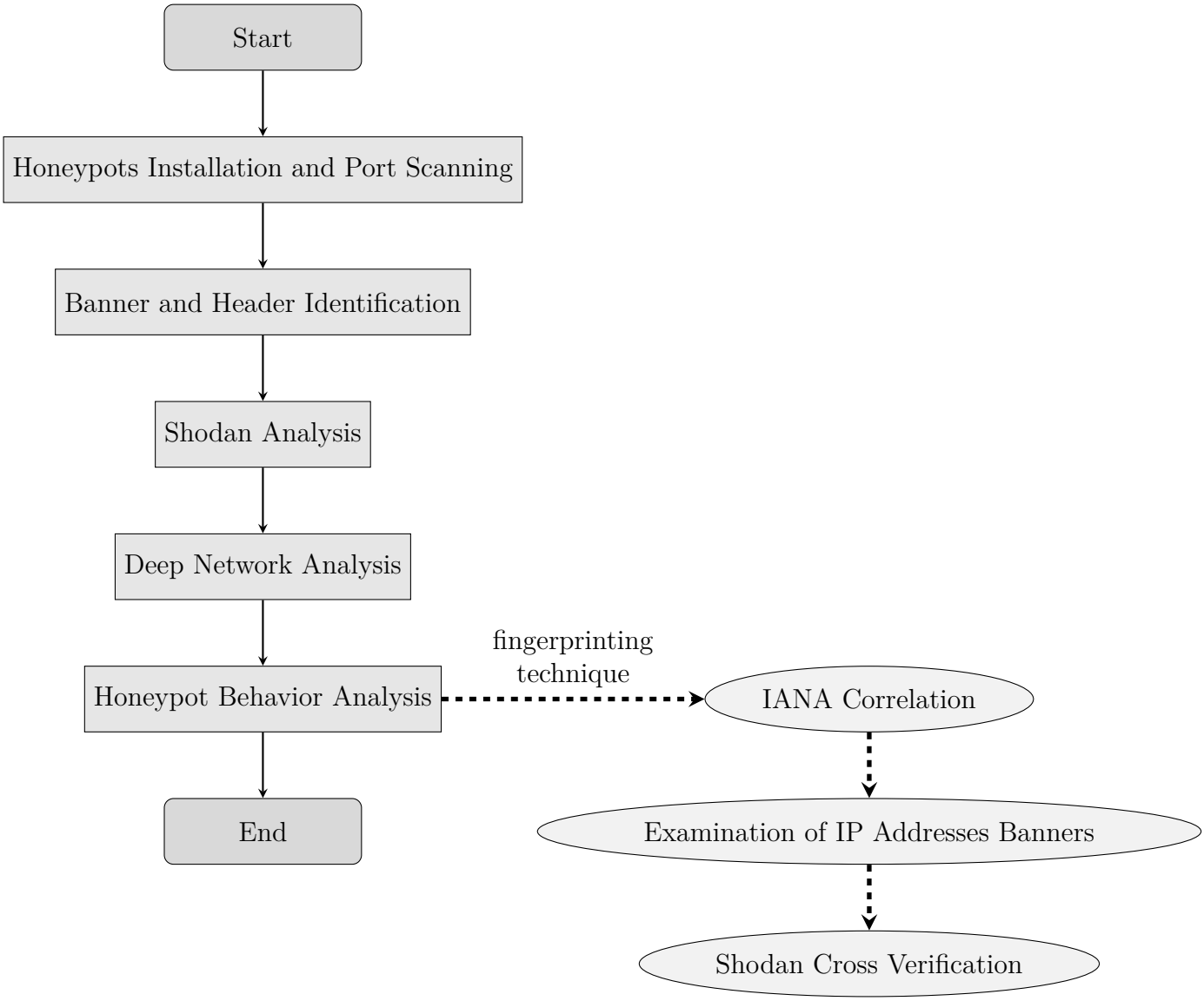

Figure 4.1: Flowchart of the Research Methodology

4.1 Honeypots Installation and Port Scanning

During the first stage of our research, we set up honeypots in controlled virtual environments. We use two virtual machines (VMs), one running Ubuntu 18 and the other running Ubuntu 22. Within these environments, we install many low- and medium- interaction level honeypots such as citrix, cowrie, dionaea, elasticpot, elasticsearch, honeytrap, and ipphoney. We follow a specific installation process for each of them.

We select those low- and medium-interaction honeypots for our study because they are easy to set up and maintain. The sensitivity of these honeypots makes them especially appropriate for our investigation. Because these honeypots only mimic usually one operating system and its services, fingerprinting attacks can more easily target them. This is really helpful for our research. We will look at their unique attributes like their ports and associated services, as well behavioral characteristics.

So, after the installation, follows a port scanning with the use of the networking tool `netstat` for each honeypot to find their open ports. Particularly, the command we use is the `netstat -nalp | grep LISTEN` that gives us the open ports in LISTEN state.

4.2 Banner and Header Identification

After the port scanning, we collect the banner and header from each honeypot's corresponding ports and analyze them. In the start of our research, we make the assumption that the banners' and headers' data can show that they are running in a honeypot (section 1.3).

A detailed examination of each honeypot is covered below.

4.2.1 Elasticsearch

Once we establish the Elasticsearch honeypot [21], our attention turns to the active ports 9200 and 9300.

Ports 9200 and 9300 in the Elasticsearch have different purposes. The default port for enabling HTTP communication with the REST API is 9200. However, Elasticsearch nodes can only communicate with one another over port 9300 and a unique binary protocol.

```
1 wget http://127.0.0.1:9200 %For retrieving the banner
2 curl -I http://127.0.0.1:9200 %For identifying the
    header
```

Listing 4.1: Elasticsearch Commands

To retrieve the banner of port 9200, as supporting HTTP communication, we use the tool `wget`. `GNU Wget` is a free software package for retrieving files using HTTP, HTTPS, FTP and FTPS, the most widely used Internet protocols. It is a non-interactive commandline tool, so it is easily called from scripts, cron jobs, terminals without X-Windows support. Additionally, to retrieve the header of port 9200 we make a HEAD request `curl -I`. With The HEAD method, the server returns the headers exactly the way it would do for a GET, but without a body.

Unfortunately, the communication is not successful with port 9300, as it is with port 9200, and we did not manage to get the information we want.

4.2.2 Elasticpot

We discover that the Elasticpot honeypot [22] is listening port 9200, which is used for Elasticsearch services.

```
1 wget http://127.0.0.1:9200   %For retrieving the
    banner
2 curl -I  http://127.0.0.1:9200    %For identifying the
     header
```

Listing 4.2: Elasticpot Commands

We identify its headers and banners using commands as shown above. The commands we use are the same with ELasticsearch's.

4.2.3 Ipphoney

When we complete the setup of Ipphoney honeypot [23], we discover that it is listening on port 631. This port is linked with Internet Printing Protocol (IPP) services.

```
1 wget http://127.0.0.1:631    %For retrieving the
    banner
2 curl -I  http://127.0.0.1:631    %For identifying the
    header
```

Listing 4.3: Ipphoney Commands

Because IPP is based on HTTP, we submit a request to the IPP server listening on port 631/tcp, in order to retrieve the banner from the printer. For the banner and header, because is a HTTP communication, we use the `wget` and `curl -I` commands.

4.2.4 Cowrie

After the Cowrie honeypot [24] is successfully installed, we discover the active ports 2222 and 2223. They are used for OpenSSH services.

```
1 nc localhost 2222    %For retrieving the banner
```

Listing 4.4: Cowrie Commands

A few words for these services is that SSH uses encrypted format for data transmission, and also it uses a secure channel. Now, we execute the command `nc` to retrieve banner from port 2222. The `netcat` command is a command-line utility for reading and writing data between two computer networks. The communication happens using either TCP or UDP.

Unfortunately, we cannot connect with port 2223 and get the information we want.

4.2.5 Citrix

When the installation of Citrix honeypot is completed [25], we detect that it is listening on ports 80 and 443.

```
1 wget http://127.0.0.1:80    %For retrieving the banner
2 curl -I  http://127.0.0.1:80    %For identifying the
     header
3 curl -Ik https://localhost:443 -k    %For identifying
     the secure header
```

Listing 4.5: Citrix Commands

Port 80 allows HTTP protocol, meaning the information remains in plain text between the browser and the server, while Port 443 allows HTTPS protocol and all the information travels between the server and the browser remains encrypted. So, for port 80 in order to grab its banner, we utlize the command `wget`, as it allows us to connect to localhost:80 and download the HTML source code to the terminal. As for its header, we make again a HTTP HEAD request `curl -I`. In port 443, we only download successfully the header. Here, we use the `curl -Ik`, because to take the header, we have to bypass the SSL/TLS security checks.

4.2.6 Honeytrap

Regarding the Honeytrap honeypot [26], we notice the ports 5900, 8022, and 8023. These ports follow respectively the protocols VNC, SSH.

```
1 nc 127.0.0.1 5900    %For retrieving the VNC banner
2 nc 127.0.0.1 8022    %For interacting with SSH port
3 nc 127.0.0.1 8023    %For interacting with SSH port
```

Listing 4.6: Honeytrap Commands

For all the three ports we use the command `netcat` to take the banner. For the SSH services we have explained previously the reason why we use this command. As for VNC protocol, is a graphical desktop-sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. We are able to see the banner information when we link netcat to the VNC service.

4.2.7 Dionaea

For the Dionaea honeypot [27], we notice the most active ports 27017, 3306, 11211, 1433, 1883, and 1723.

```
1 mongo --host localhost --port 27017    %For retrieving
    the MongoDB banner
2 nc localhost 11211                  %For interacting
   with Memcached
3 sqlmap -u "http://localhost:1433" --banner    %For
   retrieving MS SQL Server banner
4 mysql -h localhost -P 3306 --protocol=tcp    %For
   retrieving MySQL banner
```

Listing 4.7: Dionaea Commands

The above commands are all used to fetch the banners of each port. Port 27017 is specifically employed for mongoDB server, which is a module for EAP Controller. Memcached is a performance distributed memory caching system. It operates on port 11211 both for TCP and UDP protocols. To establish a connection with the SQL database instance TCP, port 1433 is utilized. Lastly, the default port for the MySQL protocol is port 3306.

However, attempts to retrieve banners from ports 1883 (MQTT) and 1723 (PPTP) are unsuccessful.

4.3 Shodan Analysis

In this phase of our research, we use Shodan, a search engine for internet-connected devices, to find devices that have matching combinations of ports, banners and headers. We carefully select them based on the data we collected before. To look deeper, these combinations are produced from each honeypot. Specifically, every banner or header and the corresponding port from each honeypot is a unique pair. Thus, we create these search string combinations and upload them to Shodan.

We employ this technique to detect devices that exhibit similar behavior traits to the installed honeypots. According to our assumption in section 1.3, these matches serve as initial indicators of potential honeypots. It is important to note that, when IP addresses align with the banners or headers of our established honeypots, it does not exclusively mean they are also honeypots. There can be cases where false positives occur. So, these matches should be seen as indicators that require more analysis, in order to determine the true nature of the devices being investigated.

Overall, we carefully consider the possibility of errors when evaluating the data from Shodan. This cautious approach highlights the significance of validating our preliminary results, using multiple methodologies and analytic approaches.

4.4 Deep Network Analysis: Beyond Shodan to Nmap Automation

Next in our investigation, we obtain a list of IP addresses from Shodan that match the banners and headers we already have. To gain an understanding of each system, it is necessary for us to go beyond the matching ports and behavior data. We must note first, that we purposely use `nmap` in a non-intrusive way. Although `nmap` can perform aggressive scans, our methodology is designed to ensure that the systems being investigated are not subjected to intensive probing. Following this approach, each IP address is scanned once. This preserves a balance in the analysis and minimizes any potential disruption to the target IP addresses. Therefore, we utilize the command `nmap` for each IP address. This examination allows us also to reveal, besides ports, other operational patterns and services, providing us with deeper insights. Among other networking tools, `nmap` is a better choice and has multiple benefits. For instance, `nmap` can scan

large networks and also due to the reason its flexible, it offers a plethora of scanning tools.

We develop a python script in order to automate the scanning process. The following script runs for each IP address the `nmap` command. The command `nmap -Pn <target_ip>` by default appears the 1,000 most frequently used TCP ports. Also, the `-Pn` flag helps us to disable host discovery. Disabling host discovery with the -Pn option, causes Nmap to attempt the requested scanning functions against every target IP address specified, even if the host is at first blocking.

```
1 # Execute Nmap scan for the current IP
2 command = ["nmap", "-Pn", ip]
3 result = subprocess.run(command, capture_output=True,
    text=True)
```

Listing 4.8: Nmap Automation Script

We ensure a thorough examination of numerous IP addresses, by collecting crucial data for improved analysis and identification of possible honeypots. As a consequence, our results are more detailed.

4.5 Honeypot Behavior Analysis

In this section, we evaluate the IP addresses with the goal to unveil a multi-faceted behavior. We strongly believe that in a single machine can run multiple honeypots, as stated in section 1.3.

In the beginning of our honeypot behavior analysis, we observe that some IP addresses exhibit a variety of additional ports. Very interesting is the fact that some of the ports are linked to our known honeypots. So, these IP addresses undergo a more in-depth analysis to study the services and operations of their additional ports. Our final fingerprinting technique is concluded of three steps. Initially, the services of the IP addresses in interest are compared with the services of IANA. For the IP addresses that have discrepancies in their services, we perform banner grabbing in order to obtain their banners and headers of

their ports. We carry out banner grabbing only to the ports that link with our established honeypots. Then, we compare all these banners and search if they match with any of our honeypots. This correlation process plays a critical role in our study, as we manage to accurately distinguish honeypots from genuine devices. Finally, to reinforce our findings, we make cross verification with Shodan's data.

Our behavior analysis of the IP addresses is analyzed in detail below.

4.5.1 Correlation with IANA

The first step of our fingerprinting technique includes comparisons with the Internet Assigned Numbers Authority (IANA). IANA is responsible for maintaining the official assignments of port numbers.

This is a thorough examination, as involves each port and service of the associated IP addresses. In particular, we examine the services of each port if aligns with IANA's records, in order to verify the legitimacy of the services. Through this process, we identify many IP addresses that some of their services do not align with IANA's official services. These discrepancies are red flags and strengthen the possibility that these IP addresses may be honeypots (section 1.3).

However, we must not consider these deviations from IANA's standards a solid proof. They are only considered as stronger indicators for the presence honeypots and we must investigate further these flagged IP addresses. This deeper analysis is crucial in order to reveal the characteristics and functionalities true nature of the IP addresses that are marked as potential honeypots during the IANA correlation.

4.5.2 Unveiling Honeypots By Examining The Banners of IP Addresses

The second step is the most crucial of all. It focuses on the IP addresses that were previously highlighted due to their services deviations on some ports. These IP addresses undergo a more intensive examination and we concentrate to discover the banners associated with these machines. Our goal is to correctly identify and classify the IP addresses, whether they are genuine systems or honeypots. So, it is necessary to find distinct patterns and traits that are identical with those displayed by the previously installed honeypots. For this reason, in

our comparison for each IP address we examine only the ports associated with our known honeypots.

With this final fingerprinting method, many machines confirm our assumption. We find out a plethora of IP addresses that exhibit additional banners matching with the established honeypots. The combination of matching banners and IANA inconsistencies not only help us to distinguish honeypots from genuine systems, but also strongly confirm that are running multiple honeypots in these machines. Every matching banner corresponds to a different honeypot.

As a result, the assumptions we make before we start, are pivotal to the process and completion of our study. The fact that we depend on the assumption, that the banners and headers of the IP addresses may indicate the presence of a honeypot, lead us to confirm the other assumption we make that in a single machine can run more than one honeypots (section 1.3).

4.5.3 Cross Verification with Shodan

We have a third and last step in our behavior analysis, with the objective to confirm the accuracy of the data and minimize the false positives. For this step, we use again the Shodan database. The reason why we prefer it among the other databases is because it constantly updates its data. Therefore, our outcomes are verified with the current state of the same IP addresses. Another advantage of Shodan is the tagging system it uses, in order to classify the devices. The tag that is useful to us is named honeypot; facilitating our cross verification.

To summarize, the collaborative approach of our results along with Shodan's large database, gives our fingerprinting analysis an extra layer of verification and strengthens the precision of our results.

Chapter 5

Results

In this chapter, we go over the findings from our investigation. There is an explanation of the results for each step in the methodology. We discover many information, including potential vulnerabilities and unique differences that will cause the compromise of honeypots. These findings shed light on cyber security threats and emphasize the necessity of increasing the number of defensive honeypots.

5.1 Open Ports Analysis

Following the installation of honeypots, we do a port scan to determine which are open ports, in accordance with our approach in section 4.1.

The table below shows the open ports for every honeypot. After the table, there is also an explanation of the ports associated with each honeypot.

	Honeypots	Number of ports	Ports
1	Elasticsearch	2	9200 9300
2	Elasticpot	1	9200
3	Ipphoney	1	631
4	Cowrie	1	2222
5	Citrix	2	80 443
6	Honeytrap	3	5900 8022 8023
7	Dionaea	6	27017 3306 11211 1433 1883 1723

TABLE 5.1: List of Open Ports Associated with Each Honeypot

Elasticsearch Ports (9200 and 9300): Elasticsearch is connected to ports 9200 and 9300. Port 9200 is used for the HTTP API calls. Any activity that makes use of an HTTP request is covered by this including monitoring, search, and aggregations. This port will be used by all client libraries to talk with Elasticsearch. A cluster's nodes can communicate with one another using a custom binary protocol on port 9300.

Elasticpot Port (9200): Likewise Elasticpot with port 9200 mimics Elasticsearch services.

Ipphoney Port (631): Ipphoney simulates a printer that supports the Internet Printing Protocol and is exposed to the Internet through the port 631.

Cowrie Port (2222): Cowrie by default operates on port 2222 in order to prevent conflicts with the machine's actual SSH or Telnet services.

Citrix Ports (80 and 443): Citrix uses port 80 and 443 for network communications. Port 80 is used for HTTP and port 443 is used for HTTPS protocol.

Honeytrap Ports (5900, 8022, and 8023): The Honeytrap is listening to ports 5900, 8022 and 8023. Port 5900 is commonly the remote frame buffer protocol used by the many variants of VNC. As for ports 8022 and 8023 use SSH protocols.

Dionaea Ports (27017, 3306, 11211, 1433, 1883, and 1723): Dionaea can collect data on malware used to compromise the system and mimic susceptible Windows environments and services. A few of its ports are 27017 (MongoDB), 3306 (MySQL), 11211 (Memcached), 1433 (Microsoft SQL Server), 1883 (MQTT), and 1723 (Point-to-Point Tunneling Protocol).

The reason why choose these honeypots over others is because they are more vulnerable. Specific, elasticsearch is a complex system and a common target. The rest honeypots are either miscofigured, or intentionally designed to be vulnerable.

5.2 Honeypots Banners and Headers

Now, according to the step in section 4.2 of the methodology, we extract the banners and header of these honeypots.

The ports, at which we identify headers and banners for each honeypot, are listed in the following table.

	Honeypots	Ports	Banners	Headers
1	Elasticsearch	9200	✓	✓
		9300	×	×
2	Elasticpot	9200	✓	✓
3	Ipphoney	631	✓	✓
4	Cowrie	2222	✓	×
5	Citrix	80	✓	✓
		443	×	✓
6	Honeytrap	5900	✓	×
		8022	✓	×
		8023	✓	×
7	Dionaea	27017	✓	×
		3306	✓	×
		11211	✓	×
		1433	✓	×
		1883	×	×
		1723	×	×

TABLE 5.2: Honeypots: Ports with Corresponding Banners and Headers

As we see from the table, the findings are promising, even we not succeeded for all ports to find banners and header. Then, to gain a better understanding of their behavioral characteristics, we display and analyze those banners and headers.

5.2.1 Elasticsearch

The Elasticsearch on port 9200 has the below banner and header.

```
{
  "name" : "ubuntu-focal",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Tk3TPhcqT1-I7b6f3qx4bg",
  "version" : {
    "number" : "7.17.10",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "
       fecd68e3150eda0c307ab9a9d7557f5d5fd71349",
    "build_date" : "2023-04-23T05:33:18.138275597Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-
       beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Listing 5.1: Elasticsearch Banner 9200

The banner of the Elasticsearch instance on port 9200 reveals important information about the setup and current state. This instance is named `ubuntu-focal` meaning that the server node is using a pseudonym or that it is connected to the Ubuntu Focal Fossa release. The cluster has UUID `Tk3TPhcqT1-I7b6f3qx4bg` and name `elasticsearch`. On `2023-04-23 the version 7.17.10` of Elasticsearch was developed as a `Debian package`. It is built on top of `Apache Lucene 8.11.1` for scaling. The title `You know, for Search` reveals that the main function of this Elasticsearch instance is for searching purposes.

```
1 HTTP/1.1 200 OK
2 X-elastic-product: Elasticsearch
3 Warning: 299 Elasticsearch-7.17.10-
    fecd68e3150eda0c307ab9a9d7557f5d5fd71349
4 "Elasticsearch built-in security features are not
    enabled. Without authentication,
5  your cluster could be accessible to anyone.
6  See https://www.elastic.co/guide/en/elasticsearch/
     reference/7.17/
7 security-minimal-setup.html to enable security."
8 content-type: application/json; charset=UTF-8
9 content-length: 543
```

Listing 5.2: Elasticsearch Header 9200

As for the header, we observe with `HTTP/1.1 200 OK` a HTTP successful communication. First, there is warning message, and then the content information are appeared, which is suspicious. It may be a strategy to attract unauthorized users. The type of the data are in `json format UTF-8 encoded` and the length is `543 bytes`.

5.2.2 Elasticpot

For Elasticpot we present the banner and header running on port 9200.

```
{
  "status" : 200,
  "cluster_name" : "elasticsearch",
  "version" : {
    "lucene_version" : "4.10.4",
    "build_hash" : "
       b88f43fc40b0bcd7f173a1f9ee2e97816de80b19",
    "number" : "1.4.1",
    "build_timestamp" : "2015-07-29T09:54:16Z",
    "build_snapshot" : false
  },
  "name" : "Green Goblin",
  "tagline" : "You Know, for Search"
}
```

Listing 5.3: Elasticpot Banner 9200

Elasticpot and Elasticsearch have similar banners. Examining the banner we find that the servers operational status is `200`. Same it is build on top of `Apache Lucene` but version `4.10.4`. Some honeypots use older versions to look vulnerable and thus provoke attackers. It mimics the `1.4.1` version of Elasticsearch with hash `b88f43fc40b0bcd7f173a1f9ee2e97816de80b19`. The instance's unusual name Goblin may indicate the presence of honeypot.

```
HTTP/1.1 200 OK
Date: Sun, 07 May 2023 16:15:20 GMT
Content-Length: 0
Content-Type: application/json; charset=UTF-8
Connection: Close
Server: Apache
```

Listing 5.4: Elasticpot Header 9200

Like Elasticsearch, the server `Apache` responds with a `HTTP/1.1 200 OK` status, indicating successful communication. In the content, we observe that is in `json`

format and UTF-8 encoded, but has no body. Finally, the connection seems to be Close.

5.2.3 Ipphoney

The Ipphoney honeypot set up on port 631 has the following banner details.

```
1  HTTP/1.1 200 OK
2  Content-Length: 0
3  Upgrade: TLS/1.0, HTTP/1.1
4  X-Content-Type-Options: nosniff
5  Content-Security-Policy: frame-ancestors 'none'
6  Server: Lexmark_Web_Server
7  Connection: upgrade
8  X-XSS-Protection: 1; mode=block
9  Cache-Control: no-cache
10 Date: Sun, 07 May 2023 17:14:27 GMT
11 X-Frame-Options: SAMEORIGIN
12 Content-Type: application/ipp
```

Listing 5.5: Ipphoney Banner 631

At first, the server Lexmark_Web_Server responds with HTTP/1.1 200 OK, which means a successful communication, but the body message is empty. The current connection is upgrading to TLS/1.0 within HTTP/1.1. The X-Content-Type-Options header is set to nosniff guiding the browser to always use the MIME type that is declared in the Content-Type header application/ipp. Lastly, the X-XSS-Protection: 1; mode=block means that, when is detecting an XSS attack, will simply render a blank page instead of attempting to sanitize the injected script. The unusual combination ofLexmark_Web_Server and the application/ipp content type can reveal the identity of the honeypot.

5.2.4 Cowrie

The banner of Cowrie honeypot is the following.

```
1  SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
```

Listing 5.6: Cowrie Banner 2222

From the information we confirm that is an `SSH server` with version `2.0`. Also, the server has `OpenSSH` software with release version `6.0p1`, meaning that the version is 6.0 and has patch level 1. It provides secure encryption for both remote login and file transfer. With `Debian-4+deb7u2` the server runs on a Debian package operation, and specifically on update 2 of Debian 7. However, these versions are outdated, and particularly Debian 7 discontinued on May 2018. This happens, probably, because honeypots are not updated frequently or for strategic purposes.

5.2.5 Citrix

The behavioral characteristics that reveal from the Citrix honeypot on ports 80 and 443 are clarified below.

```
1 HTTP/1.1 400 Bad Request
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
```

Listing 5.7: Citrix Banner 80

The banner on port 80 has `400 (Bad Request)` status code, indicating that the server could not process the `HTTP/1.1` request, due to a client error. The content type is `plain text` encoded in `UTF-8`. The connection appears as `Close`.

```
1 HTTP/1.1 200 OK
2 Accept-Ranges: bytes
3 Content-Length: 19449
4 Content-Type: text/html; charset=utf-8
5 Last-Modified: Thu, 23 Jan 2020 03:38:51 GMT
6 Date: Sun, 07 May 2023 17:50:38 GMT
```

Listing 5.8: Citrix Header 80

In the header data of port 80, the server with a `HTTP/1.1 protocol` has a successful transaction `200 OK`. The Content contains an `HTML document` encoded in `UTF-8` with length `19449 bytes`.

```
1 HTTP/2 200
2 accept-ranges: bytes
3 content-type: text/html; charset=utf-8
4 last-modified: Thu, 23 Jan 2020 03:38:51 GMT
5 content-length: 19449
6 date: Sun, 07 May 2023 18:04:24 GMT
```

Listing 5.9: Citrix Header 443

The header on port 443 is almost similar with the header of port 80. The only diffference is the `version` of `protocol HTTP`, which is `2`. The rest are all the same.

5.2.6 Honeytrap

The Honeytrap honeypot provides the following banners according to ports 5900, 8022, 8023.

```
1  ************************************************************
   **********************
2  *             Copyright(C) 2008-2015 Huawei Technologies
   Co., Ltd.              *
3  *                                All rights reserved
                                    *
4  *                    Without the owner's prior written
   consent,                    *
5  *           no decompiling or reverse-engineering shall be
    allowed.              *
6  * Notice:

   *
7  *                     This is a private communication system
   .                       *
8  *              Unauthorized access or use may lead to
   prosecution.               *
9  ************************************************************
   **********************
10 Warning: Telnet is not a secure protocol, and it is
   recommended to use STelnet.
11 Login authentication
```

Listing 5.10: Honeytrap Banner 5900

The analysis of banner on port 5900 presents a simulation of a `Huawei system`, underlying that this is a private communication. Also, the banner warns that Telnet is not a secure protocol and recommends an alternative the `STelnet`.

```
1 SSH-2.0-OpenSSH_6.6.1p1 2020Ubuntu-2ubuntu2
```

Listing 5.11: Honeytrap Banner 8022 & 8023

The banners of both ports 8022 and 8023 are identical. First of all, the banners use the `version 2.0 of SSH (Secure Shell)` protocol. The server also is running `version 6.6.1 patch level 1 of OpenSSH`. This version is considered old and may be in purpose in order to lure attackers. As for the

2020Ubuntu-2ubuntu2, does not align with the standard Ubuntu naming practises. But, what we understand from this, is that it is an Ubuntu package released on 2020.

5.2.7 Dionaea

The Dionaea honeypot has generated the next banners for the ports 27017, 3306, 11211.

```
1 MongoDB shell version v3.6.3
2 connecting to: mongodb://localhost:27017/
3 MongoDB server version: 3.4.4
```

Listing 5.12: Dionaea Banner 27017

The banner of port 27017 shows a MongoDB database interaction in a MongoDB shell. The version of the MongoDB shell is 3.6.3 and of the server is 3.4.4. Again, we have older versions. Also, the shell and server versions are different, maybe intentionally to resemble real machines.

```
1  Welcome to the MySQL monitor.  Commands end with ; or
       \g.
2  Your MySQL connection id is 1729232896
3  Server version: 5.7.16 MySQL Community Server (GPL)
4
5  Copyright (c) 2000, 2023, Oracle and/or its affiliates
       .
6
7  Oracle is a registered trademark of Oracle Corporation
       and/or its
8  affiliates. Other names may be trademarks of their
       respective
9  owners.
10
11 Type 'help;' or '\h' for help. Type '\c' to clear the
       current input statement.
12 mysql>
```

Listing 5.13: Dionaea Banner 3306

The banner of 3306 implies a MySQL monitor introduction. The server is running version 5.7.16 MySQL Community Server license (GPL). The current session has a unique id which is 1729232896. The only vulnerability, as we have said before, is the old version data.

```
1  stats
2  Response:null
```

Listing 5.14: Dionaea Banner 11211

Port 11211 banner is linked to memcached service. The command stat gives a lot of useful information about the system. The null response suggests that the system may not have any relevant data or is inactive. Perhaps, the reason behind this response, is to deceive the attacker into doing more work.

Concluding, as we look into these various types of honeypots, we see default configurations and responses that these systems typically show in real-world deployment scenarios. It is interesting that sheer number of honeypots deliberately mimics older software versions. As hypothesized at the section 1.3, this strategy might be for miscofigurations reasons, or even meant to entice attackers looking to exploit known vulnerabilities that are frequently present in these outdated systems.

5.3 Shodan Investigation Outcomes

Through the Shodan analysis (section 4.3), we obtain for each honeypot the IP addresses with matching behavioral data.

The number of IP addresses related to each honeypot is shown in the table below.

	Honeypot	**Details**	**Matching IP addresses**
1	Elasticsearch	Banner 9200	278
		Header 9200	235
2	Elasticpot	Banner 9200	221
		Header 9200	1
3	Ipphoney	Banner 631	385
4	Cowrie	Banner 2222	1,619
5	Citrix	Banner 80	858
		Header 80	2
		Header 443	5
6	Honeytrap	Banner 5900	1
		Banner 8022	21
		Banner 8023	1
7	Dionaea	Banner 3306	0
		Banner 11211	118
		Banner 27017	0

TABLE 5.3: Count of IP Addresses Matching Each Honeypot Post Shodan Analysis

From these numbers we observe that exist many machines with similar configurations across the world. The fact that banners and headers are identical does

not confirm that these systems are honeypots, but they form the foundation for a deeper analysis. In order to reveal the systems true nature and reduce the false positives, these IP addresses are thoroughly examined.

5.4 Unveiling Deep Network Analysis

In this section we present the results of our deep network analysis. As we explain in the methodology section 4.4, we conduct a non intrusive network scan with the tool nmap. This automated scan is applied to all the IP addresses we discovered on the Shodan analysis. Through this analysis many other details are revealed, including additional ports and their corresponding services. This step help us to understand deeper the under investigation IP addresses, and whether they are honeypots or genuine systems.

The results are represented with graphical illustrations. Before we continue with these visual representations, we must mention a few details that apply for all of them.

By default, the networking tool nmap scans the 1,000 most common ports. So, in the following illustrations, the number of ports of the IP addresses does not exceed the 1,000.

Also, for the number of ports in the y-axis, we apply a symmetrical logarithmic scale (symlog). In general, log scale helps us visualize the data better because some IP addresses have a lot more ports that others. In contrast to linear scale, where is evenly divided, the log scale has uneven spaces in between consecutive numbers. Then, the symlog scale is more appropriate, as it allows for handling both positive and negative values including zero. We want our graphs to maintain a central region around zero.

There is a color transition in these graphics. IP addresses, that have the same number of ports, are represented with the same color. Therefore, the shade of the color shifts with the viridis python function when the number of ports changes.

5.4.1 Elasticsearch

We start with the IP addresses of Elasticsearch, particularly with the derived data of banner and header on port 9200.

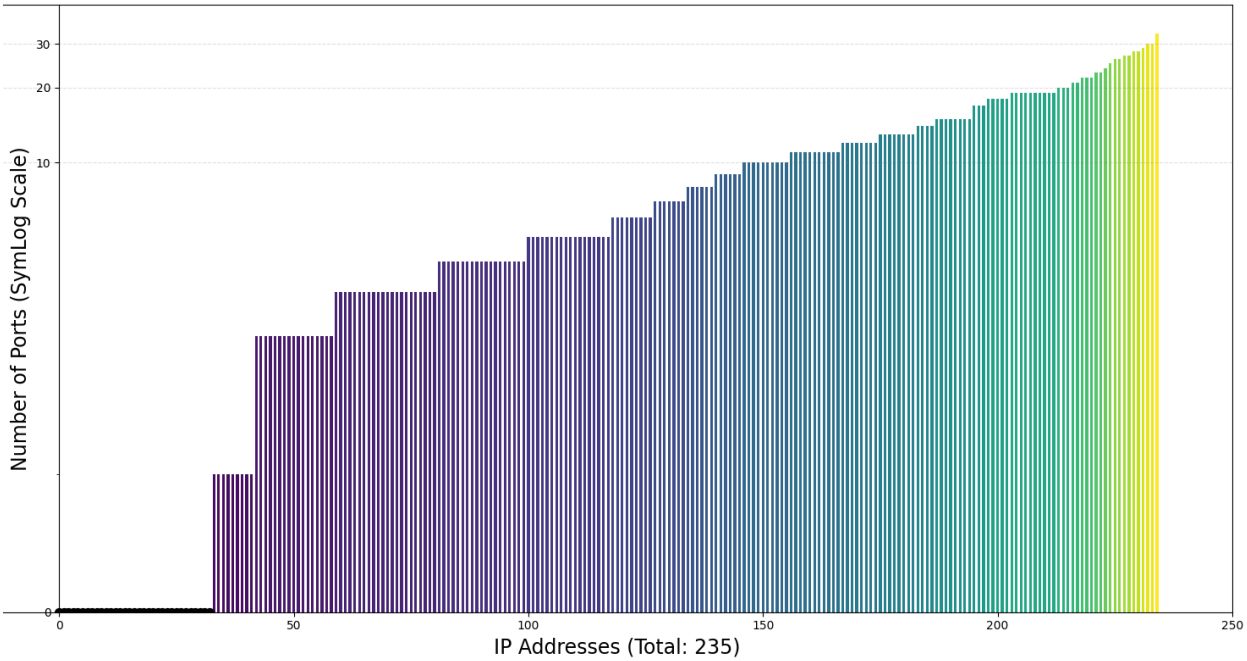

FIGURE 5.1: Port Distributions of Elasticsearch IP addresses: Data Header on Port 9200

As for the data header on port 9200 from the 235 IP addresses, we observe that 33 have no active ports. This is maybe due to strict network filters or because the host is down. The highest number of ports is 33, which is associated with one IP address. We, also, notice many groups of IP addresses that have the same amount of ports. This could indicate common setups or possible honeypots.

Beyond the port distributions of the IP addresses, it is essential to examine all the ports across the IP addresses.

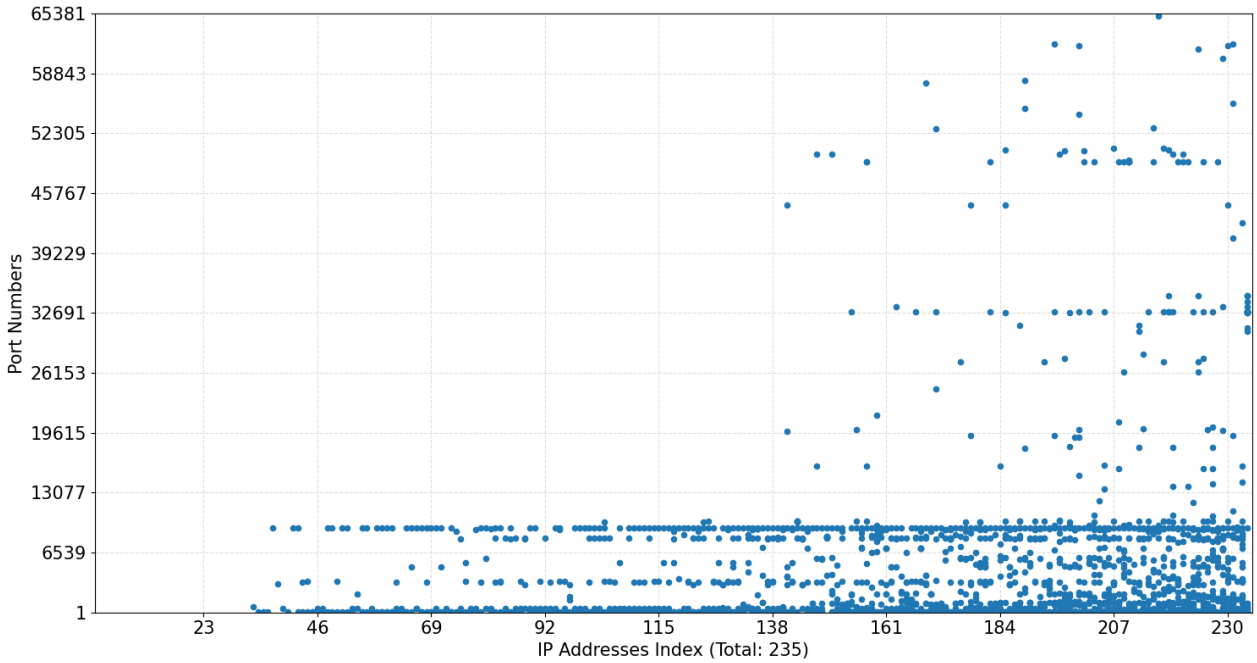

FIGURE 5.2: Dense Port Activities Across Elasticsearch IP Addresses: Data Header on Port 9200

It is visible that the activity of ports among all IP addresses is concentrated in the lower port numbers (1-13,077). Ports in range (1-1,023) are well known ports and are associated with standard services and protocols. Also, in some IP addresses a few high numbered ports exist. Dynamic ports are not secure and they are used to emulate less common services. For example, these ports may monitor malicious actions.

The graphs change a lot when we move our attention to Elasticsearch data banner on port 9200.

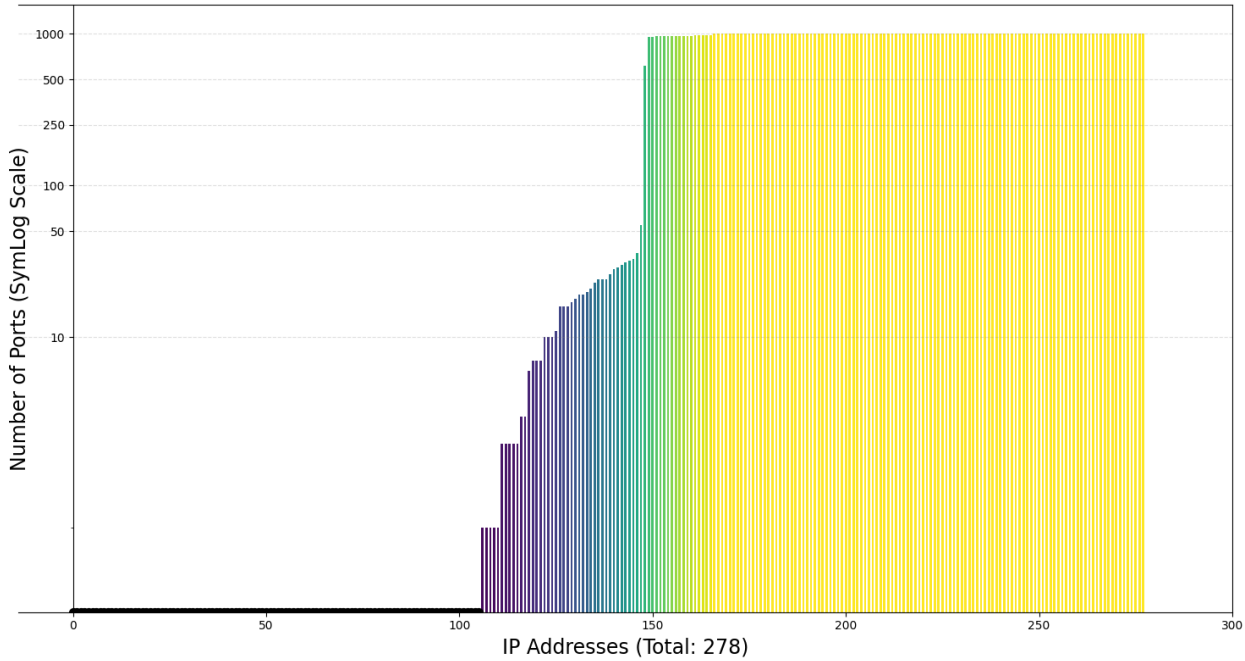

FIGURE 5.3: Port Distributions of Elasticsearch IP addresses: Data Banner on Port 9200

In banner on port 9200, 106 IP addresses out of 278 did not display active ports for the same reasons. As for the number of ports, here is increased in contrast to the previous dataset. The most of IP addresses, specifically 112 have 1,000 active ports. This could be due to several reasons. For instance, in some cases the IP addresses can host multiple services or it can be as a result of misconfiguration. Also, these IP addresses can be honeypots, and the high number of ports is on purpose to attract and study malicious activity.

As previous we evaluate the port density of the data.

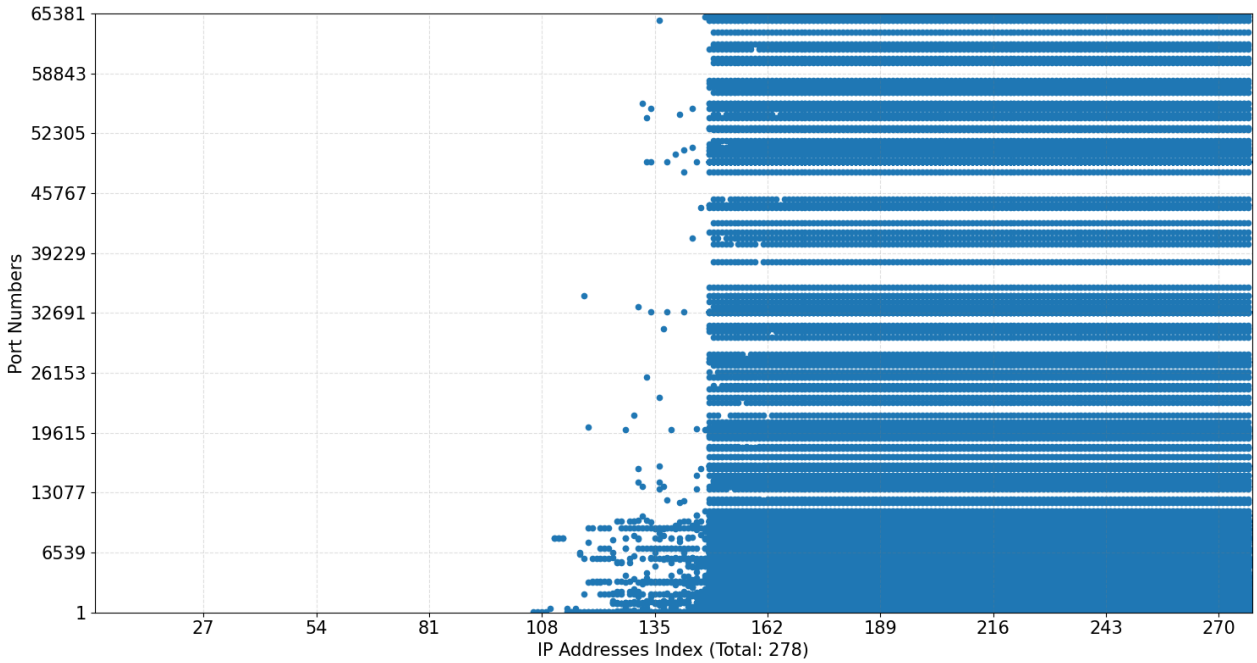

FIGURE 5.4: Dense Port Activities Across Elasticsearch IP Addresses: Data Banner on Port 9200

To a large number of IP addresses, we observe an unusual port activity among all the three port ranges; low (1,023–1,023), medium (1,024–49,151), and high (49,152–65,381). This deviates from standard network setups, where they only open the necessary ports. It is important to mention that among this activity, are many ports associated with our known honeypots. This behavior strongly suggests the presence of honeypots. Honeypots also many times open private ports either to be more attractive to potential attackers or to mask their real services and monitor malicious activity.

5.4.2 Elasticpot

We continue with the IP addresses from Elasticpot.

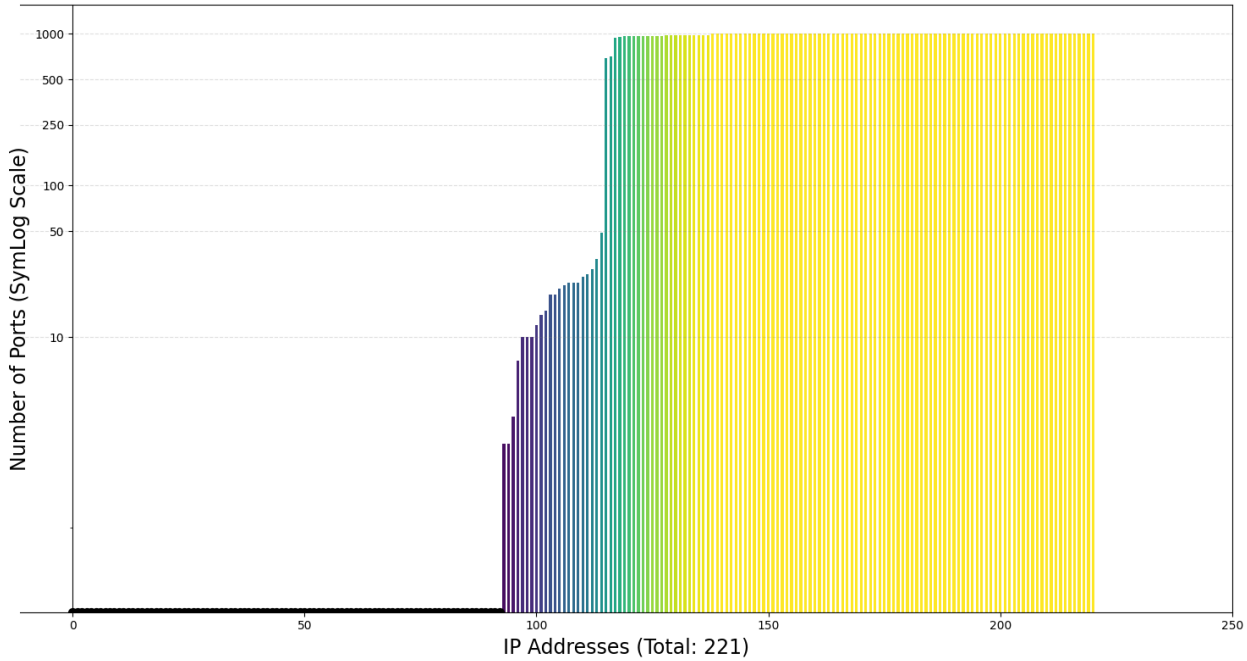

FIGURE 5.5: Port Distributions of Elasticpot IP addresses: Data Banner on Port 9200

The data derived from this dataset present many similarities with Elasticsearchs banner data. From the total 221 IP addresses, the 93 display no ports. This happens for the same reasons. Many of the hosts are down or have strict firewall rules. Additionally, the highest number of ports is 1,000, where 83 IP addresses correspond with it. Besides the IP addresses with 1,000 ports, also these with 974 or 967 ports stand out. They may mimic a variety of services or display on purpose multiple ports to look vulnerable. Their actions resemble those of honeypots.

Beyond the port distribution of active ports, its essential to examine the ports individually.

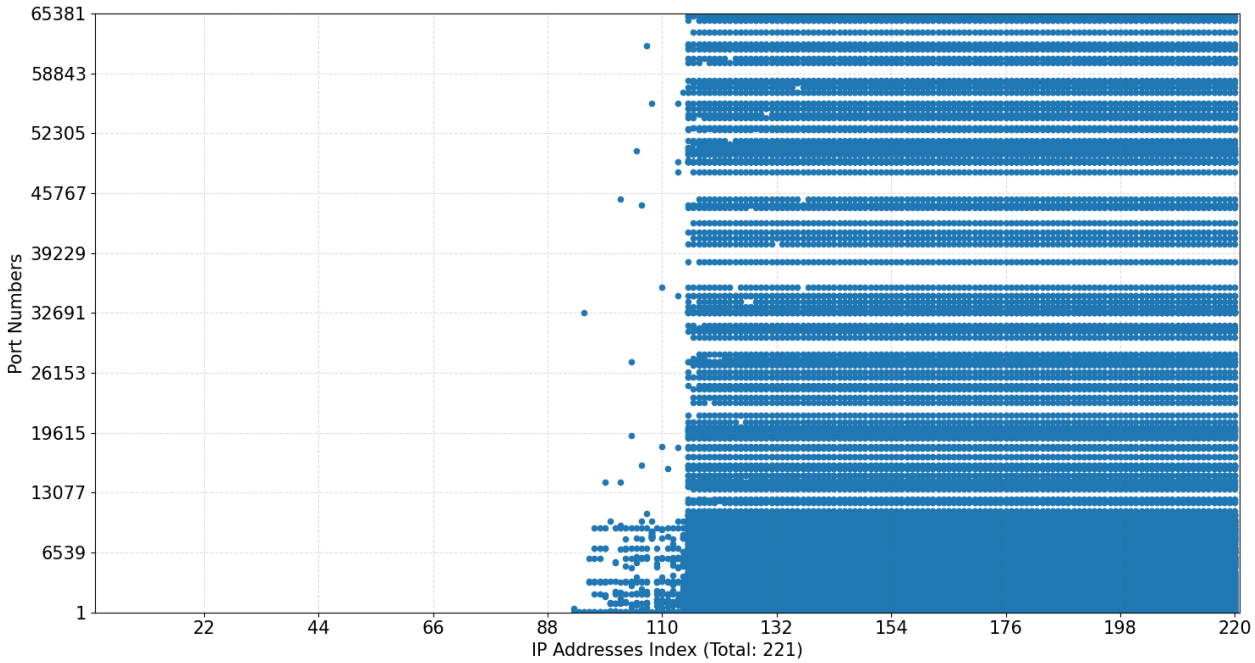

FIGURE 5.6: Dense Port Activities Across Elasticpot IP Addresses: Data Banner on Port 9200

The above graphical representation is almost identical to the Elasticsearch banner on port 9200 one. Most of the IP addresses have all the ports open. Conventional network configurations limit the number of open ports to those that are necessary for a given service. The broad range of open ports violates this rule. The ports in the high range (49,152–65,381), also known for private ports, are used for custom services as monitoring unauthorized users and studying their behavior. This, along with the ports from the honeypots we analyze together in the beginning, provide compelling evidence of the existence of honeypots.

5.4.3 Ipphoney

Below is a summary of the deep network analysis we execute on the Ipphoney IP addresses.

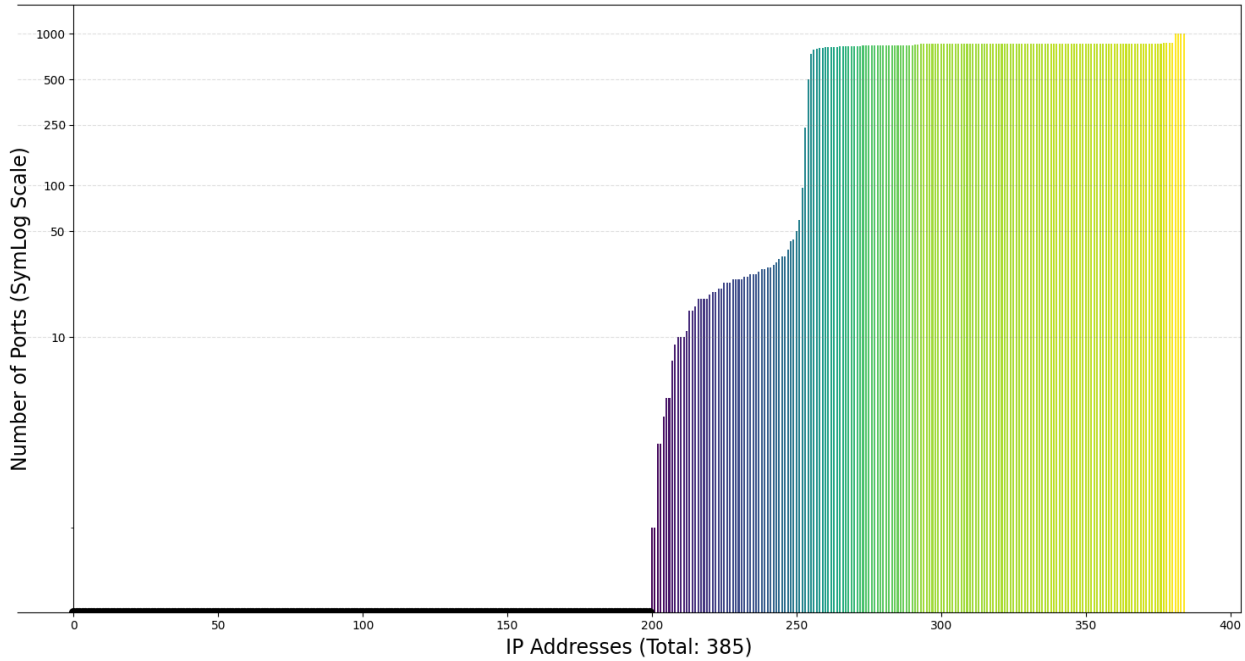

FIGURE 5.7: Port Distributions of Ipphoney IP addresses: Data Banner on Port 631

Likewise, we observe that 200 IP addresses from the whole 385 have no open ports due to deactivation or strict filtering. For the remaining 185 IP addresses, the range of active ports fluctuates from just 1 to 1,000. The IP addresses with so many active ports, including similar to our known honeypots and private ports, might in purpose mimic different services to increase the likelihood of interacting with potential threats. These actions reflect those of honeypots, and to be more specific of multiple honeypots.

We further explain the details of each IP addresses ports.

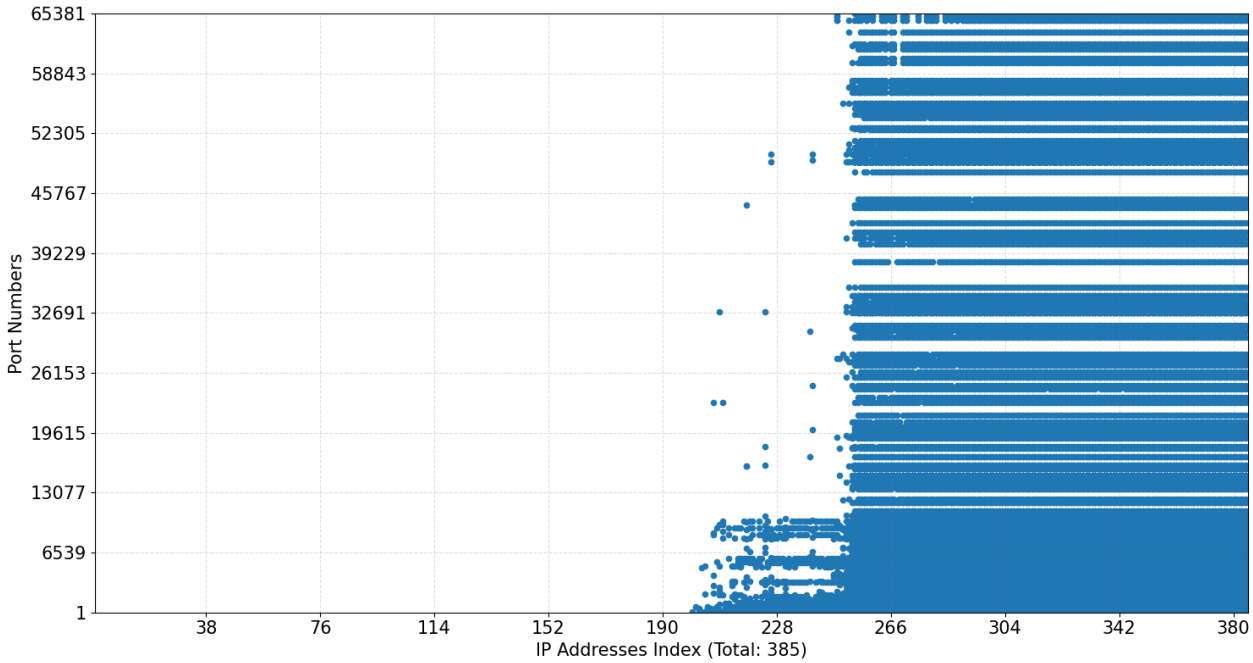

FIGURE 5.8: Dense Port Activities Across Ipphoney IP Addresses: Data Banner on Port 631

We discern a repeated pattern across all the honeypots matching IP addresses. In this case, 130 IP addresses have almost all the ports open in the range (1-65,381). In general, only necessary ports that are needed for particular services are left open for security reasons. The fact that, the ports 9200, 631, 80, 443, 2222, 5900, 8022, 3306, 1433, 1723 are visible, which associate to a few known honeypot systems, strengthens the suspicion of honeypot activity. Furthermore, the high concentration of dynamic ports in range (45,767-65,381), may also be a red flag for honeypots. As we explain in the previous analysis, the use of high-numbered ports can either mean hiding legitimate services behind, or it could be a honeypot strategy to pose as a multi-service host in order to attract possible adversaries.

5.4.4 Cowrie

The IP addresses obtained from Cowrie banner and header, provide valuable insights about their port distributions and dense port activities, that help us discover malicious activity.

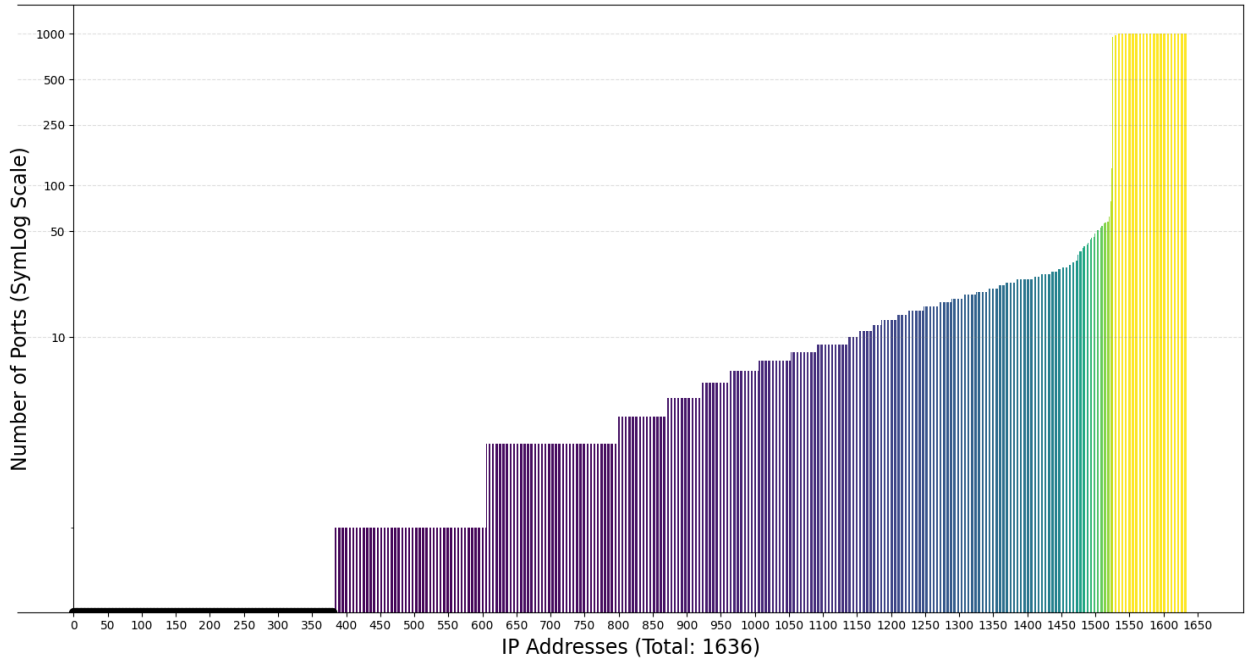

FIGURE 5.9: Port Distributions of Cowrie IP addresses: Data Banner on Port 2222

Although in a lower percentage, the phenomenon of IP addresses that exhibit no active ports exists. Specifically, the total IP addresses are 1,636 and the 381 have not active ports. The pattern continues here. We see a huge difference between the IP addresses as some display only one port, while others reach 1,000 active ports.

We, then, assess each IP addresse's ports.

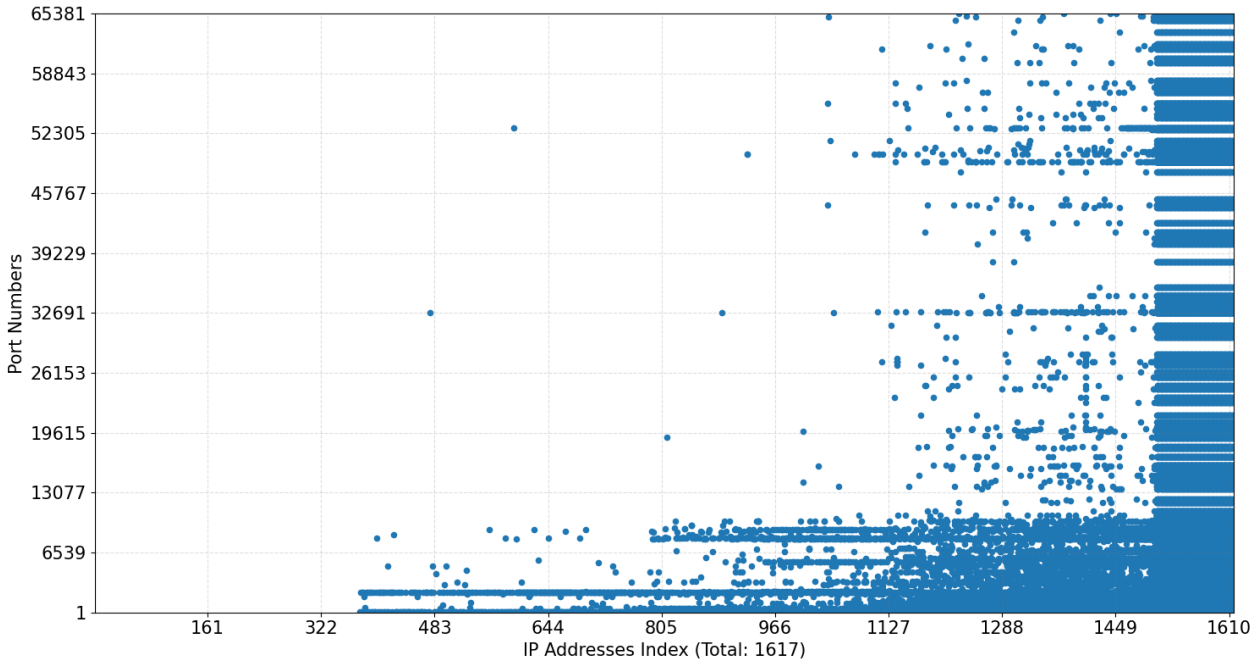

FIGURE 5.10: Dense Port Activities Across Cowrie IP Addresses: Data Banner on Port 2222

Across all the IP addresses, we observe that the average concentration of ports is higher at the range (1-13,077). Once more, several IP addresses show nearly every port that is available. This high number of open ports is a strong indicator for the existence of honeypots, because combines particular ports that align with our established honeypot and private ports.

5.4.5 Citrix

From the IP addresses derived from banner on port 80, we can discern some notable patterns.

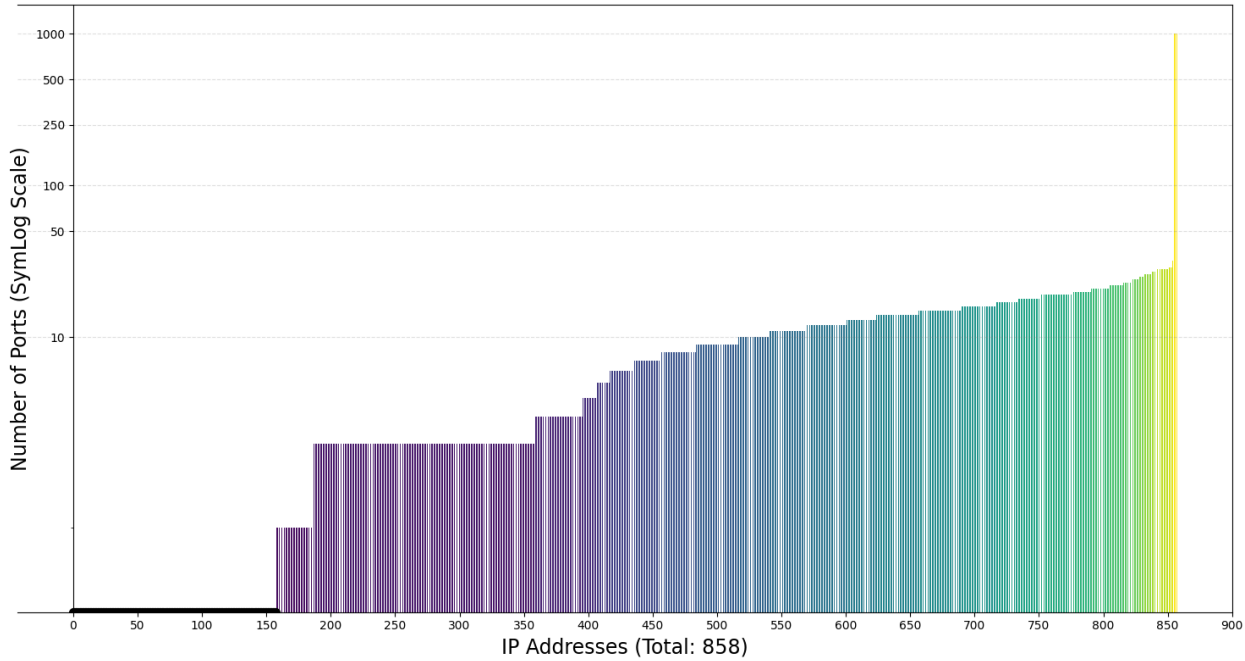

FIGURE 5.11: Port Distributions of Citrix IP addresses: Data Banner on Port 80

In the graphic representation, 158 from 858 IP addresses have no open ports, suggesting either strong security or deactivated systems. On the other hand, the average number of ports is decreased a lot and varies from (1-50). Only 3 IP addresses display 1,000 active ports. Overall, in these machines may run multiple honeypots, and thus intentionally have so many services to attract malicious users.

It is crucial to look more closely at the port activities of each IP address.

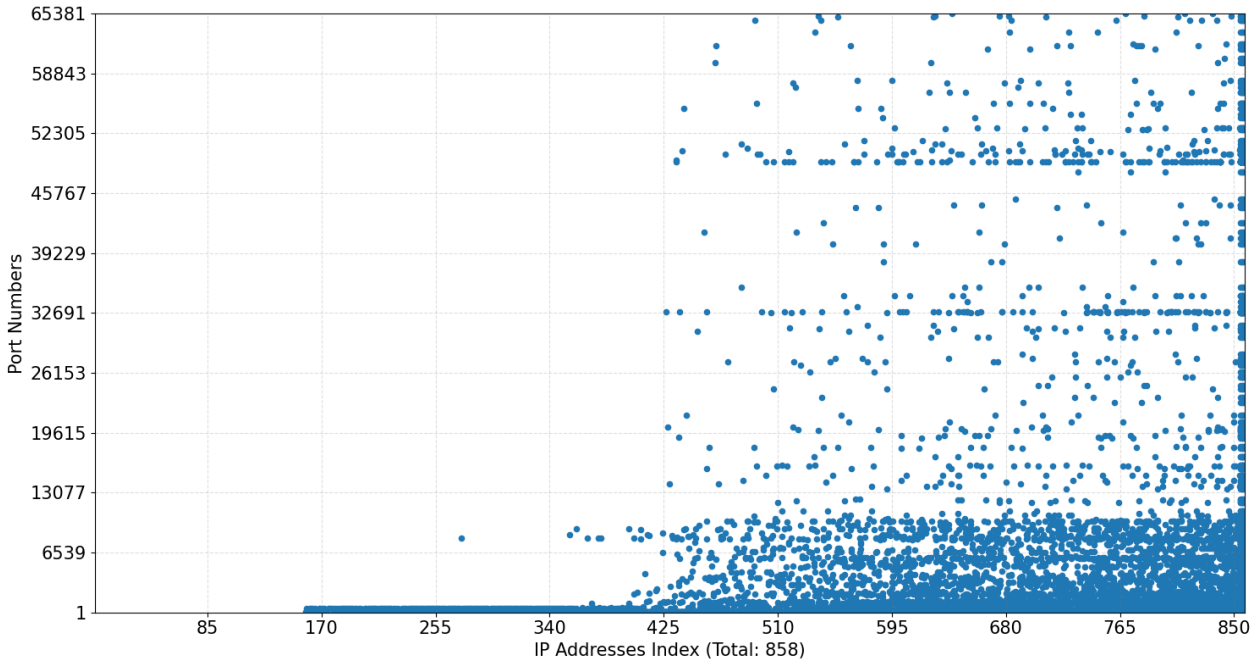

FIGURE 5.12: Port Activities Across Citrix IP Addresses: Data Banner on Port 80

The IP addresses, derived from Citrix, the port activities are not so dense, as are in the other honeypots. There are much denser blue dots at the bottom of the graph, and especially in range (1-12,000). Fewer registered (12,001-49,151) and private (49,152-65,535) ports exist. Even if the port activity is not the same, we still observe some well-known ports, as well some dynamic ones. The interaction of these ports can be a sign that a honeypot is present.

5.4.6 Honeytrap

Honeytrap IP addresses, that obtained from banner and header on port 8022, reveal important patterns.

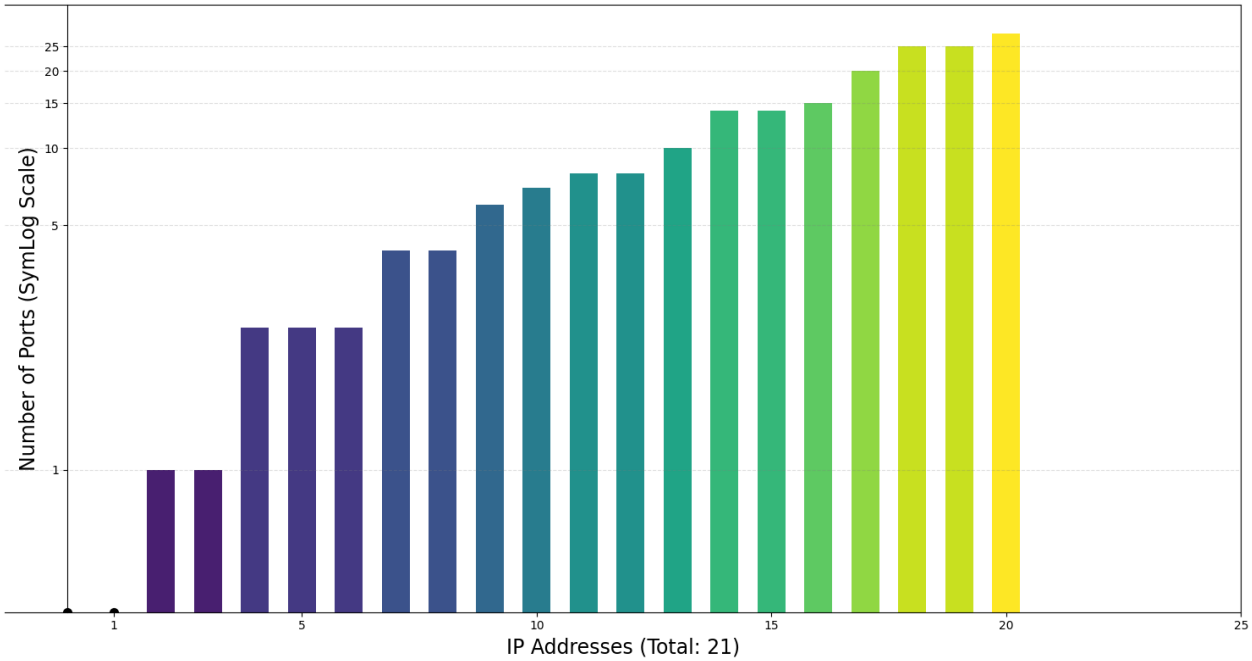

FIGURE 5.13: Port Distributions of Honeytrap IP Addresses: Data from Banner on Port 8022

Based on the graph, we observe that from the 21 IP addresses the 2 show no ports. Also, unlike the rest honeypots, these IP addresses have a significant lower number of active ports. Such port distributions do not provide a reliable indication of whether honeypots are present.

Nevertheless, still is important to examine individually the ports associated with each IP address.

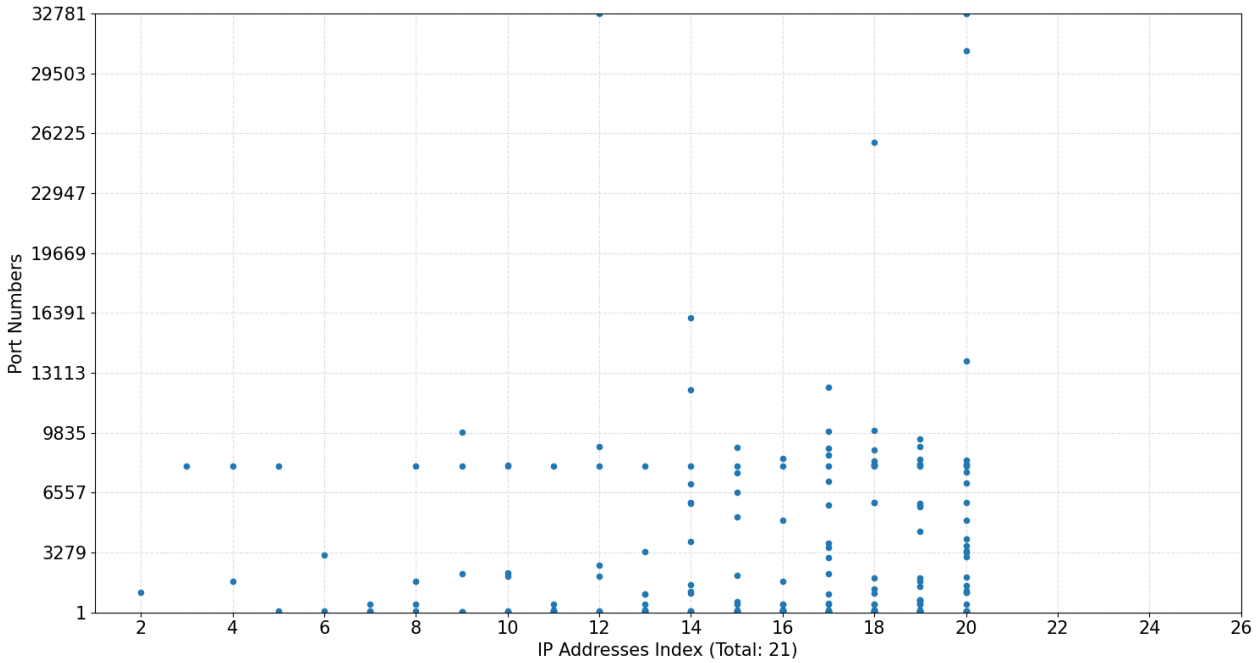

FIGURE 5.14: Port Activities Across Honeytrap IP Addresses: Data Banner on Port 8022

The port activities across IP addresses are mostly concentrated on the range (1-9,835). Some well-known ports such as 80, 443, 1433, 1723, 3306 are visible in the port activities, but not all of the ports are together in the same IP address. Also, no private ports are visible. In general, this pattern deviates from the others and it does not seem to be a warning sign for honeypots.

5.4.7 Dionaea

We, finally, display the Dionaea IP addresses, that are gathered from its port 11211 banner.

FIGURE 5.15: Port Distributions of Dionaea IP Addresses: Data Banner on Port 11211

Once more, we note that these IP addresses port distributions differ. Similar to Honeytrap, these IP addresses have a relatively small number of ports ranging from (1-18). Based on the port distribution, it seems that only the necessary ports are open, indicating a normal network configuration.

Nonetheless, it is still vital to look at each IP address port separately.

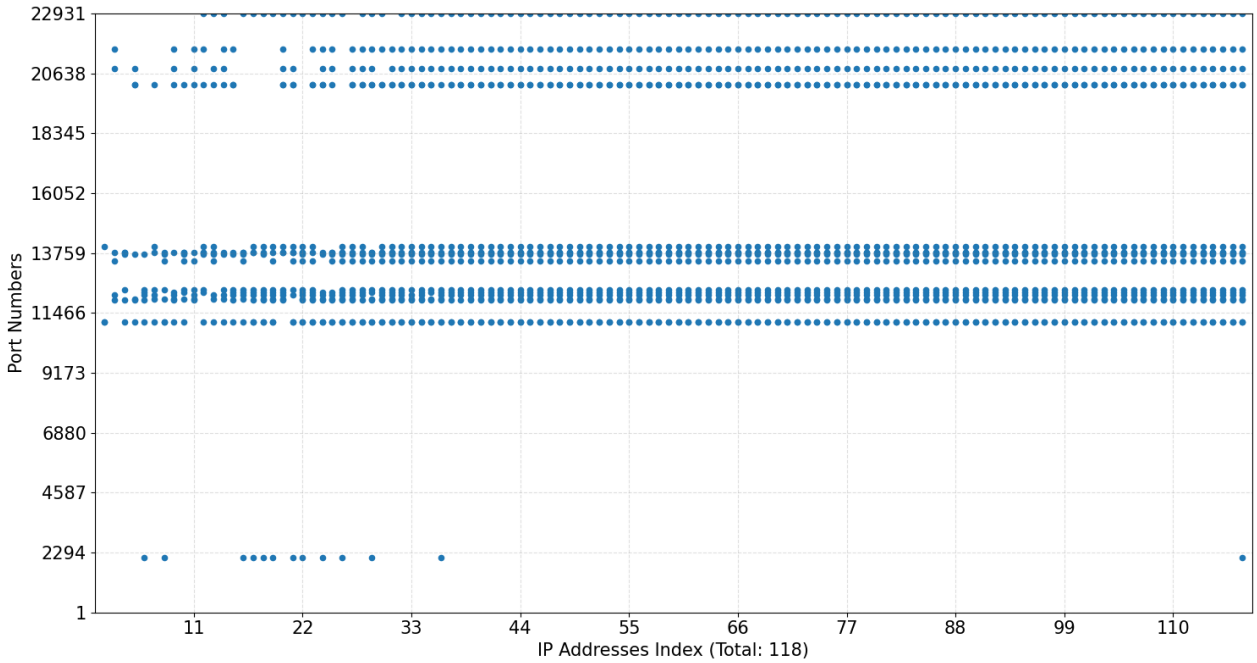

FIGURE 5.16: Port Activities Across Dionaea IP Addresses: Data Banner on Port 11211

The strange thing we observe, is that across all the IP addresses similar ports are open. Specifically, the ports are visible in the range (11,000-14,000) and (20,000-22,931). These port activities are completely different to what we expect as they not only lack ports associated with our Dionaea honeypot, but also they do not exhibit similar ports related with any other of our installed honeypots. We, therefore, believe that we may not find running in these IP addresses multiple honeypots.

In conclusion, we find that the IP addresses we collect from our honeypots Elasticsearch, Elasticpot, Ipphoney, Cowrie, Citrix show a recurring pattern of a wide number and variety of open ports. As a result, we analyze deeper these IP addresses. Regarding the IP addresses from honeypots Honeytrap and Dionaea, they do not demonstrate this unusual pattern. So, we do not look at them further.

5.5 A Deep Dive into Honeypot Behavior: Analytical Insights

We move on to the discoveries we make from the behavior analysis of the IP addresses. First, we show off the IP addresses and which additional ports they

provide based on our known honeypots. After that, we present the results of our final fingerprinting analysis. Specifically, we provide information about the correlation with IANA, as well the precise number of services that each IP address deviates from IANA standards. Last but not least, using the flagged IP addresses that came from IANA comparisons, we reveal the number of IP addresses that host multiple honeypots and which types of honeypots. Finally, we showcase the success proportion of the cross verification we execute between Shodans database and the IP addresses we identify as honeypots.

5.5.1 IP Addresses with Additional Ports from Our Honeypots

We gather for each honeypot the IP addresses that display additional known ports. We concentrate on the additional ports, which are linked to the ports from our established honeypots. Hence, we display these IP addresses and the exactly known ports we find.

From the IP addresses associated with the **Elasticsearch banner on port 9200** we are interested in 128 of them. Following the IP addresses linked to the **Elasticpot banner on port 9200,** we select 101 IP addresses. Also, 118 IP addresses from the **Ipphoney banner on port 631** stand out. As for **Cowrie banner on port 2222**, out of the total 1,636 addresses the 111 reveal multiple additional ports. Finally, from **Citrix banner on port 80** only 3 IP addresses are noteworthy.

All these IP addresses we select to investigate deeper, have a wide number of open ports that ranges from 800 to 1,000. Among the open ports, we meet some that our known honeypots use. Especially,

- **Honeytrap:** The IP addresses show open ports 5900 and 8022, which fit the characteristics of our Honeytrap honeypot.
- **Elasticsearch:** In the IP addresses is also visible the port 9200, which is the default port to emulate Elasticsearch services.
- **Dionaea:** Additionally, active ports, which associate with our Dionaea honeypot, are the ports 3306, 1433 and 1723.
- **Ipphoney:** Very important is that all IP addresses have port 631 open. This port is for Internet Printing Protocol (IPP) and is also linked to Ipphoney honeypot.

- **Citrix:** Although the ports 80 and 443 are very common, their existence may reveal Citrix honeypot.

Closing, these IP addresses have a high likelihood of honeypot activity and most importantly of multiple honeypots. For that reason, we conduct our final fingerprinting analysis to identify if these IP addresses are genuine services or honeypots.

5.5.2 Outcome of Correlation with IANA Data

The first step of the fingerprinting analysis is very important. We evaluate for each IP address the ports and services to the official records of the Internet Assigned Numbers Authority (IANA) (section 4.5.1).

The numbers of IP addresses that do not match the services that IANA listed are shown below.

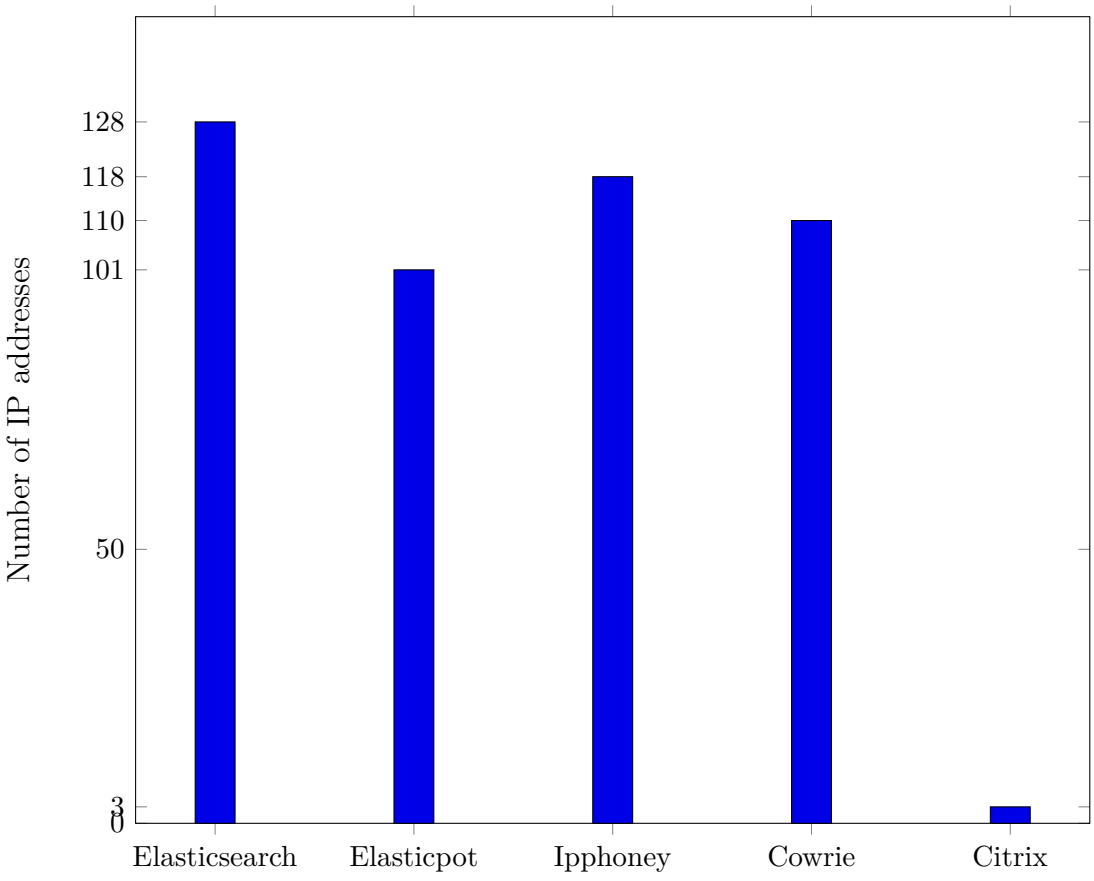

FIGURE 5.17: Number of IP addresses per Honeypot with Discrepant Services Listed by IANA

In the above figure are displayed five columns, and each one corresponds to a different honeypot. To go into details, for the **Elasticsearch honeypot**, 128 IP addresses (100%) show discrepancies in some of their services. The same is for **Elasticpot honeypot** that also has a 100% deviation. From 101 IP addresses some services deviate from IANA data. This high 100% percentage continues for **honeypots Ipphoney and Cowrie**, from which 118 and 110 IP addresses accordingly do not adhere to IANA. Last, in the **Citrix honeypot** the services of the 3 IP addresses (100%) are not the same with IANA services.

Specifically, we observe that each IP address exhibits a range of 300-470 ports that do not match. Among these differences, some ports do not comply with IANA, and others are high-numbered ports and are linked to services that remain unidentified. Such in purpose inconsistencies or unique services strongly indicate the presence of honeypots (section 1.3).

5.5.3 Insights Derived Examining The Banners of IP Addresses

The IP addresses, that are marked from the IANA correlation, are used in the second step of our fingerprinting method. For each IP address we pinpoint their banners, and compare them to the according ones from our established honeypots. The goal of this is to discover unique patterns and traits that are identical 4.5.2.

We focus on the Elasticpot and Ipphoney honeypots. We look over the 118 highlighted IP addresses that we get from Ipphoney and use an automated script to search their banners on ports 631 and 9200. From the 118 IP addresses we conclude that 91 (77.12%) of them display banners that correspond with our reference honeypots Elasticpot and Ipphoney.

The banners we find on ports 631 and 9200 are shown below.

```
HTTP/1.1 200 OK
Content-Length: 0
Upgrade: TLS/1.0, HTTP/1.1
X-Content-Type-Options: nosniff
Content-Security-Policy: frame-ancestors 'none'
Server: Lexmark_Web_Server
Connection: upgrade
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache
Date: Sun, 07 May 2023 17:14:27 GMT
X-Frame-Options: SAMEORIGIN
Content-Type: application/ipp
```

Listing 5.15: Banner: Ipphoney Dataset IP addresses with Port 631

The banner on port 631 of the highlighted IP addresses is identical with the one we find on port 631 of our Ipphoney honeypot. This similarity is confirmed, also at the start, when we query the banner on Shodan and search for IP addresses with matching banner.

```
{
"status" : 200,
"cluster_name" : "elasticsearch",
"version" : {
"lucene_version" : "4.10.4",
"build_hash" : "
   b88f43fc40b0bcd7f173a1f9ee2e97816de80b19",
"number" : "1.4.1",
"build_timestamp" : "2015-07-29T09:54:16Z",
"build_snapshot" : false
},
"name" : "Green Goblin",
"tagline" : "You Know, for Search"
}
```

Listing 5.16: Banner: Ipphoney Dataset IP addresses with Port 9200

The most important is the resemblance we find between the banner of the marked IP addresses on port 9200 and the banner of the Elasticpot honeypot on the same port. The two banners have exactly the same attributes, including the build hash, timestamp and name.

As for the remaining 27 (22.88%) IP addresses, that do not display banners, we encounter timeouts. Taking this into account, we come to the assumption that these IP addresses are also honeypots that might are compromised or turned down.

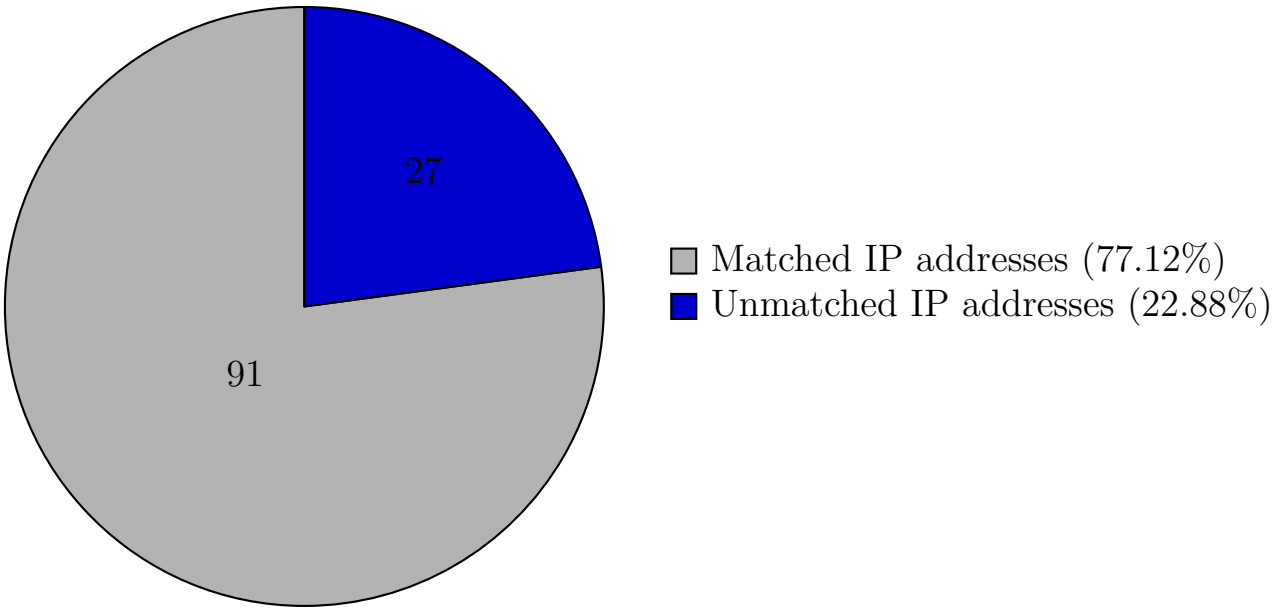

FIGURE 5.18: Matched vs. Unmatched IP Addresses as Honeypots

Given the results from the matching banners, we can declare with confidence that these IP addresses are not legitimate but they serve as honeypots, therefore verifying our assumption about the banners and headers of the services that can indicate is running a honeypot. Specifically, we strongly believe that they mimic simultaneously at least two honeypots, which are the Ipphoney and Elasticpot. This confirms, also, our hypothesis that multiple honeypots can run in a single machine. As for honeypot Ipphoney; the machines have as server the Lexmark_Web_Server, and for honeypot Elasticpot; they emulate an old version of Elasticsearch the version 1.4.1 which was published in 2015. This observation validates another of our assumptions, that honeypots may not be updated as frequent as real machines (section 1.3). We hope with these discoveries to strengthen the cyber security community and our fingerprinting technique to be used by cyber experts for identification and validation of IP addresses.

5.5.4 Results from Cross Verification with Shodan

To support our findings and minimize the false positives, we perform a cross verification with Shodan database.

During the cross verification process, we take the 91 IP addresses, which we support to be honeypots, and check them in Shodan if they are also tagged as honeypots. We discover that all the 91 IP addresses are also classified as

honeypots in Shodan. By this cross verification, our findings are confirmed with a 100% success rate, fortifying the reliability of our honeypot identification process.

Regarding the remaining 27 IP addresses, that have timeout issues, are not present in Shodan records either. This absence reinforces our hypothesis that they are also honeypots, but are compromised or completely shut down.

To sum up, the cross verification process is essential to improving the accuracy of our findings. Our fingerprinting technique receives an extra validation from the combination of the high percentage of successful verification and the absence of the unmatched IP addresses. All of these findings show the precision of our investigation, which increases our confidence in the conclusions drawn in this academic research.

Chapter 6

Limitations And Challenges

We examine, in this chapter, the limitations and challenges that affected the breadth and depth of our research. Our objective is to provide context for our investigation, and identify areas that need improvement or further research.

6.1 Limitations

During our research, we face some challenges when using the search engine **Shodan**. Although, Shodan provides an open source platform, the free version lacks the features for our needs, particularly in terms of query capabilities and accessing data.

We run into a problem, when we encounter limitations on the Shodan platfrom, about the number of times we can run queries. These restrictions make it challenging for us to gather the data for our study. Moreover, the free version of Shodan does not allow us to retrieve all the data, resulting from our queries, which affects the breadth and depth of information for analysis. In our last stage of data analysis and validation, we are also unable to utilize the tag filter, which would have assisted in automating the verification process.

To face this problem, we decide to go with a paid subscription from Shodan. This plan gives us access to retrieve the queried data. However, even though we upgrade to the paid plan, we still face difficulties with the tag filter, which remain out of reach. Consequently, automating the verification process entirely is not possible and we have to carry out the procedure manually.

To summarize, Shodan is a very valuable tool. Although, the limitations, that arise from using its free version, pose obstacles when it comes to download all

the data for our analysis. Furthermore, in the paid plans, the lack of filters adds another layer of complexity, failing to automating the data verification process.

We encounter also a problem, while working with the network scanning tool **nmap**. To prevent any violation to the IP addresses, we are examining we deliberately restrict our usage of nmap. Because nmap can perform aggressive scans, in our approach we scan each IP address only once. We opt for this method to survey ports, without causing any disruptions to the network services, linked to those IP addresses.

6.2 Challenges

Throughout the process of our analysis, we face a problem with our data. The problem is that the data are not updated over time, but they are static. This problem becomes particularly visible, when we undertake the final fingerprinting process for the 118 IP addresses from Ipphoney. The fingerprinting process is conducted about five times. On this analysis, we notice an interesting pattern: as the preliminary Shodan data grow older, there is a decrease in the number of IP addresses matching our reference honeypots. This lead us to result, that the older Shodan data, the less likely that the IP addresses correspond to one of our established honeypots.

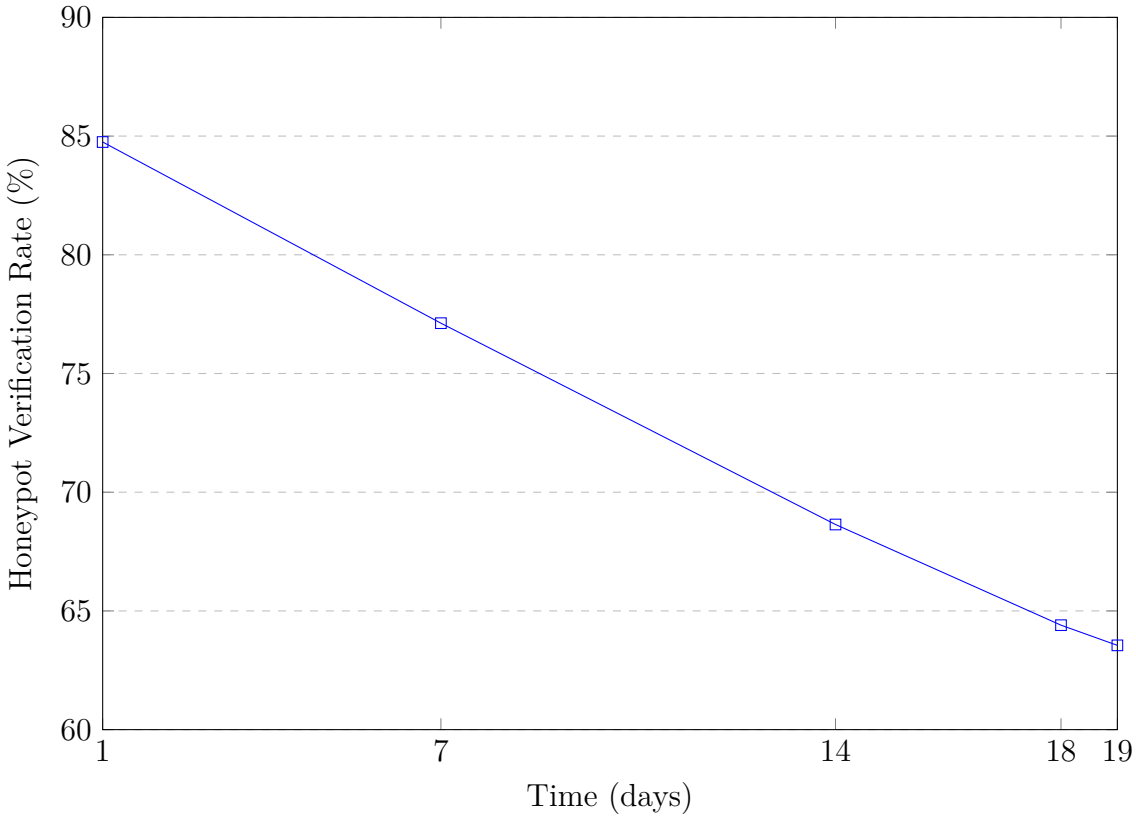

FIGURE 6.1: Honeypot Verification Rate Over Time

As we see from the graph, the percentage of matches we discover during the analysis is decreasing over time. This happens, as we said before, due to the lack of real-time data, and as a result many IP addresses shut down over time. Having access to updated data is crucial, because it will probably raise the percentage of IP addresses that match the identified honeypots. Consequently, it will improve and the results of our fingerprinting procedure.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In our research, we adopt a multi-step method to study the detection of honeypots. Initially, we set up different types of honeypots with low- and medium-interaction levels and gather information about the ports, headers, and banners, associated with each one. Next, we focus on finding, through Shodan, IP addresses that share similar behavioral characteristics. On these IP addresses, we execute port scanning again in order to reveal all the ports and their corresponding services. To delve deeper into our investigation, we conduct a final fingerprinting analysis, which is consisted of three steps. First, we examine the consistency of the services across the IP addresses, by referring to the listings provided by the Internet Assigned Numbers Authority (IANA). Through this, we discover some discrepancies that hinted at IP addresses possibly being honeypots. Follows a comparison between the IP addresses banners and headers with those from our installed honeypot. By detecting common traits, typically linked to such deceptive systems, we are able to identify IP addresses as honeypots. To strengthen our findings, a cross verification is performed using the Shodan database. The extensive indexing of internet connected devices by Shodan allows us to confirm and validate these suspected honeypots.

The results obtained through this approach are significant. A notable number of IP addresses exhibits matches with those commonly found in known honeypots. With this methodology in place our study effectively distinguishes services from honeypots. The fingerprinting technique that is developed compromises with 100% accuracy honeypots, and detects their vulnerabilities with the use of behavior characteristics.

We hope these findings to motivate cyber experts to enhance the existing weaknesses in honeypots, and ultimately boost network security.

7.2 Future Work

The results we conclude based on our study, open up different future researches in the field of detecting and analyzing honeypots. Below we outline some potential areas of interest.

7.2.1 Behavioral Analysis

The future work in this area has a lot of potential, when it comes to advancing the detection techniques for honeypots. Besides the behavioral patterns (ports, banners, headers) of honeypots we examine, the scope can be expanded by exploring also honeypots response times, packet timing and network traffic patterns. This knowledge is extremely valuable, as it can help improve the accuracy of honeypot detection methods. With these advanced detection methods, cyber experts can find the vulnerabilities of a honeypot that may reveal its presence. Essentially, behavioral analysis offers a nuanced viewpoint, which has the possibility to further improve honeypots weaknesses, in order not to be fingerprinted and therefore preserve their efficiency.

7.2.2 Expand Database Comparisons

While Shodan is accurate, it is crucial to utilize other IoT search engines and databases to cross verify our findings. This will decrease much more the false positives on identifying honeypots, and will bolster the credibility of the research.

7.2.3 Long-term Studies

In the field of research, long term studies are important. By observing behaviors and attacker strategies over a period, we gain a dynamic better perspective on how the evolve over time. These studies provide insights into how honeypots must adjust their techniques to avoid detection. This knowledge equips cyber security experts with the experience needed to stay ahead of attackers.

7.2.4 Honeypot Deployment Strategies

It is crucial for some future researches to address these weaknesses we found on honeypots and build more effective ones. This means the development of honeypots that are more difficult to detect or adapt a dynamic behavior. Ultimately, a new generation of honeypots will cause stronger network protection in the on going cyber war between attackers and defenders.

In conclusion, our study provides a basis and reveals findings on identifying honeypots. Nevertheless, it is crucial for researchers to make advancements. As analyzed before, they can explore additional behaviors characteristics, expand data sources, conduct longer studies and enhance the design of honeypots. This will boost the effectiveness of honeypots and cyber security.

References

[1] Lukáš Zobal and Dušan Kolář Radek Fujdiak. "Current State of Honeypots and Deception Strategies in Cybersecurity". In: (Oct. 2019). DOI: 10.1109/ICUMT48472.2019.8970921. URL: https://ieeexplore.ieee.org/abstract/document/8970921.

[2] Michail Tsikerdekis et al. "Approaches for Preventing Honeypot Detection and Compromise". In: (Oct. 2018). DOI: 10.1109/GIIS.2018.8635603. URL: https://www.researchgate.net/publication/328430317_Approaches_for_Preventing_Honeypot_Detection_and_Compromise.

[3] Nitin Naik et al. "Honeypots That Bite Back: A Fuzzy Technique for Identifying and Inhibiting Fingerprinting Attacks on Low Interaction Honeypots". In: (Oct. 2018). DOI: 10.1109/FUZZ-IEEE.2018.8491456. URL: https://ieeexplore.ieee.org/document/8491456.

[4] Blake Anderson and David McGrew. "OS fingerprinting: New techniques and a study of information gain and obfuscation". In: (Oct. 2017). DOI: 10.1109/CNS.2017.8228647. URL: https://www.researchgate.net/publication/321999327_OS_fingerprinting_New_techniques_and_a_study_of_information_gain_and_obfuscation.

[5] Juan David Guarnizo et al. "SIPHON: Towards Scalable High-Interaction Physical Honeypots". In: (Apr. 2017). DOI: 10.1145/3055186.3055192. URL: https://dl.acm.org/doi/abs/10.1145/3055186.3055192.

[6] Vincent Nicomette et al. "Set-up and deployment of a high-interaction honeypot: experiment and lessons learned". In: (June 2010). DOI: 10.1007/s11416-010-0144-2. URL: https://link.springer.com/article/10.1007/s11416-010-0144-2.

[7] E. Alata et al. "Lessons learned from the deployment of a high-interaction honeypot". In: (Oct. 2006). DOI: 10.1109/EDCC.2006.17. URL: https://ieeexplore.ieee.org/document/4020829.

[8] Georg Wicherski, Florian Weingarten, and Ulrike Meyer. "IP agnostic real-time traffic filtering and host identification using TCP timestamps".

In: (Mar. 2014). DOI: 10.1109/LCN.2013.6761302. URL: https://ieeexplore.ieee.org/document/6761302.

[9] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. "Remote physical device fingerprinting". In: (2005). DOI: 10.1109/TDSC.2005.26. URL: https://ieeexplore.ieee.org/document/1453529.

[10] Alexander Vetterl and Richard Clayton. "Bitter Harvest: Systematically Fingerprinting Low- and Medium-interaction Honeypots at Internet Scale". In: (Aug. 2018). URL: https://www.usenix.org/conference/woot18/presentation/vetterl.

[11] Nitin Naik and Paul Jenkins. "Discovering Hackers by Stealth: Predicting Fingerprinting Attacks on Honeypot Systems". In: (Oct. 2018). DOI: 10.1109/SysEng.2018.8544408. URL: https://ieeexplore.ieee.org/document/8544408.

[12] Wonkyu Han, Ziming Zhao, and Adam Doupé. "HoneyMix: Toward SDN-based Intelligent Honeynet". In: (Mar. 2016). DOI: 10.1145/2876019.2876022. URL: https://www.researchgate.net/publication/299794727_HoneyMix_Toward_SDN-based_Intelligent_Honeynet.

External Links

[13] "Best Honeypots for Detecting Network Threats". In: (). URL: https://securitytrails.com/blog/top-honeypots.

[14] "Iana". In: (). URL: https://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority.

[15] "Shodan". In: (). URL: https://en.wikipedia.org/wiki/Shodan_(website)/.

[16] "Nmap". In: (). URL: https://en.wikipedia.org/wiki/Nmap.

[17] "Netstat". In: (). URL: https://en.wikipedia.org/wiki/Netstat.

[18] "Fingerprinting". In: (). URL: https://www.firewalls.com/blog/security-terms/os-fingerprinting/.

[19] "A Comprehensive Guide to Banner Grabbing". In: (). URL: https://emeritus.org/blog/cybersecurity-banner-grabbing/.

[20] "What is fingerprinting in cyber security?" In: (). URL: https://securitytrails.com/blog/cybersecurity-fingerprinting#content-what-is-fingerprinting-in-cyber-security.

[21] "How To Install and Configure Elasticsearch on Ubuntu 22.04". In: (). URL: https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-elasticsearch-on-ubuntu-22-04.

[22] "Elasticpot Installation Guide". In: (). URL: https://github.com/bontchev/elasticpot/blob/master/docs/INSTALL.md.

[23] "Ipphoney Installation Guide". In: (). URL: https://github.com/bontchev/ipphoney/blob/master/docs/INSTALL.md.

[24] "Installing Cowrie in seven steps". In: (). URL: https://cowrie.readthedocs.io/en/latest/INSTALL.html.

[25] "Citrix ADC (NetScaler) Honeypot". In: (). URL: https://github.com/haxrob/citrix-honeypot.

[26] "Honeytrap Installation Guide". In: (). URL: https://github.com/armedpot/honeytrap/blob/master/INSTALL.

[27] "Dionaea Installation". In: (). URL: https://dionaea.readthedocs.io/en/latest/installation.html.