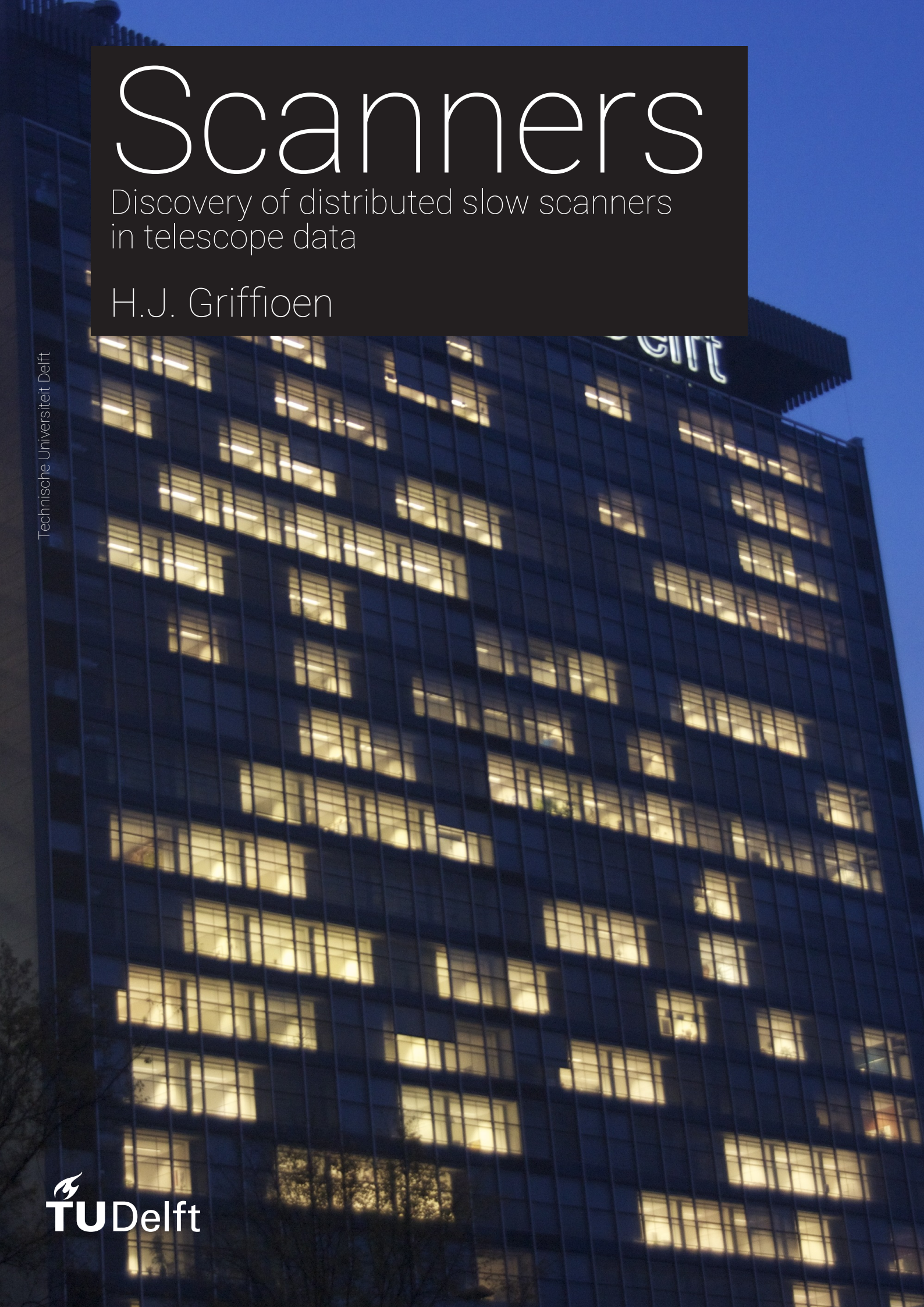


Scanners

Discovery of distributed slow scanners in telescope data

H.J. Griffioen

Technische Universiteit Delft

TU Delft

Scanners

Discovery of distributed slow scanners in telescope data

by

H.J. Griffioen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on October 30, 2018

Student number: 4303598
Project duration: November 1, 2017 – October 30, 2018
Thesis committee:

Dr. C. Doerr,	TU Delft, supervisor
Dr. ir. J. van der Lubbe,	TU Delft, chair
Prof. Dr. K.V. Hindriks,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

If you only do what you can do, you will never be more than you are now.

— Master Oogway, *Kung Fu Panda*

This thesis is the result of the research that I have started ten months ago. During this time I have obtained a thorough understanding of the state of the art in threat intelligence, as well as an increase in problem solving skills. The thesis has been written in partial fulfillment of the Degree Master of science from the Delft University of Technology.

This paper is the result of a long process of encountering and overcoming problems, as well as spending time on possible solutions that not worked out. I would like to thank everyone that has helped and motivated me during this time.

Also, I would like to give special thanks to my supervisor Dr. Christian Doerr, together with ir. Vincent Ghiette, for their excellent guidance during these past months.

Finally, I would also like to thank you, the reader, for taking the time to pick up and start reading my thesis.

H.J. Griffioen
Delft, July 2018

Abstract

The internet is rapidly growing, and with it grows the number of malicious actors. For many attacks, the attacker first scans the internet to detect vulnerable devices. In order to evade detection, the attacker distributes the scanning over a large number of machines. Because attackers are distributing this scanning and there is no way to find these scanners, we have no knowledge of what groups are actually scanning the internet and what they are up to. This thesis proposes a method to identify and fingerprint these distributed scanning groups. It does so in order to detect and analyze slow scanning groups that are actively trying to remain undetected by companies. The data used for this thesis originates from a large network telescope operated by the TU Delft, which contains packet data aimed at the TU delft network range. First, this data is analyzed in detail and several patterns are discovered. Using the analysis, a method is created to cluster the dataset without losing critical information needed to identify scanning groups. After the clustering, the resulting smaller datasets are analyzed in more detail. To do this post-processing, a new method of analyzing scanning behavior is created. This method is called XOR-analysis and works by looking at different patterns that scanners use to re-identify their packets. From the analysis, groups are extracted and fingerprinted. These fingerprints can ultimately be used as Indicators of Compromise to detect and mitigate scanning behavior in order to deny adversaries the possibility to learn about weaknesses of a system.

List of Figures

List of Tables

Contents

1 Introduction

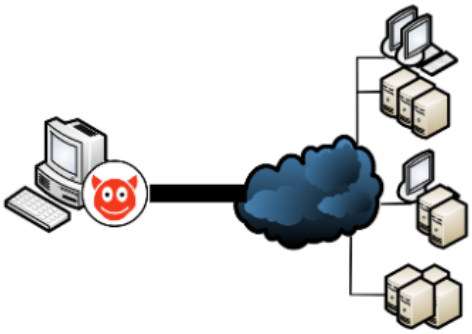
The internet is a communication medium to which most of the world is connected. There are so many companies, citizens and governments dependent on the internet that it is unthinkable to do without it. The internet has grown so big, that the last available IPv4 block has been assigned [1]. To put this into perspective, there are 2^{32} possible IPv4 addresses. This totals 4.294.967.296 public IPv4 addresses. Not all of these IPv4 addresses are used, as there are some ranges that are reserved. The total amount of public IPv4 addresses is around 3.7 billion [5]. This number might seem to represent the total amount of devices that can be present on the internet, but this is not true. Using Network Address Translation (NAT) [9], the possibility exists to connect multiple devices to the internet via one IPv4 address. Therefore the possibility exists to have more than 3.7 billion devices present on the internet, which is already the case [4].

Because the internet is so widespread that virtually everyone is connected with it, the possibility arises to "attack" companies, citizens or governments even on the other side of the earth. This gives rise to a new field of criminals, so called cyber-criminals. These criminals have incentive to hack or disrupt devices connected to the internet. These incentives could be monetary, but also political. Oftentimes, systems even get hacked just for fun.

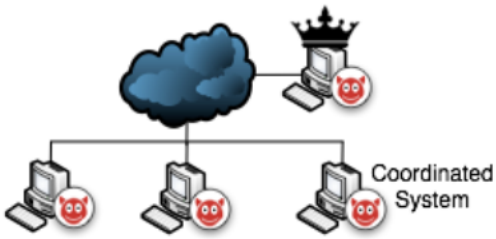
In the past, most attacks that were conducted on the internet were targeted on a specific target. Information was gathered about the target and this information was used in order to conduct an attack that was tailored for that purpose. With the rise of the Internet of Things and the millions of devices connected to the internet, the objective of attackers changed. Because the devices are often using the same protocols, they became a large surface of attack. An attacker does not have to gather information about one specific company, but can attack common protocols if an exploit is available. This gives the attacker opportunity to infect and possibly control a large number of devices, while using a single exploit. These devices can in turn be used by the attacker to conduct Denial-of-Service attacks or for other malicious purposes.

If an attacker has found a weakness in a protocol and has developed an exploit, the attacker still needs to know where devices are located that are vulnerable to this attack. This is done by scanning the internet for the specific protocol that is going to be attacked. These protocols often use specific ports in order to connect to the internet. The attacker can then scan this port to see if a device is running that uses this protocol. In some cases, the attacker might not even need an exploit. In the case of the Mirai botnet, a large number of devices became infected because they were using very weak or default passwords [7]. The Mirai botnet was afterwards used to conduct large DDoS attacks. This stands to show the severity of the threat that these botnets and exploits are posing.

Exploit or not, the commonality in this attack pattern is that every adversary has to scan the internet in order to find possible vulnerable devices. Detecting these attacks in an early stage might help in the mitigation of these attacks.



(a) Single source scanner scanning a lot of devices. This type of scanner is easily detected by an IDS.



(b) Multi-source scanner, different scanners are working in co-ordinated fashion. This type of scanner is very hard to detect because individually, the scanners remain under the thresholds of an IDS.

Figure 1.1: Two types of scanners

It is clear that organizations do not want their network scanned so people can find vulnerabilities. Therefore, scanning traffic is considered by intrusion detection systems (IDSs) as a threat, and these systems will drop scanning traffic when it is detected. Scanning detection at the moment, detects scans that are mostly originating from single sources that send a lot of packets in a small time-window (Figure 1.1a). Because attackers got detected by IDSs, they 'evolved'. Where they would first use only one host to scan at a very high speed, now they distribute this scanning over a large range of IP addresses. By doing this, all the IP addresses will only send a handful of packets, which remains under the threshold of an IDS. Therefore, scanners are currently remaining undetected, and there is not yet a reliable way to identify scanning groups from their distributed sources. Figure 1.1b shows a coordinated scanning system, in which different scanning machines are controlled by one host. This thesis will focus on detecting these slow and distributed scanning groups that are scanning the internet in a stealthy manner.

Cyber threat intelligence

Cyber threat intelligence (CTI) aims to anticipate the moves of an attacker. To do so, information is needed that allows the defending party to investigate and analyze the attacker in order to draw a sensible conclusion about the behavior of the attacker. CTI is not yet a large research area. However, with the current threats that are present in the cyberspace, the need for research in this area is high. From a defensive point of view, information is the best resource that is available. Which is what CTI aims to provide. Therefore, this thesis aims to contribute to cyber threat intelligence research by devising a novel way to classify scanners according to their behavior. The information that will be obtained by doing so can later be connected to various attackers or attacking groups.

The ultimate goal of CTI is to stop all attacks. While this is infeasible to achieve entirely, CTI can be used to make it more challenging for an attacker to achieve his objectives. The pyramid of pain (figure 1.2) shows how hard it is for an attacker to adapt given a certain defensive measure. When defending on the "domain name" level for example, it is simple for an attacker to circumvent this defense. Moving further up the pyramid will create the hardest challenge for attackers and price out most attackers of the system. The goal of CTI should therefore be to disrupt an attacker in his Tactics, Techniques and Procedures (TTPs). These TTPs are located in the very top of the pyramid. A TTP is related to the behavior of an attacker, disrupting these TTPs means that the attacker has to change the way in which he operates. In this study, we will try to contribute to CTI in order to advance the move towards disrupting the TTPs of attackers on a behavioral level.

Kill chain model

A way of analyzing cyber threats is using the cyber kill chain model, which has been presented by Lockheed Martin in 2011 [43]. The cyber kill chain model is visualized in figure 1.3. The model consists of seven stages. If an attacker is detected in any of these stages, the attacker can be stopped. It is obvious that stopping an attacker in an early stage is more preferred than stopping an attacker in a later stage. If an attacker is stopped before any damage can be done, the attack will be unsuccessful. Therefore, it is vital to detect attacks as early as possible.

Scanning belongs to the first stage of the cyber kill chain, the reconnaissance stage. An attacker has to

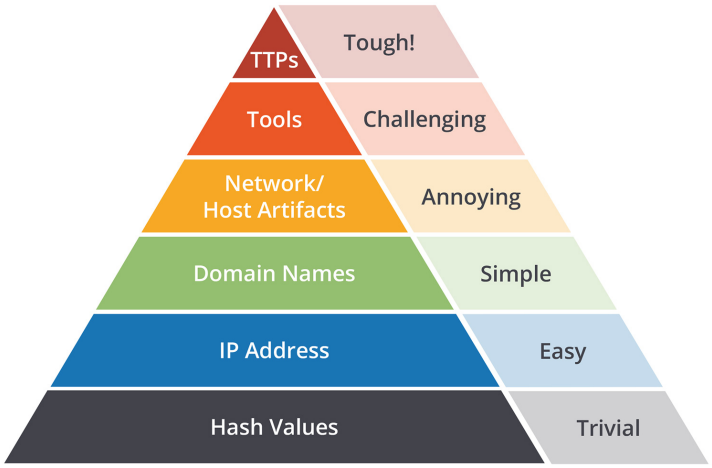


Figure 1.2: Pyramid of pain [3].

Figure 1.3: Cyber kill chain model.

gain information about the system, which can be used to conduct an attack. Detecting an attacker on the scanning level can stop the attack completely before the attacker can gain access to the system. While it is easy to understand that detecting attacks on this level can help strengthen the security of a company or government, it is currently not used in defensive strategies. The research in this area is still very limited, which means that with the current state of research, implementing a system on this level is infeasible. This thesis is relevant because detecting scanners would mean that attacks can be detected earlier on in the kill chain, the information about these attacks can then be shared as Cyber Threat Intelligence to create a more resilient cyberspace.

In order to detect and classify scanners, this thesis work studies network behavior using a network telescope. A network telescope is a set of IP addresses that do not provide any service and are not being used. All traffic that is sent to these addresses is logged. Using this so-called telescope, a method for classifying scanners based on their behavior is presented, as well as a method that can be used to find very slow scanning groups in high volumes of data. The questions this thesis aims to answer are the following:

RQ1. *How can telescope data be processed in such a way that slow distributed scanning groups can be detected?*

RQ2. *Which indicators can be extracted from the telescope data that can be used to identify and mitigate scanners?*

The thesis consists of several chapters explaining the thought process and results of this study. In chapter 2, background information is given that is required for understanding this thesis. The state of the art will be discussed in chapter 3. How the data is collected and which specific parts of the data are used is explained in chapter 4. Using the data that is gathered, an analysis is made in chapter 5. After analyzing the data, a methodology of how to process and analyze the data further is explained in chapter 6. Results obtained are listed and evaluated in chapter 7. Finally, chapter 8 will conclude this thesis and identify shortcomings for future work.

2

Background information

The goal of this chapter is to give a basic understanding of how the Internet works, and what Internet scanners are. The infrastructure of the Internet is explained, and parts of the infrastructure that are important for understanding the rest of this thesis are elaborated upon.

2.1. Internet

This section gives a concise explanation about the inner workings of the Internet. The Cambridge dictionary defines the Internet as follows: "Internet is the large system of connected computers around the world that allows people to share information and communicate with each other" [6]. The Internet allows people to communicate using various protocols. One of these protocols is highlighted in this section as it will be used as the object of study in this thesis.

2.1.1. Open System Interconnections Reference Model (OSI)

The OSI model was the first step in the international standardization of Internet protocols. The protocols associated with the OSI model are not used anymore, but the model is still very relevant today. The OSI model consists of seven layers, as depicted in Figure 2.1. In practice however, the presentation and session layers, which are the 5th and 6th layer, are merged with the transport layer. This results in a model of five layers [49].

The first layer of the OSI model is the physical layer. This layer represents the physical infrastructure of the Internet, and includes all cables, routers and other physical mediums. The second layer is the data-link layer. This layer takes the information that is coming in on the physical layer and transforms it to data frames. The third layer is the network layer, this layer is responsible for routing and ensuring that packets reach the right destination. After the network layer comes the transport layer. This layer transforms the data it receives from higher layers into small portions (packets) and sends it over to the network layer. After the network layer routes the packets to their destination, the transport layer on the other end has to piece all the information back together. The session layer is the fifth layer and is responsible for the end-to-end session of a client. The sixth layer is the presentation layer, which 'presents' the data to the next layer. The presentation layer is sometimes called the 'syntax layer'. The application layer contains protocols that are needed by users. For example, the HTTP protocol is used for the World Wide Web. This protocol can be used to request and transmit web pages [49].

2.1.2. TCP/IP Reference Model

The TCP/IP reference model also defines layers, but the layers are less strict than in the OSI reference model. This model uses four layers, which can be loosely compared to some of the layers in the OSI reference model, as depicted in Figure 2.2.

The strength of this model are the protocols that are built on top of it, the name TCP/IP is even named after its two primary protocols. The TCP/IP reference model is useful because of the various standardized protocols that are defined for the model, with the primary two protocols being of course the TCP protocol and IP(v4) protocol.

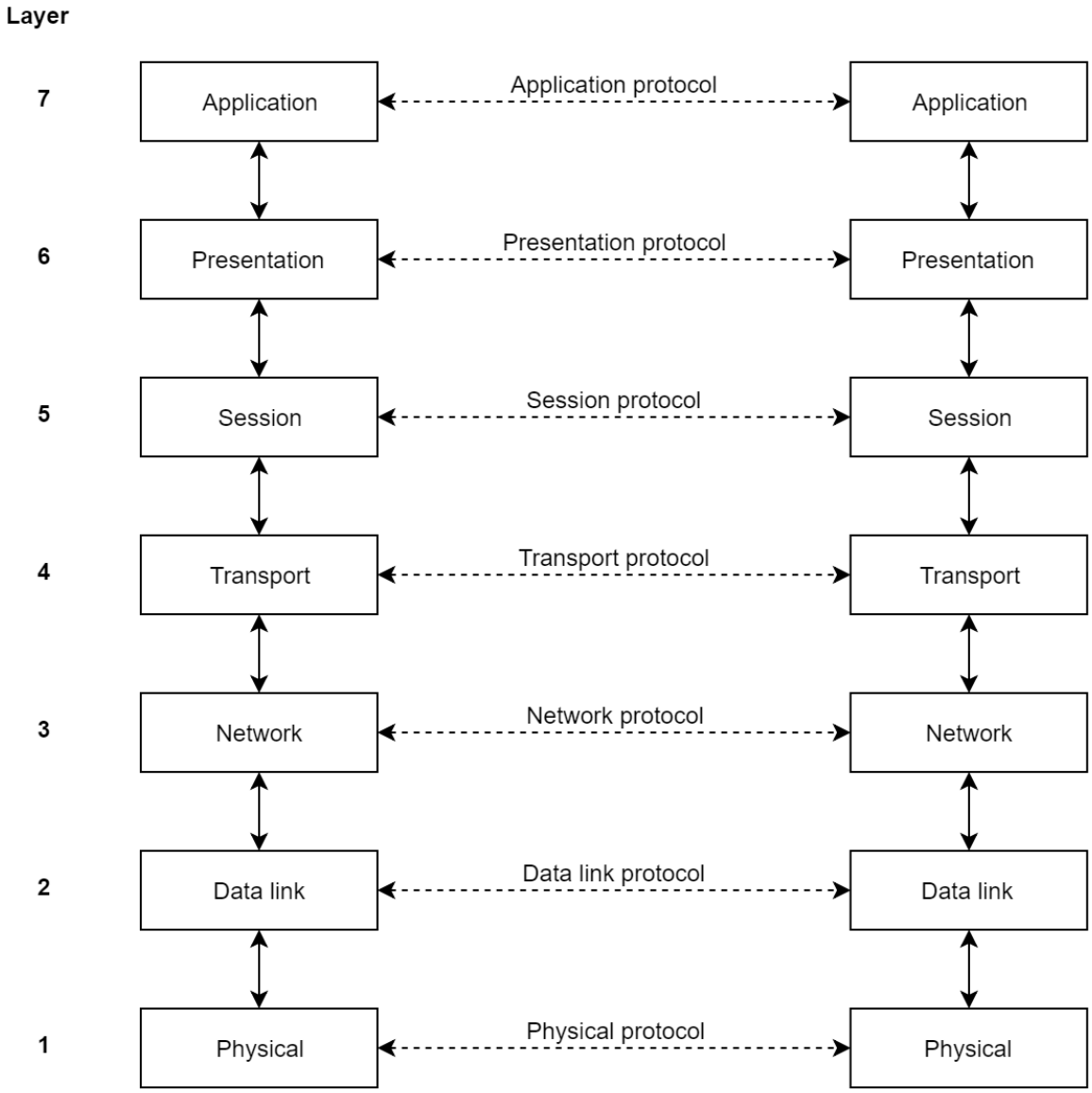


Figure 2.1: The Open System Interconnections Reference Model.

2.1.3. The model used in this thesis

In this thesis, the same model that Tanenbaum [49] uses in his book shall be used. This reference model is a combination of both the OSI and TCP/IP reference model. The reference model can be seen in Figure 2.3. In addition to this model, the IP and TCP protocols shall be extensively used. These protocols are discussed in Section 2.2 and 2.3.

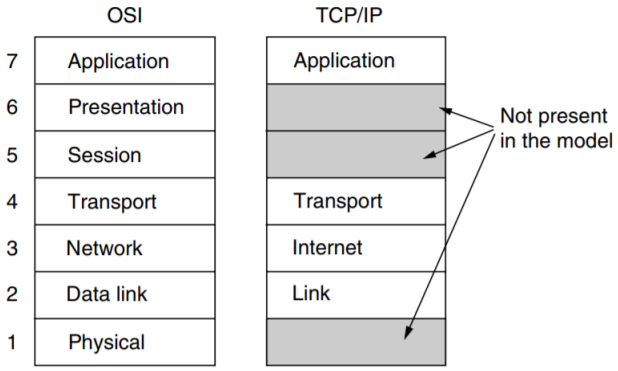


Figure 2.2: The TCP/IP reference model [49].

2.1.4. Network layer and transport layer

The most important layers that we are dealing with in this thesis are the network and transport layer. As discussed before, the network layer has to ensure that the IP packets are delivered to the right destination. In order to accomplish this, the network layer defines an official packet format and protocol. This protocol is called the Internet Protocol (IP). The most prevalent Internet Protocol that is used today is Internet Protocol version 4 (IPv4) [10]. In this thesis we will look at this protocol. IPv4 is explained in further detail in section 2.2.

The transport layer allows two hosts to communicate with each other. This layer can ensure same-order delivery, which the network layer cannot. There are two major protocols on this layer, the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). The main difference between the two protocols is that UDP is a connectionless protocol, whereas TCP is not. Since most scanning traffic originates from TCP, only TCP data has been analysed. UDP traffic is considered out-of-scope for this thesis. TCP is explained in more detail in Section 2.3.

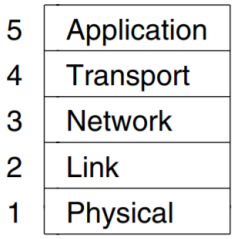


Figure 2.3: Reference model as used by Tanenbaum [49].

2.2. Internet Protocol version 4

The Internet Protocol defines a packet format that allows the network layer to read and understand what it has to do with the packet. Because the network layer is responsible for routing the packet, it contains all routing information that is needed to ensure that the packet reaches its destination. The IPv4 packet format consists of a header and a payload. The header contains the information for the network layer, whereas the payload consists of data that is used in the other layers. The format of the IPv4 header as specified in RFC 791 can be seen in Figure 2.4.

Some of the header fields are explained below, only the most interesting fields will be discussed:

- **Source address**. This is the address of the sender of the packet. Because the sender has full control over the packet, it is possible to change this address so that the receiver does not know the real address of the sender. For this thesis we will not see this as a problem, as we will explain in Section 2.4.
- **Destination address**. The destination address is the address that is supposed to receive the packet.
- **Identification**. This value represents the IP identification number (IP ID). This number helps to reassemble a stream of packets. This number is randomly chosen for packets that are not related to one another, therefore it can be considered random.
- **Time To Live**. The time to live value determines how many 'hops' a packet is allowed to make through the network. When the packet travels through the network, each device that it encounters will decrease this value by one. When the value reaches zero, the packet is discarded and the transmission has failed.
- **Protocol**. The protocol field indicates which protocol is used for the next layer.
- **Version**. In the version field, the IP version is stated. For IPv4 this field will always be equal to 4, and as we are not considering IPv6 in this thesis, this field will be 4 in all the data that we will look at.

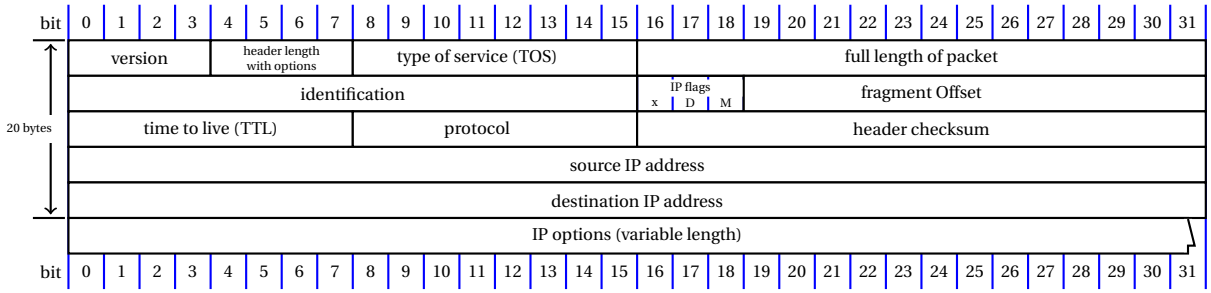


Figure 2.4: IPv4 header as defined in RFC 791 [10].

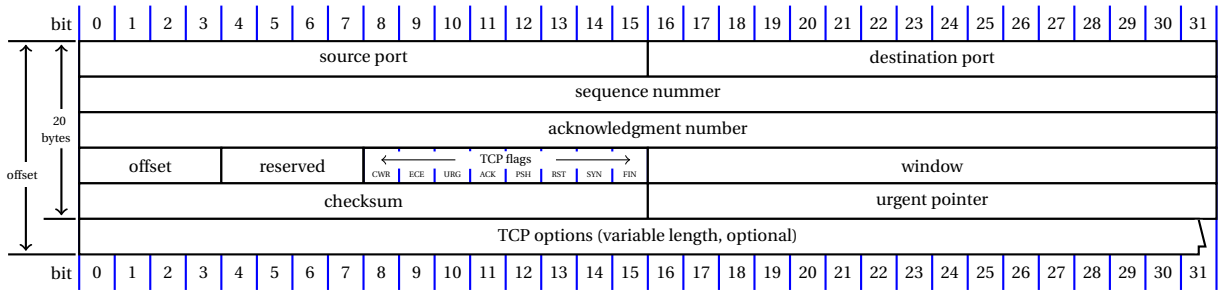


Figure 2.6: TCP header as defined in RFC 793 [12].

- **Header checksum**. The header checksum field is used to determine if the packet is corrupted. This can either be done by malicious actors or occur due to transmission errors.

As Figure 2.4 shows, both the source and destination addresses of the packet consist of 32 bits. Most people however, will associate IP addresses with their number notation, as that is more commonly used. The 32 bits are essentially four bytes, representing the bytes individually gives the commonly known representation of the IP address. An example can be seen in Figure 2.5.

The other data present in the IP header has not been used in the analysis. These values have not found to be different between various scanners and therefore they are not useful in the detection of scanners.

2.3. Transmission Control Protocol

TCP is a protocol that enables users to have reliable communication over the Internet. For received packets, the receiver will send acknowledgements so the sender knows that the packet has been successfully transmitted. A TCP packet consists of a header and a payload, a split which is similar to the IPv4 packet. The format of the TCP header is included in Figure 2.6. TCP has been standardized in RFC 793 [12].

The values in this header used in this thesis are the source and destination ports, the sequence number and the flags of the packet. The fields represent the following:

- **Source port**. The source port is the network port which the sender of the packet chose to send the packet on. This is also the port where the sender is listening on for a reply.
- **Destination port**. The destination port is the port on the IP address that receives the packet. If the port is in use, it means that there is a service running on the machine that uses this port.
- **Sequence number**. In TCP, the sequence number is a way to track a session. When the first packet of a session is sent, a random sequence number is sent with it. Consecutive packets can be tracked

Bit representation	`10000010101000011000001010101100`
Byte representation	`10000010.10100001.10000010.10101100`
Number representation	`130.161.130.172`

Figure 2.5: Representations of an IP address.

using the sequence number and acknowledgement number, which change in specific ways. After the first packet is received, the receiver responds with a packet in which the acknowledgement is equal to the sequence number increased by one. The other ways in which the sequence and acknowledgement numbers change is not important for this thesis.

- **Flags**. Flags represent control bits, identifying what the purpose of the packet is. These flags will be used in this thesis in order to select specific packets. The different flags and their uses are explained in Section 2.3.1.

For more information about the fields in the TCP header, please refer to [12].

2.3.1. Flags

As can be seen in Figure 2.6, there are different flags present in the TCP header. These are control bits that can be used to identify a packet. The different flags are explained in this section.

SYN The synchronize flag means that the sender of the packet wants to start a new session with the receiver.

ACK The acknowledgement flag means that the receiver has successfully received the packet.

URG If the urgent flag pointer is set, the packet should be treated with urgency by the network it is transported on.

PSH The push flag indicates whether or not the packet can be buffered at the receiver. If the push flag is set, the packet is pushed to the application immediately.

RST The reset flag indicates that the connection between the sender and receiver should be reset.

FIN When the fin flag is sent and received, both systems know that the session is over and can close it.

The most interesting packets in regards to scanning are the packets where the SYN flag is present. The reasoning behind this is given in Section 2.4.2.

2.3.2. Three-way handshake

In order to create a session, TCP uses what is called a “three-way handshake”. This handshake ensures that two hosts agree on the connection and initialize the sequence and acknowledgement numbers that are used to keep track of the session. The visual representation of the three-way handshake as given in RFC 793 [12], can be seen in Figure 2.7.

From the figure, it can be seen that there are three steps involved in order to establish a connection. These are steps 2, 3 and 4. In step 1, there is no connection. However, it can be seen that Host B (TCP B) is listening. This means that Host B is listening on a specific port. Host A (TCP A) is not listening and has a closed port instead. Because Host A wants to connect to Host B on the listening port, Host A initiates the three-way handshake. This starts out with what can be seen in step 2. Host A sends a synchronization packet to Host B on the listening port. Host B, receiving this packet, sends a packet back to Host A with the synchronization and acknowledgement flags set. This means that Host B has received the synchronization request. Note also, that the acknowledgement number here is the previously received sequence number plus one. When Host A receives this packet, Host A sends an acknowledgement to Host B, meaning that the packet from step 3 was received. This completes the three-way handshake and enables Host A to send data to Host B in step 5.

If the three-was handshake is not completed, Host B will not read the data that Host A is sending. This happens because TCP is a session based protocol that keeps track of the state of connections. If no packets have been sent for a certain period of time, the connection will be revoked and the three-way handshake has to be done again for the hosts to send data to each other.

If a host does receive unsolicited traffic, it can respond in a number of ways. This is mostly dependent on whether the port is open or closed and the configuration of the system. Some systems choose to drop the packet and do nothing, while others send a reset packet to the original sender of the unsolicited request.

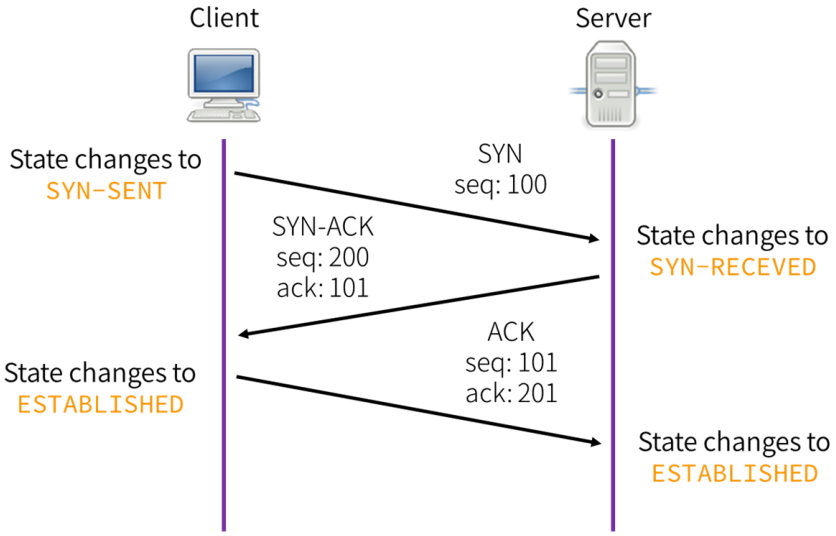


Figure 2.7: TCP three-way handshake [35].

2.4. Scanners

Another concept in need of explaining is the concept of scanners. By scanners we mean programs that scan single or ranges of IP addresses for open ports. This can be done by a variety of people, including researchers and security companies. In this thesis however, we are interested in malicious internet users that conduct port scans. Conducting port scans gives malicious actors insight in systems and potential vulnerabilities. This information can be used in order to exploit and infect a system. Therefore, detecting these port scans in an early stage gives valuable information about attackers and the services that they are interested in.

2.4.1. Scanning strategies

There are several high level strategies of scans. These scans do not only differ in their execution, but also in their objective. This section will highlight the three different scanning strategies and what the objective of an adversary executing these scans could be.

- **Vertical scan**. The vertical scan is a type of scan that is used to probe an entire system. This means that every port (or at least a large portion) of a single host or IP address is scanned. Vertical scans are often used if an attacker wants to exploit a particular system. It is the scan type that gives the most information about a system, as it scans the entire system.

- **Horizontal scan**. Horizontal scanning means that a large range of IP addresses is being scanned on a single port. This does not give the amount of information about a system that a vertical scan provides, but the scanner only has to send one packet to each IP address. Sending only one packet means that it is easier to stay undetected as the volume is low. It also means that the scanner is faster as it does not have to scan each port on a system. The objective of an adversary using this scan is to find one particular service in order to exploit it. This could mean that the adversary has a working exploit against the service on a particular port, or knows that there are many devices running the service with default or weak passwords.

- **Block scan**. Block scanning is a combination of horizontal and vertical scanning. Instead of scanning only one port, the adversary scans a range of ports. The objective is the same as with the horizontal scan, but the adversary now scans for multiple services. Most likely because the adversary has multiple exploits that target different services. For example scanning port 80, 8080 and 443 to find web servers on 1000 hosts.

In the past, vertical scanning was the most prevalent. However, with the rise of IoT and the increasing number of devices, the incentive to find as many exploitable devices as possible became higher. Therefore, the current scanning landscape shows a lot of horizontal scans that are scanning the entire Internet. The focus in this thesis will be on these horizontal scans that are scanning a large number of IP addresses.

2.4.2. Types of scans

Port scanning can, apart from the strategy, also be done in a number of ways. There are different techniques that can be used in order to detect whether a port is opened or closed. This section will list some of the techniques that are being used in order to scan the Internet for open ports.

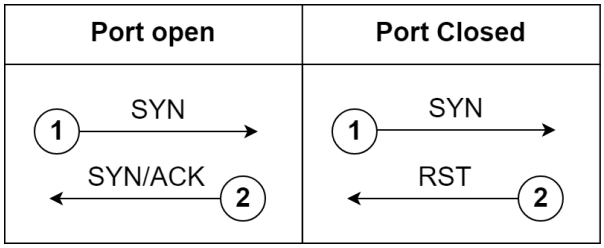


Figure 2.8: SYN scanning

The first and most prevalent technique that is being used in TCP scanning is the SYN scan. This scan uses the working of the three-way handshake protocol as described in Section 2.3.2. Using this technique, the scanner sends a SYN packet, and waits for the scanned host to return the SYN-ACK packet that the three-way handshake protocol returns. If the port is closed, the scanned host will send a RST packet in return, as the port is not open for connection. Therefore, the SYN is unsolicited. This can be seen in Figure 2.8. As can be seen in the figure, if the scanning host (Host 1) sends a SYN packet, the scanned host (Host 2) responds differently when the port is closed as opposed to when the port is open.

The second scanning technique that we will discuss is the FIN scan. FIN scanning relies on (as the name suggests), the FIN flag of a packet. RFC 793 states that a packet to a closed port should always be replied to with a RST, whereas if the port is open and the received packet does not have the SYN, ACK or RST flag set, it should be dropped [12]. Therefore, when sending a FIN flag to an open port, there should be no response. A closed port however, does respond. The FIN scan is depicted in Figure 2.9. In some more recent implementations of the TCP/IP stack, the FIN scan does not work anymore. When implementing this stack, one can simply make both responses the same in order to not disclose this information.

There are many other types of scans, and FIN scanning is only one of the examples of a scan that is different from a more traditional and straightforward SYN scan [23]. In this thesis, for the sake of simplicity, we will mostly focus on the SYN scan. The algorithms and mechanisms to detect SYN scans will also be applicable to most other types of scans, so there is no need to go in to every other scanning type separately.

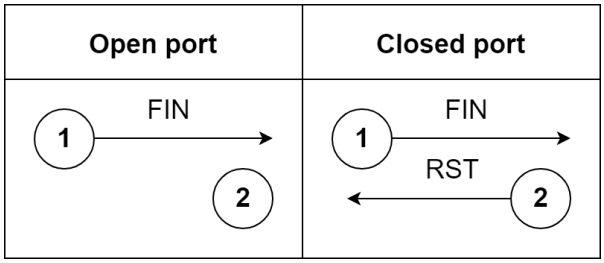


Figure 2.9: FIN scanning

As can be seen from Figures 2.8 and 2.9, the scanner has to be able to receive the response of the server. Therefore, it is hard to spoof the source IP address, which is present in the IP header (Figure 2.4), to show an address which is not owned by the attacker. Because in doing so, the attacker would not be able to receive the response generated by the server.

2.4.3. Types of scanners

As with almost all types of tools, there are multiple tools available that can be used in order to conduct a port scan. The most well known and commonly used scanning tools are called NMap, Zmap, Masscan and Unicorn [24, 33, 41, 44]. The thesis of ir. Vincent Ghiette [32] shows the difference in behavior that these scanners have and the different purposes for which they are commonly used. These tools make up most of the scanners that are scanning the Internet and have very specific fingerprints. In this thesis however, we are interested in different persons or groups that are scanning the Internet. Multiple groups can be using the same tool, in which case the previous work is not sufficient in detecting the different scanning behaviors. Most of the available and commonly used tools are open source. It is therefore possible for attackers that have little knowledge to create and use their own tool by simply altering the available code base. These tools can then also have distinct fingerprints. Detecting these tools immediately gives extra information to defenders and helps classifying new actors who have created their own tool.

2.4.4. Distributed scanners

Scans do not have to be coming from the same IP address for all scanned hosts. An attacker could distribute the scanning process over multiple IP addresses in order to not only speed up the process, but also to avoid detection. Most intrusion detection systems aggregate data by source IP. When a lot of packets come in from an IP address in a small amount of time, the intrusion detection system will detect this and flag or block the IP address. When the process is distributed however, multiple IP addresses are sending the packets. It is then harder for the intrusion detection system to block IP addresses as none of the scanning IP addresses are classified as malicious. Because there is a large amount of data transmitted to and from an organization, it is hard to keep track of all data and match the scanning IP addresses. This process of distributed scanning can therefore avoid detection in most cases.

2.4.5. Slow scanning

Most tools that are mentioned in Section 2.4.3 are capable of scanning at very high speeds. They achieve this by injecting their scanning packets directly onto the network layer, instead of first going through all other layers. Another optimization which lets scanners scan very fast is an efficient look up algorithm to track which IP addresses have already been scanned.

When tools are going very fast however, they are easy to detect. A large stream of unsolicited packets coming in to your network in a short time will most likely be picked up by a intrusion detection system if it is in place. Therefore, some scanners opt to send packets at a low frequency. These scanners will remain under the radar more often, as intrusion detection systems will not keep a very large backlog due to space constraints. It will therefore become harder to detect these slow scanners. It is even harder if the scanner is not only scanning slowly, but also distributed. In this case, the scanner will generate very little data points, which are hard to relate to one another. Finding these scanners is the problem we aim to solve in this thesis.

3

Related work

Detection of distributed slow scanners is not a large field of research. Most research in scanners is either the creation of new scanning techniques, or the detection and analysis of very specific scenario's. Even though not entirely about the same subject, the research is still related to scanning and the effects thereof. Therefore, work about different kinds of scanners will be included in this Section. The goal of this Section is to provide an overview of the research that has previously been done and to show the gaps in the current research. One of the research gaps that was found during this thesis work was also the lack of literature about how to visualize telescope data in order to analyze the data. Therefore, this Section will not only discuss detection methods, but also visualization methods.

Bhuyan et al. [17], have provided an extensive overview of port scans and how to detect them. The authors identify two distinct port scanning scenario's, the single-source port scan and the distributed port scan. The first category might not seem relevant to this research, as this research aims to detect slow distributed scanners, but identifying these scanners helps to thin out the dataset. Therefore, single-source port scanners are also relevant to this research.

In addition to the two categories, the authors of this survey paper identify five subcategories of scanner detection. Namely Algorithmic, Threshold-based, Soft Computing based, Rule-based, and Visual. These subcategories apply to both the single-source and the distributed scanner categories, except for the rule-based category, which does not apply for distributed scanners.

3.1. Single-source scans

For single-source port scan detection, most of the proposed detection methods are algorithmic. One of the algorithmic approaches has been proposed by Robertson et al. [45]. This work uses packet header data from organizations, which contains both malicious and benign traffic. In the proposed method, the authors rely on grouping IP addresses which are close together in the same subnet. For example, 10.10.10.10 is close to 10.10.10.11, whereas 10.10.10.10 is far away from 11.10.10.10. The reason for this is that the likelihood of multiple IP addresses are used by the same organization or person is high if the IP addresses are close together. Given the total space of IP addresses, the likelihood of two separate entities probing your network with IP addresses that are really close or even adjacent is low.

In most of the proposed solutions surveyed ([17]), detecting slow scanning single port scans is infeasible due to memory constraints. The paper by Rong-sheng et al. [48] tackles this problem by using a bloomfilter to reduce the memory footprint. The drawback of using a method like this is the increase of false positives when dealing with large data. In the case of this paper this methodology works, because the proposed algorithm only detects single-source port scans. When using such a method for distributed scans, the loss of information will be significant and the false positive rate will rise.

Jung at al. [38] use sequential hypothesis testing in order to separate scanning behavior from benign traffic. For every IP address, the probability of it being benign is calculated. Every time a new packet comes in, this probability is updated. This threshold based approach has as a drawback that it does not work in real-

time. Also, this work does again not use a network telescope, but rather a production environment.

Other threshold based approaches [19, 28, 39, 46, 53] are designed to be used in real time, but all of the listed solutions do not keep a large backlog due to memory constraints. Therefore, they cannot be used to detect slow scanners as they will remain hidden under the threshold. Furthermore, a scan that is distributed enough will also remain under the radar as every single IP will remain under the threshold. While every single IP is undetected, the entire group can be over the threshold. The threshold based solutions have no means to link these distinct IP addresses together.

Proposed soft-computing approaches [26, 40, 47] all operate on packet level. The detection rate in these approaches for single-source scanners is very high. Drawbacks of these methods are that they do not work in real time and are not feasible to use in very large datasets.

It can be concluded that single-source scanner detection is mainly focused on production environments. This makes sense as it is vital for organizations to know that there is a scan going on. Nevertheless, detecting slow scanning single-source scanners can still pose a challenge due to the amount of data that can be analyzed.

3.2. Distributed scans

In order to detect distributed scans, most proposed works try to extract footprints from what the attacker is trying to obtain. One of these approaches is proposed by Gates et al. [30] and uses set cover to find relations between IP addresses. The set cover problem is a NP-complete problem [29], meaning that it takes a long time to solve when the dataset is sufficiently large. Therefore, using this algorithm is infeasible for high volumes of data. Also, the limitations of this method include that only groups that hit 95% of the telescope can be detected.

Robertson et al. [45] detect distributed scans as long as the IP addresses participating in these scans are close together. It makes the assumption that the IP addresses are closely related when they are close together. This assumption might hold in a number of cases, but it is not always valid. For example, scans originating from botnets or attackers with a larger IP space separation will remain undetected.

In the work by Yegneswaran et al. [52], the authors identify co-ordinated behavior by looking at destination ports and destination IP addresses. It is found that a large amount of scans are co-ordinated in nature, meaning that IP addresses elicit the same behavior. Only the destination ports and IP addresses are evaluated, meaning that the amount of information gathered from this analysis only indicates that there are multiple sources scanning the same ports on multiple destinations. There is some degree of temporal analysis, as the IP addresses have to be scanning in the same hour long windows. However, the paper does not identify groups or provide information about what ports certain groups are scanning for. Additionally, the limited temporal analysis and identification using only destination ports and IPs is bound to give a lot of false positives. Nevertheless, the paper shows that there are instances of co-ordinated scanning and succeeds in finding relations.

In the work of Durumeric et al. [25], the amount of scanning on the internet is analyzed, as well as what scans are most prevalent and which protocols are scanned for. It is found that horizontal scan traffic is the most prevalent form of scan traffic, with a volume of about 80%. This type of scanning can be easily distributed and according to the authors, there is a need for a way to correlate these distributed scans. The authors state: "*While we are able to estimate broad patterns in scanning behavior, we excluded scanners that operate at under 10 packets per second or targeted fewer than 100 hosts in our darknet. This likely excludes slow, massively distributed scans [25]*". This work analyzes network telescope data, but does not identify scanning groups nor does it find slow distributed scanners.

Some work actually analyzes very slow scanning groups. In the paper by Dainotti et al. [22], a very slow and distributed scan is analyzed. The scan consists of a very large amount of IP addresses that scan the entire internet. This traffic was captured in a network telescope and analyzed. Some of the IP addresses participating in the scan only sent 1 packet during the entire duration of the scan (to the network telescope). The authors of this paper did not find a way to correlate these scans however. The reason that the scan was found

was the sudden rise in scan traffic on a certain port in the entire network telescope. This lead to the discovery of this large group of co-ordinated scanners. When such a coordinated scan hits a port that has a high amount of traffic overall, these types of scans would go unnoticed even in a network telescope. Dainotti et al. [22] do however show that such co-ordinated scans do exist and provide an overview of how sophisticated these scans can be.

Distributed scanning is harder to detect than single-source scanning. Currently, there are no matching algorithms that can reliably match such scanners together. The current research focuses on the bigger picture, surveying all port scanning activity as a whole. Some research tries to detect groups in the data, but this is prone to false positives as the correlation is done by using only a very limited amount of identifiers. Research shows however that there are very sophisticated and stealthy scans going on, the problem is that there is currently no way to reliably find these groups.

3.3. Visualization

Visualizing data from telescopes in order to analyze and find scanning patterns is useful as it can help to select distinguishing identifiers. Unfortunately, there is limited research available in this area. Some visualization techniques that are available for network telescopes aim to visualize DDoS backscatter or other malicious software. According to Fachka et al. [27]: "*the visualization of darknet data is the smallest part of our darknet taxonomy*". From all visualizations of telescope data, the majority of techniques aim to visualize threats like malware and DDoS attacks. While a network telescope is arguably the best source to detect scanning traffic, the visualization techniques of scanning traffic in a telescope are lacking.

One way to visualize telescope data in order to see scanning patterns is InetVis, which was designed by van Riel et al. [50]. InetVis plots packets in a cube in order to detect patterns from source to destination IP addresses on certain ports. The representation can be seen in Figure 3.1. From this representation, an analyst can detect scans by looking for lines in the graph. Horizontal scans from one address to the network telescope can be detected using this method, as well as vertical scans and horizontal scans originating from closely related source IP addresses. It does however not show the relation between scans. A heavily scanned port has a lot of traffic, but there is no way of visualizing how related two scanners scanning the same port are. This can be contributed to a lack of identifiers, as only the source and destination IP's, as well as the destination port is used.

Fachka et al. [27] acknowledge the lack of visualization and state that there is a need for better and faster visualization methods. This can help in the analysis of telescope data by showing relations that have gone unnoticed up till today.

3.4. Research gaps

In this Section, current research has been reviewed. Current research shows a trend in which the focus lies on detecting scans rather than classifying scans. Data used to conduct this research mainly consists of live production data, which is reviewed as either packet or net flow data. Very limited research has been done in actually classifying these scanners. Therefore, the first research gap that should be noted is the limited amount of research into the classification of scanning groups.

The second gap that becomes apparent is the lack of methods to correlate distributed scanners together. The best proposed solution so far is correlating the destination ports and IP addresses in order to make a selection of IP addresses that could belong together [52]. This might work in some cases where the destination port is not commonly scanned, for example in the case of a slow distributed scan in the work of Dainotti et al. [22], but this method is not able to tell scanning groups apart if they happen to scan the same port. A better way of correlating scanners is needed in order to create a distinction between separate groups.

The final gap that we have identified in the current research are the current visualization approaches. In order to gain insight into the data and the patterns that emerge in the data, solid visualization approaches can help tremendously. Therefore, a way of visualizing the data in order to find interesting relations between scanners is helpful for this thesis work. While not the main area of interest, this is deemed as an important part to find the necessary relations. This thesis will therefore also have a small focus on visualization of telescope data.

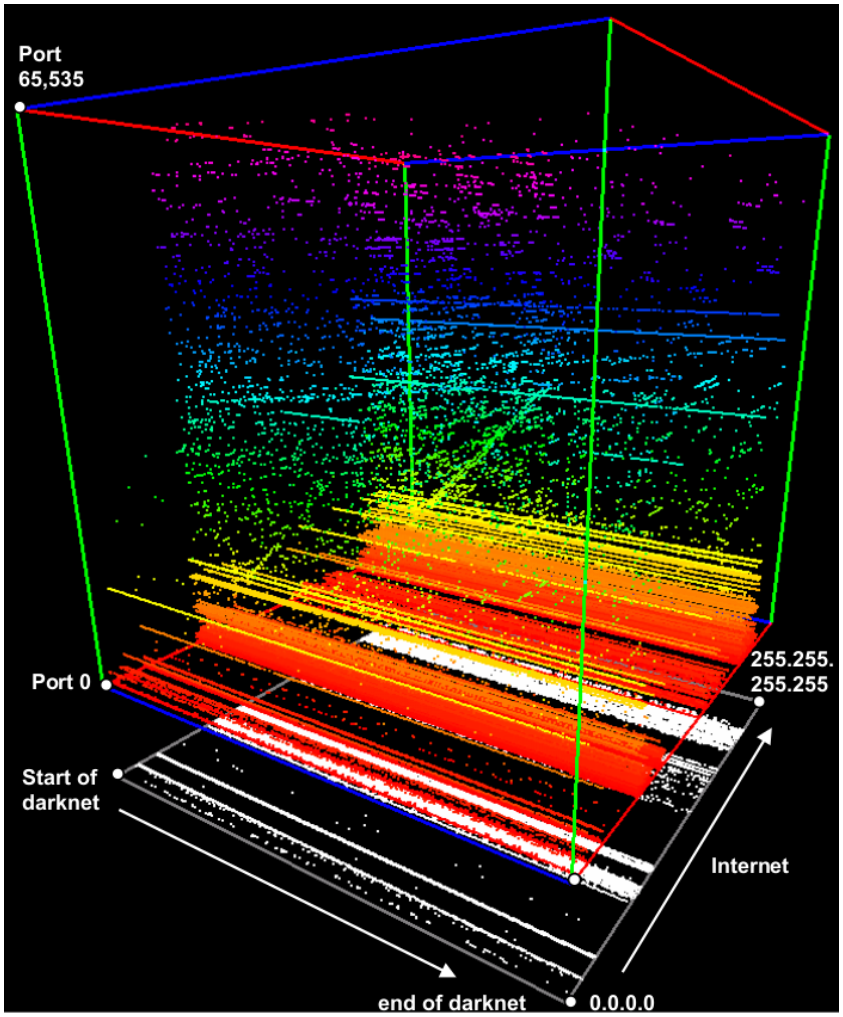


Figure 3.1: InetVis telescope visualization [50]. Lines perpendicular to the 'darknet' axis correspond to horizontal scans originating from one source IP address. From the Figure it becomes clear that there is indeed some interesting information embedded in the visualization, but the visualization is not easy to read. For single-source scanners, conclusions can be drawn, but no conclusion can be made about distributed scanners.

4

Data collection

This Chapter explains the data that has been used in this research and where the data is collected. How the data is collected will also be explained. Not all data that is available will be used in this research, this Chapter will give the reasoning behind this in Section 4.3.

4.1. Dataset

The dataset used in this thesis has been provided by the TU Delft. It is a large collection of network telescope data generated at the internal network of the TU Delft. First, the concept of telescope data will be explained. After that, what the telescope at the TU Delft looks like will be shown.

4.1.1. Telescope data

A network telescope is a collection of unused IP addresses. Because this network is unused, one would not expect data to be sent to these addresses, as no services are running and no requests are originating from these addresses. However, this is not the case. Unused IP addresses are still receiving packets, originating from DDoS backscatter for example, or from horizontal scanning campaigns. Telescopes have an interesting use when it comes to detecting horizontal scanners, but are useless in the detection of vertical scanners. Because the telescope is not connected to any service, a vertical scan targeted at the telescope does not make sense. In horizontal scanning however, people are scanning the entire IPv4 space. This means that there are scanning campaigns going on that are hitting network telescopes. Because a network telescope is a collection of unused IP addresses, there will never be any responses originating from the telescope. Therefore, it is also referred to as a Darknet.

4.1.2. TU Delft network telescope

The network telescope of the TU Delft consists of three IP ranges. Specific ranges are listed in Table 4.1. The first two ranges in the table are used by employees of the TU Delft, as well as for the network equipment like servers. The third range is the Eduroam range. This range is mainly used by students and visitors of the TU Delft. As can be seen in the table, the size of each of the ranges is /16. A /16 network has 65.534 usable addresses, which results in a total network size of 196.602 IP addresses. Not all these addresses are used in the telescope, as a large part of the addresses is in use, but it shows the size of the network that we are operating in.

The IP ranges of the TU Delft are dynamic, meaning that the usage of IP addresses changes over time. When a system logs out, its IP address goes back into the pool of unused IP addresses. The effect of this transition is that the traffic sent to the IP address becomes unroutable. This unroutable traffic is logged in the network telescope, as the IP address is unused.

A dynamic setup has upsides, but it also has downsides. An upside is that it is impossible for an adversary to map the network telescope at the TU Delft. The only way to avoid being logged in the telescope is to evade the network of the TU Delft entirely. A downside however, is that the dynamic range can introduce noise in the telescope. Take for example a host that makes a request to a server. After making the request, the host goes offline. Then, the response returns, but this is now unroutable traffic and will be logged in the

Table 4.1: IP ranges of the TU Delft

	IP Range	Subnet size
1	130.161.0.0 - 130.161.255.255	/16
2	131.180.0.0 - 131.180.255.255	/16
3	145.94.0.0 - 145.94.255.255	/16

network telescope. This results in benign traffic coming in to the network telescope. The network telescope can therefore contain noise.

The data in the network telescope is captured using Tshark [20], which is a network analyzer which is commonly used with Wireshark [21]. The raw traffic of the telescope is stored in the ".pcap" file format. This raw traffic gives an accurate representation of the data, as it is stored bit-by-bit. Therefore, it is a perfect copy of what is coming in to the telescope. The files that are being collected by the telescope are ±200 megabyte in size. One file corresponds with roughly 20 minutes of data. In one day, this means that the telescope will collect about: $200 * 3 * 24 = 14.400 = 14,4\ GB$ of data each day.

4.2. Initial analysis

The goal of this Section is to give a better view of what the data looks like and what information it contains. This analysis is also used in order to identify which parts of the data are interesting for this research (more on this in Section 4.3).

For simplicity, the data that is used in this Section are three captured telescope files. These files have a total size of 600 megabyte, and have been collected between April 1, 2018 at 17:55 - April 1, 2018 at 18:45. The files contain a total of over 6 million packets. While the data might look different during other parts of the day, this data will give a rough overview of the protocols and the intensity of scanning. The purpose of this analysis is to give a feeling for the data and to help select interesting parts. Therefore, this data will be sufficient for this initial analysis.

First, we will look at the protocols that are present in the data. The protocols that we will look at first are the protocols that are running on the link layer as defined in the reference model (Figure 2.3). The most protocols that run on the link layer are included in Table 4.2. As can be seen from the table, the most prevalent protocol by far is the Ethernet protocol. The other protocols are ARP and LLC. ARP stands for Address Resolution Protocol and is used for discovering link layer addresses. This is originating from the internal network and is probably a result of the noise that has been discussed in Section 4.1.2. The LLC protocol stands for logical link control and originates from the switches in the network.

Table 4.2: Link layer protocols

Protocol	Count	%
Ethernet	6152617	99,979
ARP	100	0,002
LLC	1170	0,019

The next layer present in the reference model (Figure 2.3), is the Internet layer. From Table 4.3, it can be seen that the most used protocol is the IPv4 protocol. Almost all traffic captured on the telescope uses this protocol. The table shows that there are IPv6 addresses present in the dataset, but the TU Delft telescope only offers IPv4 destination addresses. The IPv6 addresses that are targeting the telescope do so via a tool called Teredo [2]. Therefore, this can also be classified as IPv4 traffic, because Teredo is a tool that tunnels IPv6 over IPv4. There are also ICMP packets present, ICMP stands for Internet Control Message Protocol and is defined in RFC 792 [11]. This protocol is mostly used to send a ping to a system. If a system responds, the sender of the ping knows that the system is online and using a specific IP address.

Table 4.3: Internet layer protocols

Protocol	Count	%
ICMP	39647	0,644
IPv6	536	0,009
IPv4	6112434	99,347

To analyze the transport layer, we will contrast the use of TCP versus UDP. These are the most predominantly used protocols in this layer. The results can be seen in Table 4.4. While TCP takes is the most prevalent, UDP takes up a sizeable chunk of the data. Wireshark [21] gives a lot of extra information about the packets. For UDP, it can be seen that there are 164 different application protocols present in this portion of the dataset. The most prevalent application protocol in UDP is the eDonkey protocol, which is a peer-to-peer sharing application [15]. Memcached is also targeted a lot on UDP, Memcached can be used to conduct

DDoS-amplification attacks [31], which is probably why scanning packets to Memcached are coming in on the telescope as attackers are trying to find these services.

The last analysis to make in this Section is the analysis on the different telescope ranges. As stated in Section 4.1.2, the TU Delft telescope uses three different IP ranges, which serve different purposes. Because these ranges are different, the data collected might also be different. Adversaries might only scan part of the Internet, which leads to the adversary showing up in one range, but not in the others. The results of the analysis on different protocols is collected in Table 4.5. As can be seen from the table, the first two subnets have more or less the same distribution in terms of protocols. The Eduroam range however, which is the 145.94.0.0/16 subnet, has a different distribution of protocols. Upon closer inspection, this range receives a lot of peer-to-peer network traffic. Because this is the student range, an explanation for this might be that the students are using peer-to-peer software on the network. Once a student closes his/her laptop, the IP associated with the student will be added to the telescope. The peer-to-peer protocol will then ask if the host is still available, which will be captured in the telescope.

Table 4.4: Transport layer protocols

Protocol	Count	%
TCP	5.167.807	84,54
UDP	944.943	15,46

We can therefore conclude that the Eduroam subnet is the most unstable, which is a consequence of the dynamic allocation of the telescope. This is the network range where most users connect and disconnect, because students are rarely spending their day sitting in the same place, but are busy going to different lectures and working in between. This means that the IP address allocation in this subnet will change a lot during the day. This generates a lot of noise as discussed in Section 4.1.2.

Table 4.5: Protocols used per subnet

Protocol	Total	130.161.0.0/16	131.180.0.0/16	145.94.0.0/16
ICMP	39.647	16.417 (0,65 %)	7.931 (0,80 %)	22.655 (0,86 %)
TCP	5.167.807	2.367.513 (93,85 %)	933.109 (94,14 %)	1.867.185 (70,56 %)
UDP	94.4943	138.669 (5,50 %)	50.170 (5,06 %)	756.104 (28,57 %)
Other	1.490	28 (0,00 %)	20 (0,00 %)	202 (0,01 %)

4.3. Data selection

In Section 4.2, the different protocols that we see in the telescope have been analyzed. For the analysis of scanners in this data, we do not need all this information. This Section will explain the decisions that have been made in order to select and reduce the data.

4.3.1. Protocols

A telescope receives all kinds of data that are sent to it, spanning a large range of protocols. While most of the traffic using different protocols is sent in order to scan for a particular service, detecting scans on different protocols are separate tasks. Among all these different scans, the most prevalent scan is the TCP scan as explained in Section 2.4.2. Due to time constraints, this thesis will focus on the TCP scans, as they are the most prevalent and most services of interest use this protocol. Different protocols will not be evaluated in this thesis.

4.3.2. Flags

Section 2.4.3 has shown different types of scans. There are many forms of scanners, but in this thesis, the main focus will be on the SYN scan. The SYN scan is the most used and easiest scan, and the noise polluting the data using only the SYN flag is low. Therefore, packets that have not set the SYN flag, or that have set additional flags are discarded. The large majority of the remaining data will be scans. Selecting this part of the data will remove network artifacts such as DDoS backscatter, which is known to be present in the network telescope [18]. Selecting only the data with these flags to find scanning groups will not impact the result of the research, as long as the method that is devised will use identifiers that are agnostic of the flag.

4.3.3. IP range

As stated in Section 4.1.2, the TU Delft network telescope consists of three dynamic ranges. This dynamic use of the IP addresses can introduce noise in the telescope, which is undesired. The noise can be clearly seen in

Table 4.5, in which the first two ranges are rather close together in terms of protocol usage. The third range however, does not have the same distribution of protocols. Looking at for example the UDP usage, the table shows that the *145.94.0.0/16* range received a significant higher percentage of UDP packets than the other two ranges. This can be contributed to unfinished connections of a user when the IP address enters the telescope after being used. Because this range is prone to noise, it is excluded from the data.

4.3.4. Destination ports

In an ideal scenario, no destination ports would be filtered. Filtering on these ports will result in data loss as the scanners for that particular port can not be analyzed. Port 23 however has been filtered by the TU Delft, as it was prone to so many scans and attacks that the IT security department deemed it wise to remove all packets to port 23 on TCP. Therefore, no packets sent to this port from outside the network are present in the telescope data.

4.4. Non scanning packets after data selection

By selecting the data as stated in this Section, we assume that almost all of the data is scanning data. DDoS backscatter and similar artifacts are filtered out, as well as the noise that was present on the Eduroam network. There are however still cases in which the packet that is received does not contribute to a scan. This could be a connection request from a computer that has gone offline, as well as a machine with a faulty configuration that tries to connect to an IP address in the telescope by accident. Unfortunately, we cannot filter this data reliably. One way to filter this would be to drop all IP addresses that have sent only one or two packets in a specific time period. By doing this however, we risk dropping the very slow scanners that only send a small amount of packets. This is of course not desired as we would like to find these scanners. Therefore, the noise that remains is left in the data, which will either drop out on correlating IP addresses together, or provide a small error in the result.

5 Data analysis

In this Chapter, the selected data in Chapter 4.3 will be further analyzed, a thorough overview of the data will be given and the data will be visualized. Note that the data selection made sure that only SYN packets are present in the current data set, almost all of this data will therefore be scanning traffic.

5.1. Single fields of the packet header

In this Section, the data will be analyzed by looking at single fields present in the header of the packets. During this analysis, odd occurrences of values are found in the data. These occurrences are stated and explained and later used as part of the methodology to detect scanning groups.

5.1.1. Destination port

The destination port shows which service the packet is sent to. In Figure 5.1 can be seen that the ports in these lower ranges are receiving more traffic on average than the ports in higher ranges. This happens because the ports in higher ranges are not reserved and are not commonly used in widely used services, whereas in the lower ranges, ports are more widely used by specific services. For most attackers, the amount of devices they are able to compromise is important. Therefore, most scans will target the commonly used services.

In Table 5.1, the top five targeted ports are shown. The most targeted port in the data is port 22. This port is reserved for the SSH protocol [14]. There is high interest in this service as gaining access to either of them will very likely result in the control of the targeted system. It also runs at a high number of systems, thus has a high attack vector. This makes the port a highly interesting port to scan for when scanning the Internet.

Table 5.1: Most targeted destination ports.

Port	% Packets
22	2,99
80	1,97
2000	1,94
81	1,63
8545	1,46

The second most targeted port is port 80. This port is reserved for the HTTP protocol, which is used to run web servers. The World Wide Web uses this protocol in order to communicate between servers and clients. High demand for scans on this port can be explained by looking at the sheer size of the World Wide Web. If a web server is misconfigured or has not been updated in a while, an attacker might be able to attack the web server to gain access to the underlying infrastructure.

5.1.2. Source IP

The source IP shows the presumed sender of the packet. In case of scans, it is likely that the system using this IP is controlled by the attacker (Section 2.4.2). In Table 5.2, the top five source IP addresses with the most packets sent are shown. The amount of packets sent by these source IP addresses to the telescope in a limited time frame is high. These can be easily classified as scanners as they are probing a lot of different destination IP addresses. Not all source IP addresses are sending such amounts of packets, most IP addresses are sending a very low amount of packets. These IP addresses are performing much slower than others, which could

Table 5.2: Source IP with the most traffic.

Source IP	% Packets
5.188.87.9	4,68
185.222.211.45	3,77
181.214.87.221	3,02
5.188.11.37	2,18
113.207.80.11	1,96

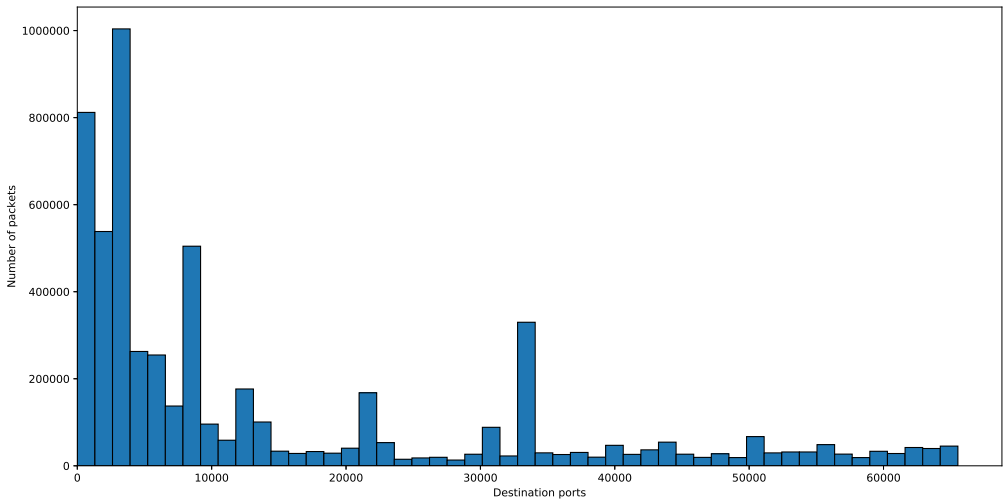


Figure 5.1: Distribution of targeted destination ports.

mean that these IP addresses are used in distributed slow scanning operations.

5.1.3. Source port

The distribution of source ports captured in the telescope is included in Figure 5.2. A noticeable split can be seen around 32.000, where the higher range sends on average a larger number of packets than the lower range. This happens because the operating system will present a port to a program when asked, the port that is presented will come from the dynamic port range. This dynamic range does not include the reserved ports for common services, but only the higher ports. So when a scanner asks the operating system for a port, it will be returned a high port instead of a low port.

Table 5.3: Most used source ports.

Port	% Packets
40659	14,02
47654	11,59
6031	7,95
44174	7,45
161	6,37

In Table 5.3, the top five used source ports are shown. Port 40659 is used over 14% of the time, which can mean that there is a specific attacker with a lot of resources scanning the Internet using this port. It could also mean that there is a tool that many people use which elicits this behavior. Searching the Internet does not reveal any information about the use of this port, which could mean that there might be a group with a lot of resources scanning using this source port. The other source ports in the table are also used quite heavily. It is clear that the source port distribution is not uniform, whereas most parts of the range should be approaching a uniform distribution according to the design.

5.1.4. IP identification number

The IP identification number is used, as stated in Section 2.2, to reassemble streams of packets. These values should be randomly chosen by the kernel. From Figure 5.3 however, it can be seen that the distribution of these IP identification numbers is not random. The figure shows a large number of IP identification numbers around 54.000 and 33.000. Table 5.4 shows this in more detail, from the table it can be seen that the most used IP identification number is 54321. This IP identification number is used by the tool ZMap and is hard coded instead of random. The scanners using this identification number are mostly ZMap based scanners [32]. As seen from the table, five identification numbers are responsible for almost 22% of the packets captured by the telescope. From the rest of the data in Figure 5.3, it can be seen that the distribution is quite constant if we do not consider the peaks. This might be

Table 5.4: Most used IP identification numbers.

IPID	% Packets
54321	8,58
33441	7,43
0	3,36
33716	2,51
256	0,40

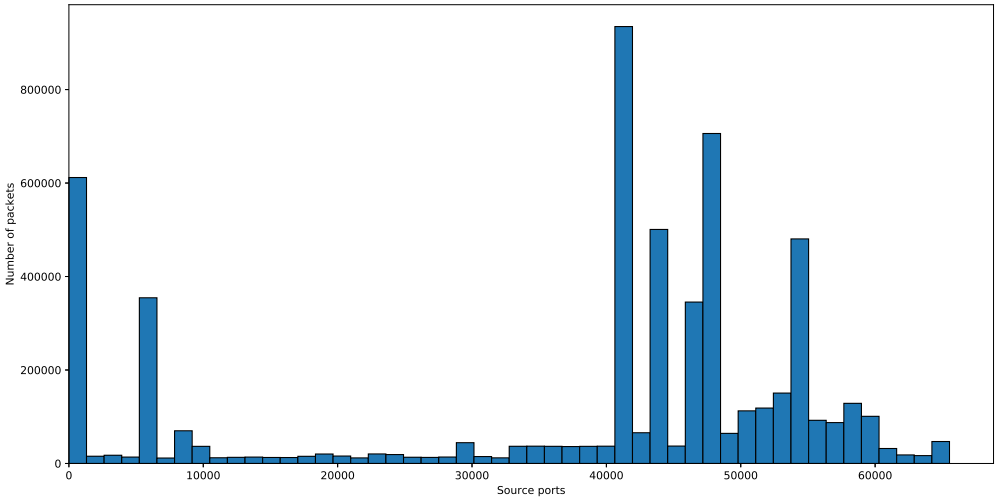


Figure 5.2: Distribution of used source ports.

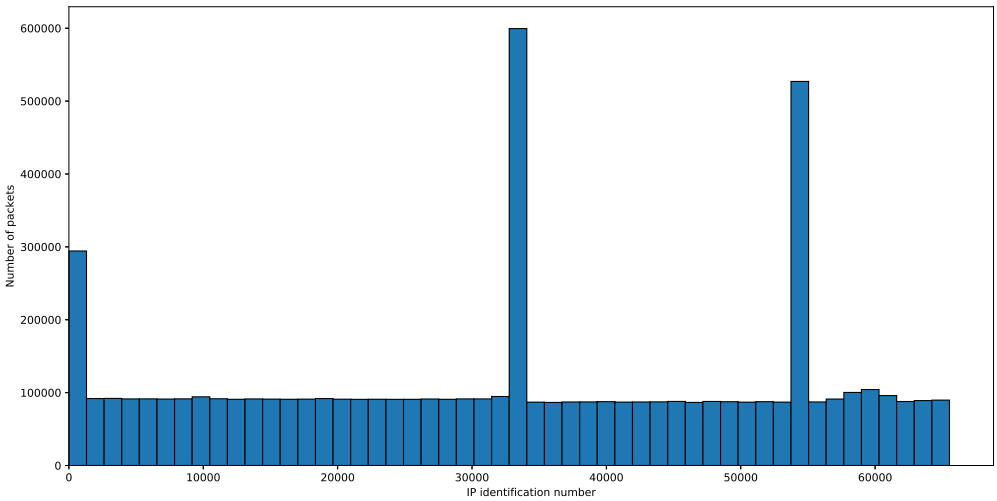


Figure 5.3: Distribution of used IP identification numbers.

due to scanners that do choose a random IP identification number for each packet that is sent. Attackers will try to stay undetected by blending in in this distribution. But when looking at the figure, we can see a very slight difference between the first half of the IP ID distribution, and the second half. Looking at the left side of the big peak in the middle, it can be noticed that it is slightly higher than the right side. This might indicate another relation.

On closer inspection of the data, it becomes clear that there are indeed more packets with an identification number that lies between 0 and 2^{15}. We know that the total space of IP identification numbers is 2^{16}, which means that there are more packets sent where the most significant bit is 0 than packets where the most significant bit is 1. The distribution of the other bits in these two sets are more or less the same. One explanation for this phenomenon could be that there is a tool or adversary that only randomizes the last 15 bits, by randomizing a positive signed integer instead of an unsigned integer. A positive signed integer would only use the 15 least significant bits. Whatever the reason for this discrepancy is, mistakes like these can provide us with indicators which can lead to the identification of different groups and allow us to link IP addresses that elicit this behavior.

5.1.5. Sequence number

The sequence number, as stated in Section 2.3, is random by default. A sequence number consists of an integer, which ranges from 0 up to $2^{32}-1$. This means that there are 2^{32} different sequence numbers possible. This Section analyzes close to 2^{25} different packets. Given the total space and the number of packets that are being analyzed, large overlaps in sequence numbers are unlikely. The probability of a packet having a specific sequence number S is: $\frac{1}{2^{32}}$. Given the birthday paradox, small overlaps can be very likely. However, large overlaps are not expected.

Table 5.5: Most used sequence numbers.

Sequence number	% Packets
0	6,83
791101	1,24
1376002373	0,86
2006491667	0,54
30000	0,43

In Table 5.5, the five most used sequence numbers are shown. From the table it can be seen that there are five sequence numbers that are used for over 9% of the packets. The expected percent of packets received per sequence number in a perfectly random distribution would be $\frac{1}{2^{32}} * 100 = 2.33 * 10^{-8}$. The values that are stated in the table show that the distribution is not random. While there might be tools that create random sequence numbers, there are also tools that do not. An example of a scanning tool that does not randomize the sequence number is Unicorn scan [44]. Using this tool, sequence numbers are generated using the following formula:

$$sequence\_number = sessionKey \oplus srcIP \oplus ((src\_port << 16) \oplus dst\_port)$$

It is clear that this number is not randomly chosen, but rather depends on a generating function [32]. Knowing this, it becomes clear why the sequence number distribution is not uniform. ZMAP on the other hand uses AES as its sequence number generating function, with the source and destination IP addresses as input. Because the output of AES is designed to be random-looking, such techniques are hard to detect.

The reason for not randomizing the sequence number is speed. By using an XOR operation, the scanner does not have to keep track of which IP addresses it has scanned in order to link packets back to the original packet that was sent. In Section 2.3.2, the TCP handshake was discussed. In the handshake, a random sequence number is sent in the first packet. For the response, the initial sequence number is incremented with 1 and sent back. Therefore, the attacker can subtract 1 from the sequence number he receives and obtain the original sequence number which he used in the first packet. By doing so, all information is encapsulated in the sequence number and can be easily extracted.

From Table 5.5 it can be noted that sequence number *0* is used large amounts of the time. This might indicate the usage of a tool that, similar to ZMap with the IP ID, uses a constant sequence number. The same can be said for the other sequence numbers in the table.

These relations in sequence numbers, be it constant, some form of XOR, truly random or even an encryption mechanism, the relations provide information about attackers and tools used. Finding relations such as the one for the Unicorn scan provided above, will give more insight in what tools are out there and provide another identifier for finding groups.

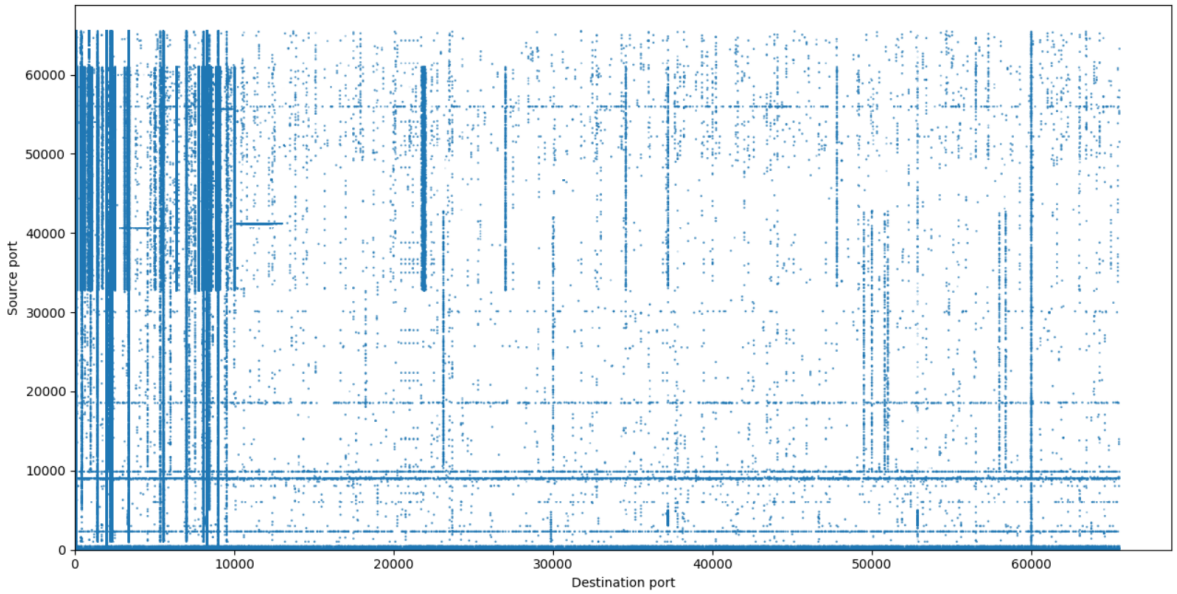


Figure 5.4: Scatter plot of source and destination ports.

5.1.6. Destination IP distribution

The destination IP addresses are addresses in one of the used ranges in the telescope. These are the IP addresses that are being scanned. As stated in Section 4.3, only two of the three ranges present in the telescope are used in this study. In these ranges, the availability of IP addresses in the telescope is dynamic. Therefore, given the IP addresses that receive packets over time, the exact telescope range can be estimated in a specific time period.

When looking at scanners, a highly coordinated scanner will most likely scan each IP address in the telescope only once. This might be used in the detection of very advanced scanners. To identify those scanners, a subset of IP addresses has to be found, which scan the total telescope range exactly once without overlaps. Using this method would however mean that the Set-cover problem has to be solved, as this is an instance of Set-cover. Unfortunately, the Set-cover problem is NP-complete [37]. With the amount of data that the telescope produces, this analysis is unfeasible to perform on the data.

5.2. Two dimensional patterns

Individual fields of data show some anomalies that are useful for data analysis and scanner detection. But only looking at individual fields gives a one dimensional view of the data. While all fields give valuable information, combining these fields together can show underlying relations that were not seen before. This Section will show how patterns emerge from graphs and what kind of patterns occur in the telescope data.

5.2.1. Source and destination ports

In Figure 5.4, used source ports have been plotted against the targeted destination port. In the figure can be seen that there are a lot of different combinations, but also patterns emerge. For example, the source ports around 35.000 up to around 60.000 are used a lot in the traffic. This is to be expected, as it is the free range where operating systems may distribute source ports from. As can be seen from the figure, these source ports are used to scan a lot of different ports.

Another interesting aspect of the figure are the vertical “lines” ranging from 0 up to 65536 (the whole source port range). This behavior is not expected as there are many reserved ports in the range, which are being used to scan single destination ports that have nothing to do with the reserved protocol of the port. For example destination port 60.000, it is clear that the entire range of source ports has been used to target this port, but the reserved ports have nothing to do with this port. A distribution like this could mean that a tool has been used that randomizes the source port with each packet sent. Another range lies between source port 10.000 and 42.000. Looking at destination port ±50.000, three “lines” can be seen where the source port

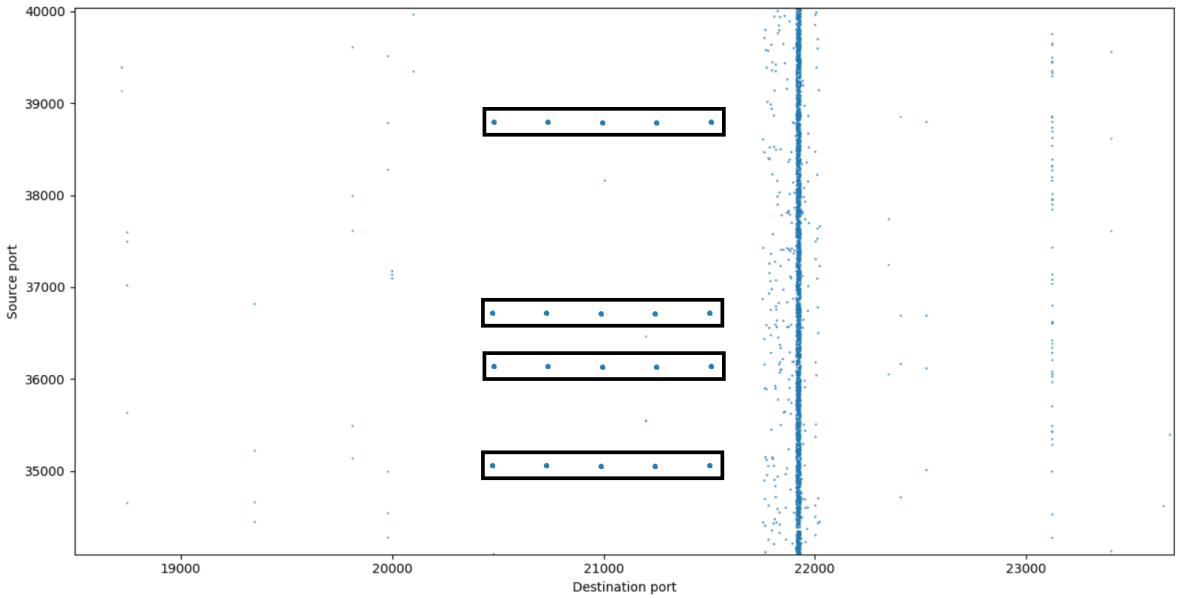


Figure 5.5: Zoomed scatter plot of source and destination ports. Black boxes show an interesting area.

is randomized over a certain range. Because this range occurs frequently, chances are that this is the behavior of yet another tool that scans the Internet.

Not only the source port distribution is interesting, also the destination port distribution shows patterns. Around source port number 10.000 in the graph, a very clear horizontal "line" can be seen. This line means that there is a certain source port that is used to scan a very large number of destination ports. This could mean that a tool is used where the source port is either hard coded or assigned at the start of the program. A scan like this, where all destination ports are scanned, we call a vertical scan 2.4.2.

More advanced scans can also be spotted in this data. One more advanced campaign is located between destination ports 20.000 and 22.000. A close-up of this data is included in Figure 5.5. Black rectangles are added to show the interesting part of the data. The figure shows groups of five destination ports that are scanned by the same source port. Every source port is used to scan all five destination ports, but after these are scanned, the source port changes. Source ports used in this behavior span the entire range, but consecutive source ports are not used. This could mean that a tool is being used in order to scan these ports, in an organized fashion. According to this behavior, this could be a block scan (Section 2.4.2).

5.2.2. IP identification number and destination port

As stated in Section 2.2, the IP identification number is designed to be random. But, as seen in Section 5.1.4, the identification numbers of packets captured by the telescope are not distributed randomly. This Section will look at the IP identification number together with the destination port to find more interesting relationships.

In order to visualize the IP identification number and destination port in one visual representation, a bubble chart is used. The result can be seen in Figure 5.6. The size of the bubble shows the amount of packets that have been received with the combination of values. This means that the packets using a combination of values that does not occur a lot are not visible in this plot, as they are too small. Not visualizing this "noise" however, gives a better view of the stronger relations in the data.

Section 5.1.4 showed that the most used IP identification number is 54321. Looking at Figure 5.6 however, the IP identification number 54321 is not actually used to scan all ports. 54321 is mainly used to scan the lower ports (below 10.000), but looking at the size of the bubble, it can be seen that a lot of packets are sent to these ports. The tool *ZMap*, which is responsible for the usage of IP identification number 54321, is thus probably mainly used for this purpose, scanning ports in the lower range.

Usage of *ZMap* is in stark contrast with the supposed tool that uses IP identification number ±33.000. From the figure it can be seen that that tool is mainly used for the ports that fall outside the first 10.000 port range.

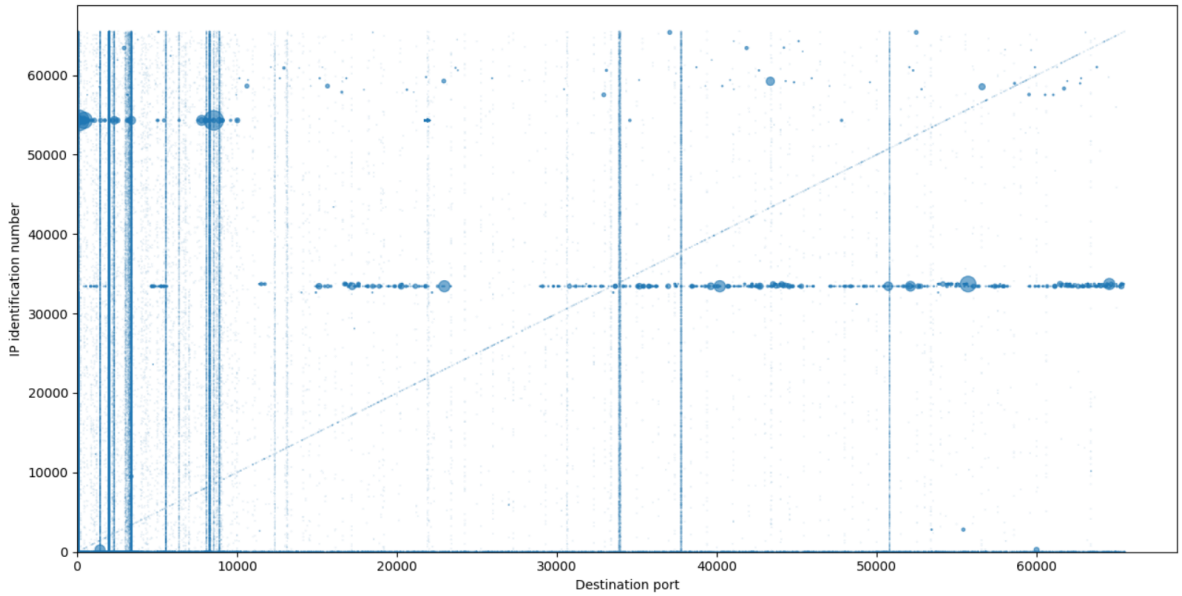


Figure 5.6: Bubble plot of IP identification number and destination ports. Size of bubbles shows the number of packets.

It can be seen from the figure that the IP identification number is also sometimes randomized. In the figure these can be seen as vertical lines. As this is intended by design, the traffic that comes into the telescope with randomized values is harder to classify, as it cannot be linked to a specific tool just by looking at the IP identification number.

The earlier relation that was found in the data analysis in Section 5.1.4 can not be seen here from the figure. The absence of this relation in the graph shows that combining different identifiers into one visualization can help to gain new information, but it can also remove visual information from the graph.

Another interesting relation that is shown by this graph is a linear relation between the destination port and IP identification number. A diagonal line can be seen in the graph. This shows that there is a tool that has quite a lot of traffic, and does not randomize the IP identification number, but chooses the destination port as the identification number. It is also a good example of why visualizing the data using multiple dimensions is useful, as this would not stand out when looking at only the destination port and IP identification number. In the one dimensional distributions, this would show as a uniform distribution, while it actually holds an interesting relation.

5.2.3. IP identification number and source port

Visualizing the source ports and IP identification numbers is done using a bubble chart. The result can be seen in Figure 5.7. The graph shows vertical lines for source ports where the entire range of IP identification numbers has been used. As can be seen from the graph, this happens quite often. More interesting are the larger bubbles that are visible in the graph. From the analysis in Section 5.1.4, it is known that IP identification numbers 33441 and 33716 are used a lot. From the figure it can be seen that there are large bubbles around the area of these numbers. These bubbles imply that there is a relation between the source port and IP identification number for these packets.

It is known that ZMAP uses IP identification number 54321. Given this information, the source port behavior of ZMAP can be explored using the figure. The only horizontal line in the figure is located at IP identification number 54321. The ports used by ZMAP range from 32.000 to 60.000, meaning that ZMAP requests a source port from the operating system. In the range where the operating system returns a source port from when requested (32.000 - 60.000), it can be seen that the distribution in the graph becomes more dense. This leads to believe that most tools request the operating system for a source port to use in their scanning.

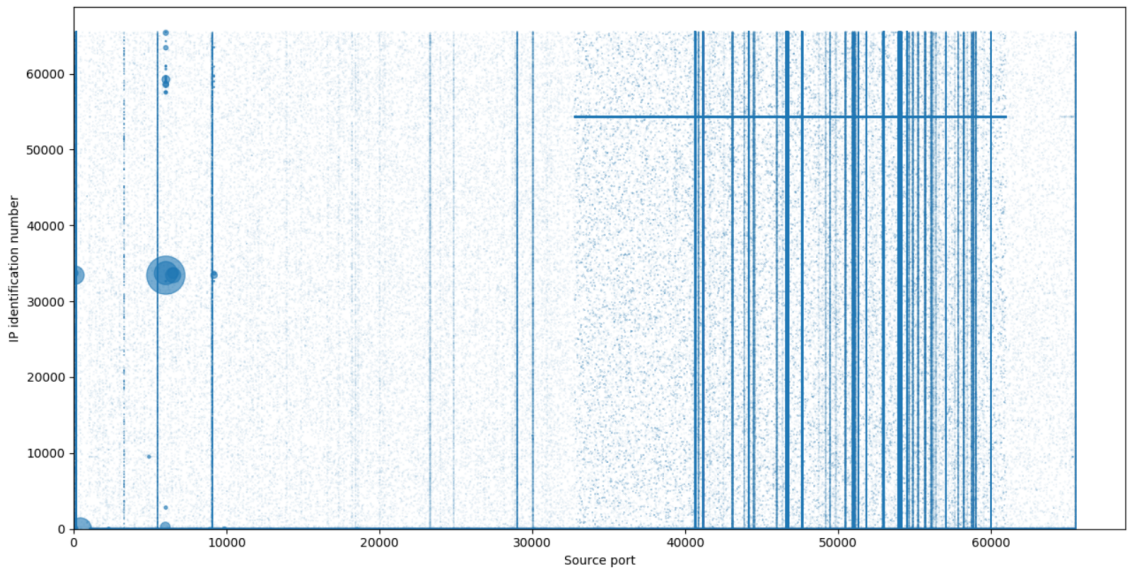


Figure 5.7: Bubble plot of IP identification number and source ports. Size of bubbles shows the number of packets.

5.3. Multi-dimensional patterns

In the previous sections, one dimensional and two dimensional patterns have been evaluated. There are interesting relations that can be found in this data, but it would be more beneficial to find patterns in even more dimensions. Finding relations between more than two fields allows for more thorough inspection of the data. Finding these relations also strengthens the hypothesis that some behavior is generated by a tool or at least by a cooperating group rather than different disjoint groups or individuals. In order to obtain a thorough understanding of the data and the patterns that emerge in the data, it is useful to devise a method that allows the analysis of the data as a whole and the analysis of data that has been identified as belonging to one specific scanning group.

Take for example IP identification numbers 33441 and 33716. From Figure 5.6 and 5.7 we can see that there is a relation between the source port and IP ID as well as between the destination port and IP ID. It would be beneficial to have a method for analyzing the relationships in multiple dimensions. Multi-dimensional visualization could show relations that consist of the three values instead of only the two tuples.

One way to visualize more dimensions in a graph is to move from two-dimensional graphs to three-dimensional graphs. This uses the same method as used in the InetVis paper [50], but with other values for the axes. This type of visual representation can be very useful for detecting clusters. However, when looking at the result in Figure 5.8, it can be seen that the visualization is unclear and unusable. It is hard to see what the specific position of a point is and whether points are changing on the X and Y axis or solely on the Z axis (or any other combination). Diagonal lines along two axes, meaning that there is a linear relation between two of the three axes, look the same in the 3d graph as a uniform distribution along one axis. For example, a linear relationship on axes X and Y looks the same as a uniform distribution on axis Z with a constant X and Y. Moving in to even more dimensions using this representation is even harder, as visualizing the fourth dimension and up is a hard problem in itself.

Marghescu et al. evaluate a large number of visualization techniques on multi dimensional data [42]. From the paper, we have chosen the Scatter Plot Matrix method in order to represent the data. The method was chosen because of the nature of the data, we know the data is strictly numerical, and we know that scatter plots already hold valuable information as seen in Section 5.2. A scatter plot matrix can be seen in Figure 5.9. Every row represents one variable, and one column also represents one variable. This enables the analysis of multiple dimensions of the data at once.

As seen in Section 5.2, the data contains interesting relations between variables. In order to enable the analysis of these relationships on more dimensions, an interactive tool has been created. The tool visualizes subsets of the data using the scatter plot matrix method. Screenshots of the tool can be seen in Figure 5.9.

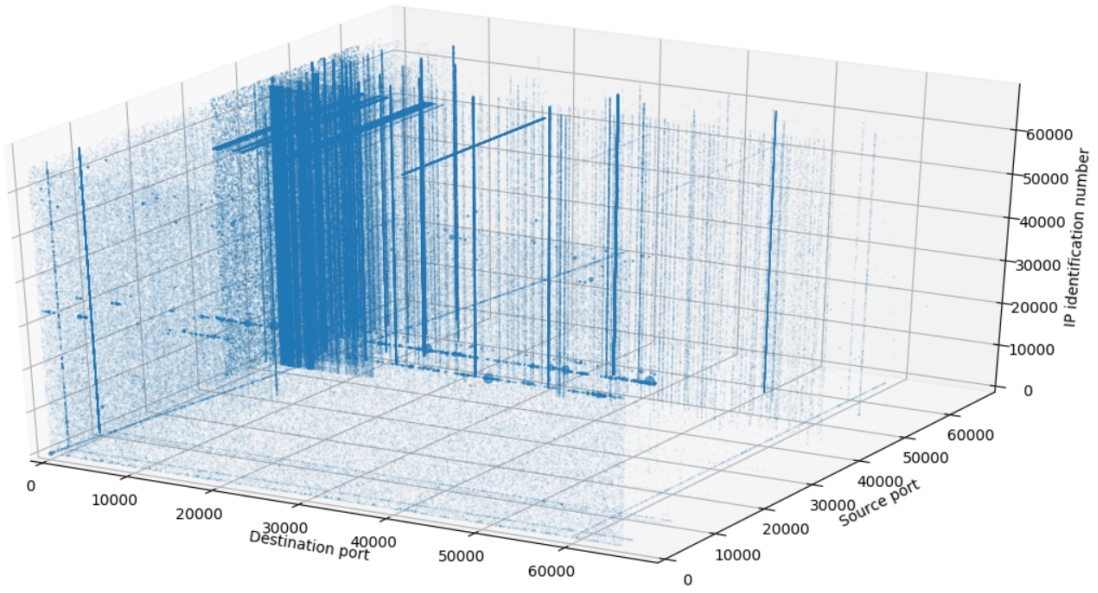
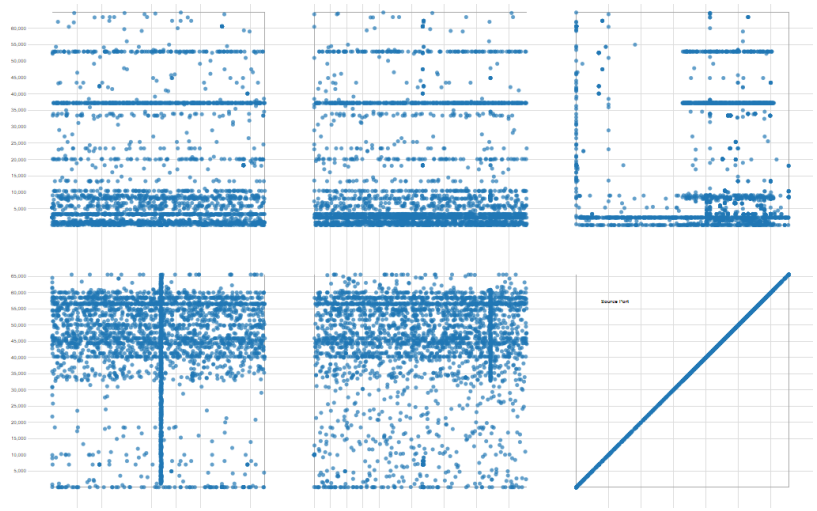


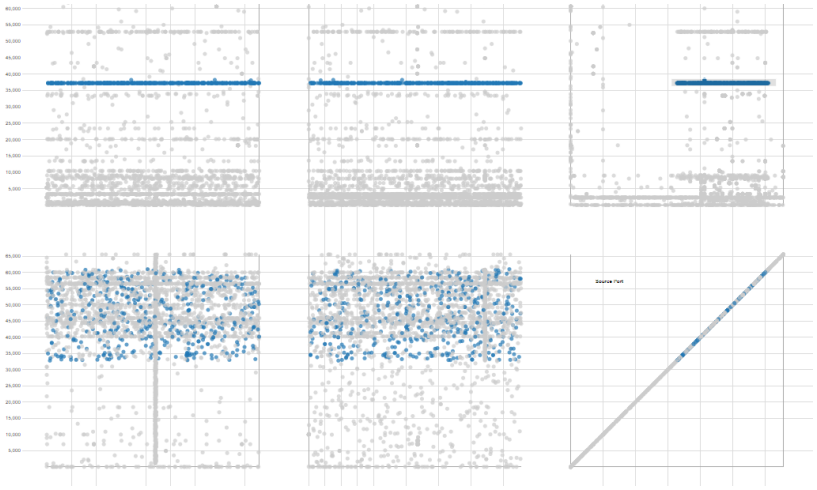
Figure 5.8: Three dimensional scatter plot of destination port, IP identification number and source port.

This tool can be extended to add more dimensions in order to obtain a thorough overview of the different relations present in the data. Using this tool, one can select an interesting relation in a graph in order to find underlying relations in the data.

As this is not the main subject of study, we will not evaluate the results that can be obtained with the visualization tool. However, we do indicate that these kinds of visualization can be very beneficial for analysts to make sense of this data.



(a)



(b)

Figure 5.9: (a) Tool without selection. (b) Tool with selection.

6

Detecting scanners

In the previous chapters, data provided by the TU Delft network telescope was analyzed. This Chapter will use the analysis made to formulate a methodology on how to detect slow distributed network scanners. After explaining the used methodology, a detection method has been devised. To counter the shortcomings of the proposed detection method, a method is created to split the data to processable chunks without losing important connections within the data. Finally, the detection method is evaluated.

6.1. Evading detection

To reliably find scanners and evaluate the results of this findings, we need an ontology of how scanners would try to evade detection. In this section, we will create a model in which we capture how advanced a scanner is given the evasion tactics that are used.

Given the functioning of IDS systems, which trigger alerts on scanners based on the frequency of sending packets, the most obvious way of avoiding detection is to send such an amount of packets that the threshold is not reached. However, by doing do, the scanner will become unsuitable for Internet wide scanning, as it will traverse the Internet very slowly. Therefore, an adversary will spread this scanning over a number of IP addresses. These IP addresses can either be very distributed, or located in the same subnet. Figure 6.1 shows these three dimensions in a graph, in which the point at $(0,0,0)$ shows very unsophisticated scanners. When going up in the chart to point $(1,1,1)$, the scanners become increasingly more sophisticated.

While Figure 6.1 only shows three dimensions in which we can measure sophistication to explain our method, there are many more. Ways for scanners to 'hide' are listed below.

1. **Source IP address:** Scanning from only a single source is easily detected, as the amount of probes to a network will be very high when scanning at a reasonable speed. Distributing the scanners over a large amount of IP addresses will evade detection, as long as the scanners are not in the same subnet, otherwise they will be detected by [45].

2. **Sending low amounts of packets:** IDS systems will detect a high number of probes coming in at the network, so in order to hide, each IP address would send a low amount of data.

3. **Destination IP addresses:** A scanner can skip part of the destination IP space in order to not be detected by methods proposed in [30]. This means that the scanner loses some information, but to tackle this, a scanner could also create a larger overlap in scans in order to increase the chance that the detection method finds false positives.

4. **Randomizing packet fields:** From Chapter 5, we have found that a some scanners do not randomize their packet fields. A good method to 'hide' in the data is to randomize the fields present in the packet. This way, the scanning traffic will ultimately be indistinguishable from normal network traffic.

   Randomizing each field would mean that the scanner has to keep track of every packet it has sent out, in order to match them back to the scan. For Internet wide scanning, this is undesired, as there are a lot of packets to track. Therefore, some scanners embed data in their packet so they can match it back to the scanning campaign (Section 5.1.5). As scanners become more advanced, this relation will be harder, if not impossible to detect.

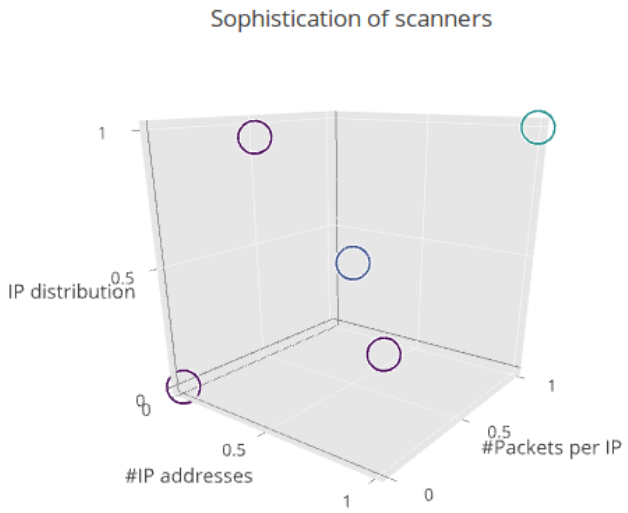


Figure 6.1: Graphical representation of sophistication of scanners. This image has three dimensions, but the problem has many more. Circles show locations of different sophistication, bottom left is unsophisticated, top left is very sophisticated. A scanner can get more sophisticated in each of these axes.

5. **Not creating time-based patterns:** When a scanner group scans a specific target port at a certain time, after which it completely disappears, the IP addresses in this group can be matched. Therefore, in order to scan in a very hidden way, a scanner could scan in a random looking time pattern, where it gradually switches on or off.

Given these categories, we can use our multi-dimensional visualization tool that was proposed in Section 5.3 (Figure 5.9). From this visualization, we obtain a way of analyzing how different hiding techniques are used by different groups.

6.2. Methodology

Distributed slow scanners do not generate a lot of data points. Finding these data points is one challenge, linking the data points together is the second. Problem with the telescope data is that all data looks the same. When looking at only a few data points, they appear to be random in the distribution. The data points that are generated by slow distributed scanners are not outliers either, the slow distributed scanners are hidden in the larger picture. This makes the 'signal' that the scanners leave in the data very weak. The goal of this Section is to propose a method in which the weak signals of slow distributed scanners can be detected. To achieve this, a systematic way is shown that is used to find these weak signals in the large pool of data that the telescope provides.

The method that we propose is a method that has been popularized by Vern Paxson, in the work he calls "*Finding Very Damaging Needles in Very Large Haystacks*" [8]. This work aims to detect stealthy intruders in a system, which suffers from the same challenges as this work, namely the overabundance of data and the low frequency of the events of interest.

Processing the data to uncover hidden scanners will be done using an iterative approach, where the dataset will be reduced in size after every iteration. The reason behind this is that once we know what a scanning group is doing and can fingerprint its behavior, there is no need to keep analyzing the same group. From the fingerprint, the data points can be removed to show underlying relationships that were previously obscured due to other data. This process is visualized in Figure 6.2, in which step 2, 3 and 4 create a loop that will execute until no new scanning groups can be found in the data.

Extraction of fingerprints is explained in Section 6.3. Every time a fingerprint is found, the fingerprint is checked and then all datapoints that are coherent to that particular fingerprint are removed from the data. After removing all values with the fingerprint, the algorithm is executed again to find another fingerprint.

Using this particular method will enable us to shrink the dataset, which is useful for both a human analyst, as well as for automatic analysis. Given the sheer size of the data, analysis is very time consuming. Pruning the data in each step will result in smaller datasets, but the initial steps will still be time consuming. To counter

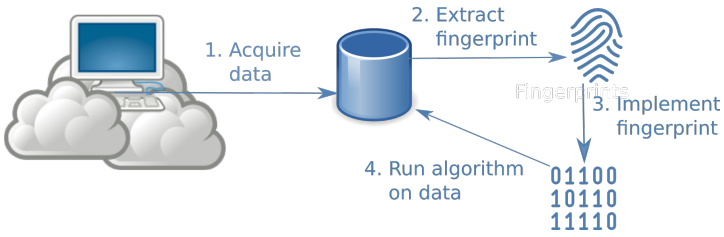


Figure 6.2: Iterative processing method of the data. After each loop, the dataset shrinks to reveal other relations.

Header	Packet field	Returning packet field
IPv4	Destination IP	Source IP
IPv4	Source IP	Destination IP
TCP	Destination port	Source port
TCP	Source port	Destination port
TCP	SYN flag	SYN & ACK flag
TCP	Sequence number	Acknowledgement number (+1)

Table 6.1: Packet fields that are present in the returned packet

this, the data can be pre-processed with the insights gained from Section 5, in which different strange patterns in the data were found. Source IP addresses that elicit very different behavior might be eliminated as being part of the same group, if there are different combinations of these IP addresses that show the same behavior. Therefore, in a pre-processing step, the data can be split in different disjoint sets. After the pre-processing step, only IP addresses in the same set have to be compared to each other. This will speed up the process as there are less comparisons to be made. The method used for splitting the data is explained in Section 6.4.

6.3. Scanner detection

In this Section we will look at different patterns that can emerge within scanning groups. Also, the extraction of fingerprints is explained, which has the form of an equation that is used to create sequence numbers for packets.

6.3.1. Xor analysis

Various tools use Exclusive OR (XOR) operations in order to generate a random looking sequence number to use in a packet. This has been highlighted before in Section 5.1.5. Because tools are using specific methods to generate sequence numbers, these methods can also be used as a fingerprint. When for a pair IP addresses the sequence number is generated in the same way every packet, they are most likely part of the same scanning group with its own custom tool. Finding these XOR relations will not only show which IP addresses belong together, it will also give a way to fingerprint the group. In this Section, the XOR analysis will be explained. These XOR relations will be used as fingerprints for different groups.

Fields used for XOR relations

Using a specific relation to compute the sequence number is done to link the packet back to the scan, without having to save a table of sent packets. Therefore, to check if the packet originated from the scan and is not just noise, the sequence number is compared against the fields present in the packet. This means that the fields that are part of the relation of the sequence number should be present in the response to the scanning packet in the handshake (Section 2.3.2).

Several fields of the packet header are encoded in the response to the scanning packet. We will look at two headers present in a TCP packet, the IPv4 header and TCP header (discussed in Chapter 2). Looking at the protocol, there is only a handful fields that are preserved in the returning packet. Table 6.1 lists the fields that are returned in the response on a SYN packet and which fields of the response hold this data. Other fields are not preserved in the response and therefore should not constitute to the equation.

Given the fields that are preserved in the response, only the SYN flag does not make sense to use in the equation, as it is a flag, which is only a single bit. This bit could also be flipped by default in the equation and this will be detected by the method described in the next Section. Other fields listed in the table could be part

of the equation, allowing the attacker to check if the packet was part of the scanning campaign.

An interesting field that is not present in the table, is the IP identification number. While not preserved round trip, it was found to be part of the equation for certain groups. This does not make sense from an operational perspective, as then there would be no way to connect the packet back to the scanner without creating a lookup table. Therefore it might be that the IP identification number is generated by using the other fields in a similar fashion as the sequence number, the equation for this could not be found. The IP identification number is also evaluated in the detection step, as it might lead to the discovery of more groups.

Types of XOR patterns

XOR relations can consist of various different relations. Which means that there are a large number of possible XOR patterns that adversaries can use. This Section will review possible XOR patterns in order to show the type of relation that we are looking for.

The part of the packet where it makes sense to use an XOR is the sequence number. As we have seen from Figure 2.6, the sequence number is a 32-bit unsigned integer. The output of the XOR operation will therefore most likely be a 32-bit integer. There are countless ways in which this number can be created. A simple relation could look as follows:

$$sequence\ number = destination\ IP \oplus destination\ port \oplus (source\ port << 16)$$

In this case, the destination IP address is a 32 bit integer, whereas the destination and source ports are both 16 bit integers. With the source port shifted 16 bits, the source and destination port combined will result in a 32 bit integer. XOR-ing this with the destination IP address will then result in a seemingly random number, while there is an underlying relation.

The previous example is really simple, but it gives an idea of what can happen. In the example, one of the 16 bit ports was shifted 16 bits, but this does not have to be the case. To create a new integer, any number of shifts will do. Therefore, the equation can also look like this:

$$sequence\ number = destination\ IP \oplus (destination\ port << k_d) \oplus (source\ port << k_s)$$

In the equation, k_d and k_s are arbitrary numbers in the range $[0,31]$. The destination IP can also be substituted with the source IP, or both can be used in the XOR operation. IP address fields can also be shifted, just like the ports. This shows that there is a very large space in which we can try to find relations. When session keys are added, the equation becomes obfuscated even more. Also, there are more values that might be used in the XOR relation. Most of the other values that can be used do not make much sense to use, as most of these values are not preserved in the response of the TCP handshake, which was explained in the previous Section.

Finding XOR relations

In the previous Section, we have found different fields that can be part of the XOR equations used by different tools. This Section will explain how the different fields have been used in order to find XOR relations. Adding all fields together, we obtain the following equation:

$$\begin{aligned} sequence\ number = destination\ IP\ \oplus\ source\ IP\ \oplus \\ (destination\ port << k_d)\ \oplus\ (source\ port\ << k_s)\ \oplus\ (IP\ ID << k_i) \end{aligned}$$

The goal is to establish the equation, as well as the fields that make up the equation. But simply taking the XOR from only the $(destination\ port << k_d)$ with the sequence number for example does not work, because the bits that we are XOR-ing with at the moment might be dependent on multiple fields. Table 6.2 shows this in more detail. In order to detect XOR patterns, we have to find the relation that creates the sequence number, or part of the sequence number. Part of the sequence number means that given a few values, a part of the sequence number can be explained. This is also the case in Table 6.2.

Given the equation above, some fields that make up the relation could be 'missing'. A group could for example not use the IP identification number in the equation. Therefore we have to check 32 combinations per value (31 unique shifts and 1 check for the absence of the value). This would mean that for the source port, destination port and IP identification number, all combinations have to be checked. For packet, this would mean that there are $32 * 32 * 32 = 32.768$ comparisons needed. When adding source and destination IP addresses to the equation, the number of comparisons grows with a factor 4 to 131.072 comparisons per packet. Doing this for every packet is unfeasible.

Sequence number in bits	01011001 11001110 01110010 00011011
Destination port « 16	00101101 10101010 00000000 00000000
Source port « 16	01110100 01100100 00000000 00000000
XOR source and destination port	01011001 11001110 00000000 00000000

Table 6.2: XOR relation between source and destination port. Both fields are part of the equation for the first 16 bits in the sequence number, but on their own they do not show any relations.

To reduce the amount of comparisons that have to be done, we look at pairs of IP addresses. When two identical values are XOR-ed with each other, the result is 0. This property can be used by comparing two packets and XOR-ing their sequence number. What this will do is cancel out any common values that the two IP addresses from the equation. Take for example two packets that use the following equation to generate their sequence number:

$$sequence\ number = destination\ IP \oplus destination\ port \oplus (source\ port\ << 16)$$

The fields of the two packets are shown in Table 6.3. As seen from the table, both packets have the same IP ID and destination port. This means that we do not have to account for these values in the equation anymore. Take the destination port for example, given the equation of the sequence numbers, XOR-ing for the two packets would mean the following:

$$sequence\ number_1 \oplus sequence\ number_2 =$$

$$(destination\ IP_1 \oplus destination\ port_1 \oplus (source\ port_1\ << 16)) \oplus$$
$$(destination\ IP_2 \oplus destination\ port_2 \oplus (source\ port_2\ << 16)) =$$

$$((destination\ IP_1 \oplus destination\ IP_2) \oplus (destination\ port_1 \oplus destination\ port_2) \oplus$$
$$((source\ port_1 \oplus source\ port_2) << 16)$$

Given this equation, we can see that when one of the fields is the same, it will zero out. So in this case, the equation becomes:

$$((destination\ IP_1 \oplus destination\ IP_2) \oplus ((source\ port_1 \oplus source\ port_2) << 16)$$

While we do not know the equation, we can disregard the IP identification number and the destination port in our analysis, as we know these values will fall out of the equation. Therefore, when checking for the relationship, there are less comparisons that have to be made. In order to find the relations, we have to take the XOR of the fields that could be unknown parts of the equation. In this case, we would take the XOR of both source and destination IP addresses, and the XOR of the source port from both packets. We have to do this because the resulting value from XOR-ing the sequence number now contains the XOR-ed fields that are part of the equation. In the case from Table 6.3 and the equation above, this method reduces the number of comparisons to 132 instead of 143.748.

In order to go even further, one of the fields that is likely shifted can be eliminated. By considering only one shifted field, we can establish whether the number that we are comparing to is a shifted version of this number. This can be done by using a logarithm operation. The reason for this logarithm to work is that when bit shifting, a number will always grow with a power of 2. By dividing the sequence number with the original number, we get the factor with which the number has grown. Taking the Log in base 2 for this number will give the amount of bits that the number has been shifted. This leads to the following equation:

$$log_2(\frac{seqnumXOR}{value\ to\ check})$$

When a whole integer k is returned from this function, the number was shifted with k places. If the algorithm returns a fraction, the resulting sequence number was not a shift of the value that was checked.

Using this method, the comparisons decrease even further. When there are two common fields in the packets as in Table 6.3, the number of comparisons goes down to 8. When there is only one common field in the source port, destination port or IP ID, the total number of comparisons that have to be made is 264.

Xor relations with session key

A session key means that the XOR value is further obfuscated. Finding these session keys is hard as they are generally randomly generated. This means that between different source IP addresses, these session keys are different. But because session keys are usually generated on start-up of a scanner, the session keys are constant per source IP address. We know from Section 6.3.1, that constant values can be filtered out of the equation by XOR-ing the sequence numbers together. This means that the session key can be filtered by XOR-ing two packets originating from the same IP address. When doing so, the source IP address is also out of the equation. Using this method, we can find groups of IP addresses if the equations are the same between two IP addresses.

There is a drawback for this method however, as there is no way to check if the source IP address is part of the equation. When a session key is used, it will cancel out, but so will the source IP. When the rest of the equation is known, we can trace the XOR back to the session key, or the $session\ key \oplus source\ IP$. There is no way of telling whether it is one or the other, as XOR-ing the result with the source IP will give a value either way. Therefore, we cannot know if the source IP address is part of the equation. If the IP address always scans the same port, the destination port also becomes unknown.

Minimal amount of packets for XOR analysis

Given the total space of possible sequence numbers, it is very unlikely that these relations happen by chance. Sequence numbers are supposed to be randomly drawn from a range of $[0, 2^{32} - 1]$. This means that the chance to randomly stumble upon a specific sequence number is $\frac{1}{2^{32}}$. A match from a sequence number to a specific XOR operation is therefore very unlikely, but there are multiple ways of XOR-ing the data present in the packets. This makes the chance to randomly stumble upon any XOR operation higher. In order to verify that an IP address is indeed using a XOR operation to generate the sequence number, at least two packets are used in the analysis. Even when the first packet randomly matches, the second packet will have to match the specific XOR sequence which it will with a probability of $\frac{1}{2^{32}}$. After collecting a fingerprint however, checking the fingerprint for IP addresses that have only sent one packet to the telescope becomes feasible. Because we have a rule which the sequence number should match entirely, the sequence number matches the rule with a probability of $\frac{1}{2^{32}}$. This chance is deemed a low enough to be certain enough that an IP address is part of a specific group.

Finding XOR patterns in the data is not trivial. Even given a set of input variables, the resulting XOR can be made difficult to find. Inputs can be bitwise shifted in an unknown way, and not every field might be part of the equation. Also, as seen in Section 5.1.5, a session key might be used. This session key is a random number generated at the start of a program. Because the number is random, the outcome after the XOR operation is also seemingly random. This operation is comparable to the operation used in the one-time pad cipher [16]. In the one-time pad cipher, the operation is impossible to break if the random string is not known. This also holds for the XOR patterns that use a session key in our data. Luckily, session keys are reused over multiple packets, which invalidates the rules of the one-time pad. This strengthens the need to analyze multiple packets per source IP address, as this relation can not be broken given only one packet.

Runtime of the algorithm

For every packet in the dataset, there needs to be a comparison with every other packet. The run time of this algorithm is therefore:

$$n * (n-1) \rightarrow \mathcal{O}(n^2). \text{ Where } n \text{ is the number of packets.}$$

This means that for a time period of 1 hour, ± 3 million packets have to be compared against each other. Such an amount of packets leads up to $\pm 9 * 10^{12}$ comparisons that have to be done every hour of data. To put

Field	Packet 1	Packet 2
IP ID	1000	1000
D-port	433	433
S-port	20000	18999
Destination IP	1.2.3.4	4.3.2.1
Source IP	2.2.2.2	3.3.3.3

Table 6.3: Fields of two packets in the dataset

this into perspective, when doing 1 million comparisons per second, this would take ± 104 days to process. So even while the algorithm has a polynomial run time, it is unfeasible to run on the entire dataset.

To decrease the number of comparisons that we have to make, we use clusters that are created using the algorithm described in Section 6.4. The assumption here is that the groups that belong together are put in the same cluster. Normally, all packets would have to be compared against each other, while now only the packets inside a cluster have to be compared. Comparing only within clusters of the data means that the total number of comparisons that has to be made goes down. This drastically reduces run time and makes the proposed algorithm feasible.

6.4. Splitting the data

To reduce the number of computations that have to be done on the data, we can split the data into disjoint sets of IP addresses. An example of these disjoint sets is the following: An IP address that is solely scanning port *50.000* is most likely not related to an IP address that is only scanning port *22*. Therefore, we do not have to compare these IP addresses to each other as it is clear that they do not belong together.

From these splitted data sets, further analysis only has to be done within each cluster. All IP addresses with the same behavior will be in the same 'cluster' after the split, so therefore there is no need to compare IP addresses in different clusters as they are within disjoint sets. This Section explains how these clusters are made and what the thoughts behind the proposed clustering method are.

6.4.1. Evaluation of clustering methods

Clustering can be done in various ways. This Section lists a few methods that are considered and the reason why they are selected for the final system or not. This list is by no means exhaustive, and there are many clustering approaches that are not listed in this Section. While they are not listed, a lot of clustering methods have been evaluated. This Section aims to give an overview of the evaluated algorithms and methods, but will not list all available algorithms for this problem. Only the most important algorithms are shown.

Many modern clustering approaches make use of machine learning based methods. We have chosen not to use these machine learning approaches because they take away the control we have over the data. Given that there is no data with ground truth, and currently no way to create this data ourselves, means that supervised learning is out of option. Therefore, when tackling this problem with machine learning, unsupervised learning has to be used. Examples of such clustering techniques are: k-means clustering, affinity propagation or hierarchical clustering. While these methods can be used for clustering and classifying data, it also comes with a lack of control over the clustering. Because of the nature of the data, in which all data looks alike so much, control over what is being clustered is vital. If the algorithm wrongly identifies some part of a slow scanner as being part of a group, whereas another part of the slow scanner is not, a lot of data is being lost. This result is hard to evaluate as there is, as stated, no ground truth. Also, because of the sparsity of data points provided by slow scanners and the current inability to classify them, it is infeasible to evaluate if they are wrongly added to a cluster.

When looking for a clustering method, we spoke with researchers from the University of Hamburg. In their paper [34], they use the Clique Percolation Method (CPM) in order to detect multi-step attacks in network data. The method showed to work really well for the clustering of this type of data, which is very similar to the data that is captured by the telescope.

Clique Percolation

CPM works by finding k-cliques in a graph, each of these cliques is a cluster. Because there can be multiple cliques in one portion of the data, overlap can occur. This overlap of cliques means that some points in the graph are clustered in several clusters. Having this overlap in the data might be beneficial for scanner detection, as it prevents the misclustering of data. We want to retain the integrity of the data in order to have enough datapoints to do post-analysis, the noise that is generated by the overlap in clusters is less important as it will be filtered out in the post-processing steps. Therefore, this overlap is considered a very beneficial feature.

A drawback of clique percolation is that it clusters the data based on *k-cliques*. In order to do so, the algorithm has to find these cliques, which is a NP-Complete problem and therefore takes a long time to complete with large amounts of data. Because our dataset is large, this method is not feasible for us. Additionally, the

result of this method is highly dependent on the size of the cliques k. As slow scanners generate only little datapoints, the size of the cliques would have to be very small, which also creates large overlaps and false positives in the data.

However, considering that this method has been proven to work very well for network data, we have limited our scope to algorithms that cluster highly connected parts of the graph. Additionally, we do not know the number of clusters, so the algorithm of choice should not make any prior assumptions on the amount of clusters.

Higly Connected Subgraphs

Highly Connected Subgraphs (HCS) [36] is a clustering method that finds portions of the graph that are highly connected. HCS relies on the minimum cut of a graph to identify how similar elements are to each other. A cluster is defined as a subgraph for which holds that the minimum cut consists of more than half of the vertices in the subgraph. The algorithm runs in polynomial time and does not require prior knowledge of the amount of clusters, but does not support overlaps in clusters. Because the overlap in clusters would be beneficial for this analysis, HCS is not used as part of the proposed clustering method.

Speaker-Listener Label Propagation

Speaker-Listener Label Propagation (SLPA) [51] is an algorithm that behaves similar to clique percolation. The algorithm is based on the Listener Propagation algorithm, but in contrast to the Listener Propagation algorithm, this algorithm supports overlapping communities. This overlap is created by adding multiple labels to vertices, whereas there was only one label per vertex in the Listener Propagation algorithm. The run-time of this algorithm is polynomial, it does not require prior knowledge about the number of clusters and it supports overlap. There are alternatives to this algorithm, such as CONGA, Copra and EAGLE [51] , but literature points out that these algorithms do not perform as well as SLPA.

We have chosen SLPA as the clustering method for this thesis. SLPA satisfies all the constraints that we have set beforehand, in that it clusters highly connected parts of the graph, is agnostic of the amount of clusters, supports overlap of clusters and is fast enough to process our dataset.

6.4.2. Identifiers

Data clustering depends on the identifiers given to the algorithm. From literature study, the current best clustering approach identifies coordinated behavior by using only the destination IP and port [52]. This neglects the other possible identifiers present in the data. From chapters 5 and 5.2, we can find that there are a lot more data fields that can help as features for the clustering. This will provide us with a broader set of values with which we can identify scanning behavior.

Source IP

The source IP address is used as a special identifier. Given the time frame in which we process data (Section 6.4.4), we assume that the owner of the source IP is constant. This means that the behavior of the source IP reflects the behavior of the scanner itself. Therefore, clustering will be done on a per source IP basis, where the input of the algorithm is a set of source IP addresses, which all have lists of identifiers. Consequently, all the correlations of identifiers will be done on lists instead of on single values.

Destination port

This field is used as the main identifier. When two source IP addresses scan different destination ports, we assume that there is no correlation between the two source IP addresses. The reason behind this is that attackers that are scanning a system will target the ports that they have a working exploit for. Therefore, similar scanners will most likely scan the same ports. It is still the case that the same group might target two different ports with two disjunct sets of IP addresses. When this happens, the two sets of IP addresses are classified as two different scanners because they are not scanning the same port(s). While this result might seem counter intuitive as the scans originate from the same group, it are in fact two different scanners. Therefore, this result is deemed correct.

Note that an IP address that scans port 80 and 8000 differs from an IP address that is only scanning port 80. For the same reason as for the case described above, this is deemed to be a correct result.

Further identifiers

Further identifiers are not as strict as the destination port. In these identifiers, partial overlap in values is allowed. Because some of these identifiers are random by design, matches on these identifiers might not even occur for similar scanners. When two IP addresses that belong to the same scanning group randomize their sequence number, the chances of having an exact match between these numbers is very low. Therefore, other identifiers cannot be as strict as the destination port.

- **Source port**
  Figure 5.2 shows that an unexpected distribution of source ports is captured by the telescope. The distribution shows that certain source ports are used far more often than others. If scanners are using the same source IP address over all their IP addresses, we can use this information to cluster them. Therefore, the source port field will be used as identifier, as it has discriminating features.

- **Sequence number**
  The sequence number field is arguably the most valuable field in the data, as it can contain a lot of information. These relations were shown in Section 5.1.5 and consist of, for example, an XOR from other fields. Apart from these XOR relations, the tool described in Section 5.3 showed interesting relations between the sequence number and other identifiers that can be used as discriminating factors.

- **IP identification number**
  Given that known scanners (ZMap) are using static IP identification numbers, other scanners might do the same. In our analysis in Section 5.1.4, we have seen that some IP identification numbers are used so often in the data that they cannot be contributed to chance. Therefore, this field is included as an identifier.

- **{#Sequence number, #IP identification number, #Source port} / #packet**
  While performing the clustering, it became apparent that some IP addresses elicited common behaviors when it came to selecting new sequence numbers, IP identification numbers and source ports. For example, a set of IP addresses always sent three packets with a constant source port, and then switched to using a different port. This kind of behavior were so common in the data that it was added as a discriminating identifier. Reason being that such similar behavior will probably result from the same tools.

Destination IP

The destination IP address will not be used in this stage of the clustering, as no feasible correlation can be made at this point. Given a set of packets and source IP addresses, we would have to find a set that covers most of the destination IP set. This is however infeasible for the amount of data that we are processing. Therefore, this will not be included in the initial clustering step, but can be done in the post-processing steps.

Other identifiers

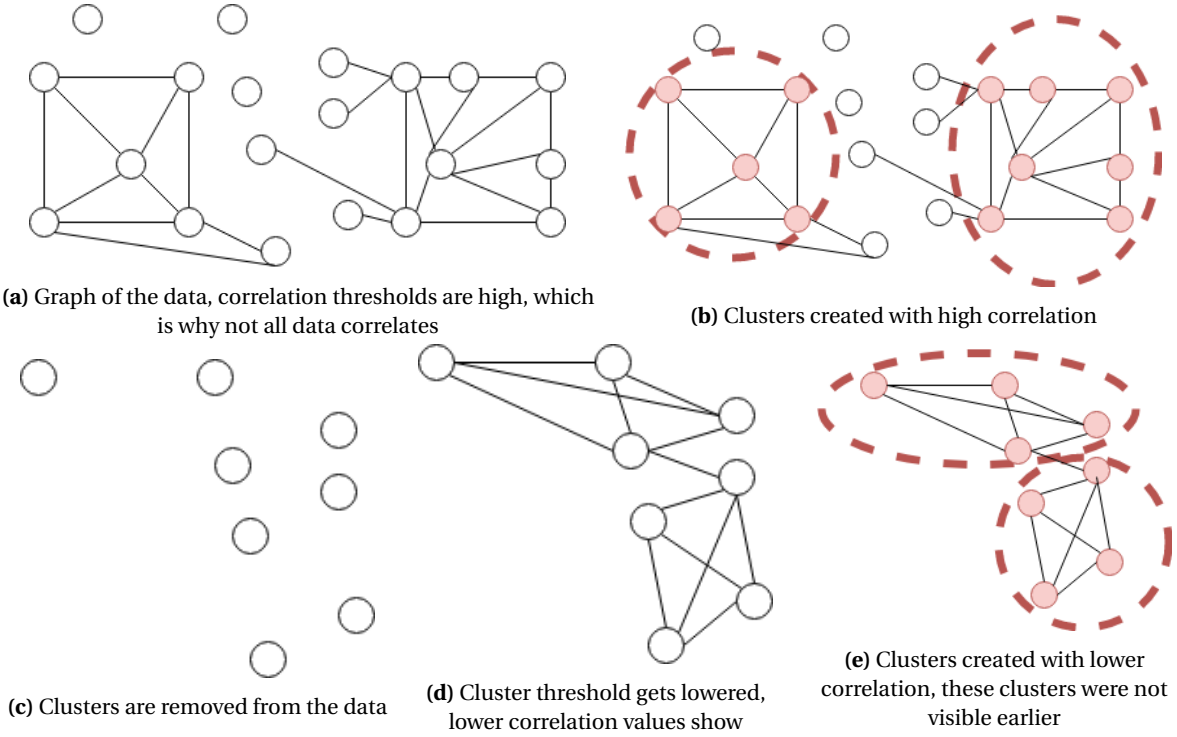
Other identifiers such as Fragment Offset, Type of Service and Header Checksum (Section 2.2) are not used as identifiers as they were not found to be a discriminating factor in the clustering. Sometimes these values caused clusters that should be together to break, which is an undesired result. Therefore, these values are not included in the clustering.

6.4.3. Clustering approach

This Section will explain the way in which the clustering is performed. Correlations between different source IP addresses have to be generated, and IP addresses with similar behavior have to be grouped. The identifiers specified in Section 6.4.2 are used in order to calculate the correlation between different source IP addresses. First, the way in which correlations are computed between source IP addresses is explained. After that, the way that the clustering is performed is elaborated upon.

Correlation

To identify how much certain IP addresses behave the same, a correlation algorithm has to be created. In this work, calculating correlations is done by comparing the values of every identifier to each other. If an identifier matches perfectly between two IP addresses, a score of ‘1’ is added to the correlation. If the identifier matches partially, a score between ‘0’ and ‘1’ is given to the identifier, depending on the match. After calculating the



(a) Graph of the data, correlation thresholds are high, which is why not all data correlates

(b) Clusters created with high correlation

(c) Clusters are removed from the data

(d) Cluster threshold gets lowered, lower correlation values show

(e) Clusters created with lower correlation, these clusters were not visible earlier

Figure 6.3: Visual representation of proposed clustering method. Pseudo code for this clustering can be found in algorithm 1. As seen from the Figure, removing high correlating clusters shows new interesting correlations

total correlation between two IP addresses, the score is normalized by dividing it by the total number of identifiers used. This will result in a number in the range $[0,1]$. Examples of correlation scores for two IP addresses are shown in Table 6.4. Correlations are computed for every pair of source IP addresses.

Normally, high correlation says something about how related two items are, while low correlation means that the items are most likely unrelated. In this case however, this is not necessarily the case. A correlation of 1 means that the values of identifiers that two IP addresses are using are exactly the same, while a correlation of 0.4 means that the values of the identifiers of two IP addresses are quite different. When looking at what we have found in the data analysis Section (5), we know that there might be some identifiers that match in the behavior of a scanner, while other identifiers are randomized by the scanner. An example of this is ZMap. As stated in Section 5.2.3, ZMap always uses IP identification number 54321. This value is therefore fixed and will correlate with a value of 1 between scans originating from ZMap. Other values however, like the sequence number, are not static when using ZMap. Therefore, the total correlation will not be 1, but the behavior of the tool is exactly the same. Thus, low correlations might give interesting information in this case. Large distributed scans do not need high correlations between IP addresses, the correlation could be very low, which the clustering method has to take into account.

Clustering

The output of the correlation is for every pair of IP addresses a score between 0 and 1. As explained in previous paragraph about correlation, simply taking all the IP addresses that strongly correlate into clusters does not work. Doing this will neglect the information that can be inferred from the correlation scores. Therefore, a method was created that gives precedence to higher correlating clusters, but will also cluster lower correlating clusters.

To solve this problem, we use the method of Figure 6.2, where data is iteratively removed from the dataset and underlying relations will show. To achieve this, the clustering algorithm consists of multiple rounds. In each round, every IP address that is already clustered will be removed from the set. After removing the IP addresses, the correlation threshold will be lowered. This ensures that highly correlating IP addresses are clustered first, and they will not create false matches with other IP addresses that cluster on a lower level of correlation.

To perform the clustering, a graph is created in which every IP address is represented as a single node. This

preserves the structure as described in Section 6.4.2, in which the data is grouped by IP address. The second step is to create the edges. Edges will be updated each round and correspond to the correlation between IP addresses. If the correlation between a pair of IP addresses is higher than the threshold of the current round, an edge is added between the IP addresses. This is done for every pair of IP addresses in the set, which results in a graph with nodes that are connected if (and only if) the correlation between the IP addresses is higher than the threshold. Within this graph, the Speaker Listener label Propagation algorithm is used.

The algorithm returns a set of clusters. IP addresses present in this cluster are removed from the dataset. After removing the IP addresses, the threshold is lowered and a new graph is created. In this graph, the Speaker Listener label Propagation algorithm is used again. This cycle continues for correlation from 1 down to 0.2. The lower bound has been chosen because for lower values, the IP addresses are found to not behave the same anymore. A visual representation of the intuition behind the algorithm is presented in Figure 6.3. Pseudocode for the entire clustering algorithm can be found in Algorithm 1.

Algorithm 1 Clustering algorithm. The set S represents the set of source IP addresses.

```
1:  procedure CORRELATE
2:      for all s1 ∈ S do
3:          for all s2 ∈ S do
4:              Correlations ← correlate(s1, s2)
5:  procedure GRAPH CLUSTERING
6:      for i = 1; i ≥ 0.2; i = i − 1 do
7:          for all correlation ∈ Correlations do
8:              if correlation ≥ i then
9:                  Add IP addresses to graph if not present.
10:                 Connect IP addresses with an edge.
11:         Clusters ← Run clustering algorithm on the graph.
12:         Remove all clustered IP addresses from the Correlations set.
        return Clusters
```

Time complexity

The calculation of correlations between IP addresses is done for every pair of source IP addresses. This is only done once, as the correlation does not change after this step. This means that in total, for n IP addresses there are n^2 comparisons needed for the correlations. For the graph clustering, given n IP addresses, the graph that is built can consist of a maximum of n nodes and $n * (n-1)$ edges. The runtime of Speaker Listener label Propagation is linear in the amount of nodes, so the amount of edges does not matter for the algorithm [51]. Because the time complexity of reading the data and grouping by source IP is also linear, the time complexity of this algorithm is $\mathcal{O}(n^2)$.

Cluster indicators

Clusters are created with with the use of several identifiers. When these clusters are created, we can extract the values of the identifiers that lead to the creation of the cluster and use this as a fingerprint. For example, when a cluster is created in which the values of the source port and IP identification number are always the same, we can use this as an indicator for other packets on a live connection. This allows for real-time clustering as we have 'learned' how to classify packets.

6.4.4. Sliding data

As discussed in Section 4.1.2, 200 MB of packet data is captured every ±20 minutes. Even with the filtering done in Section 4.3, a lot of data remains. The proposed clustering method has a quadratic time dependency, but processing the telescope data as a whole is not feasible. Therefore, the clustering is performed on single files of 200 MB. This allows for semi real-time clustering, as the clustering takes shorter than the time it takes the telescope to collect the data. While the clustering is faster than the data generation, the clustering can only be done after enough data has been captured to perform the clustering. In this thesis, we will use single files of ±20 minutes, as it makes it easy to experiment with the telescope data. Therefore the clustering is lagging behind for ±20 minutes. Although we use data of ±20 minutes, this threshold can be adapted to the situation. Fingerprinting IP addresses with fingerprints that are previously seen is of course doable in

Field	IP 1	IP 2	Correlation
Destination port	22	22	-
IP ID	54321	54321	1
Sequence numbers	2,3,4	3,4,5	$\frac{2}{3}$
Source port	5000	8000	0
IP ID / packet	$\frac{1}{3}$	$\frac{1}{3}$	1
Seqnum / packet	1	1	1
S-port / packet	$\frac{1}{3}$	$\frac{1}{3}$	1
Correlation	-	-	0.77

Table 6.4: Correlation example for two IP addresses that both sent 3 packets

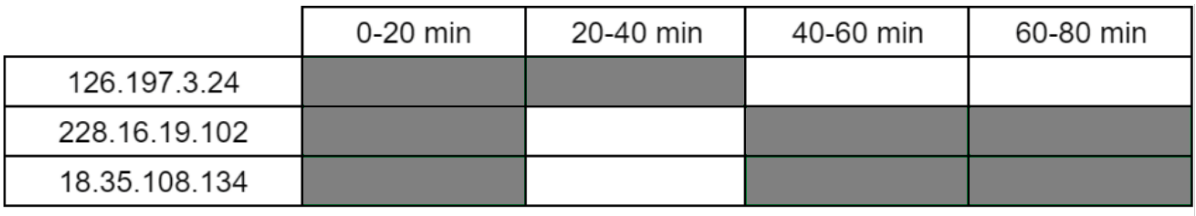


Figure 6.4: Time based analysis for three IP addresses. One can see that the last two IP addresses match completely. This analysis is used as a discriminating factor after clustering. Only 80 minutes of data are shown in this Figure.

real-time.

6.4.5. Time-based analysis

The output of the clustering is a collection of partially clustered datasets for data of approximately 20 minutes. This data format is ideal for time based analysis. The goal of time-based analysis is to identify IP addresses that are mostly scanning in conjunction. When two IP addresses always come up and go down during the same time periods, chances are that the IP addresses are used by the same group.

Because the data is clustered, there is no need to check every IP against every other IP. The clustering is done in such a way that unrelated IP addresses are not clustered together, so if IP addresses are not in the same cluster, they elicit significant different behavior. Therefore, only IP addresses that belong to the same cluster have to be compared. This significantly decreases the run-time of the time-based analysis, without compromising the correctness of the outcome.

The intuition behind time-based clustering of IP addresses is shown in Figure 6.4. For a time frame of 10 hours (30 files), the occurrence of an IP address is checked. This gives a window of 10 hours with 30 data points per IP address. The occurrence of an IP address in a file of 20 minutes is denoted with a '1', whereas a '0' shows that an IP address was not present in the given file. This results in a binary string representation of the occurrences per IP address. Figure 6.4 shows the desired result, in which the last two IP addresses can be linked together according to their time-based behavior.

Correlating the data

As discussed above, the result of our time analysis is a set of bit vectors. The similarity of the bit vectors has to be calculated in order to group the IP addresses. There are a number of ways in which this correlation can be done. The methods we have considered are the Hamming distance, Cosine similarity and Euclidean distance. Because we are only interested in complete matches, as will be explained below, these distance metrics are all considered equal. In the end, Hamming distance has been chosen as a metric, as it is the most simple and easy to implement.

Hamming distance works by taking in this case two bit vectors, and counting the times that both vectors differ. So for the 30 data points in the vector, count the times that one of the vectors has a '1', while the other has a '0'. This results in a number, which is normalized by dividing with 30. The resulting number will then be a number in the range $[0,1]$. For simplicity, we subtract the resulting number from 1. This ensures that a full match results in a 1. When the resulting number is 0, there is no similarity between the vectors at all. In Figure 6.4, the first two IP addresses will have a similarity of $\frac{1}{4}$, whereas the last two IP addresses have a similarity of 1.

Figure 6.5: Time based analysis for three near perfect matching IP addresses. In order to preserve data integrity, this is not solely used as a discriminating factor to cluster the data.

Given the similarity scores, IP addresses are grouped together. When there are perfect matches, the IP addresses are grouped by default. Given the time span in which we are doing the analysis (10 hours), we deem it to be very unlikely that perfectly similar behavior happens by chance. Even if the behavior of two IP addresses happen to be the same, for example when they are both constantly scanning, this method does not remove valuable information from the data. The IP addresses with similar behavior will still be grouped together. After identifying the perfect matches, there are also near perfect matches (eg. the similarity score is almost 1). These near perfect matches are hard to classify, as there might be multiple IP addresses with behavior that slowly diverges from a given point. Given three IP addresses: A, B and C, the similarity between A and B might be $\frac{3}{4}$, as might the similarity between A and C, while the similarity between B and C is only $\frac{1}{2}$. This relation is shown in Figure 6.5. These items are hard to classify, as they elicit nearly the same behavior as some entries, but when we would cluster these together, there might be entries that strongly diverge from one another. As we want do not want to break up slowly scanning IP addresses that are into different clusters, the near perfect matches will not be split into clusters further. Thus, only matches with a similarity of 1 will lead to a split of the cluster in the time-based analysis.

The clusters resulting from the time-based analysis will be compared at a later stage when fingerprints are made to ensure that IP addresses that might have had downtime during one of the 20 minute windows will also be correctly identified as part of a group.

Diverging clusters over time

When a cluster is found that splits up in multiple clusters over time, or when multiple clusters are found that later on merge, a decision has to be made to group the clusters or take the sub clusters. In this case, the clusters are taken as the whole group in order to preserve data integrity. This might prolong the run time in the post processing steps, but it makes sure that groups in the cluster are not accidentally broken up.

6.5. Method verification

Verifying the detection method is an integral part of the design, as this shows the validity of the proposed method. However, validating on real world data is hard, given that there is no ground truth available. We can however make an estimation on how these scanning groups would behave. Therefore, verifying the proposed method will be done by manually injecting data that can correspond with slow scanning groups. The algorithm should at least cluster the IP addresses that are intended to be part of a group in the same cluster. By doing so, we can validate that the problem space is becoming smaller while no data about specific groups is lost in the process.

6.5.1. Single file verification

Adding artificial data that resembles slow scanners is a tedious task. From [22] the behavior of one slow scanner can be studied, but this is far from exhaustive. In order to get a more detailed overview of what the behavior of slow scanners can look like we have reasoned about how they would act. Because the way that scans are regularly performed is known, as well as the techniques that can be used for scanning, an overview can be made as to which behavior can be expected from slow scanners.

From the overview of different hiding techniques in Section 6.1, we have created an overview of different scanning techniques that could be used by attackers with different sophistication levels. The overview can be found in Table 6.5. For the verification, artificial scanning groups have been added to ten 20 minute files, for almost every combination in the table there is one group. We could not make a group for every combination, as it does not make sense to have a single-source scanner that is highly distributed for example. In total, we have added 3060 groups to our data. To explain in an intuitive way, five of the groups that are added to the data are shown in Table 6.6. We will use these groups to illustrate the verification.

Note that we also add groups that have a low amount of source IP addresses and send a low amount of

Hiding method	Scale	Values
#Source IP addresses	Numerical	1, 10, 100, 1000, 5000, 10000, 100000
#Packets per source IP	Numerical	10000, 1000, 100, 50, 20, 5
#Destination IP	Nominal	Overlap, no overlap
Randomizing packet fields	Nominal	None, 1 field, 2 fields, all fields with one random per IP, all with one field per 3 packets, all fields
Same subnet	Nominal	Same subnet, multiple different subnets, no subnet
Sequence number relation	Nominal	Static, XOR relation, random

Table 6.5: Different levels for 'hiding' scanners. For every combination, an artificial scanner is added in the data. Values are sorted from low evasion to high evasion.

Group	#IPs	Packets per IP	Random fields	Dst IP	Subnet	Seq num
A	100	5	1 field	No overlap	Yes	Static
B	5000	100	1 field	No overlap	No	XOR
C	5000	1000	2 fields	Overlap	No	Static
D	1	10000	All fields	Overlap	-	Random
E	100000	5	All fields	Overlap	No	Random

Table 6.6: Examples of groups that have been added to the data to resemble slow scanners. These groups are added to the data in the verification of the clustering algorithm.

packets. 10 IP addresses sending only 5 packets per 20 minutes would scan the telescope in about one month. This is not a realistic scenario for attackers, as hosts will come up and go down a lot in this time. However, it is useful in validating our method, so we have chosen to keep this in the analysis.

Results of the verification of five groups are shown in Table 6.8. As can be seen from the table, all the IP addresses of the added groups are clustered in the same cluster. This means that no groups have been broken and no relations have been lost. Some IP addresses are added to multiple groups, which is expected (explained in Section 6.4.3). From the table it can also be seen that there are groups that are clustered entirely in their own cluster. Therefore, some slow scanners can be captured using only this clustering approach.

In total, our method is able to detect 2312 groups out of all the groups that were added to the data. This means we are able to detect 75,55% of the groups that we have added in the data. These groups are the groups that are exactly matched in one cluster. If we relax our measure a bit, by counting groups in which at least 95% of a cluster is one group, we detect 2692 groups. This means that when we allow 5% of noise in the clusters, we detect 87,97% of the added groups. From this result we can see that there is a gradual decrease in detection. The cumulative density function of the noise level versus the group detection is shown in Figure 6.7. In the figure it shows that from 0% to 5% there is a large increase in detection, after which the detection rate only gradually increases. Therefore, there is a tradeoff between the amount of noise and the amount of detection the method achieves. We find that for 5% noise, we obtain a 16,4% increase in the groups that are detected. After this, the detection rate increases slower than the added noise. Therefore, we will take a 5% noise limit as another variant of our method, in which analysts can trade a small noise threshold for increased detection.

To better understand the result, we have to look at what we do detect, against what we do not detect. Making this analysis will show how sophisticated scanners have to be in hiding until we cannot detect them anymore. Figure 6.6 shows how well our method detects groups that use various ways to remain hidden.

In the figure, we can see that the exact clustering falls down when high evasion tactics are used. The noisy method does perform better, but mostly because it counts all the groups where 100.000 IP addresses are in the same cluster as a group. This means that there can be up to 5000 other IP addresses in the cluster with these groups and they would still classify as a group. From the other results however, we can see that the noisy method actually performs equally or better on any group. Therefore, this method is a valid improvement on the recall of groups, but comes at the cost of losing at most 5% precision, as this is the noise that is allowed.

For the sequence numbers in Figure 6.6b, we see that almost all scanners containing an XOR relation are all found and correctly clustered by our method. This validates the XOR-analysis method that we use to identify these patterns in the data. Secondly, we can see that we correctly identify most of the scanners that are using a static sequence number. However, when a random sequence number is used, the detection rate goes down significantly.

#IPs	Packets per IP	Random fields	Dst IP	Subnet	Seq num
1	5	2 fields	No overlap	Yes	Static
1	5	all	No overlap	Yes	Random
100	100	all	No overlap	No	Random
100	100	all	No overlap	Yes	Random
10.000	20	all	No overlap	No	Random

Table 6.7: Undetected groups after running our algorithm, not all undetected groups are listed here.

Group	Size	Grouped in one cluster	Additional IPs in cluster	IPs also in another cluster
A	100	yes	0	2
B	5000	yes	0	0
C	5000	yes	0	0
D	1	yes	0	0
E	100.000	yes	3461	5178

Table 6.8: The result of running the clustering algorithm with the artificial data.

As with the sequence number, the percentage of detection that the method obtains when there is a low amount of packets, or randomization of all fields, also drops significantly. While here there is a significant drop, in order for an attacker to evade detection, the attacker has to be adept in most different hiding techniques. The results do however indicate that while we are able to detect many different levels of sophistication, the very sophisticated scanners have a high chance of remaining undetected even with our method. In Table 6.7, we have listed several groups that were undetected by both our detection methods. Not every group is listed, but it can be seen how groups could evade detection.

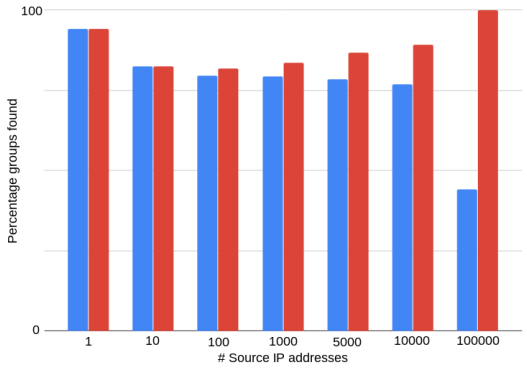
6.5.2. Time-based analysis verification

Verifying the clusters after time-based analysis is done by generating cluster data for 10 hours of data. The files are modified to contain the same groups as used in the verification of the clustering method (Table 6.6). For every group we first added a group with a time-based fingerprint in the data, and in a second run a group that cannot be clustered according to the time-based matching. The groups could not be added during the same run as they might interfere with each other. The results of the verification for five of the artificial groups are included in Table 6.9.

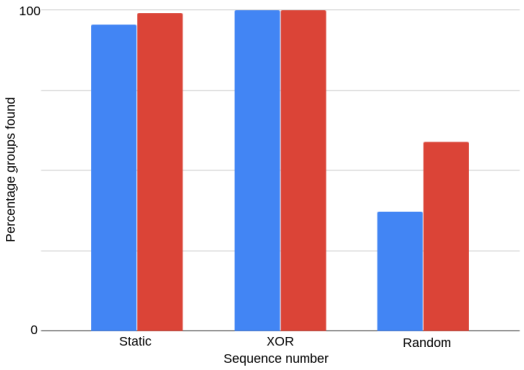
As can be seen from the results, most groups are clustered correctly after the time-based analysis. Groups that were first part of bigger clusters are now, after the time-based analysis, part of smaller clusters in which only IP addresses belonging to the group are present. In cluster E_{Time}, wrongly clustered IP addresses in the cluster are clustered correctly after the time-based step. Without time-based patterns, cluster E grew over time, as more IP addresses that matched showed up in other files and no time-based analysis could be made. In this case, the amount of data grows which obfuscates the group, but in more general cases this might be desirable as IP addresses of the scanner might not be present in only one window of 20 minutes. We are however not able to reliably detect group E in this case, as there is too much noise. This means that there is

Group	Cluster broken	Additional IPs in cluster	IPs also in another cluster
A	no	0	7
B	no	0	0
C	no	0	0
D	no	0	0
E	no	6458	12742
A_{Time}	no	0	0
B_{Time}	no	0	0
C_{Time}	no	0	0
D_{Time}	no	0	0
E_{Time}	no	0	0

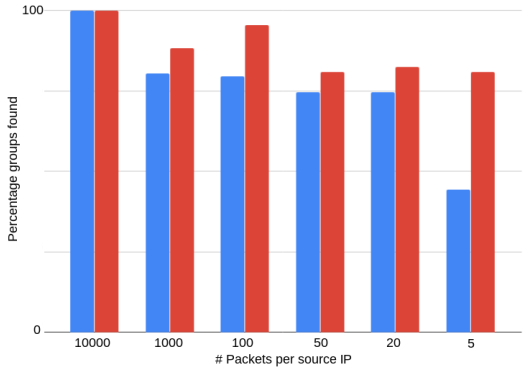
Table 6.9: Verification of time-based clustering. Artificial groups that are added in the first run do not have a time-based fingerprint, the second run contains groups that do have time-based fingerprints.



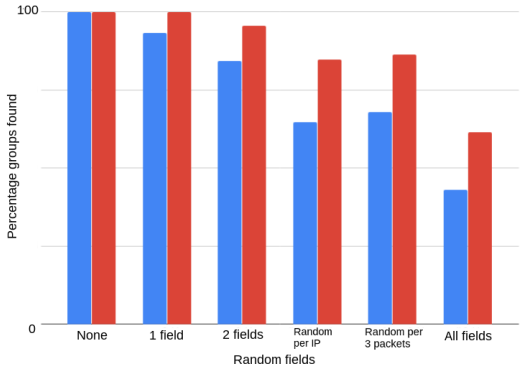
(a) Groups found with respect to how many different source IP addresses are used.



(b) Results of groups found with respect to the XOR analysis in the sequence number.



(c) Groups found for the rates in which they send packets.



(d) Groups found in terms of randomization in the packet fields.

Figure 6.6: Clustering results in terms of various evasion tactics. The blue bars show the exact method, the red bars show the method that allows 5% noise.

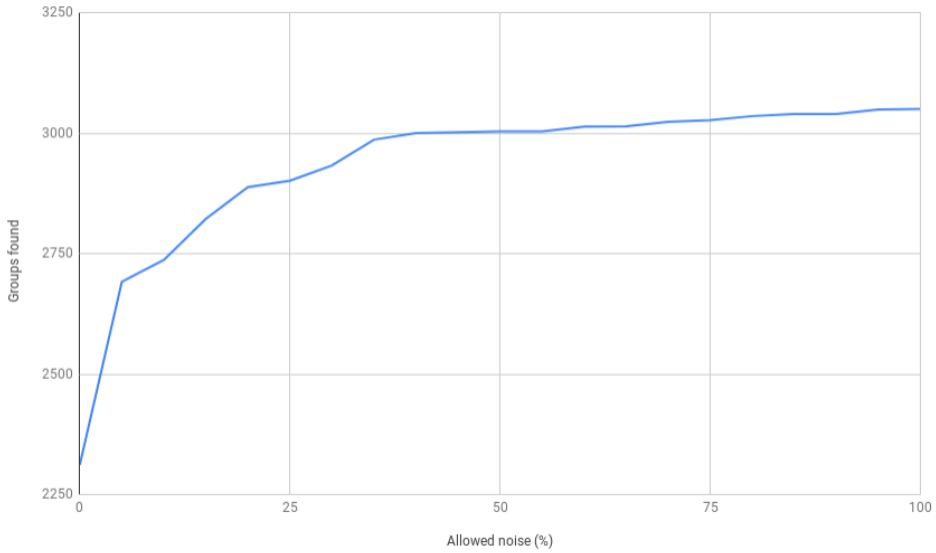


Figure 6.7: Cumulative density function of the impact of allowed noise in clusters on group detection measured in steps of 5%. Vertical axis starts at 2250.

Method	Groups found	Recall	Precision
Proposed method (exact)	2312	75,55%	100%
Proposed method (5% noise)	2692	87,97%	98,14
Subnet method	983	32,12%	100%
Destination IP (no added data)	509	16,63%	100%
Destination IP	0	0%	0%

Table 6.10: Results of the comparison between the proposed method and the state of the art. In this table, only the groups that are added in the data are analyzed. The destination IP method is run on a theoretical basis, without real data.

a boundary until which we are able to detect scanners, when they are too hidden in multiple dimensions, we are unable to detect them.

Groups that did have a time fingerprint are all correctly clustered without any additional IP addresses in the cluster. These clusters resemble coordinated scans that are only conducted at specific times. A clear cut can be made between the detection of scanners that are scanning in a very timely manner and the scanners that do not, and we see that time-based analysis is able to capture all groups that are eliciting the same behavior and are scanning at the same time.

6.5.3. Comparison to the State of the Art

An important validation step of the analysis is to compare our proposed method to the current state of the art. To make this comparison, we will again use the different ways a scanner can use to circumvent detection methods to estimate if, and how our method performs better. The methods that we will compare against are the method by Gates et al. [30] and Robertson et al. [45]. These methods are respectively clustering groups by finding a Set Cover of destination IP addresses, and by grouping IP addresses per subnet. Because these methods do not support time-based verification, we have excluded this from the comparison. In the comparison, we use 10 files where we have added the artificial campaigns from Table 6.5.

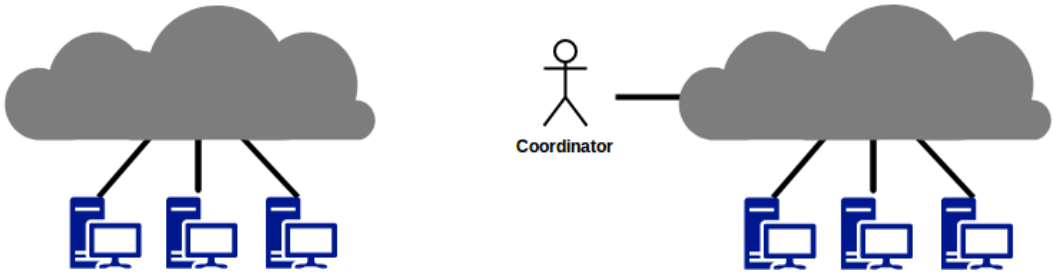
For the destination IP analysis mehod, we ran an approximation algorithm on our data that computes a set cover in destination IP addresses. In a dataset of 20 minutes, we have not been able to find any groups using this method, let alone slow scanning groups. The reason behind this is that the dataset contains so much 'noise' from other scanners that it is not feasible to do this kind of analysis. Because the detection rate of this method is always 0, we cannot validate our method against this method in a meaningful way. In order to make a comparison between this method and the proposed method, we will run the destination IP algorithm on only the group, without any other data in the test set. While this is not a fair comparison, we can analyze how the theoretical result of the destination IP cover method compares to our method. Even though we are comparing with this method, this flaw clearly shows that this method is unusable in practice. We have used a set cover size threshold of 95%, as this was used in the original work.

Grouping IP addresses by subnet will still need a threshold for which it will detect the group. We took a threshold of 60 packets per hour, which is a very generous threshold as it will not be sustainable in a production environment due to the amount of data you will have to store. A threshold of 60 packets per hour means that we have a threshold of 20 packets per 20 minutes. Some group sizes do not fit into one subnet, as one subnet is only 256 addresses large. Therefore, we split these groups over multiple subnets and will mark the group as detected if at least 95% of the IP addresses are detected.

From our comparison, we obtain the results listed in Table 6.10. In the table we see that our method outperforms the current state of the art quite significantly in terms of the artificial data, which corresponds to the hiding techniques listed in Section 6.1. The subnet approach can not detect all parts of this data, as not all scanners are configured to use the same subnet. While this is now done with artificial data, we will see in Chapter 7 that there are a many groups that do indeed distribute their scanning activity over a large number of subnets, which means that the subnet method will not be able to find this. The method of finding a destination IP set cover does not perform well in the theoretical scenario, as seen from the table, and cannot be used in a practical scenario as it is prone to false positives. Therefore, we conclude that our method outperforms the state of the art both in feasibility and detection rate.

6.5.4. Bias in verification

Inserting artificial data in the telescope data to verify the clustering method can result in a bias for the verification result. Although it is known for some slow scanning groups ([22]) how they behave, there is no guarantee that every group behaves this way. While an exhaustive verification has been attempted, this analysis method



(a) Scanner group without strict coordination, overlaps in destination IPs are more common

(b) Strictly coordinated group with a 'master'. Destination IP addresses will overlap less

Figure 6.8: Coordination in scanning groups

might not be exhaustive for all slow scanning groups.

6.6. Result verification methods

Due to the absence of ground truth, other methods for verifying the obtained results are needed. In this Section, two methods that can be used to verify the results are proposed. The first verification method is based on the distributed scanner detection method by Gates et al. [30], where the distribution of destination IP addresses is evaluated. The second method relies on implementation characteristics, sometimes these are mistakes by the programmer so that strange behavior shows, and sometimes this is a programming decision.

These methods will be used in Chapter 7 to analyze whether IP addresses that are clustered together are actually one group or multiple.

6.6.1. Destination IP analysis

Gates et al. [30] coined a distributed scanner detection method that works by analyzing the destination IP addresses which are scanned. Given a highly coordinated group, one would expect that the destination IP address distribution from that group does not have any overlaps. Figure 6.8 shows two kinds of coordinated groups. The graph in Figure 6.9 shows the information gain for an attacker given the progress of the scan. When an attacker uses a highly coordinated scanning structure, the information gain is linear to the progress of the scan. This means that this level of coordination can be detected in the telescope by finding a set of IP addresses that exactly cover the entire telescope with their scans. However, there are chances that these relations are spurious, as there are so many scans going on that there are chances for IP addresses to form a set cover even though they belong to different groups. For less coordinated or sophisticated groups, these set covers are even harder to find as they might not even exist.

While there are chances that spurious set covers that are found in the data, this chance can be minimized. This is done by using the result of the earlier clustering approach. Because the set covers are being evaluated over a lower amount of data, namely only inside a cluster, the amount of IP addresses that can interfere with the scanning group is minimized. Using the clusters also results in a feasible dataset to run the set cover algorithm, which is an NP-complete problem.

For groups that are very sophisticated and scan 100% of the telescope, this detection method works. Even given packet loss, these groups can be detected using the method described by Gates et al. [30]. This is done by setting a threshold that allows detection when a group scans 95% of the telescope. Less sophisticated groups are however undetectable via this method, as they might not scan 95% of the telescope. Still, this method is considered a good addition to the detection 'chain', as it might be able to detect some groups.

Next to detecting scanning groups, this method of set cover can be used to verify groups and estimate their coordination pattern. For a group that was found using another method, it can be verified what its set cover is. Verifying a set cover is doable in polynomial time, which means that even for very large groups, this will not put much strain and computation time on the system.

6.6.2. Implementation characteristics

In Section 5.2, a lot of different patterns in the data have been shown. Some of these patterns are the result of programming decisions and in some cases mistakes. An example of a decision is the diagonal line as was seen in Figure 5.6, where a linear relationship has been created between the IP identification number and

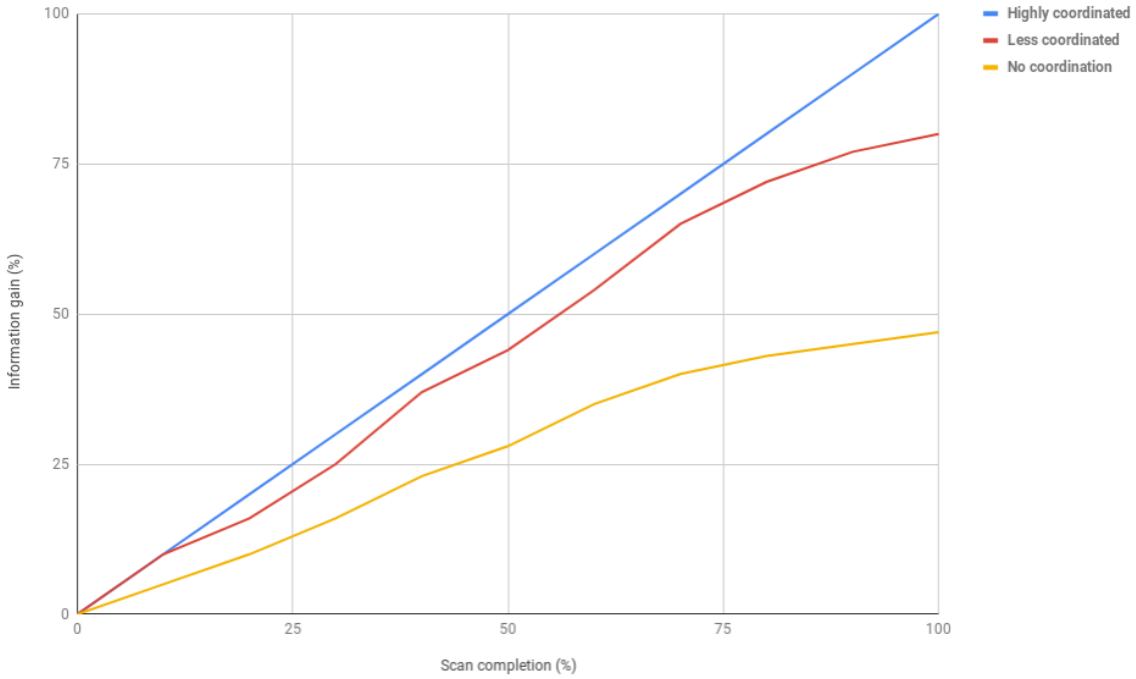


Figure 6.9: Information gain for a scan in various sophistication scenario's.

destination port.

Mistakes can be for example the usage of a signed short to randomize the IP identification number. When a positive number is generated using a signed short, the most significant bit will not be used. This means that the space in which the randomization takes place is half of the total space. These small mistakes can be used to detect groups, but will currently rely on manual labor as there is no method to detect these mistakes in packet data yet.

For now, this method will be used in order to verify the clusters that were found in the data. But given the proposed methodology, in which the dataset is pruned every step to show underlying relations, it becomes more feasible for human analysts to find these implementation mistakes. Given visualization methods proposed in Section 5.2, this analysis can be improved to create an easier and more streamlined workflow for the analyst. By using the same workflow as proposed for fingerprinting, in which the data is pruned after every detection step, a human analyst can filter the data further. To validate this method, further research is needed to verify the result of this visualization workflow.

7

Results & Evaluation

In this Chapter, the results of the method described in Section 6 are discussed. First we will look at groups detected and some characteristics on which these groups were found. Secondly, some of the groups that are found are highlighted to show how groups behave. Finally, this Section will verify the result that was obtained by analyzing the result in order to detect groups of scanners that are known to be in the data.

For this Section, analysis has been done on roughly 2.5 weeks of telescope data. Note that no IP addresses of the groups that were found will be listed for ethical reasons. If we list IP addresses, we will obfuscate them.

7.1. Groups found

Given the analysis done using the clustering from Section 6.4 and post-processing from Section 6.3, various groups have been identified. For all these groups, fingerprints could be extracted which uniquely identify the group. This Section also shows the advancement of using this method as opposed to previous proposed methods. For all the groups listed in this Section, we show if the group would have been detected using any of the previously proposed methods. The evaluated methods are: 1. the method of Gates et al. [30], in which the coverage of destination IP addresses is analyzed and 2. clustering the IP addresses in the same subnet as in [45]. More detailed analysis of groups and why the previous methods fail to detect them is given in Section 7.2.

7.1.1. Tools found using XOR-analysis

In this Section we will discuss the results of the XOR analysis in finding different tools with regards to the sequence number used in the packets. In total, we have discovered 8 distinct tools that use different combinations in their sequence number. The tools that we have found are listed in Table 7.1.

As stated before, we are not able to know whether the scanner uses the source IP together with the session key. However, for Unicorn we only know that the source port is part of the equation because we know the tool. This might also encapsulate IP addresses that removed the Source IP from this equation. Therefore, Unicorn in the Table might be comprised of IP addresses that use two tools.

We have identified different groups using certain tools. Groups were mainly split on the port that was probed and the time-based analysis. When no sensible conclusion could be made about the number of groups, we have concluded that there are many people probing the internet using these tools. In Section 7.2, some of the found groups are investigated in more detail.

7.1.2. Groups found using clustering

Next to the groups found with XOR-analysis, we also find groups using the clustering method described in Section 6.4. The groups that contain a XOR relation in the sequence number are not in the data anymore, which leaves us with the clusters that do not contain the relations that we can find with our algorithm. For some of these clusters, we manually inspected the outcome in order to find if the clustering algorithm worked as intended and if a human analyst could make sense of this data when doing analysis. Also, we investigated if we could find some strong indicators from these groups, with which they could be blocked.

In total, the algorithm created 1249 clusters. We have manually inspected some of these clusters for our analysis. In total, we have inspected 50 clusters, of which 43 were considered distinct groups after manual

Tool	Fields in equation	Prev. methods detect	# Groups
Unicorn	Session key, *Src IP*, Src port, Dst port	yes ([32])	many
Destination IP	Dst IP	no	2
Dot zero	Shifted Dst IP	no	1
IP ID	Dst IP, IP ID, Dst port	no	1
All	Session key, Dst IP, Dst port, Src port	no	many
Dst session key	Session key, Dst IP	no	1
Dst IP & ports	Dst IP, Dst port, Src port	no	many
Src IP & ports	Src IP, Dst port, Src port	no	5

Table 7.1: Tools found using XOR-analysis. Tool name is chosen as a unique identifier of the group. We have also indicated how many groups we have manually found with the fingerprint.

inspection. For some of these clusters, results are given in Table 7.2. We can see that in most clusters, the algorithm has actually clustered distinct groups. However, for some groups we are not able to do this. Take for example the ZMap #1, this group is not clustered on its own as everyone in this cluster is actually using the same tool, which is ZMap. When there are no concrete time-based relations found within the groups, there is no way to distinguish between the different source IP addresses. Within the ZMap #1 group we have seen some subnet clustering, but we do not know if the different small subnet clusters belong to the same group or not.

For most other clusters however, we have found that the clusters actually contained only one group. For ZMap #2 for example, the number of packets sent per IP address was exactly the same, which means that there are exactly the same amount of IP IDs, sequence numbers and source ports used per packet. Because we use this as an identifier, this group is clustered together. Also, the port scanned in this group is so distinct that we concluded that it was probably one group.

Block scan #1 is rather interesting, as we have seen this scan before. In Figure 5.5 this behavior was highlighted because it stood out in the graph. Here we can clearly see the added value of good visualization. Previous methods would not have found this group, as it does not generate enough probes to be easily detected. With our method we find this group as the probed destination ports are very distinct.

Random group #1 is a group that contains items that failed to be clustered, as all values are seemingly randomized and the destination port is not distinct. Therefore, only the destination ports match. However, after clustering so many other groups, this group is not that large anymore. This makes other analysis feasible. In this case subnet analysis [45] and destination IP analysis [30] might be feasible as the dataset has been significantly reduced and therefore the analysis will be less likely to turn out with false positives. From further analysis in this pruned dataset we have found another group that uses a distinct method of choosing the IP identification number. This group is analyzed further in section 7.2.5.

In random group #2 we have actually identified a distinct scanner group, as the group only came online for one and a half hours before going offline again. This behavior was captured by our time-based analysis technique. Given this, together with the fact that the same port was targeted, we have concluded that this is most likely the work of one scanning group. Within this time, the overlap in destination IP addresses was ±10%, which is not enough to attribute this scan to two groups, as then the overlap should be closer to 100%.

7.2. Group analysis

From 7.1 it is clear that many groups have been found that are supposedly working together. In this Section, some groups of the many that have been found will be analyzed in more detail. This is done in order to show the coordination patterns that exist in groups, as well as to verify by random sampling that the groups found are actually working together.

We will compare the results that we have found to existing methods and explain why the previous methods would or would not find these groups. The methods that will be compared against are the method proposed by Gates et al. [30], which uses set cover. As well as the method proposed by [45], in which IP addresses that are close together (in the same subnet) are clustered.

7.2.1. Destination IP group #1

The first group that this Section will highlight is the group that we call the destination IP group. From the data that was analyzed, there were 20.621 IP addresses that fit in the behavioral profile of this group. These

Group	Corr.	Identifiers	IP addresses	Prev. methods detect	Distinct group
Static #1	1	all	220	no	yes
Static #2	1	all	20	yes (same subnet)	yes
Heavy hitter #1	0.4	Src port	1 (single-source)	yes (# packets)	yes
ZMap #1	0.3	IP ID	859	-	no
ZMap #2	0.5	IP ID, #packets	14	no	yes
ZMap #3	0.3	IP ID	20	yes (same subnet)	yes
Block scan #1	0.5	#packets	27	no	yes
Random #1	0.2	none	11275	-	no
Random #2	0.2	none	129	no	yes
Three dest IP	0.6	IP ID	3	no	yes

Table 7.2: Groups analyzed from the clusters. Group name is chosen as a unique identifier of the group. The correlation between IP addresses in the cluster has been added in the Table. Whether this group is probably only one group is added in the Table. Note that all groups are at least split on destination port, because the clustering algorithm only considers IP addresses that are scanning the same ports.

IP addresses are distributed over the whole IPv4 space. Because there are so many IP addresses, this group can scan in a very slow manner. IP addresses in this group send at most 20 packets to the telescope per 60 minutes. This rate of sending packets would stay under the radar of any currently used IDS system we are aware of.

For why this group is dubbed the destination IP group, this is because the sequence number exactly matches the destination IP. By using this sequence number, the packet that is returned to the scanner is easily verified by subtracting 1 from the sequence number (see Section 2.3.2 and 5.1.5). The equation for the sequence number is therefore:

$$Sequence\ number = Destination\ IP\ address$$

As mentioned before in this thesis, the destination IP distribution can indicate how coordinated a group is (6.6.1). For this group, there is some overlap in the scanned destination IP addresses. The most overlapped IP address has 24 packets send to it by this group in total. This indicates that there is loose coordination, but the group is not setting off any alarm bells in an IDS by probing an IP address too much. Because the scanner is sending at most 20 packets per source IP address, it does not stand out when looking at the packets that have been send from a particular host. Also, given the strategy with which sequence numbers are chosen, no sequence number stands out in analysis.

The target of this group is destination port 5555. This port runs numerous protocols, so it is hard to know what the group is scanning for specifically. For the IP identification number and the source port, no relation could be found. These values seem to be randomized as intended. For the source port it does hold that it is randomized every time that a packet is sent from an IP address, but this is expected behavior given the time between the packets originating from one machine.

While the method of choosing the sequence number is simple, it is a full copy of the destination IP address, this does go undetected by current scanner identification methods. Analyzing the network traffic manually will also not show this relationship, as IP addresses are visualized in the IP address notation (Figure 2.5), whereas the sequence number is represented as a single integer. Also, because of the distributed nature of source IP addresses and the amount of packets per destination IP address that are sent by this scanning group, it does not create significant outliers in the data.

Next to this group, we have found another group using the same XOR strategy. This is classified as another group because the port scanned is different, which means that the objective is different. In this case, the port scanned is 3389, which is the port used by the Windows Remote Desktop protocol. Because we applied the clustering method before performing the XOR-analysis, these groups are split automatically.

Previous detection methods

For this group, the source IP addresses are very distributed, with some IP addresses in the same subnet, but most IP addresses outside other subnets. Therefore, the method of Robertson et al. [45], will not find this group.

Given the set cover of this group, the group scans all destination IP addresses. There is however a large overlap in the scanned destination IP addresses. This would mean that the algorithm proposed by Gates et

al. [30], would only find subgroups of this group, as some source IP addresses are redundant. Also, there is a high chance that other IP addresses that are not part of this group are clustered with this group, as this group has so many source IP addresses that there will be set covers with other IP addresses as well.

7.2.2. IP identification number group

The second group that will be analyzed in more detail is the IP identification number group. This group uses the IP identification number in the XOR equation of the sequence number. For this particular group, only two of the four bytes in the sequence number have been attributed. For the last two bytes of the sequence number, the equation if known, which is:

$$Sequence\ number = Destination\ IP \oplus IP\ identification\ number \oplus Destination\ port$$

For the first two bytes of the sequence number, the equation could not be found. Numerous different ways of constructing the sequence number have been tried, such as XOR-ing the first part of the IP address with the second part, or reversing the order of the bits. All the different equations did not find a match. This could be due to either this part of the sequence number being randomized or created via a different function like a block cipher.

The amount of source IPs attributed to this group in the analysis is 153. This is not a large number of IP addresses, and they are scanning in slow fashion. One source IP address is only scanning one destination IP address once. Different destination IP addresses will however be scanned by the same scanner. An interesting finding from this particular group is that a very particular strategy for choosing the destination IP address to scan is used. Each source IP address is scanning three IP addresses in a subnet before moving on to the next subnet. We do not yet know how these destination IP addresses are chosen. The fastest scanning IP address in this group sends 84 packets in 20 minutes to the telescope, which can be categorized as a slowly progressing scan which will remain undetected by current IDS solutions.

As for the destination ports of this scan, multiple are targeted. The scanner targets 6 different ports on each host, with the most scans going to port 22 (the SSH protocol). All source IP addresses are scanning all the destination ports on different destination IP addresses. Because each source IP will probe each destination IP only once, the total range of ports that this scanner targets is scanned with 6 different IP addresses for each destination IP address.

The IP identification numbers seem to not be random as well. The IP identification number is the same for every packet that is sent to the same destination IP and port. The source IP address does not matter in this case, as different source IP addresses use the same IP identification number if the same port and IP address is scanned. From these observations we can assume that the destination IP address and destination port are part of the equation that makes the IP identification number, but it is unknown what the equation is. Also, there is probably no session key involved, as the chosen value is the same for different source IP addresses.

Source ports are less interesting, as they are randomized every packet. No notable relationship could be found in this, but this does not rule out the possibility that there is a relationship between this and other fields. Just as for the IP identification number, the equation can be unknown, but there might be an equation.

So for this group, a fingerprint can be made with which the group can be mapped. One indicator is the sequence number, which has two bytes that can be predicted, while other indicators are scanning behavior (only scanning a destination once) and destination ports. Also, given a destination port and IP address, the IP identification number used will be known if the combination has been seen before.

Previous detection methods

The source IP addresses in this group are very distributed, with no IP address located in the same subnet as another IP address. Therefore, method [45] would never find this group.

As for the method of [30], the group does scan the whole telescope, and could therefore in theory be found using this method. However, there is a lot of overlap because an IP address only scans another IP address once, but a lot of destination ports are scanned which leads to high overlap. Therefore, only a subpart of this group would be found. Additionally, the group is scanning in such a slow pace, that before the whole telescope is scanned, there is a very large amount of data to go through. This group will therefore not be found as there are too many false positives that can occur and the processing load is very high.

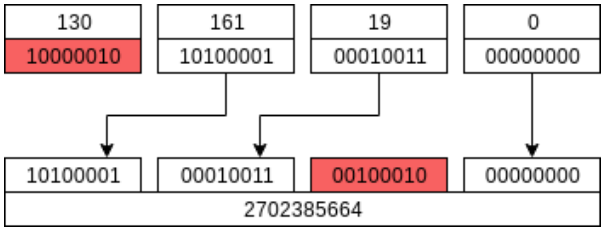


Figure 7.1: Sequence number generation for the *Dot Zero* group. No relation has been found for the third byte of the sequence number. The first byte of the destination IP address seems to be unknown.

7.2.3. Dot zero group

The dot zero group is one of the stranger groups that have been uncovered. This group targets only IP addresses where the last byte is zero. For example: 130.161.122.0 will be scanned, but 130.161.122.11 will not. Why only these IP addresses are scanned, and not other IP addresses, is currently not known. Further study in what can be achieved by scanning these particular kinds of IP addresses should be done. It might be targeting internet gateways for example, but it is unclear what this would accomplish.

The speed of this scan is very low, with each IP address sending between 1 and 20 packets a day. While it is very slow, there are still multiple ports that are being scanned. The scanner mainly targets port 130 and 2434 which host *cisco FNATIVE* and *pxc-epmap* respectively [13]. These are both protocols that are used in routers and firewalls. Also, port 131 and 2435 are scanned, albeit less intensively. These ports are also related to routers and firewalls, which might therefore be the target of this particular scanning group.

The source port used by this scanning group is particular, as it is constant over the same subnet. Take for example two IP addresses in the same subnet: *a.b.c.100* and *a.b.c.200*. These IP addresses will use the same source port. An IP address in a different subnet: *c.b.a.100* will use a different source port, but also the same source port as any IP address in its own subnet. This raises the question whether the source port is related to the first three bytes of the source IP address, or that maybe one machine is responsible for all the scans in a subnet. An adversary might use only one physical system in order to scan, while multiple IP addresses are used to stay under the radar.

The sequence number of this group is also not random, but is generated using part of the destination IP address. The third byte of the sequence number is unknown, no relation could be found for this. It is also different between scans, which might mean that the source IP address is part of the equation, or the IP identification number is. Figure 7.1 shows the relation of the sequence number in this particular group.

Previous detection methods

In this group, several different subnets are part of the scanning group. These subnets will be detected by the method proposed in [45]. However, these subnets would not be linked together, which means that the method will detect several different scanner groups while in reality, it is only one.

In the data that was analyzed, the group did not scan the entire telescope range, but only the IP addresses ending in *".0"*. Therefore, the method proposed in [30] would not find this scanning group with the threshold as is. To detect this group, the threshold should be lowered to be at least $\frac{100}{256} = 0{,}34\%$, as there are 256 IP addresses in a subnet and the scanner only scans one per subnet. This suffers again however from the amount of 'noise' that would be generated. By doing this we would see many false positives, as there are a lot of combinations of IP addresses that will generate a set cover that spans such a small portion of the telescope range. It is obvious that this would be totally unfeasible.

Another interesting fact to note, is that with a naive XOR brute forcing approach, this group would not have been found. Not only is this unfeasible, but shifting the destination IP address as done in this group would not have been found. To accommodate a brute forcing approach that would find groups with a relation like this group would mean that even more relations have to be checked, as we have to try every permutation of the raw bytes. This could mean that we also find a lot of false positives.

7.2.4. Static group #1

Static group #1 has not been found using XOR-analysis, but in post-analysis inside the clusters. This reason this group is labeled as 'static' is the absence of any randomization. Every packet by this group uses the same source port, same IP identification number and same sequence number. Therefore, the correlation between the IP addresses within this group is 1.

Field	Static #1	Static #2	Close match #1	Close match #2	Close match #3
Dst port	3389	3389	3389	3389	3389
Src port	4935	65351	random	4935	4935
IP ID	9496	9496	9496	random	9496
Seq num	2406000322	2406000322	2406000322	2406000322	random

Table 7.3: Closely matching groups found in the clusters. All groups seem to be a variation of each other.

The group does scan the entire range of the telescope, with a small overlap in destination IP addresses. The port that is scanned is port 3389, which is the Remote Desktop protocol. It does so in a very slow manner, with the packets sent per hour being at most 30. In the total amount of data, this behavior remains undetected by IDS systems. The source IP addresses that are part of this group are very distributed, with no IP address being in the same subnet as another IP address.

Table 7.3 shows the values 'Static group #1' uses in the packet header fields. Additionally, groups that were found that closely match this static group are listed in the table. We can conclude that there are multiple variations of this scanner, all interested in the same port. Whether this is one group that uses variations of their own tool, or different groups that have made small alterations to a shared tool is unknown.

Previous detection methods
Although this group uses a naive approach in selecting the packet fields, it is not detected by previous methods. Previous methods do not look this far into the packets. The distribution of IP addresses for this group is very distributed, which makes subnet analysis impossible. Between the different variations, there is much overlap in the scanned destination IP addresses. Therefore, finding a set cover of destination IP addresses, even without looking at other data, will give a lot of false positives.

7.2.5. Incremental IP identification number group

After analysis of a large random group, we discovered a particular strategy in selecting IP identification numbers used by one group. The group increments the IP identification number by one each time a packet is sent. In one hour, the IP addresses with this behavior send a maximum of 24 packets to the telescope.

From this group we can sometimes see that an IP address scanned a destination that is not present in the telescope. We can see this because there is a gap in the increments of the IP ID, in which we see the IP ID suddenly increase by more than 1.

Previous detection methods
Previous methods would not have been able to find this group, as there is no subnet relation and not all destination IP addresses are scanned. Note that our method did not find this distinct group either, but only after manual inspection of a larger cluster.

7.3. Results against known scanners

To verify the results of the method proposed in this work, the result is tested against a known group. Next to analysis of known groups in the data, the previous Section (7.2) has analyzed different groups in more detail. Because of the coordination patterns that emerge from the analysis, this can also be seen as a verification of the result.

The main groups that have been analyzed in the data are the University of Michigan, which uses ZMap in order to scan common ports on the internet, and Shadowserver, which is a group of volunteers that provide threat intelligence free of charge. This group is not trying to hide as slow scanners do, but the group does use common tools that are used by a large number of people. Therefore, this group will be 'hidden' in the total number of groups that is scanning the telescope using ZMap.

Detecting whether an IP address belongs to the University of Michigan is trivial, as it originates from the same subnet. Note that because of this, the method of [45], in which scanners are clustered by subnet are detected, will find the University of Michigan. However, this analysis is made to verify the result of our detection method. In particular because our method is agnostic of the subnet in which the IP addresses reside.

Port	#IP addresses	Wrongly clustered	Other IPs in cluster
20.000	16	0	0
502	16	0	1
143	16	0	2
7	14	2	0
80	80	0	2
8888	16	0	1
143	16	0	14
22	16	0	17
53	16	0	0

Table 7.4: Clustered scans conducted by the University of Michigan.

Port	# IP addresses	Equation tool	Wrongly clustered	Other IPs in cluster
22	20	Dst session key	0	0
22	20	Unicorn	0	4
22	20	Dst IP	0	0
22	20	Dst IP & ports	0	14

Table 7.5: Artificial scans added in the data for validation. Results are also in this table.

Note that we cannot analyse this scanning group using XOR-analysis, as ZMap is not detected by this analysis method. Therefore, the clustering algorithm has the task to identify a ZMap scanner that scans a particular port. As the ZMap scanners will all have the same behavior, the only way to detect a group within the scanners on a certain IP is to use time-based analysis.

The results of the clustering of the University of Michigan is shown in Table 7.4. From the Table we see that there is a very small number of misclassifications. In the case of the University of Michigan, the method of [45] works better, in the sense that all IP addresses in the subnet will be clustered. When randomizing the IP addresses however, so that the scanner resembles a distributed scanner, the subnet method will find no group at all. We are in that case still able to detect this group with the same accuracy as seen in the Table.

To formally validate the results, we compute the precision and the recall of our proposed clustering method. Precision is the percentage of relevant items that are in the cluster, whereas recall is the amount of relevant items that are selected. In laymans terms, precision is the percentage of clustered IP addresses that are actually part of the scanning group, whereas recall is the percentage of clustered IP addresses that are part of the group in terms of the total clustered items. In terms of precision, the clustering method obtains $\frac{206}{208} = 0.99$. In terms of recall, the clustering method scores $\frac{206}{206+37} = 0.85$. This means that we are very precise in clustering the data, but we do have some noise inside the clusters. Remember from our clustering method that we wanted to have clustering in which the integrity of the data was preserved, which could be at the cost of some noise (due to overlap) in the clusters. Therefore, this result is the result that we were looking for.

The University of Michigan scans the internet using the ZMap scanner, and we were able to successful cluster their scans even though there are many scanners using the same tool. If this is the case for ZMap, we argue that this is also the case for other tools that are used. To validate this, we have added four artificial scans to the data, in which a sequence number relation is present that is the same as for the tools found in Table 7.1. The added scans and the results have been listed in Table 7.5. From the Table it can be seen that the results are comparable to the results obtained from the University of Michigan scanning campaign. This means that if we are able to identify a tool used by different scanning groups, we are actually able to take apart a portion of the groups that are using this particular tool. By doing so, we identify scanning groups in the data that we had never seen before.

From our analysis, we have identified IP ranges operated by Shadowserver, which only scan port 80. They do so with 95 hosts, which were all clustered in the same cluster without additional IP addresses. Interesting to note about Shadowserver is that they use what appears to be ZMap, as the IP identification number is 54321, but the sequence number is a constant, which means that they might have changed ZMap.

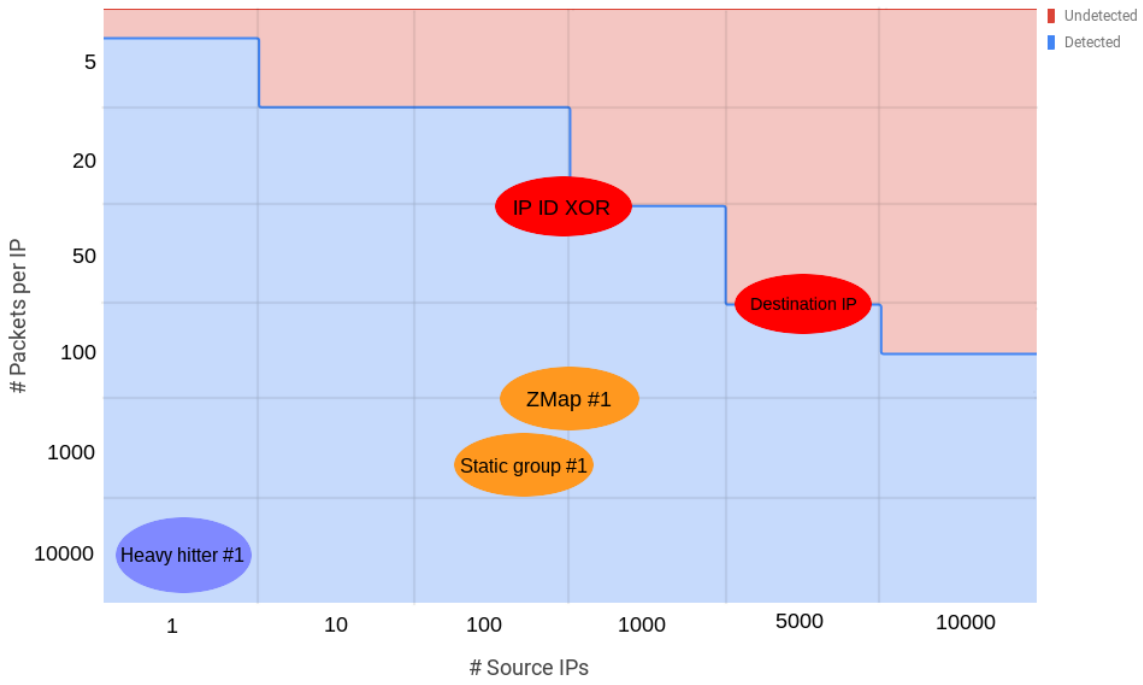


Figure 7.2: 2D visualization of the detection boundary of our detection method, the blue area is what we can detect, the red area is what we cannot detect. This is a representation, as the actual boundary is multi-dimensional.

7.4. Discussion

In the previous chapters, a method for detecting distributed scanners is proposed. In this Section we will first look at what we expect to still be in the data. Secondly, we will explain which indicators we have so far generated from the data.

7.4.1. Undetected groups

As we have seen from Section 6.1, there are many techniques that an attacker can use to hide scanning behavior. We have also seen in Section 6.5, that there is a limit to what our method detects. With the results obtained from this chapter, we can identify if there might be groups that are so hidden that they remain undetected, and how sophisticated groups were that we have detected. Note that we have multiple dimensions which an attacker can use to hide, and when we can detect the attacker in one dimension, we can detect them in all dimensions. In the method validation step in Section 6.5, we were able to see at which level of hiding our method stopped being able to detect groups. While this detection 'boundary' is multi-dimensional, we show what this looks like in 2D in Figure 7.2. Note that the figure is 2D and therefore only a representation of the real boundary, we have chosen the number of source IP addresses per group and number of packets per IP address, as this combination shows that some of the groups found are actually on the boundary of our detection capabilities.

The circles in the figure denote different groups that have been found in the real data. From these points we can see that we have detected groups that are at the boundary of our detection capability, these groups are represented with red circles. Circles in blue are very unsophisticated scanners, as their level of hiding is really low. Orange circles are deemed moderately sophisticated. While this is only a 2D representation, it reflects the data in multiple dimensions, as we have found groups that were at the boundary of what we are able to detect.

Because we detect groups at the limit of our method, we think that there are groups in the data that we do not detect. Most groups that we detect at our limit would have remained undetected if they took one step further in one hiding dimension, which might be what certain groups have already done. While we cannot make a sensible quantification of how many groups we have missed, we are fairly certain that we are not yet able to detect every group.

7.4.2. Indicators of Compromise

Finding slow and distributed scanner groups gives the knowledge that these groups are out there and interested in various protocols. What we want to do to make this intelligence actionable is shifting from knowing that the groups are scanning the internet to indicators of compromise against these groups. This Section discusses three ways of creating these indicators of compromise. These indicators will allow us to diverge from intrusion detection of scanners, and instead go in the direction of intrusion prevention.

IP address indicators

The result of the detection method are clusters of IP addresses that elicit the same behavior and are contributed to scanning groups. IP addresses in itself can be used as Indicators Of Compromise (IOCs), as they belong to an adversary that is trying to scan the network. Chances are that if the telescope catches this scanning behavior, other parties do as well. If the other parties have access to the list of scanning IP addresses, they would be able to drop the scanning traffic in order to ensure that the scanner would not get any information.

Thinking back to the pyramid of pain (Figure 1.2), changing IP addresses is considered 'easy' for the attacker. Obtaining new IP addresses from which the scan is conducted can be done by either registering a new IP or infecting more devices. Therefore, blocking on IP level will probably not stop knowledgeable adversaries. Also, it might be possible that the telescope has only captured part of the IP addresses that the scanning group is using. Other parts of the internet might be scanned using different IP addresses.

So while IP addresses are widely used as an IOC to stop attacks, it might not be the right fit to use in this particular case. Also, it is desirable to go higher up in the pyramid of pain, as this means that it becomes harder to mitigate the defenses for the adversary.

XOR fingerprint indicators

As stated in Section 6.3, fingerprints can be made of scanners in various ways. The most exploited way to create these fingerprints in this work is the XOR analysis. But other than the XOR analysis, fingerprints are extracted based on behavioral characteristics given by the different fields available in the packets (IP identification number, Source ports, Destination ports, etc..). These fingerprints identify the tools used to scan the internet, as different combinations of the packet fields are used by different tools. These fingerprints can be shared as IOCs to mitigate scanning in a much broader way.

From the pyramid of pain (Figure 1.2), it is clear that changing used tools is 'challenging' for the adversary. The adversary needs the knowledge to do this and the time to constantly adapt the tool. Therefore, detecting scanners on the 'Tools' level will improve defenses against scanners and therefore will decrease the information that an attacker is able to gain of a system.

Another upside of the fingerprinting IOC is that it is independent of the used IP by the adversary. This will solve any 'blind spots' in the telescope where IP addresses are not used to scan the telescope range. When another party is now using the IOC, the IP address with which the scanner is scanning does not matter.

Cluster indicators

Given the result of the clustering method, we can create a fingerprint that fingerprints the values of the fields. If there is a packet that contains exactly this combination of fields, the packet can be classified to be in a cluster. For clusters that have a low correlation score, more packets might be needed in order to make sure that the packets are actually in the cluster, but this can be done by waiting on more packets originating from the IP address. In a live environment, the packet could be stopped, in which case a re-transmission would occur from the source, after which we can identify from the two sent packets whether or not this packet belongs to a group.

This method of fingerprinting has the same upsides as the XOR fingerprinting, in that it is also not on IP address level. Therefore, this method can be useful in detecting scanners agnostic of their IP address.

7.4.3. Ethical considerations

Fingerprinting the scanners via the information contained in the packet can prove effective. However, there is a debate about "deep packet inspection", whether or not this is ethical. The proposed method does however not look into the contents of a packet, but only at the fields of the packet in the header. Also, nothing is dependent on the length of the packet, which ensures that nothing would be learned from the packet data. Therefore, we argue that this approach can be considered ethical as the packets only have to be inspected on values that should be randomized anyway. Also, when the handshake is conducted between two IP addresses,

no further inspection has to be done on that connection, as scanning traffic will not complete the entire handshake, and the initial packet would have been detected from the fingerprint before.

8

Conclusion & Future Work

Organizations, governments and everyone else who is using the cyberspace is currently a possible target for attacks by malicious actors. Current technologies are trying to reduce this risk by stopping these attacks on various levels, but the limitation in the current technology is however that it does not allow detection of advanced attacks during the very first stage of the attack, which is the actual gathering of data about vulnerable systems.

Very unsophisticated scanners are detected by current technologies, based on the number of packets that are sent by the scanner. More sophisticated scanners however, are less prone to detection, as they actively limit the amount of packet that a host sends. These scanners use a distributed set of IP addresses and devices to scan other devices for targeted services and vulnerabilities so that every system only sends a few probes, by combining the information of all the scanning systems, an adversary can learn the total attack surface without being detected. The aim of this project was to devise a method with which distributed slow scanners can be found, and evaluate if there are indeed groups that are using these evasion techniques. The research questions that were derived from this problem are:

RQ1. *How can telescope data be processed in such a way that slow distributed scanning groups can be detected?*

RQ2. *Which indicators can be extracted from the telescope data that can be used to identify and mitigate scanners?*

In this work, we have created a method that is able to identify slow scanners using only data collected by a network telescope. This data was obtained from the network telescope at the TU Delft, which in total spans around 100.000 IP addresses. Clustering these slow scanners is difficult, as slow scanners create very weak signals in a dataset in which they strongly correlate with unrelated points.

To detect distributed slow scanners in the data, we have proposed XOR pattern analysis, which is a new method of distributed scanner linking. It relies on tools that use XOR patterns in their sequence numbers, for which it is known to happens in some open source tools, and from this work it follows that it is also present in previously unknown tools.

Next to the XOR-analysis to detect tools, an efficient graph clustering algorithm has been proposed that allows clustering of groups with different correlations. The proposed algorithm uses a high density clustering algorithm called Speaker Listener Label propagation in a number of rounds. First, clusters with high correlation are detected and removed from the data so clusters with lower correlations can be detected. It does so by using different fields present in the TCP/IP protocol, of which was shown that they contain identifying information, which was not done by similar research.

Techniques for a scanner to hide itself have been presented, and proposed detection methods were evaluated against the state of the art in detecting these various levels of stealthiness. The method proposed in this work was found to be a significant improvement on the state of the art, but not yet perfect, as some levels of stealthiness still remain undetected.

Given the detection methods that are proposed, eight new tools have been identified, and 1249 clusters were

extracted. From these 1249 clusters, we have manually verified 50, in which we found 43 distinct groups. To our knowledge, most of these tools and groups have not been previously found. When found, we can fingerprint these groups with the relation that is present in the sequence number, or the information on which the group was clustered. This allows for real time scanner detection and mitigation.

For some of the groups, more thorough analysis has been provided. In this analysis, we have seen clear coordination patterns of scanners that are scanning at a very slow rate.

Coming back to the research questions: *How can telescope data be processed in such a way that slow distributed scanning groups can be detected?* In this work, we have devised a method that enables us to detect slow scanning distributed scanners using information from the packet headers. Using the clustering approach from this work, the dataset that needs to be analyzed will become smaller without discarding useful data, which makes further analysis feasible. Clustering is done by looking at fields in the packet header that attribute to specific behaviors, which improves on previous research by adding more behavioral analysis in the detection. Also, this method is agnostic of whether the source IP addresses are highly distributed or close together. Various previously proposed methods did not allow for this and were therefore unsuitable to find these kind of scanners. Therefore, we conclude that we have provided a way to process telescope data in an efficient manner that enables us to detect slow and distributed scanning groups.

Which indicators can be extracted from the telescope data that can be used to identify and mitigate scanners?. We were able to obtain Indicators of Compromise that not only consist of IP addresses, but identify specific tools. Fingerprints can be made based on XOR relations and header values used by specific scanners, and can therefore be used to identify and mitigate said scanners. By doing so, we will shift from detecting scanners, towards stopping scanning on the internet. This means that we will be able to create a more pro-active defense.

8.1. Main contributions

In this Section, the main contributions from this work will be listed, which are the key takeaways from this work.

- First of all, a clustering method has been created that is able to cluster distributed groups in a semi-real time manner. After clustering, this allows for more feasible computations in order to find very slow scanners. The proposed method improves on other works by not actively splitting the data based on subnet. Many distributed scans are conducted using IP addresses that are highly distributed, which would not be detected if only IP addresses in the same subnet are considered. Thorough analysis has been done in order to verify if the proposed method is better than the state of the art, which indicated that we significantly improve on the current state of the art in scanning detection.

- Secondly, the presence of very sophisticated scanner groups has been detected that were, to our knowledge, not currently known. Previous methods were unable to find these scanning groups. While the results might not show all the sophisticated scanners in the data, it does show that there are adversaries out there that are collecting information and are very capable. This shows that there is a risk and calls for more and better mitigation methods.

- A method for obtaining Indicators of Compromise for scanning groups has been created, which targets specific tools. This is challenging for the attacker to change according to the Pyramid of Pain (Figure 1.2). Previous methods would only result in IP addresses as indicators of compromise which would lead to 'blind spots' when used in other networks that are scanned with a different IP range.

- Finally, an analysis method has been created and implemented that finds groups using peculiar XOR patterns in packets. When found, this method allows for easy fingerprinting of a group, and can be used to mitigate scanning behavior.

8.2. Future Work

While we consider this work a step in the right direction, more work can be done to advance the information that this intelligence provides. This section will discuss the future work that can be done on top of this research, as well as parts of the research that might be improved by future work. Of course there are other

directions that research can take from here, but we identify the following directions as the main topics from which this research will benefit and will help develop the technology further.

8.2.1. Attack attribution

In this work, a method has been proposed to detect distributed and stealthy scans on the internet. Currently, there is no way of mapping these distributed scans to actual attacks. Given the overlap of all sorts of scans, practically anyone could have attacked a target. In cyber threat intelligence, the goal is to know as much from the attacker as possible, so attack attribution would be a very good addition to this work.

8.2.2. More discriminating rules

While distributed and stealthy groups have been detected, there is no guarantee that all of them have been detected. Some groups might have slipped through the detection. When more discriminating factors have been found that can distinguish scanners, it would be a great addition. When ground truth datasets can be provided, AI techniques might start to be useful for classification of the data in a changing threat landscape.

8.2.3. Threat intelligence sharing

In order for this work to apply to industry and companies, which might not have access to a darknet, this intelligence should be transferable. There are bodies in the industy that provide threat intelligence feeds which can be bought to gain intelligence. Whether or not this is the way that the industry will use cyber threat intelligence, the Indicators of Compromise that are obtained from identifying scanners should be transferable through this, or other ways. In order to do so, research has to be done to find a way in which this data can be shared securely, and in a private manner.

8.2.4. Visualization

As stated before in this thesis, visualization of telescope data is lacking. The current state of the are is defined by Inetvis [50], and lacks the ability to clearly show patterns emerging from the data. From section 5.2 we have seen that there are a lot of different patterns that can be extracted using visualization of the data. More research in this area can improve the way that analysts are working and their ability to detect threats. These visualizations can also be used in order to further develop the detection criteria in this method.

8.2.5. Implementation mistakes

In section 6.6.2, it is explained that sometimes, mistakes are made in creating the scanners. Currently, these mistakes are only found and verified by hand, which requires a lot of manual labor. An automated way of finding mistakes could be created that can be used to fingerprint scanners when implementation mistakes happen. Also, human analysts can be supported by new techniques that allow for better analysis of the data, be it in the form of visualization or new algorithms that provide better insights.

8.2.6. Analysis of different behaviors

While the proposed methods currently detect a large variety of groups, this might not always be the case. The threat landscape is ever changing, and this might result in scanners that evade detection. Therefore, more behavioral analysis of scanners should be done in order to further reduce the strategies a scanner can use without risking detection.

8.2.7. Different scanning methods

This research has only analyzed TCP SYN packets in order to obtain a dataset that mostly contains scanners. However, there are more scanning methods on TCP that attackers can use in order to obtain data about potential victims. The proposed method is considered as universal on all TCP scans, but research has to be done in order to analyze if this is truly the case.

8.2.8. UDP

Only TCP has been analyzed in this thesis, and much of the detection relies on the TCP sequence number. This number is not present in the UDP protocol, which means that the XOR method will not work. However, the clustering and timing based approach will work on this data. Still, other methods should be created in order to identify slow and distributed scanners in UDP traffic.

Final thoughts

Given the contributions that have been made in this thesis, we believe that this research actively advances the current state of the art in scanner detection. Proposed methods provide novel ways in which scanners can be detected and classified, which provides useful intelligence that can be used to mitigate attacks before they even happen. Using this method, slow and distributed scanning groups have been found, which provides intelligence about actors and what the actors are after.

Although it is an advancement on the state of the art, we believe that there are scanning groups that still remain hidden. Future work is needed to uncover these actors, which will provide useful intelligence and maybe even mitigation methods for these very advanced adversaries.

In order to create a more secure cyberspace, in which we minimize attacks, we need to further look in to how attackers gather information. If we deprive these actors from their information sources, attacks cannot be as sophisticated and dangerous as they are today. In this thesis, we have taken a step towards depriving attackers of scanning information, but other forms of reconnaissance should also be looked at. We cannot secure our systems and information if we openly show to attackers where they can attack, and this is what we need to work on.

Bibliography

[1] Reaching the last /8 block. Retrieved from: https://www.ripe.net/publications/ipv6-info-centre/about-ipv6/ipv4-exhaustion/reaching-the-last-8, at 04-07-2018.

[2] Teredo tunneling service. Retrieved from: https://docs.microsoft.com/en-us/windows/desktop/Teredo/teredo-addresses, at 07-07-2018.

[3] Dave bianco blog. Retrieved from: http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html, at 03-07-2018.

[4] Number of iot devices on the internet. Retrieved from: https://www.gartner.com/newsroom/id/3165317, at 02-07-2018.

[5] Internet assigned numbers authority. Retrieved from: https://www.iana.org/, at 25-06-2018.

[6] Definition of the internet. Retrieved from: https://dictionary.cambridge.org/dictionary/english/Internet, at 14-06-2018.

[7] Mirai botnet source code. Retrieved from: https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/, at 02-07-2018.

[8] Finding very damaging needles in very large haystacks. Retrieved from: http://cs.ucsb.edu/ kemm/MURI/Presentations/paxson.pdf.

[9] Rfc 2663 - network address translator (nat), . Retrieved from: https://tools.ietf.org/html/rfc2663, at 02-07-2018.

[10] Rfc 971 - internet protocol, . Retrieved from: https://tools.ietf.org/html/rfc791, at 18-06-2018.

[11] Rfc 972 - internet protocol, . Retrieved from: https://tools.ietf.org/html/rfc792, at 18-06-2018.

[12] Rfc 973 - transmission control protocol, . Retrieved from: https://tools.ietf.org/html/rfc793, at 18-06-2018.

[13] Port allocations. Retrieved from:https://www.speedguide.net/, at 14-08-2018.

[14] Ssh port allocation. Retrieved from: https://www.speedguide.net/port.php?port=22, at 09-07-2018.

[15] Chapter 10 - edonkey and emule. In Paul L. Piccard, Brian Baskin, Craig Edwards, George Spillman, and Marcus H. Sachs, editors, *Securing Im and P2P Applications for the Enterprise*, pages 267 – 283. Syngress, Burlington, 2005. ISBN 978-1-59749-017-7. doi: https://doi.org/10.1016/B978-159749017-7/50015-X. URL `http://www.sciencedirect.com/science/article/pii/B978159749017750015X`.

[16] Steven M Bellovin. Frank miller: Inventor of the one-time pad. *Cryptologia*, 35(3):203–222, 2011.

[17] Monowar H Bhuyan, Dhruba Kr Bhattacharyya, and Jugal K Kalita. Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581, 2011.

[18] Norbert Blenn, Vincent Ghiëtte, and Christian Doerr. Quantifying the spectrum of denial-of-service attacks through internet backscatter. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, page 21. ACM, 2017.

[19] Paxson V Bro. A system for detecting network intruders in real-time. In *Proc. 7th USENIX Security Symposium*, 1998.

[20] Gerald Combs. Tshark—dump and analyze network traffic. *Wireshark*, 2012.

[21] Gerald Combs et al. Wireshark-network protocol analyzer. *Version 0.99*, 5, 2008.

[22] Alberto Dainotti, Alistair King, Kimberly Claffy, Ferdinando Papale, and Antonio Pescapé. Analysis of a/0 stealth scan from a botnet. *IEEE/ACM Transactions on Networking (TON)*, 23(2):341–354, 2015.

[23] Marco de Vivo, Eddy Carrasco, Germinal Isern, and Gabriela O. de Vivo. A review of port scanning techniques. *SIGCOMM Comput. Commun. Rev.*, 29(2):41–48, April 1999. ISSN 0146-4833. doi: 10.1145/505733.505737. URL `http://doi.acm.org/10.1145/505733.505737`.

[24] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *USENIX Security Symposium*, volume 8, pages 47–53, 2013.

[25] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An internet-wide view of internet-wide scanning. In *USENIX Security Symposium*, pages 65–78, 2014.

[26] Wassim El-Hajj, Fadi Aloul, Zouheir Trabelsi, and Nazar Zaki. On detecting port scanning using fuzzy based intrusion detection system. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*, pages 105–110. IEEE, 2008.

[27] Claude Fachkha and Mourad Debbabi. Darknet as a source of cyber intelligence: Survey, taxonomy, and characterization. *IEEE Communications Surveys and Tutorials*, 18(2):1197–1227, 2016.

[28] Mark Fullmer and Steve Romig. The osu flowtools package and cisco netflow logs. In *Proceedings of the 2000 USENIX LISA Conference*, 2000.

[29] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified np-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[30] Carrie Gates. Co-ordinated port scans: A model, a detector and an evaluation methodology. 2006.

[31] Vincent Ghiëtte and Christian Doerr. How media reports trigger copycats: An analysis of the brewing of the largest packet storm to date. In *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*, pages 8–13. ACM, 2018.

[32] V.D.H. Ghiëtte and C Doerr. Classifying scanners: Mapping their behaviour. *Master thesis TU Delft*. DOI: uuid:8b842a9e-563e-4a44-b46f-becfb6e8af18.

[33] Robert David Graham. Masscan: Mass ip port scanner. *URL: https://github.com/robertdavidgraham/masscan*, 2014.

[34] Steffen Haas and Mathias Fischer. Gac: graph-based alert correlation for the detection of distributed multi-step attacks. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 979–988. ACM, 2018.

[35] Fu-Hau Hsu, Yan-Ling Hwang, Cheng-Yu Tsai, Wei-Tai Cai, Chia-Hao Lee, and KaiWei Chang. Trap: A three-way handshake server for tcp connection establishment. *Applied Sciences*, 6(11):358, 2016.

[36] Falk Hüffner, Christian Komusiewicz, and Manuel Sorge. Finding highly connected subgraphs. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 254–265. Springer, 2015.

[37] David S. Johnson. The np-completeness column: An ongoing guide. *J. algorithms*, 3(2):182–195, 1982.

[38] Jaeyeon Jung, Vern Paxson, Arthur W Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 211–225. IEEE, 2004.

[39] Shijin Kong, Tao He, Xiaoxin Shao, Changqing An, and Xing Li. Scalable double filter structure for port scan detection. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 5, pages 2177–2182. IEEE, 2006.

[40] Dai-ping Liu, Ming-wei Zhang, and Tao Li. Network traffic analysis using refined bayesian reasoning to detect flooding and port scan attacks. In *Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on*, pages 1000–1004. IEEE, 2008.

[41] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[42] Dorina Marghescu et al. Evaluating multidimensional visualization techniques in data mining tasks. 2008.

[43] Lockheed Martin. Cyber kill chain®. *URL: http://cyber. lockheedmartin. com/hubfs/Gaining_the_Advantage_Cyber_Kill _Chain. pdf*, 2014.

[44] David Myers, Ernest Foo, and Kenneth Radke. Internet-wide scanning taxonomy and framework. In *Proceedings of Australasian Information Security Conference (ACSW-AISC), 27-30 January 2015*. Australian Computer Society, Inc, 2015.

[45] Seth Robertson, Eric V Siegel, Matthew Miller, and Salvatore J Stolfo. Surveillance detection in high bandwidth environments. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 130–138. IEEE, 2003.

[46] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999.

[47] M Zubair Shafiq, Muddassar Farooq, and Syed Ali Khayam. A comparative study of fuzzy inference systems, neural networks and adaptive neuro fuzzy inference systems for portscan detection. In *Workshops on Applications of Evolutionary Computation*, pages 52–61. Springer, 2008.

[48] Rong-sheng Shan, D Li Xiao-yong, and Jian-hua Li. An adaptive algorithm to detect port scans. *Journal of Shanghai University (English Edition)*, 8(3):328–332, 2004.

[49] Andrew S Tanenbaum et al. Computer networks, 4-th edition. *ed: Prentice Hall*, 2003.

[50] Jean-Pierre van Riel and Barry Irwin. Inetvis, a visual tool for network telescope traffic analysis. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 85–89. ACM, 2006.

[51] Jierui Xie, Boleslaw K Szymanski, and Xiaoming Liu. Slpa: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 344–349. IEEE, 2011.

[52] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions: global characteristics and prevalence. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):138–147, 2003.

[53] Yu Zhang and Binxing Fang. A novel approach to scan detection on the backbone. In *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*, pages 16–21. IEEE, 2009.