Scanner Clustering

How can clustering techniques be applied to classify and identify slow scanners based on their behavior and attributes?

Scanner Clustering
Mick Koertshuis

Delft University of Technology

TU Delft

Scanner Clustering

How can clustering techniques be applied to classify and identify slow scanners based on their behavior and attributes?

by

Mick Koertshuis

Student Name	Student Number
Mick Koertshuis	4355059

Instructor:	Dr. ir. H. Griffioen
Project Duration:	October, 2023 - July, 2024
Faculty:	Cybersecurity, Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

In this research, we propose a new method for detecting slow, distributed port scanners by utilizing clustering techniques based on the behavioral characteristics of scan sources. Traditional methods often rely on identifying sources within the same subnet and using frequency-based algorithms; however, our method operates independently of source address correlations. We assign numeric features based on these characteristics to packet sources observed through our network telescope. These features are then processed using a clustering algorithm to identify distributed scans without depending on the origin of source addresses.

Our findings demonstrate the effectiveness of this method in accurately clustering and identifying slow scanners. We verified the results by comparing the detected scan clusters against the expected behavior derived from our simulations. This approach not only reveals complete distributed scans but also offers insight into different scanning strategies employed across the internet.

The implications of our research are significant for network security, enabling early detection and mitigation of potential cyber threats. Our methodology lays a foundation for further research and optimization, offering a valuable tool for both understanding and securing internet infrastructure against scanning techniques.

Contents

1

Introduction

Every time you visit a website or use an online service, you connect to a host. Most websites openly advertise their hosting servers online. Accessing a website's URL allows for a relatively straightforward connection to its host. This accessibility extends to various other services such as Secure Shell (SSH), File Transfer Protocol (FTP) and Telnet. When you connect to an online service, your computer sends a connection request to a host. Ideally, the host responds with the requested information. These exchanges between users and hosts are described as packets, the exchange of these packets is what we refer to as internet traffic. Typically, this traffic is solicited, as hosts have advertised their presence to users. However, an examination of internet traffic reveals that this is not always the case. Unwanted traffic, even directed toward destinations lacking active hosts, is what we call background radiation. Background radiation is comprised of the following:

- The spread of network worms (malware replicating across the internet)
- Denial-of-service attacks
- Backscatter, which involves the transmission of packets with spoofed (fake) IP addresses
- Address space pollution, characterized by faulty requests (e.g., typographical errors like typing "169.0.0.1" instead of "196.0.0.1")
- Third-party network configurations, arising from outdated setups or haphazardly adopted settings from tutorials or obsolete documentation

And, of course, non-targeted port scans of the global network contribute to this radiation [36]. A port scan is a process where a client probes a port on a host server to see if ports are open. Where, a port serves as an endpoint to a specific service on a host. If a port is open, the associated service is accessible to the internet. This process is useful when you want to access a service from an external location, such as accessing a service at home from work. If a port is closed, the service will not be accessible to the internet, and you would need to connect locally (e.g., at home). When you open a port, it doesn't actively advertise its openness online, but it can still be probed by others to check its status. For this reason, network administrators often check their ports to ensure that only desired services are open to the internet. Administrators conduct scans on their network for this purpose. However, attackers may also scan networks to check for open ports. By identifying open ports and associated services, attackers can discover vulnerabilities and potentially launch attacks against the host. As a result, port scans constitute a significant portion of internet traffic. Since many people are scanning, it's crucial to secure your services. This is mainly done by keeping them updated. But it's also important to spot scans before they turn into attacks. You might even be able to identify attackers by the scans they've done before. So, by spotting scanning attempts early, you can beef up your defenses and stop potential threats in their tracks. Understanding scanning patterns can also help figure out who the attackers are and how they operate. In short, spotting scans isn't just about protecting your systems, it's also about stopping cyber threats before they become serious problems. Scanning is typically conducted on a large scale, either through partial or full internet-wide scans [24]. In the past, scanning activities were

predominantly carried out by botnets. However, the emergence of tools like NMap, MASSCAN, and ZMap has made scanning much easier, faster, and more accessible [15]. As a result, the frequency of scans and the number of scanners have also increased significantly.

Detecting scans requires a large number of hosts to monitor network activity effectively. This allows for the identification of whether an adversary is targeting a substantial network or even the entire IPv4 address space. This large-scale monitoring network is commonly referred to as a network telescope. A network telescope captures all packets sent to any of its hosts. Typically, this is achieved using tools like TCPDUMP. In the case of our network telescope, it isn't associated with any active hosts, meaning there's no solicited traffic. As a result, all recorded packets are considered background radiation. Using a network telescope enables the detection of both full internet-wide scans and partial internet-wide scans if they target our network. In this paper, our focus will be on internet-wide scanners.

An internet telescope is essentially a large-scale monitoring network comprised of numerous hosts. Its purpose is to observe and record internet traffic, particularly unsolicited packets that are indicative of malicious activities such as scanning attempts or denial-of-service attacks. By analyzing the volume of incoming packets, internet telescopes can provide valuable insights into cyber threats and the current state of the internet. Our telescope uses TCPDUMP to record all the packets.

If we only look at scanners we see some distinctions between different methods. The first distinction is heavy hitters. Heavy hitters are scanners that try to scan the network in a short amount of time by sending a lot of packets. These are easy to set up as this feature comes with most toolchains. Heavy hitters are also quite easy to detect. A simple way to detect them is with some sort of a leaky bucket algorithm. Which is an algorithm that is based on the analogy that a bucket that constantly leaks will overflow if the intake is more than the amount that is leaking. If a source is sending packets to a network a leaky bucket is created and each packet sent by the source will be added to the bucket. Every couple of seconds some packets will be deleted out of the bucket. If the number of packets in the bucket comes above a set threshold the source of the packets is blocked, usually for a certain amount of time. This detection algorithm is fairly easy to set up and it is effective against most heavy hitters and denial-of-service attacks. That is because it is most effective for limiting bandwidth and countering bursts of packets. Because the buckets are always leaking this algorithm will also automatically free up space. While this is not an issue for denial-of-service attacks it can be an issue for port scanning. A common technique to circumvent this algorithm is by slowing down the rate you send packets to a network. If an adversary has a single host available then it would need to slow down the rate of its source drastically. But an easier method would be to divide their scan over multiple hosts. Each host will create a different leaky bucket on a network that is being monitored and therefore divide the flow each bucket is receiving by the number of hosts the adversary uses. This way the adversary will have a lower chance to be detected but will still get their information in the same amount of time. Scanners who use these techniques are what we call slow scanners. In this paper, we will focus on these slow scanners and provide a method on how to cluster and identify them.

There has been done some research about detecting distributed slow scanners, but most of these techniques rely on the distributed scan to be in the same \24 (or class C) subnet [22, 24, 8, 25]. In this paper, we propose a method that works independently of the correlation of the addresses of the sources in a distributed scan. The basics of our method rely on characteristics that distributed scans show. For an adversary, it is straightforward to make a script to launch its scan from multiple hosts with the same settings. This causes sources to start being active around the same time and will have the same scan frequency. We can then use clustering techniques to cluster sources with the same characteristics together. While we will not use the source addresses in our clustering models, these addresses can still be used. Mainly for validating our method because the results of previous research still hold [22, 24, 8, 25]. Our method needs to detect both scans originating from the same subnets as others originating from different subnets. This is also needed because some adversaries use hosting providers to distribute their scans and even hosting providers have multiple subnets available for their servers. Clustering algorithms have been used in cyber security but not necessarily for detecting slow scanners. It has however been used to identify sources as malicious [33], but sources are tagged independently. Clustering has also been used to identify background radiation in general [17]. But this is used to classify each source and not cluster them together in scans.

In this paper, we will propose a detection system for slow scanners by assigning features to different

sources based on their characteristics and clustering them to identify other sources that cooperate with their scanning. This is possible because of the settings used for distributed scans, this causes each source in a scan to behave similarly.

First, we will explore the background of scanning in section 2, where we will explain what port scanning is and how it is used. We will also provide a brief history of scanners, discuss scanning in general, and describe the modern characteristics of scanners. Additionally, the background section will delve into different scanning tools, primarily ZMap, and explain the concept of clustering.

Next, we will review previous research and related work in section 3. Following that, we will talk about our dataset, how we acquired it and its limitations, this will be covered in section 4.

We will then explain our methodology in section 5, providing a detailed explanation of the chosen features for the sources and the clustering process. Here, we will also devise a simulation to test our method. Our findings will be analyzed in section 6, followed by a discussion of our strengths and limitations in section 7. Finally, we will concluded our thesis in section 8.

2

Background

In our introduction in section 1, we outlined our goals. However, we only briefly explained some of the concepts needed to fully understand our research. In this section, we will clarify everything you need to know before delving into our methodology.

We will start with an explanation of what ports are and how they are used in transport protocols. This will lead to our explanation of port scanning, how it is performed, and its uses. Then, we will provide a brief history of port scanning, discuss how administrators have responded to different port scanning techniques, and offer some information about scanning detection.

Next, we will present a more modern view of scanning, discussing new tools developed to make scanning easier and more stealthy.

Lastly, we will explain the basics of clustering, which we will use later in our methodology in section 5.

2.1. Ports

If you want to connect to a server you first need the IP address of that server, this IP address functions the same way as geological addresses work. Imagine a server as a warehouse where the IP address will lead you to it. Inside this warehouse are different rooms each with their own function. If a door is open you can enter one of such rooms and see what is inside. In networking this door is called a port. Just like people can enter and exit the room through a door, an internet port allows data to enter and exit a device. A port has a number associated with it to identify the kind of data it has available, just like a warehouse has certain rooms with different kinds of purpose. These doors have numbers assigned to them. Let's look at Transmission Control Protocol (TCP), one of the most common transport protocols. If we browse the web, we typically use ports 80 and 443, associated with HTTP and HTTPS respectively. But if we use our email we usually use ports 25 and 465, SMTP with or without encryption. As the number on a door helps us identify the purpose of the room, the port helps us identify the service that is running on a server. Ports can range from 0 to 65535 and the Internet Assigned Numbers Authority (IANA) is responsible for the registration of commonly used TCP (and UDP) ports. The lower ranges from 0 to 1023 are well-known ports and are hard to get registered. But IANA also maintains an official list from ranges 1024 to 49151. The remainder of these ports are not officially registered but some applications are still known to use some of these ports. Server administrators are free to pick whichever port for any of their services, but using the standard port for the service helps with configuration and compatibility. So administrators typically use the standard ports for their services. This means that if you want to access a service you know which port you need to look for.

2.2. TCP

To control the data that can enter through these ports we need a transport protocol. This data is transported using network packets, which are formatted units of data, basically envelopes with a message inside of them. The two most common transport protocols are Transmission Control Protocol

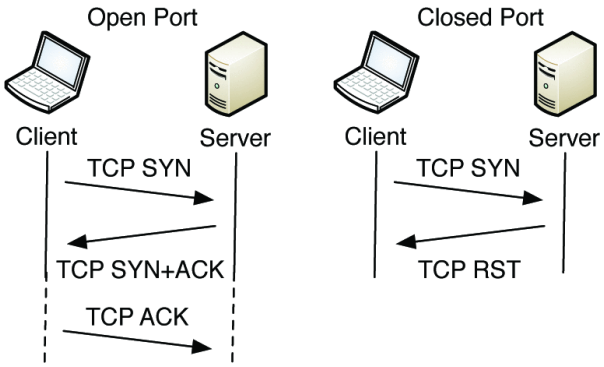

Figure 2.1: Establishing a connection using the three-way handshake [16]

(TCP) and User Datagram Protocol (UDP). For reliable data, we typically use TCP while UDP is used for unreliable time-sensitive data. In this paper, we will focus on TCP because we will look at clustering scans and while UDP scans are possible they are less common and harder to do than TCP scans [10, 26].

We mentioned that TCP is reliable, so let's see how they achieve this. In table 2.1 a TCP segment header is given. This header is pasted in front of every TCP packet and consists of several 32-bit "words". The first 32 bits are used for the source port and the destination port. The destination port is the port of the service we want to access. The source port is an ephemeral port which means that they are dynamic. These are used in communication but in normal cases do not correspond with a service. We can ignore them for now. The next 32 bits are used for the sequence. This is where the reliability of a TCP packet comes in. The sequence number is used for ordering the data in a conversation. The acknowledgment number is used together with the sequence number to confirm or "acknowledge" packets back to the sender. For the remainder of the headers, we will only look at the TCP flags the others are not relevant to this paper. The flags of a TCP header tell the receiver what the sender wants to accomplish with this packet. These are bits that can be set, the ones relevant are:

- ACK: "Acknowledge field is significant" [23]. If it is set, the acknowledgment number will be the sequence number the sender of the segment is expecting to receive.
- RST: "Reset the connection" [23].
- SYN: "Synchronize sequence numbers" [23]. If set, the sequence number will be the initial sequence number.
- FIN: "No more data from the sender" [23].

When a connection with a service needs to be established the "three-way handshake" is used. The process is shown in figure 2.1. When a client wants to establish a connection with a server it will send a SYN packet to a port on the server. If the port is open the server will respond with a SYN+ACK and if the client receives the response it will send an ACK. If the port is closed the server will respond with a RST. If a connection is established packets are free to be sent and the sequence and acknowledgment numbers will be used to put the packets in the correct order.

Bit	0	8	16	24
0	source port		destination port	
32	sequence number			
64	acknowledgment number			
96	offset | reserved	TCP flags	window	
128	checksum		urgent pointer	
160 | 448	Options			

Table 2.1: TCP segment header

2.3. Port scanning

In section 2.2 and figure 2.1 we have seen that it only takes two packets to check if a port is open. This is the basis for port scanning. A port scan is a technique used to discover which ports on a target system are open or closed. The three most common types of TCP port scans are as follows:

1. SYN Scan: The client does not care about actually establishing a connection. If a port is open and the SYN+ACK is received as shown in figure 2.1, the client will not respond with an ACK but with an RST, breaking the connection. The client already knows that the port is open if they receive a SYN+ACK from the server instead of an RST [10]. This is sometimes referred to as a half-open scan. Custom packets need to be made for this so super-user privileges are needed. Which is on most machines not a problem.
2. Full Connect Scan: Establishes a full connection with the three-way handshake. This scan does not use custom packets and can be achieved with the system call CONNECT() and therefore does not need super-user privileges [10].
3. ACK Scan: Does not use the three-way handshake but only sends an ACK to the server. If the server is open it will respond with an RST otherwise the packet just gets dropped.

Usually, a list is kept of which servers have which ports open. If new open ports are discovered this list will get updated. There are two main distinctions between port scanning strategies; vertical and horizontal scans. A vertical scan will scan multiple ports on a single machine and a horizontal scan will scan a single port on multiple machines.

Port scanning is a crucial part of network reconnaissance. It is both used for legitimate and malicious purposes. It is possible to scan your own network or server and check which ports are open. But together with the knowledge of common ports as we discussed in 2.1 it is also possible to scan a lot of servers for a particular service. This often happens, particularly so when a new vulnerability of a particular service is discovered. When this happens lots of scan groups can the entire IPv4 address space of the internet in just a few hours [11]. When a malicious group scans your network and vulnerabilities are found then an attack usually follows [32].

2.4. Brief history

In the earlier years, port scanning was mostly done with botnets. The first big increase in scanners was around 1998 and with the introduction of worms, the next big increase was around 2001. This was because of the Code Red and Nimba worm outbreaks [3]. Scanning was mostly done with heavy hitters, meaning that single sources were scanning a lot of destinations in a short amount of time. Because of common intrusion detection systems introduced that detected these heavy hitters, more slow scanners showed up. Scanners will distribute their scans so the rate per host would decrease together with the total packets sent per host. Currently, we also see in our database that a lot of scanners use hosting providers to distribute their scans over multiple virtual private servers (VPS).

2.5. Scanning detection

There are several intrusion-detection systems available for detecting scanners. As we discussed in section 1, the easiest way is with some sort of frequency based algorithm like a leaking bucket algorithm. This can be done on a single server or a network with the help of a software-defined network (SDN) which is a way of managing a network by centralising control, so packets going to and from servers on the network can be easily monitored by the controller. A commonly used tool for monitoring your server or network is Snort [30], which is an Open Source Intrusion Prevention System (IPS). With Snort you can set up rules to detect half-open SYN scans (see section 2.3). It does this by not only limiting the rate of packets but also identifying different kind of packets. The effectiveness of your parameters to detect certain scans can be validated, while it works great for heavy hitters, it does not detect slow scanners with a high probability [32]. Smarter intrusion detection exists by taking a more proactive stance than Snort does [6, 27]. But they still focus on single sources and not on distributed scans.

2.6. Network telescope

The use of active intrusion detection or prevention systems requires the software to differentiate between attackers and users. Most of the time they keep a blacklist and potential attackers get flagged and banned to use the network. This makes the detection of internet events harder. This is where network telescopes [20] come in. A network telescope is a group of routed IP addresses where little or no legitimate traffic occurs. This happens if IP addresses do not have a host attached to them. This means that when an IP address has no host, all the packets it receives are forwarded to a single machine that functions as the telescope. The telescope usually just monitors and saves the packets with the use of tools like tcpdump. The idea is that if the telescope is big enough (a large amount of IP addresses), it can detect network wide events. In particular large events like Denial-of-Service Attacks, Internet Worms and of course; internet-wide port scans. It remains a problem that some attackers might use a blacklist that lets them ignore known network telescopes [11].

2.7. Scanning Toolchains

There have been different tools throughout the years but the most popular ones are NMap, MASSCAN, Unicorn and ZMap [15]. These are all free and open source. It is therefore easy for adversaries to implement. NMap is the oldest of these tools and is still widely used but lacks the speed of the newer tools [11]. There are also some functionalities missing in NMap that other tools use. For example, ZMap supports distributed scanning called sharded scans. Where you can send an instruction to multiple ZMap instances that work together for their scan. Because of the usage, performance and popularity of ZMap together with their own published papers [11, 12] we will primarily focus on ZMap for our research.

2.8. ZMap

By default, ZMap will always set their IP identifier to 54321 making it easy to detect out of the box. However, because ZMap is open source this feature can easily be changed, even randomized. We also see this on our network telescope. Even so, we still found a lot of packets with the ZMap identifier that we used to design our method. ZMap also uses the TCP handshake to scan for open ports but they will never actually connect with servers, after a SYN+ACK is received ZMap will send an RST automatically closing the connection.

ZMap also has some features that make it interesting to use. With parallel address generation, their blacklist algorithm and zero-copy NIC access, it can send nearly 15 million probes per second, achieving nearly 10Gbps throughput [11]. As mentioned before, with this speed scans can be easily detected. But one way to circumvent this is by distributing your scan activity. Durumeric, Bailey, and Halderman calls this a sharded scan [11]. "Where a shard is a partition of the IPv4 address space that can be iterated over independently from other shards; assigning one shard to each thread allows for independent, mutex-free execution." The sharding focuses on the distribution of destination addresses between all the hosts. This will cause the number of destinations to be nearly equal between the different hosts with no overlap in destination addresses. To use a ZMap sharded scan you will need multiple hosts and send the following example command to them:

```
zmap -p[PORT] [SUBNETS] --shards=N --shard=n --seed=seed
```

Where N is the total number of shards and n is the identifier for the shard where $0 \leq n < N$. Together with the random seed, this causes all the shards to perform their part of the bigger scan without communicating with each other. You then would only have to combine the output of all the ZMap instances.

2.9. Clustering

Because of the settings used in ZMap as described in 2.8 we will look for characteristics that scanners perform so that we can cluster them together. A cluster analysis or clustering is the task of grouping sets together that have similar features. There are many clustering algorithms [34] with different techniques. In our case, we have multidimensional data with a lot of noise, uneven cluster sizes and irregular shapes. This is why we opt for DBSCAN [9, 29]. This is a clustering algorithm that uses the distance between

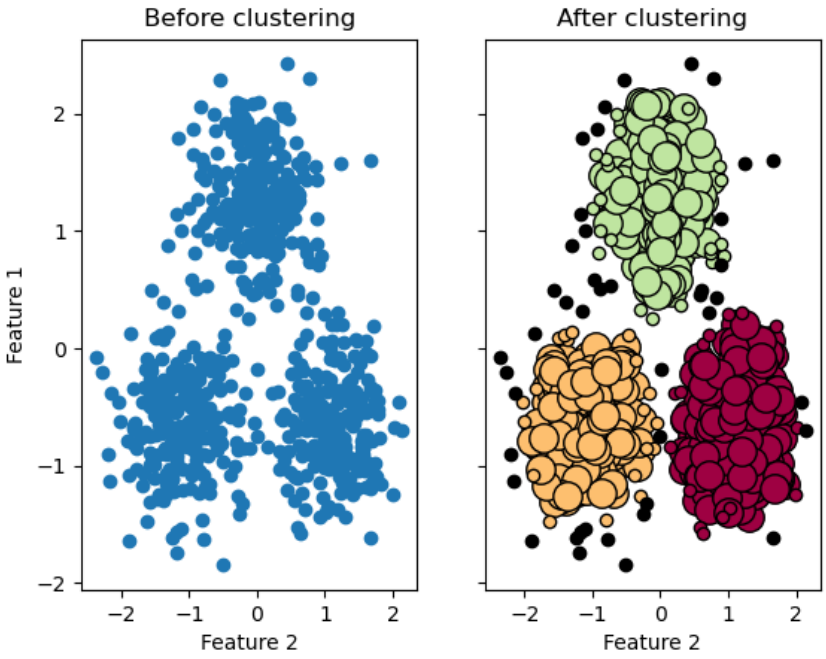

Figure 2.2: Example of DBSCAN to identify an unknown number of clusters.

points and separates areas by low density. An example is shown in figure 2.2. Here we see some random data with two different features. In this figure we see that the algorithm has identified three different clusters. The black spots in the figure is what DBSCAN considers noise.

DBSCAN primarily needs two different parameters, $min\_samples$ and ϵ. An example is shown in figure 2.3. The $min\_samples$ is the minimum number of samples inside each neighborhood and ϵ is the distance of each neighborhood. DBSCAN will look in a distance of ϵ around each point and if there are at least $min\_samples$ around that point the point will be added as a core point for that cluster, which are visualized as the bigger points in figure 2.3. Points that are part of a neighborhood of another point but do not have enough points in their own neighborhood are called edge points, which are the smaller points in figure 2.3. A cluster in DBSCAN consists of core and edge points. Other points are considered noise, which is the black point in figure 2.3.

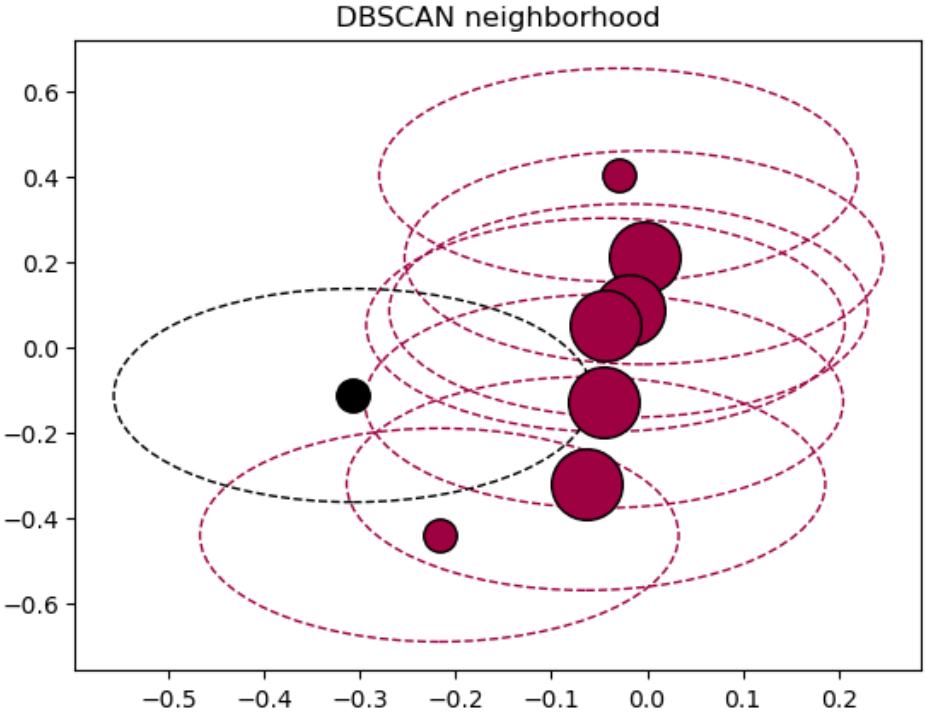

Figure 2.3: Neighborhood selection of DBSCAN. Where $\epsilon = 0.2$ and $min\_samples = 3$

3

Related Work

Zolotykh has done a classification of Internet Background noise [36]. Which is unsolicited network traffic. They define the most popular ones are:

- Non-targeted scan of global network
- Distribution of network worms
- Backscatter from attacks using IP-spoofing
- Address space pollution
- Using of third-party configurations

They acknowledge that there are other less frequent reasons of background noise and that they change depending on the development of the global network, like discoveries of new attacks. They suggest a classification based on header field combinations which allowed to distinct the malicious noise sources from the victims of remote attacks. They also find that IANA port recommendations are followed poorly.

Allman, Paxson, and Terrell have done research on the history of scanning [3]. They acknowledge the diversity of background radiation and the challenge to characterize it. They focus on scanning and examine the scanning activity through time. Their dataset consists of scanning activity beginning in 1994 and up until 2006. Their subsets consists of 2.4 millions distinct IP addresses performing 623 million scans. However, they lack the data on distributed scans.

Most people use a network telescope to detect scanning, which we also use for our research. Moore et al. have written a technical report on network telescopes [20]. They have found that data collection on the internet for internet-wide phenomena is nearly impossible without the use of a network telescope. They make a distinction from previous research on network telescopes that there is no systemetic analyses if a local observed event correctly portrays a global event. The paper does not focus on scans but rather on internet worms and denial-of-service attacks. They state that a larger size of a telescope corresponds to a better accuracy, providing more detail. With enough information it is possible to accurately extrapolate the true start and end times of host activity.

De Vivo et al. has done a review on basic port scanning techniques [10]. They focus on TCP port scanning techniques and give an explanation the three most common:

- Full TCP connection
- Half open scanners
- ACK scanning

They also go in-depth how these scans bypass firewalls analysis and why they work. Roslan has done more research on these different scanning techniques and makes a comparison of the performance of

those techniques [26]. They have found that TCP SYN scanning (or half open scanning) has the lowest response time.

Durumeric, Bailey, and Halderman have done an internet-wide view of internet-wide scanning [11]. They examine the practice of internet-wide scanning and explore the impact of faster tools. This research is done after the introduction of faster and better like ZMap and Masscan. They present a broad view of the scanning landscape around that time and what protocols they target. They find that scanning practices has changed over the years and provide their findings. They acknowledge the rapid responses from adversaries if new vulnerabilities are found and show some of these network events. They also show that the majority of network administrators do not see scanning as a significant threat. They find 80% of non-solicited traffic is coming from horizontal scanning and that the internet landscape is ever evolving. Richter and Berger have also done a research focusing on the behaviour of internet scanners [24]. They acknowledge that previous research lacks in the size of data they have. Primarely due to a small network telescope and not a big enough spread of IP addresses, normally containing a single network or just using university or research networks. They have done research on a telescope containing 89000 hosts over different vantage points. They find that a spread is necessary to view localised scanning activity. Pletinckx and Ghi have also done research on port scanning [22]. They acknowledge that network administrators can get suspicious of scanning activiteits and already mitigate an attack before it happens. This is why adversaries use different techniques to circumvent detection during scanning. Pletinckx and Ghi take these stealthier scanning techniques into account in their research and classifies distributed port scan reconnaissance strategies based on empirical data [22]. They report on non-overlap and overlapping in destinations for scanning techniques and acknowledge the use of different address blocks being used for some distributed scans.

One of the techniques used is what we call slow scanning. And like Pletinckx and Ghi say, it is the response by adversaries on intrusion detection systems. This trend has also been previously researched [22]. Barnett and Irwin have given some insight in scanning techniques and how existing network intrusion detection work and lack to recognise all forms of scanning traffic [4]. Dabbagh et al. have also researched scanning detection techniques and focus their research on slow scanners [8]. They propose an approach for detecting slow scanners on a busy network by following conversations. For our research that focuses on detecting slow scanners on a network telescope this method is not relevant. But this paper gives great insight in the definition of slow scanners and the difference between horizontal en vertical scans. Nisa and Kifayat also did research on detection techniques for slow port scanning [21]. They acknowledge that the main difficulty in detecting slow port scanning attacks is uncertainty. They try to overcome this by categorising single sources as suspicious of malicious when they are active.

The detecting of certain toolchains has also been researched. Ghiëtte, Blenn, and Doerr detects the toolchains by their header [15]. With default configurations, you can detect Nmap, MASSCAN, ZMap and Unicorn by looking at the values in the SrcIP, SrcPort, Seq, session key and DstPort. They also mention that it is possible to look at the concurrent destinations of ZMap scans to identify them, but you need to know the internal state of ZMap. Mazel and Strullu actually expands on this idea [19]. They use a cryptographic approach to detect the primitive root used in the pseudo-random destination cycle generation [11]. This also works for sharded scans. However, they mention that with custom blacklists, jitter and packet losses, it is very hard to make it work.

If we focus on ZMap we can look at the papers written by its creators. Durumeric, Wustrow, and Halderman J have accompanied ZMap with a paper which addresses previous hiccups with older scanning tools and explain the goals of ZMap [12]. They also give insight how ZMap works and what it can accomplish. They also explore the new attack vectors that come with ZMap and how one might consider those when defending their own network. Adrian et al. have also done a follow-up paper about new features of ZMap later released [2]. Where they primarely talk about the optimization. But this research also gives more insight into sharded scanning which is the focus of our research.

Vugrin et al. also have done a more recent study on port scanning detection and focusses on model validation [32]. This research focusses on Nmap but they demonstrate how mathematical modeling and emulation can be used together to validate and develop models.

Griffioen and Doerr have done research more focussed on detecting slow scanners not by detecting hosts out of a gigantic pool of all IP addresses and incoming packets that complement each other to

a plausible degree, but by identifying custom port scan tooling and coordinated scanners based on artefacts containing in the header fields [16].

Furthermore a big part of our research will be about clustering. Clustering network traffic with clustering techniques has been done before. While not necessarily clustering distributed scans together Liu, Li, and Li uses K-mean clustering to categorize network traffic [18]. They also use multiple dimensions for their clustering. They use the following features for their clustering algorithm:

- Start-time
- End-time
- Duration
- Flows

Iglesias and Zseby have used clustering for classifying internet background radiation [17]. Their main contribution is the proposal of the AGM vector format. Which is a numerical vector for network trafic representation. They use the following features for their clustering algorithm:

- Number of destination ip addresses
- Number of source ports
- Number of destination ports
- Number of protocols
- Number of TTL
- Number of flags
- Number of different lengths
- Number of total packets

We have also looked through some papers regarding clustering algorithms to chose our own. Xu and Wunsch have done a survey on clustering algorithms [34]. They provide a comprehensive and systematic description of commonly used clustering algorithms in computer science. They conclude that there is no perfect algorithm to solve every problem, but they give some insight in choosing an algorithm for certain needs. For the use of DBSCAN we can review the accompanied paper by Ester et al.[13]. Where they describe their focus on large databases, arbitrary shapes and minimum requirements of domain knowledge to determine the input parameters. This paper is however a bit old, but a more recent paper by Schubert et al. have determined DBSCAN as still relevant and why it is still a very good algorithm for certain problems [29]. They also give some recommendation on how to choose parameters correctly. Moreover the parameters, Starczewski, Goetzen, and Er has also recently done a paper on DBSCAN where they propose a new method for automatic determining of the parameters [31]. They base this on the detection of sharp distance increases generated by a function which computes distances between each element of a dataset and its k-th nearest neighbor.

As we can see, clustering has been used to classify and categorize in different fields of background radiation. But as far as our knowledge goes, the actual clustering of potential scanner origins to group distributed scanners together has not been researched. The papers that we have found only focus on single source scanners or, if the scan from multiple sources, on centralized scanners that scan from the same subnet. They acknowledge that some distributed scans can be decentralized but there is no research done on how to group these scans together. We have discussed some papers that cluster types of packets together to either classify them as malicious or to group different types of background radiation together, but this means that detection results are not always accurate as it is not possible to know if you detected complete scans. This leaves a gap in research where it is needed to detect distributed and decentralized slow scanners.

4

Dataset

In this section, we delve into our network telescope and dataset, explaining how it operates and its size. Additionally, we address the limitations inherent in our telescope and propose strategies for overcoming these constraints. Finally, we outline the intended utilization of our telescope.

The network telescope of TU Delft consists of three class B (or /16) subnets. Every IP address on the telescope only dumps TCP packets while the addresses are not assigned to a user. This will in theory give us a total of $3 \cdot 2^{16} = 196608$ observers. In reality, we observe only 65321 active observers in June. This would also suggest that these observers are not reliable because they can be used by users on the network. Luckily, the 65321 observers that we see have very little downtime.

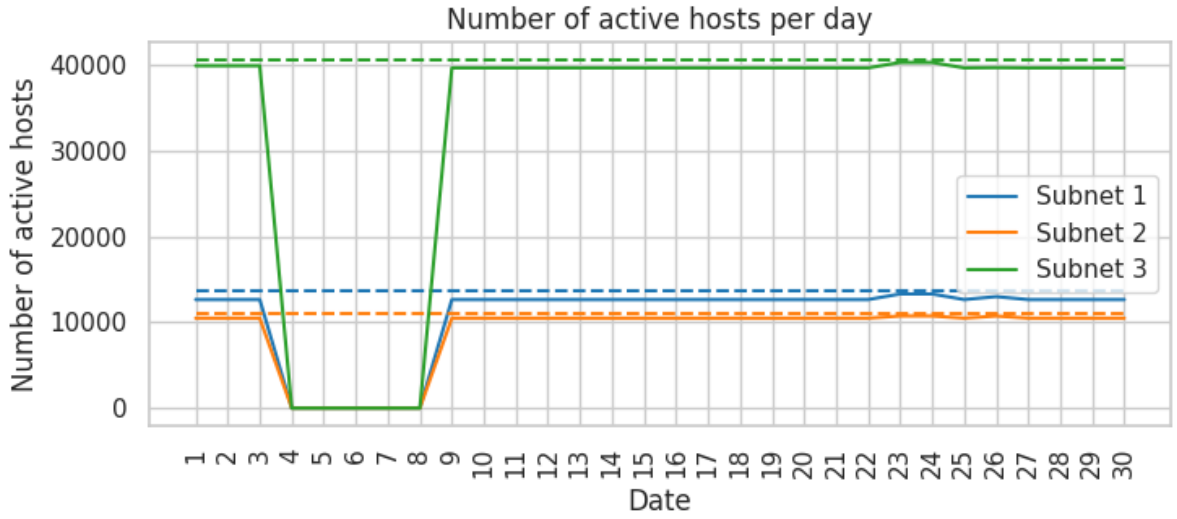

Figure 4.1: The sum of all the active observers per subnet per day. The dotted line represents the sum of unique observers in each subnet over the month.

Our hosts do not keep a log of their uptime. So to check the uptime of the hosts on our telescope ourselves we can use internet background noise. If we see no noise for a long time then one of our hosts is currently not capturing packets. In the setup of our telescope, this happens if an IP address is assigned to a user. In June we have captured $10.2e9$ packets. To check the uptime of every of the 65321 active observers with this amount of packets would take a very long time. To decrease the computation time we only check for certain periods. The period needs to be big enough to confidentially say that the host is down, but small enough to correctly assign features as we will explain in section 5. We also acknowledge a period as active if we find a single packet targeting that source, so we can stop measuring after the first packet to also save on computation time. We choose our sampling period to be a minute.

The results are shown in figure 4.1. Here we plotted the active hosts in June per subnet. The dotted line is the sum of all the unique observers during the whole month, so the theoretical maximum number of active hosts. We see that between the 4th and the 8th of June, the tcpdump was offline. We also see that the 23rd of June has the most activity. To ensure a consistent stream of data it is therefore needed to only take data between the 9th and the 30th. Later when more data is available it is also possible to compare our results to other months.

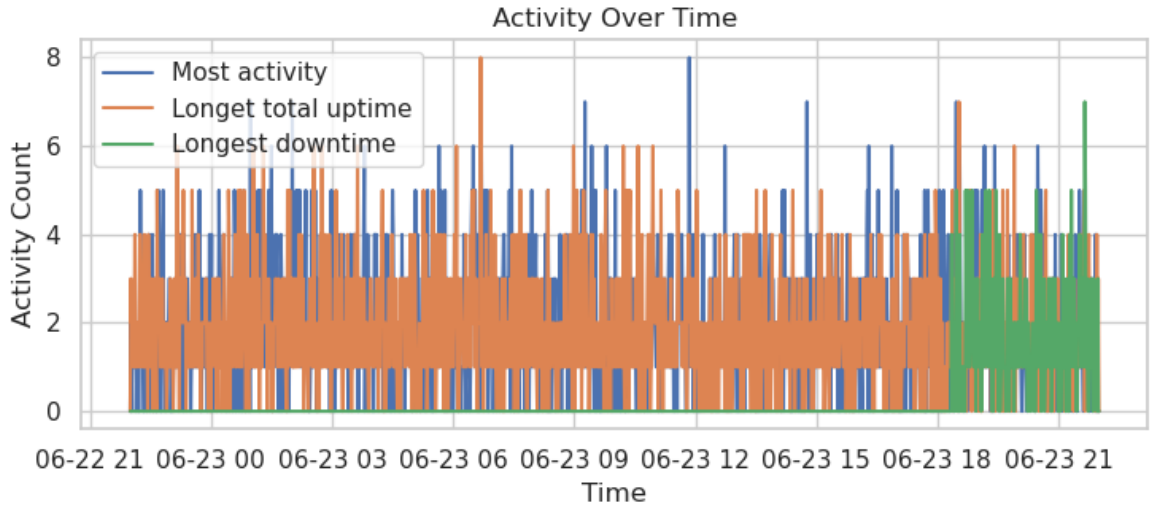

Figure 4.2: Three outliers of data in a single day.

To get a grasp on how our dataset differs we also zoom in on three outliers in our data and look at their daily activity. In figure 4.2 these three outliers are as follows:

- Observer 1: The most activity;
- Observer 2: The least downtime;
- Observer 3: Longest downtime.

To filter out some noise we will only look into ZMap data for this plot. We plot the number of ZMap packets each observer gets every second. It might be a bit hard to see but observers 1 and 2 receive nearly every second 1 to 4 ZMap packets. When looking at our data we notice that most observers look similar to observers 1 and 2. If we look at observer 3 we see a long period of downtime and thereafter similar activity as observers 1 and 2 receive. Only a handful of observers behave similarly to observer three as it is an outlier. This is also shown in figure 4.3 where we have plotted the boxplots for our activity. Here we see that the activity and uptime for all the hosts is about the same. If we look at the downtime we see that hosts with downtime are outliers, most hosts have little to no downtime. In table 4.1 the uptime of all the observers is shown after the 8th of June. Here we see that most observers have an uptime of over 97%.

Uptime percentage	Observers
97.9	62883
3.90	1552
≤3.16e-2	630

Table 4.1: Uptime percentages and corresponding number of observers.

In this paper, only TCP packets are considered. We dump nearly all of them with the only exceptions being port 23 (telnet) and port 445 (Shared Message Block), which are blocked on our network. These are the limitations set by our network but we will also set some limitations ourselves to help with our methodology. Because we design our method around ZMap we will make sure that we have a high chance that we only observe ZMap packets. We will therefore filter for $IPid = 54321$ [12]. This might

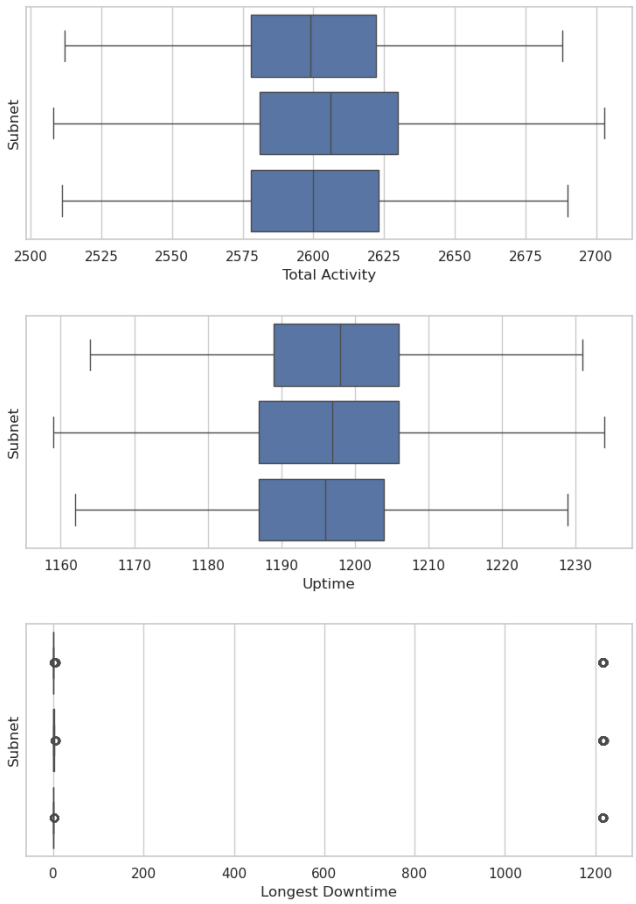

Figure 4.3: Distribution of activity, uptime and longest consecutive downtime. Split over the different subnets.

still produce noise because other packets are free to have their identifiers assigned to 54321, but most of our data should be from ZMap. As mentioned earlier, ZMap scanners can change this identifier but our method can still be used for the whole data. This limitation is only for the designing phase.

We have looked at our network telescope and have seen it's limitations and have come up with ways to overcome these limitations. When we will assign features later in section 5 we will only look at the packets received after the 8th of June to bypass the downtime of our network telescope. Furthermore, we have seen that our telescope has 62883 observers with an uptime of above 97%. This number of observers and uptime percentage will be used for further calculation in our methodology in section 5.

5 Methodology

Because of the use of scanning tools and methods to avoid common scan detection, we will observe unintended characteristics from the different hosts of a sharded scan. We hypothesize that we can assign these characteristics to features to be used in clustering algorithms to detect slow scanners. In this section, we will go over the necessary steps to characterize and cluster different scan groups together. Not necessarily similar scanners but to identify sharded scans as explained in section 2.8.

We will first build a simulation in section 5.1 which we will use to understand ZMap better. This simulation will also be used to test the features we will assign to the different hosts we observe in section 5.2. With the features, we could later replicate the scan using our simulation to check the algorithm.

Because of the size of the total data our telescope gets in a month, we will talk about bin sizes in section 5.3. There we would explain some necessary optimizations to make our method more scalable.

After the assigned features we would need some preprocessing and scaling, which we will discuss in section 5.4. With the preprocessed data we can go to the actual clustering in section 5.5. There we would also explain how to get viable parameters for scanning this kind of data.

And lastly, we will discuss a method to check the method in section 5.6 by looking at IP address correlation and looking at the feature values to check if the correspond with the number of found shards in a sharded scan.

5.1. Simulation

As discussed in Chapter 2, several tools facilitate scanning tasks, which are openly accessible due to their open-source nature. This accessibility contributes to their widespread usage. To comprehend scanners thoroughly, it's important to grasp both scanning processes and the functioning of scanning tools. However, it is not ethical to start scanning the internet on our own. Online, various IP blacklists exist, comprising addresses that do not want to be scanned. While these blacklists can mitigate undesired scanning activities, their completeness remains uncertain. Hence, we use simulations to aid in comprehending the scanning process without actually scanning the internet. Notably, ZMap incorporates this feature [12], coincidentally being one of the more popular scanning tools [15]. During ZMap execution, a "dryrun" flag can be set, causing ZMap to refrain from sending packets to their actual destinations but instead printing them to stdout. It does so in the following form:

```
tcp { source: 40719 | dest: 80 | seq: 1798838993 | checksum: 0XFFFF }
ip { saddr: xxx.xxx.xxx.xxx | daddr: yyy.yyy.yyy.yyy | checksum: 0XEEEE }
eth { shost: ff:ff:ff:ff:ff:ff | dhost: ee:ee:ee:ee:ee:ee }
```

Where it prints three layers. It does however not print the whole packet. Primarily what stands out is the absence of the identification field (IPId). But the information in the stdout are the only variables that change per packet. The packet does also not have a timestamp but this we can add ourselves. To make sure that the packets printed are in order of the original cycler we also set the simulation to run on a

Scan State	Shard State	Soure Address	Destination Address
Int64	Int64	Int32	Int32

Table 5.1: Save format of a simulated packet

single thread per ZMap instance. With this, we also know the state of every scan [12, 19]. Furthermore, we use the sharded function of ZMap to simulate the distributed (or sharded) scans. This is done by telling ZMap how many shards there are in total and telling each shard its number. Each instance of ZMap is then run inside a docker container. To make sure that no leaking occurs, the docker containers are also isolated from the internet.

If we would write this output directly to a file it would be massive because it is in string format. With the above example, it would be around $205B * 2^{32} = 880GB$ of data. With a blacklist and with shortening IP addresses the file size would decrease but we are not that interested in all that data anyway. The goal is to simulate an internet-wide scan, but we only want to evaluate if a telescope can cluster the shards together. So from all those packets we only save the ones that would be sent to a telescope. For the simulation, we have chosen to save packets that would have been sent to our telescope. Lastly, we also don't need the whole packet so we use a combination of GREP and AWK to filter the stdout and only save packets in the format as shown in table 5.1.

Because we know the shard state of every packet sent we can assign a starting time and simulate a timestamp for every packet. We also don't save the source port because it is randomized every time and we don't save the destination port because we already know that when initializing the scan. With this simulation setup, we can assign features to our results. With our results, we could also argue how big a telescope needs to be to correctly assign features.

5.1.1. Destinations

First, we look at the number of destinations each shards gets assigned. In ZMap, to make sure the destinations are unique, they iterate over $\frac{\mathbb{Z}}{(2^{32}+15)\mathbb{Z}}^{\times}$ to make sure every address is targeted and because $2^{32} + 15$ is a prime number the iteration will be cyclic [12]. This makes sure that every address is exactly targeted once. To randomize the sequence ZMap assigns a random primitive root g of $2^{32} + 15$ together with a random starting offset for the cycler. An example can be seen in table 5.3 where we have a prime number 7 and a step size of 3, which is a primitive root of 7. We have the iteration function $x_i = g \cdot x_{i-1}$ and assign an offset of $x_0 = 3$.

$$x_0 = 3 \tag{5.1}$$
$$x_1 = 3 * 3 \mod 7 = 2 \tag{5.2}$$
$$x_2 = 3 * 2 \mod 7 = 6 \tag{5.3}$$
$$x_3 = 3 * 6 \mod 7 = 4 \tag{5.4}$$
$$x_4 = 3 * 4 \mod 7 = 5 \tag{5.5}$$
$$x_5 = 3 * 5 \mod 7 = 1 \tag{5.6}$$
$$x_6 = 3 * 1 \mod 7 = 3 \tag{5.7}$$

Every number is visited in a pseudo-random order. This iteration formula does however not cover 0, but this is by design as 0.0.0.0 is blacklisted by ZMap anyway. If ZMap is configured to do a sharded scan these variables are the same over all the shards. Because all the shards have the same variables and they know how many shards there should be, the number of destinations per shard is divided by the total number of destinations. This means that without a blacklist the number of destinations per shard is equal. Because this could vary with the length of the blacklist we have done our simulations with the default blacklist to represent to most common user.

Now let us look at how big the telescope needs to be. We first calculated a theoretical value and then looked at our simulation to see if the measured value corresponded with the equation. For the following sections, we define some variables to be consistent. These variables are as follows:

- **H**osts; The source addresses (or shards) of a scan.

Subnet	Measured Mean	Calculated Mean	Measured Std	Calculated Std	Observers	Hosts
1+2+3	774.05	774.05	29.80	27.82	196608	254
1	258.02	258.02	16.06	16.06	65536	254
2	258.02	258.02	15.26	16.06	65536	254
3	258.02	258.02	15.94	16.06	65536	254

Table 5.2: Measured and calculated number of destinations on our telescope per host. For the Network the total IPv4 address space is used.

0	1	2	3	4	5	6
0	1	2	X	4	5	6
0	1	X	-	4	5	6
0	1	-	-	4	5	X
0	1	-	-	X	5	-
0	1	-	-	-	X	-
0	X	-	-	-	-	-
0	-	-	O	-	-	-

Table 5.3: Iteration of ZMap where we iterate over prime number 7 with a primitive root 3.

- **O**bservers; The destination addresses that we can observe.
- **N**etwork; The network that is targeted by the scan. Normally consisting of the entire internet.

We know that the distribution of targeted addresses is uniformly random over N. Our observers are part of N so $O \subseteq N$. The probability that a packet send to N will reach O will therefor be $p = \frac{|O|}{|N|}$. A single scan that will send a packet to every destination on N will therefore also send a packet to every observer O. If we want to have an estimation of the number of destinations that we see on our observers from a single shard we have to divide the number of experiments (in our case $|N|$) by the number of hosts H. If we recognize this as a binomial distribution this will give us the following estimations of the number of destinations we observe per host:

$$p = \frac{|O|}{|N|} \tag{5.8}$$

$$n = \frac{|N|}{|H|} \tag{5.9}$$

$$\mu = np = \frac{|O|}{|H|} \tag{5.10}$$

$$\sigma = np(1-p) = \sqrt{\frac{|O|}{|H|} \cdot (1 - \frac{|O|}{|N|})} \tag{5.11}$$

In table 5.2 a full scan is run. We observed three different subnets and calculated the mean and standard deviation of the destinations that every source of the scan has on our subnets. Then we also calculate the theoretical values according to 5.10 and 5.11. We see that these values correspond with each other. This also means that we can fairly easily predict the number of hosts total in a scan according to the destinations we observe on our telescope. We have also plotted the equations 5.10 and 5.11 in figure 5.1 with different host sizes of 8, 16, 32, 64, 128 and 256. This gives a rough visualization of the effects of the number of hosts in correspondence with the number of observers in the telescope.

5.1.2. Timing

As some scanner might scan with the same amount of host, we want to define more features to differentiate between scanners. This will increase the effectiveness of our clustering. In this section we will look at another important aspect which we can use in assigning characteristics: timing. When a

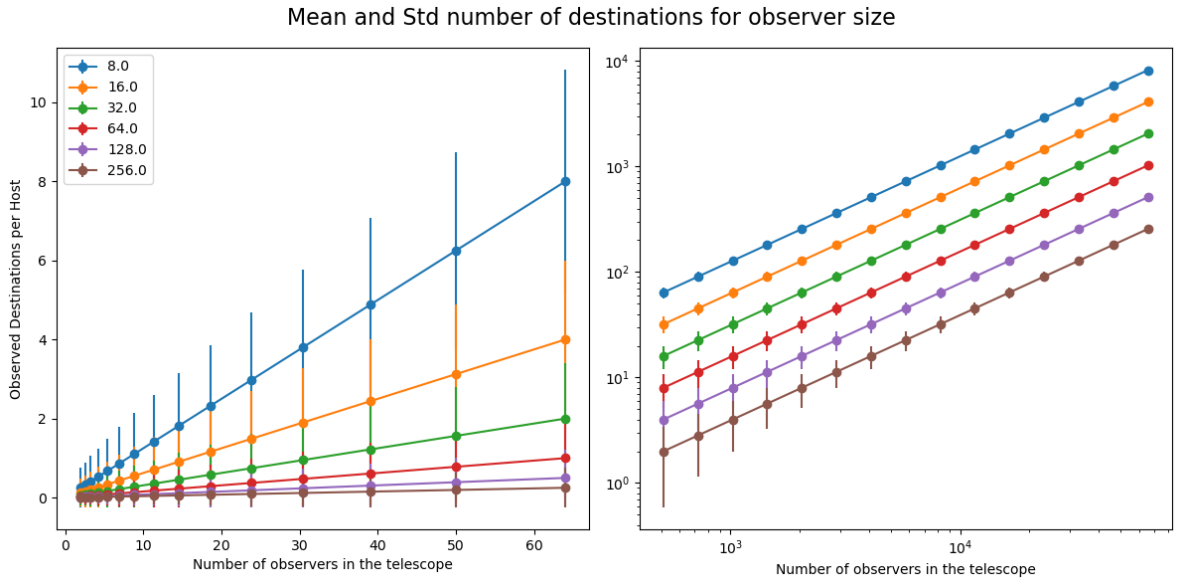

Figure 5.1: Mean and Std of the number of destinations on our telescope divided per number of hosts in a scan.

Subnet	Observers	Measured Mean	Measured Std	Calculated Mean	Calculated Std
1+2+3	196608	86.00	86.61	86.01	85.50
1	65536	258.00	260.36	258.02	257.52
2	65536	258.02	257.85	258.02	257.52
3	65536	258.01	258.05	258.02	257.52

Table 5.4: Calculated and measured frequency of a scan according to the scan state.

scan is started it is set to a particular rate. This rate is of course the rate that each H sends a packet to N. If we define the probability p as the probability of the observers seeing a packet sent by any of the hosts, we can give a probability of the frequency that this happens. This is still related to the rate the scan is set to, but if the rate is constant across the hosts then it scales linearly with the calculated rate below. We can use a geometrical distribution to calculate the rate the hosts will send packets to our telescope:

$$p = \frac{|H||O|}{|N|} \tag{5.12}$$

$$\mu = \frac{1}{p} = \frac{|N|}{|H||O|} \tag{5.13}$$

$$\sigma = \sqrt{\frac{1-p}{p^2}} = \sqrt{\frac{1 - \frac{|H||O|}{|N|}}{\frac{|H||O|}{|N|}^2}} \tag{5.14}$$

This is considering the shard states but they scale linearly with the actual rate of the packets sent per host. With the same simulated scan we ran in 5.1.1 we can compare the measured frequency with the calculated frequency. The results are shown in 5.4, where we find that formulas correspond well with the measured data. Because the earlier mentioned evenly distributed destination addresses between scans we also assume that the scan rate of each shard is also around the same. Also when the shards are started around the same time and have the same rate, the first time they appear at our telescope should be the same.

Another characteristic that follows from this is the scan duration. When doing a single scan, the scan will finish after all the destinations are approached. We can therefore also measure the activity length

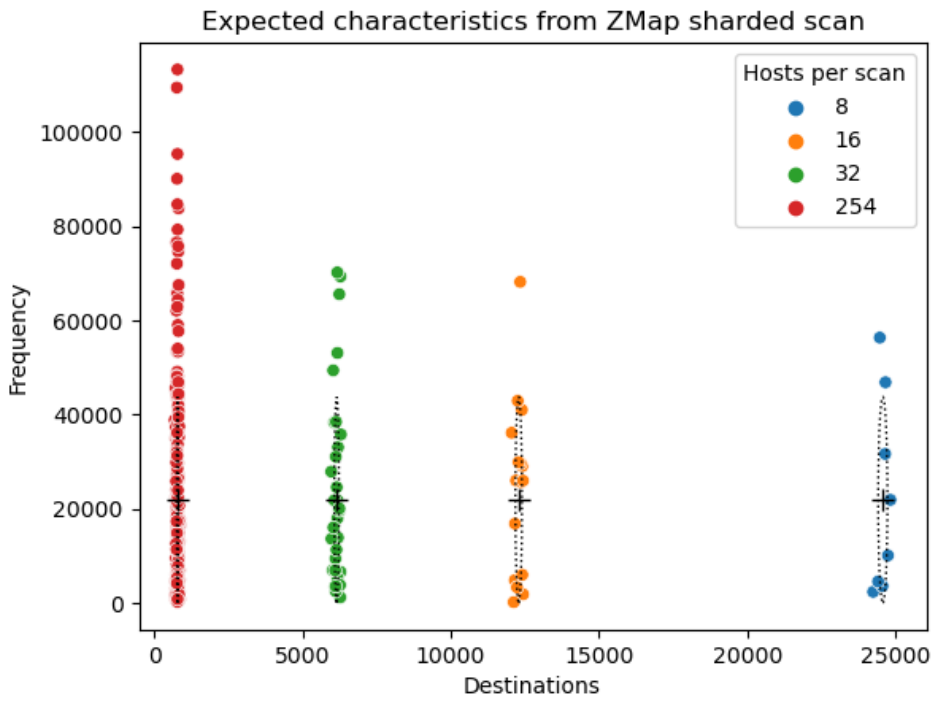

Figure 5.2: The expected vs the measured data from a few ZMap sharded scans. The crosses and dotted lines are the expected values of the different simulations.

as a feature.

5.1.3. Synthetic data

We have run some simulations as described in the beginning of section 5.1. This gives us synthetic data that we can compare to the equations 5.10, 5.11, 5.13 and 5.14. We can use these calculations to estimate the characteristics of the scan and then measure the synthetic data and see how it corresponds. In figure 5.2 we have plotted the number of destinations versus the frequency of both the measured and calculated data. The crosses and dotted lines represent the calculated expected mean and std of these features and the dots are all the hosts in each of the scans. We see that the data roughly corresponds with the exception with some outliers. These outliers are especially apparent in the frequency. We have only looked at two characteristics but because we can already see some outliers, more characteristics are necessary to make the clustering algorithm more effective. In our simulation we can also look at the scan durations for each host. These are visualised in figure 5.3. We see that these are closer together.

5.1.4. Visualization

In 5.1.3 we have only seen a few dimensions but if we increase the number of features for the clustering algorithm, we also increase the number of dimensions that represent the hosts. Above 2 or 3 dimensions it will be really hard to visualize the hosts on paper. Because of this, we will use t-SNE [14] to represent our data. An example of this is given in figure 5.4. Here we have given each host the following characteristics:

- Frequency
- Destinations
- Durations
- Offset

Where the offset is random for each scan. So we have 4 dimensions that will be represented in 2 dimensions. In figure 5.4 we can see that t-SNE puts the different hosts of each scan together with enough room in between for us to differentiate between scan. Note that the axes do not directly represent the

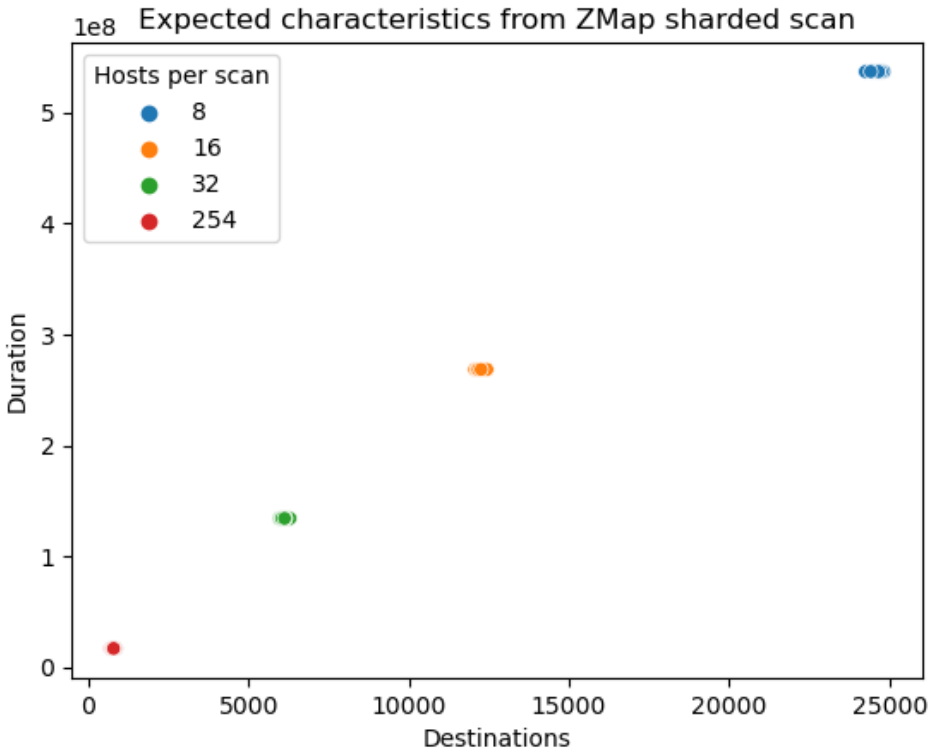

Figure 5.3: Destinations vs duration of the simulated data.

different features, it is purely for visualization. This visualization technique is stochastic and therefore non-deterministic. It will therefore only be used to visualize the clusters and not for the clustering itself as it can influence the algorithm.

5.2. Features

In section 5.1, we discussed certain characteristics of ZMap sharded scans related to the scan settings, including the destinations and timing of a single sharded port scan. In this section, we will expand on these characteristics by examining the sources over a month. This approach allows us to measure not only individual scans but also multiple scans from the same group. Consequently, we will assign additional characteristics related to their overall behavior, beyond the scanning settings of a single scan.

Number of unique destinations

For ZMap, the number of destinations for a source in a sharded scan is evenly distributed over all the shards [12, 19]. We have also seen that this is true ourselves in our simulation (see section 5.1). For single scans, this is a good indication as it is a feature with a known standard deviation which, depending on the size of the network telescope, can be quite small. For multiple scans, however, there might be some inconsistencies. This depends on the scanners using the same random seed for successive scans. If the same seed is used but only the port is changed then the different sources will have the same unique destinations every scan. But if the seed is different each scan this will not be the case. The number of unique destinations over multiple scans can therefore vary a bit. But we will still use this metric because if we would look for every packet sent then we would have even more inconsistencies because of missed packets and retries.

Scan Frequency

In section 5.1, we observed that the frequency at which we detect a scan depends on both the scan rate and the number of shards. This is because the rate at which a packet is sent to our telescope, rather than to a random IP address we can't observe, is defined by equation 5.13 and is then multiplied by the

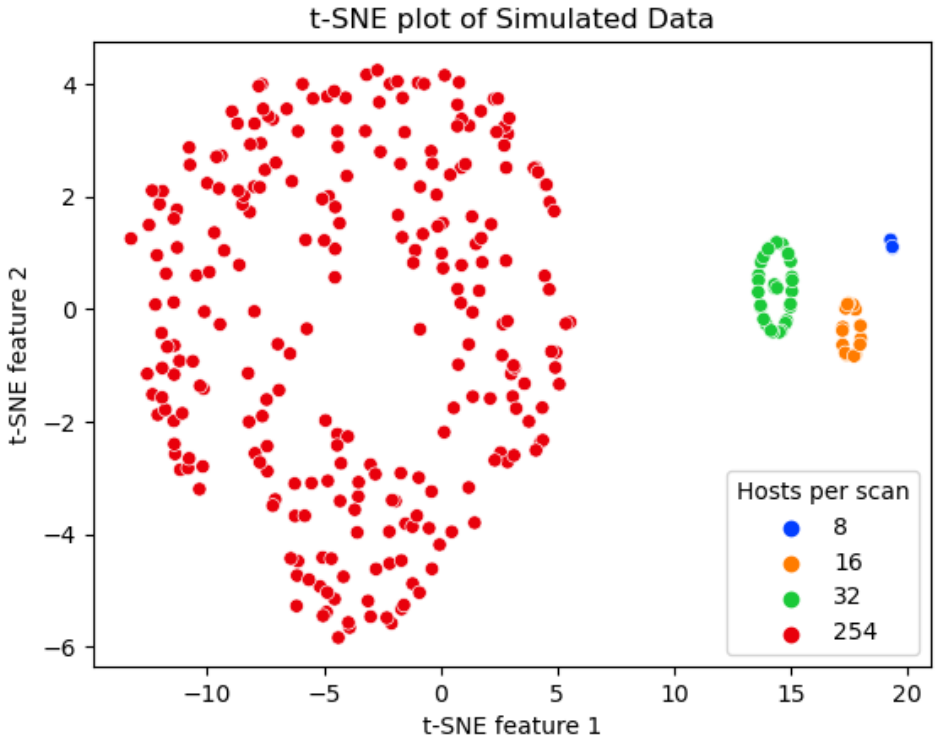

Figure 5.4: t-SNE representation of simulated data.

scan rate itself. This frequency would vary more if we measured multiple scans from the same group. However, we found that groups performing multiple scans tend to maintain a constant frequency, whether it's daily, weekly, or immediately after the previous scan, as we will demonstrate in section 6. If these two frequencies are constants, we can derive a consistent composite frequency for use in clustering.

Start time

When starting a sharded scan, it is convenient to use scripts to send ZMap instructions to all the shards simultaneously, so they start around the same time. The hypothesis is that if a scan is initiated, we will observe a peak of activity from the same group on our network telescope. Therefore, we assign a start time for each source based on the first time it appears on our network telescope. While this feature might not provide much information for internet noise and continuous scanners, it offers valuable insights for slow scanners that are not always active. Additionally, if there are continuous scanners, their start time will be the same as the start time of our telescope, ensuring they are still clustered together. It's important to note that if a new vulnerability is discovered, most groups scan the entire IPv4 address space within a few hours [11], which could result in some false positives when considering only this feature.

Total Activity

This feature measures the total active periods. This can be either the duration of a single scan or the cumulative duration of multiple scans. If the number of destinations is approximately the same and the rate of the packets sent is set equal for each shard, the activity time on the telescope should also be approximately the same for each source.

Number of unique destination ports

A ZMap scan is a horizontal scan that targets a single port. However, during our initial reconnaissance, we observed that many groups perform multiple scans for different ports. When a group conducts multiple scans, each source in the scan targets the same ports, resulting in the same number of unique

ports being scanned. Therefore, we can use the number of unique destination ports as a feature for our clustering algorithm.

We choose the number of unique destination ports because each feature must be numerical. However, we can later verify the accuracy of the clusters we identify by examining the actual ports scanned by each source.

5.3. Bins

One of the features we have used is frequency, calculated by the difference in arrival time for each packet from a source. However, ZMap sometimes sends a few retries if an error occurs, which can skew the frequency value. Therefore, it's necessary to filter out these retries.

Additionally, we need to analyse data spanning a whole month, which can be time-consuming. We therefore need to make the algorithm more efficient. To address both issues, we use bins. Bins are periods during which we measure activity, and we count a source as active if we see it once within this period. This approach allows the algorithm to avoid looking further within the period, similar to adjusting the resolution of a photo. With lower resolution, the algorithm finishes earlier, but some data might be lost.

Choosing the right bin size is crucial: a larger bin size decreases computation time but can result in information loss. However, for slow scanners where activity is spread over a long period, a larger bin size does not harm the clustering method as long as the scan frequency is smaller than the bin size. For our experiments, we chose a bin size of 5 minutes.

5.4. Preprocessing and Scaling

With the different features assigned, we still need to prepare them for clustering, as the scales of the features are not correlated. For instance, the number of unique destination ports for most slow scanners may be just a few, while the start time could be a six-digit number if we measured over a month because it is in UNIX time. In such cases, the number of ports could be ignored if spatial clustering is used. An example of this is shown in figure 5.5, where we use KMeans to cluster the three blobs we have seen earlier in section 2. Since feature 2 has a much bigger range than feature 1, the clustering only follows feature 2. However, if we scale the features appropriately, the clustering functions as expected.

There are various options for scaling the features. The most basic method, the "StandardScaler", removes the mean and scales the data to unit variance. However, outliers can heavily influence this method, as it scales all the values linearly, causing the average data to be compressed and more prominent for spatial clustering.

Instead, we use a robust scaler, which is based on percentiles and not influenced by outliers. Essentially, it scales the features more fairly by focusing on the larger portion of the data, while still treating outliers as outliers.

5.5. Clustering

We use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) for our clustering algorithm as discussed in 2.9. DBSCAN allows us to identify clusters of different densities and irregular shapes. This is perfect for our case because the features of different clusters can be very broad. DBSCAN only uses two parameters $min\_samples$ and ϵ. For $min\_samples$ Sander et al. suggests that it should be $2 \cdot dim = 10$, a higher number can be chosen if we have a noisy dataset, which we have, but a too high of a number also means that smaller clusters will not be recognized [28]. For the parameter of ϵ we want the value to be as small as possible [29]. We used the knee method [31] to assign a value to ϵ. For this method, the distance between the nearest k neighbors is plotted and it is suggested to use $k = min\_samples$ [31]. The knee would then be the point where the largest increase in distance occurs. After this point, the risk of overfitting will increase. Tweaking and optimizations of these parameters can still be done to improve the results. For this research we have chosen ϵ to be slightly below the knee because this improved our results.

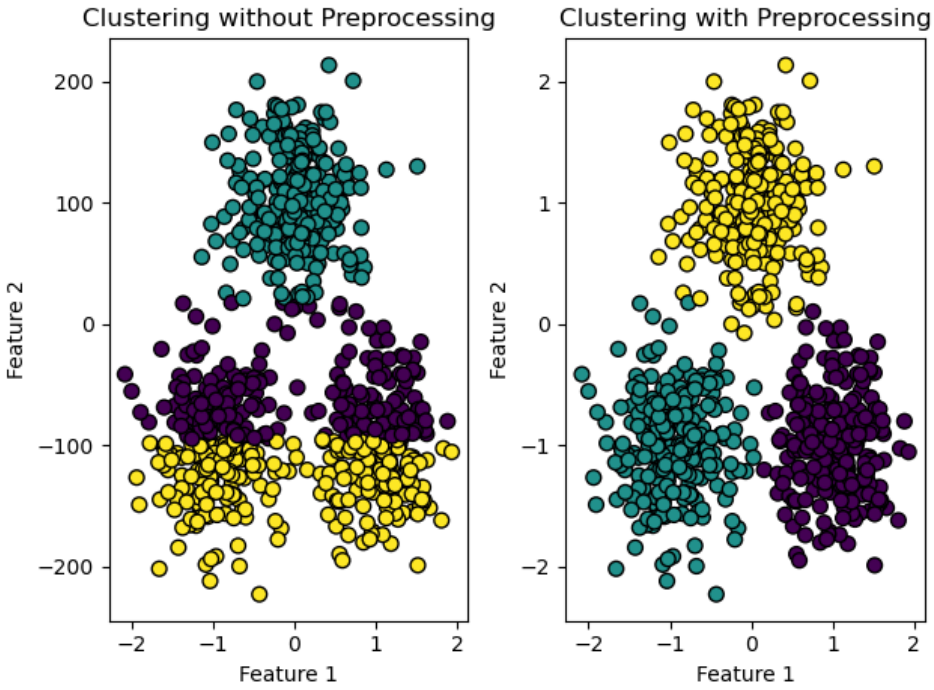

Figure 5.5: Example of why preprocessing is necessary. KMeans clustering is used on raw data and scaled data.

5.6. Confirmation

Our method allows us to identify different hosts in sharded scans. Past research suggests that scans originating from a common group often come from the same network subnet [22]. However, this assumption isn't always accurate. To verify the effectiveness of our clustering algorithm, we can cross-reference various factors such as network subnets, geographical locations, and organizational affiliations.

To strengthen the case for identifying clustered scans, we need to compare scans from both within and outside the same subnet. This comparative analysis will provide additional evidence to support our clustering approach.

Additionally, for our clustering algorithm we have used the number of destinations ports. To further validate the clusters we've identified, we'll analyze the lists of ports scanned by these hosts and compare them to the lists of ports scanned by other hosts in the same cluster. If they corresponds with each other we can enhance the credibility of our findings.

Furthermore, it is possible to use the calculations from section 5.1 to evaluate the correctness of the found clusters. If we look at equation 5.10 we can rewrite this to calculate an estimated number of hosts:

$$E[H] = \frac{|O|}{\mu} \tag{5.15}$$

We know from section 4 that our active number of observers for the whole month is 62883 and we can optimise this value if we look at more specific periods. The μ in equation 5.15 will be the mean number of unique destinations on our telescope that each host in the clusters scans. If the estimated number of hosts corresponds with the found number of hosts we know that we have found a complete sharded scan.

6

Analyses

In this section, we will look into the results yielded by the clustering algorithm, looking at the question of whether we can effectively classify and identify slow scanners based on their behavioral patterns and attributes. Before this analyses, we provide an overview of the clustering results. We first look at an examination of the origins of all recorded sources and the primary ports that are targeted.

Following the data overview, we will continue into a detailed analysis of the results. We begin with a summary of the data, where we explain the clustered datasets and outline our approach to classifying the distinct clusters. Moreover, we examine the geographical locations and organizational affiliations associated with the sources in these scans.

Subsequently, we delve deeper into the examination of the significant differences observed among the various clusters and classify them. We select representative clusters and provide insights into their origins, while also assessing the correctness of the alleged scans that we have found.

Finally, we conclude this section with a series of case studies, offering some examples of particular clusters that have some oddities and try to explain those.

6.1. General Overview

In total, we have $10.2e9$ packets captured in June of 2023. As explained in chapter 2 this is all background radiation. To make the progress of designing our method easier we have chosen for this research to only cluster alleged ZMap data. This is done by setting the IPId to 54321. This will result in our set containing $4.0e9$ packets. This means that at least 40% of the captured data is coming from ZMap instances. This percentage can vary a bit due to some packets getting randomly assigned the IPId of 54321 but on the other hand, some ZMap instances might be altered to change their IPId.

In table 6.1 we see the most popular targeted ports by the background radiation. Here it shows that they roughly contain the same set except that the ZMap packets have more HTTP alternative ports like 8000, 8088 and 8089.

For the remainder of this overview we will look at only the ZMap data. We begin by calculating a rough estimate of the percentage of heavy hitters in the ZMap data. However, the definition of heavy hitters varies per research. Our definition is that heavy hitters are easily recognizable scanners because they scan the whole network as fast as currently possible. An important distinction with the definitions of heavy hitters is that we, for our research, only care about scanners. The term heavy hitters in a broader sense can also mean adversaries that target the same destination over and over, as is done by DDOS attacks. These are easier detected by frequency-based algorithms [5, 35]. We ignore this category of heavy hitters and focus on scanners that scan the network as fast as possible. A by-product of this is that a heavy hitter will also scan our whole telescope, or at least close if we account for downtime. Our total number of active observers is 62883 (as shown in chapter 4). With this, we can calculate the percentage of every source that targets the telescope. We have sorted this and plotted this in figure 6.1. Our window is very broad, so we will see multiple scans from some of the groups. Because of this, a

Total		ZMap	
DstPort	Count	DstPort	Count
22	176165960	8080	102649785
80	163455008	5555	80071491
8080	153740272	80	70184194
3389	133298875	443	62532182
443	113312915	8443	47045070
5555	95427863	3128	46155976
8443	67856231	22	44720709
81	54383832	81	38847161
8081	51940348	8081	36081557
3128	51730870	8888	20016711
2375	51424683	9080	19994858
2323	38889597	8088	17784510
52869	38886635	1080	17584152
2376	37647694	3389	17051070
1433	37229963	9200	15552493
2222	31169712	8000	14991215
8888	30731798	8090	14432758
5985	30525111	7000	12638492
27017	29997373	7443	12632947
3306	28195009	21	12629106

Table 6.1: The most targeted ports.

single source can come close to targeting the whole network at least once. If we look at figure 6.1 we see a sharp knee around 95%. After the knee, we have the obvious heavy hitters. It might still be possible that some slow scanners have also gone past the knee because our window is so big but this will give us a rough estimation of the number of heavy hitters. The percentage of heavy hitters that use ZMap as detected on our telescope was around 17% as of June 2023. Before the clustering, we will filter out these heavy hitters because we are not interested in those for this research. Filtering more before scaling also allows the scaling to be better as potential outliers will be removed.

We also look at the organizations and countries of the ZMap data. This can later also help us by evaluating the correctness of the clustering algorithm. This data is acquired with MaxMind's GeoLite2 database which is free and can be run locally to optimize speed. In table 6.2 we see the country origins of all the ZMap packets. Which roughly corresponds with recent previous research [24, 11, 36]. The difference with previous research is that we see a rise in scan source origins from Singapore.

We also look at the organizations by looking at the Autonomous System Number (ASN) associated with the IP addresses. The top organization's origins from ZMap packets are shown in table 6.3. What immediately stands out is the massive percentage of packets originating from DIGITALOCEAN with 70% of packets coming from them. DIGITALOCEAN together with nearly every other organization on this list offers Virtual Private Servers (VPS). What DIGITALOCEAN offers which might explain their popularity by scanners is that they are scalable and will bill hourly, it is therefore possible to temporarily rent multiple servers, scan for a few hours and then shut down, saving on costs.

6.2. Cluster Summary

In table 6.4 we have given a summary of all the found clusters using DBSCAN and our assigned features for every source. We have chosen the following relevant variables per cluster:

- Sources: The number of sources in the cluster.
- Class A, B and C subnets: The number of /8, /16 and /24 used by the sources.
- Countries: The number of origin countries of the sources.
- Organizations: The number of organizations where the sources originate from.

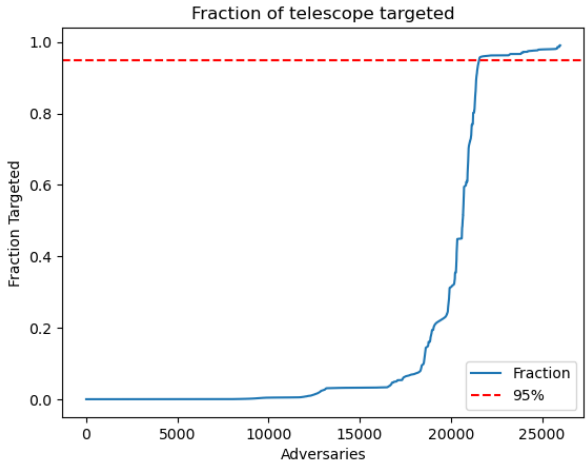

Figure 6.1: Percentage targeted by all the ZMap adversaries during the month of June.

Country	Percentage
United States	0.25
Singapore	0.12
Germany	0.11
India	0.09
United Kingdom	0.09
Canada	0.08
Netherlands	0.08
Australia	0.06
Mongolia	0.03
Italy	0.01
China	0.01
Other	0.07

Table 6.2: Top percentage of countries from the ZMap scans.

Organization	Percentage
DIGITALOCEAN-ASN	0.70
Akamai Connected Cloud	0.03
GEMNET LLC	0.03
SkyLink Data Center BV	0.03
CARINET	0.03
UK-2 Limited	0.01
Other	0.14

Table 6.3: Top percentage of organizations from the ZMap scans.

- Jaccard: The Jaccard index of each cluster. The set of destinations of each source is compared with the destinations of all the other sources in the same cluster. If there is no overlap then the index will be 0. If there is complete overlap the value will be 1.

All these variables are taken from the complete cluster except the Jaccard index. The Jaccard index is calculated from a single scan inside a cluster. This single scan is approximated by using the first peak in a kernel density plot of a single port used by the cluster. The values of the Jaccard index can vary for some clusters as this single scan is not always obvious and hard to automate. If there is some overlap in scans then the Jaccard index can also give some slight errors. If the Jaccard index is 0, or a really low value, we have a good indication that we defined a good portion of a single ZMap sharded scan.

The other variables can also be good indicators to evaluate the clusters. A single organization used is for example a strong indicator as this would also be for adversaries a simple thing to do. We see in our data that if people use hosting providers they will use the same provider for their scan. If multiple organizations are used it might still be possible to be the same scan but the certainty degrades. Another possibility is if the organization is 1 the cluster only uses their own organization, this of course is also a giveaway that it belongs to the same scan. We also look at the subnets and number of sources, in particular, is the number of sources is the same as the number of available addresses If we look at clusters 34 and 61 for example. The expected number of sources we can detect from a single /24 subnet is 254 because there are 256 options and 0 and 255 are reserved. These clusters contain 252 and 253 hosts, which would suggest that we could have missed a few hosts, but in section 6.3 we will show you that these clusters are actually complete. These two clusters in particular are from two different providers and we can cross-reference their IP address range on our telescope.

In figure 6.4 we see all the clusters that our method has found. In this plot, the heavy hitters and noise are omitted. The biggest clusters, clusters 0 and 1, are shown snaking around most of the plot. Also if we look at the data in table 6.4 we can conclude that these clusters are probably a combination of multiple scanners and/or noise. This we can also see in figures 6.2 and 6.3, where we have plotted the packets sent from a set of sources to our telescope. The different colors represent different ports and the size of the points represent the number of retries. If we look at figures 6.2 and 6.3 and in particular 6.2 we can identify some sort of similar behavior with some of the points in these figures, like the two straight lines scanning 2 different ports on the 16h and 17th of June, but the clustering algorithm has not correctly identified them. Most other clusters in the table are however correctly identified. In the next sections we will categorize them and explain their behavior.

6.3. Cluster Details

In section 6.2 we have seen two noisy clusters. In this section, we will go in-depth over the other clusters. Particularly, we will describe similar behaviors among the clusters that we have found. Here we will select some representative clusters and classify them, putting them in different categories that represent different scanning behavior.

6.3.1. Periodic single port scanner

First, we look at periodic single-port scanners, which describes the scanning strategy of scanning a single port periodically. The scanners will scan the same port multiple times over a long period of time, likely to update their database. One of these is shown in figure 6.5. This scanner in particular uses a single /24 subnet to scan the same port every 7 days. We have identified this scanner as the University of Michigan that scans port 7 periodically, which belongs to the Echo Protocol. We have cross-referenced the clustered data with the complete data of this subnet and we concluded that the cluster algorithm has not missed any sources coming from that subnet.

Another periodic scanner identified is cluster 12 as shown in figure 6.6. This scanner looks nearly identical in behavior to cluster 51 but these sources do not belong to a single subnet as cluster 51 does. These all come from different DIGITALOCEAN servers from different countries so not possible to cross-reference the subnet to see if the algorithm has missed any sources. However, if we look at the activity in figure 6.6 we can see that the activity per source is nearly identical around the same periods. Furthermore, it is possible to use equation 5.15 instead of the subnet to check if the algorithm has found all the sources belonging to this scan. In table 6.5 we have divided the data we gathered from cluster 12

Cluster	Sources	Class A subnets	Class B subnets	Class C subnets	Countries	Organizations	Jaccard
0	1385	90	171	800	34	62	2.04e-03
1	3615	134	390	2486	46	150	3.89e-03
2	276	74	124	227	31	59	1.21e-03
3	11	11	11	11	9	11	1.26e-02
4	108	4	4	4	2	2	0.00e+00
5	21	15	19	21	8	2	0.00e+00
6	29	10	11	22	2	1	0.00e+00
7	739	44	79	610	9	2	0.00e+00
8	17	13	17	17	8	14	2.92e-03
9	10	3	5	10	1	1	1.35e-01
10	144	25	39	114	7	2	0.00e+00
11	22	11	12	19	4	1	0.00e+00
12	50	20	29	47	7	1	4.78e-04
13	26	6	7	17	3	1	0.00e+00
14	20	10	13	19	6	1	0.00e+00
15	32	15	16	20	8	2	5.29e-04
16	10	4	4	8	1	1	0.00e+00
17	59	8	9	9	8	9	3.78e-03
18	10	5	6	9	1	1	0.00e+00
19	10	3	3	9	1	1	0.00e+00
20	30	8	12	27	2	1	0.00e+00
21	10	4	5	10	1	1	0.00e+00
22	21	12	17	21	7	1	0.00e+00
23	10	1	1	6	1	1	0.00e+00
24	28	8	9	26	3	1	0.00e+00
25	29	10	10	17	3	1	0.00e+00
26	126	1	1	1	1	1	0.00e+00
27	125	1	1	1	1	1	0.00e+00
28	12	10	12	12	6	10	2.90e-03
29	15	11	15	15	6	10	2.91e-03
30	10	9	10	10	4	8	2.61e-03
31	10	9	9	10	4	7	3.66e-03
32	10	10	10	10	3	8	3.22e-03
33	10	10	10	10	5	8	2.91e-03
34	252	1	1	1	1	1	0.00e+00
35	128	1	1	1	1	0	4.01e-03
36	19	5	6	8	3	1	0.00e+00
37	39	2	3	3	1	2	7.63e-02
38	27	6	9	18	2	1	0.00e+00
39	10	1	1	2	1	1	0.00e+00
40	28	7	7	17	3	1	0.00e+00
41	14	8	10	14	3	1	2.22e-01
42	10	4	4	6	1	1	0.00e+00
43	25	7	7	13	3	1	0.00e+00
44	26	7	7	20	2	1	0.00e+00
45	10	2	2	6	1	1	0.00e+00
46	27	7	7	13	3	1	0.00e+00
47	10	6	6	9	1	1	0.00e+00
48	10	3	3	3	1	1	0.00e+00
49	10	2	2	3	1	1	0.00e+00
50	28	6	7	15	3	1	0.00e+00
51	10	1	1	1	1	1	0.00e+00
52	10	3	3	5	1	1	0.00e+00
53	10	2	2	3	1	1	0.00e+00
54	12	4	4	9	4	2	0.00e+00
55	10	2	2	2	1	1	0.00e+00
56	10	2	2	5	1	1	0.00e+00
57	10	1	1	8	1	1	0.00e+00
58	10	3	4	8	1	1	0.00e+00
59	10	2	2	8	1	1	0.00e+00
60	10	2	2	8	1	1	0.00e+00
61	253	1	1	1	1	1	0.00e+00

Table 6.4: Summary of the clusters gained with DBSCAN.

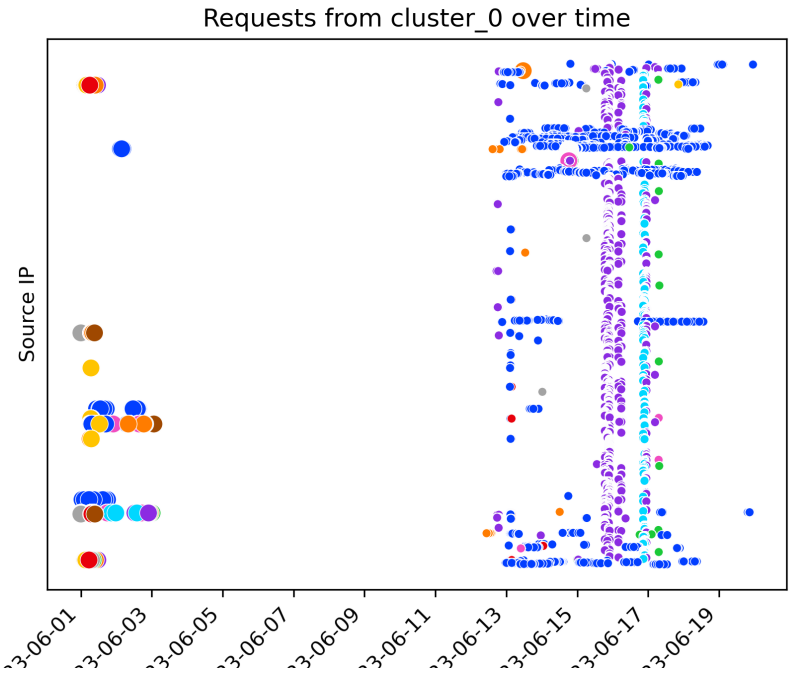

Figure 6.2: Noisy cluster 0 where a different color represents a different destination port.

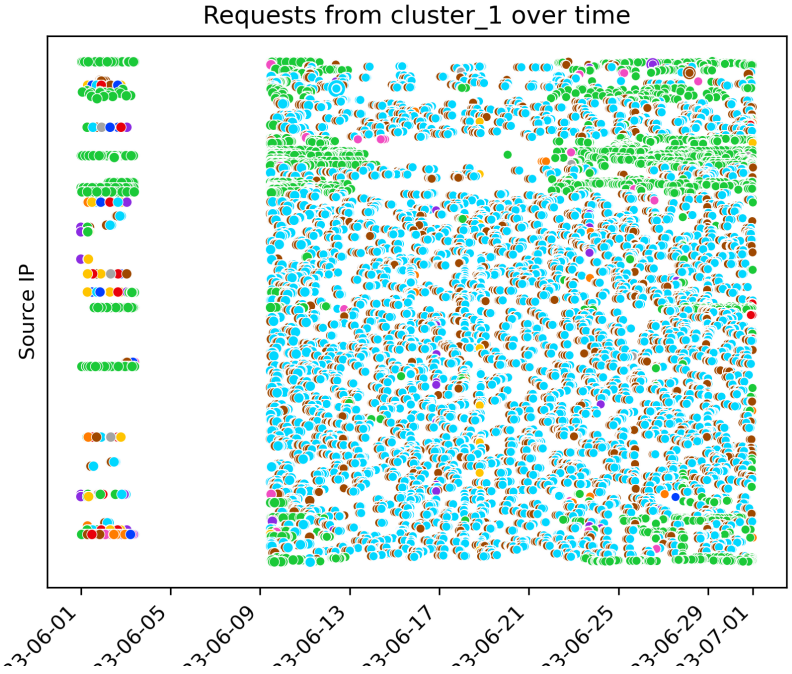

Figure 6.3: Noisy cluster 1 where a different color represents a different destination port.

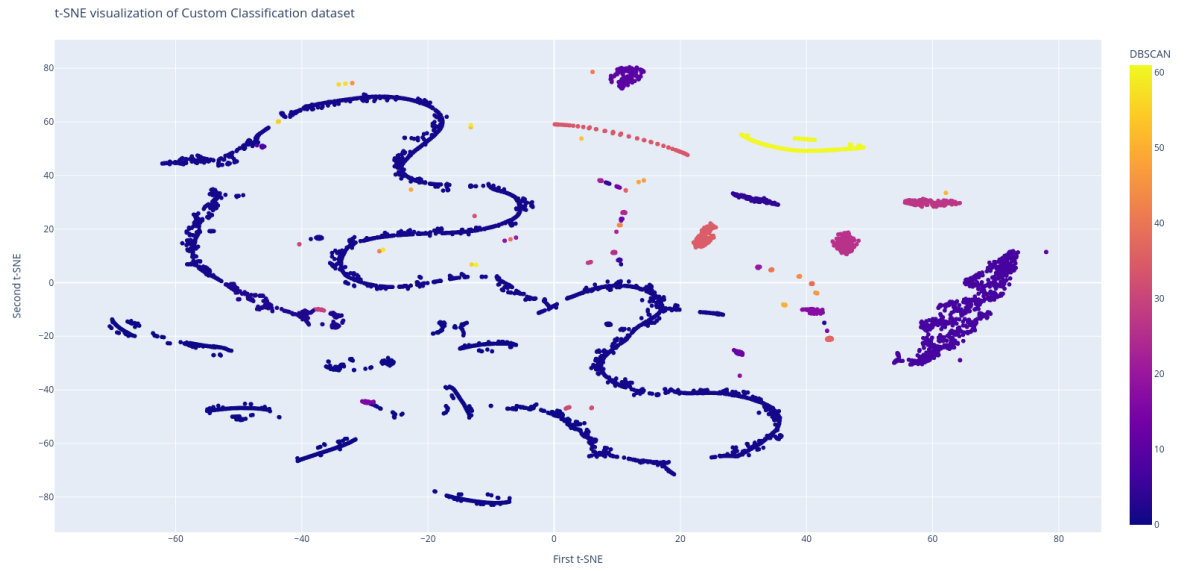

Figure 6.4: The clusters without heavy hitters and noise.

Scan	Unique Dest	Start time	Duration	Dest/Host	Jaccard
0	24561	2023-06-13 14:24:45	0 days 00:35:53	491.22	0.0000
1	24604	2023-06-13 19:44:23	0 days 00:35:04	492.08	0.0000
2	62628	2023-06-15 14:41:06	0 days 01:32:19	1252.56	0.0000
3	62626	2023-06-16 14:25:24	0 days 01:32:12	1252.52	0.0000
4	62646	2023-06-18 09:38:59	0 days 16:07:06	1267.90	0.0005

Table 6.5: Features of the different scans of cluster 12

in what we believe are the different scans. The first two peaks, however, are misconfigured scans as they do not target the whole telescope. Even grouped together, they would have a high Jaccard Index, so we know that they should have been separate scans. The last peak also has some noise around them which contributes to a non-zero Jaccard index. So let's use scan 2 and 3 for our calculation. We see that each host in the scan targets around 1252.5 destinations on our telescope. We know from chapter 4 that we on average 62883 active observers. The estimated number of hosts that this scan should have would therefore be:

$$E[H] = \frac{|O|}{\mu} = \frac{62883}{1252.5} = 50.2 \tag{6.1}$$

Which corresponds to the number of found sources in this scan, as it is also 50. We can therefore determine that we have found the complete group.

6.3.2. Slow Sweeping port scanner

The next category of scanners is the one that we have encountered the most. These scanners will do a complete scan of a single port and after that scan is finished they will go to the next port. An example of this is shown in figure 6.7. This cluster is using DIGITALOCEAN to scan ports 3000, 3790, 5000, 7443 and 22533 over 3 days originating from different countries. When we consider the default ZMap source code then we can conclude that each port is its own scan. In table 6.6 we have given the mean number of destinations visited per port. We see that ports 3000, 3790 and 5000 have the exact same number of destinations and the other ports have fewer. This also corresponds to figure 6.7. We consider ports 3000, 3790 and 5000 as completed scans. With equation 5.15 we can estimate the number of hosts the scan needs to have to scan the complete telescope. During this period our telescope had 62883 active observers. This will give us $E[H] = \frac{|O|}{\mu} = \frac{62883}{6287.9} = 10.00$, this is the same as the actual number of sources

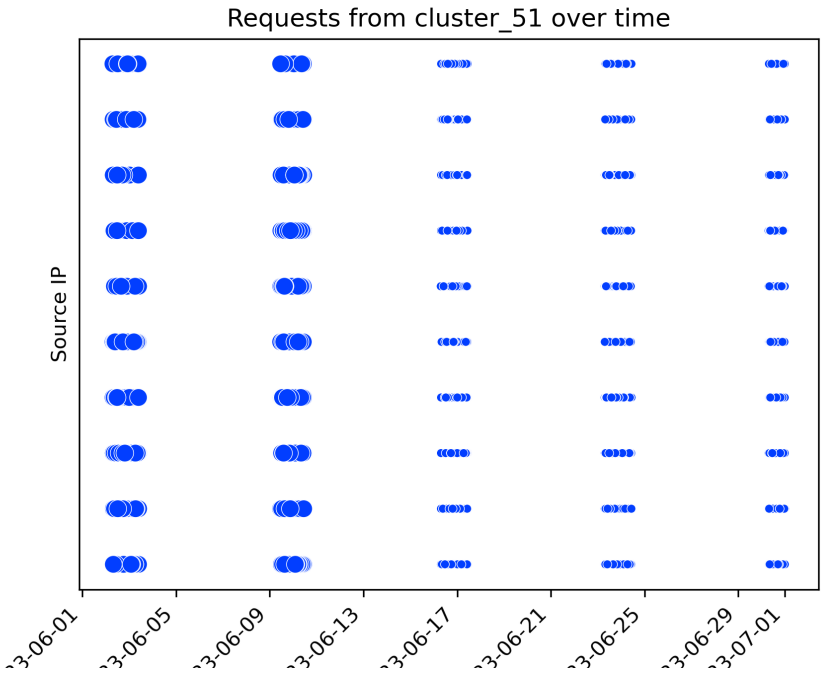

Figure 6.5: Periodic single port scanner from a single subnet

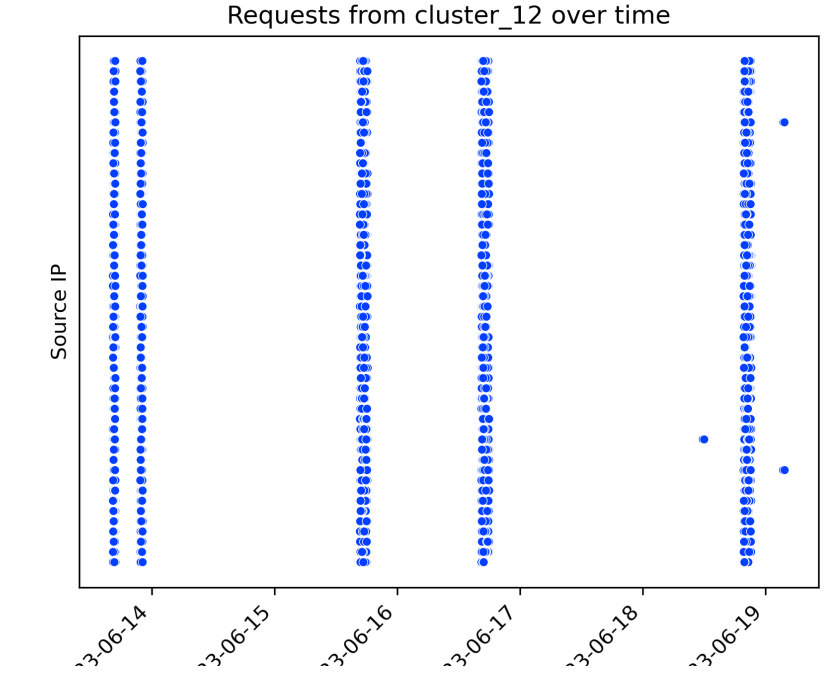

Figure 6.6: Periodic single port scanner form different subnets

DstPort	DstIP
3000	6287.9
3790	6287.9
5000	6287.9
7443	250.0
22533	541.1

Table 6.6: The mean number of destinations visited per port scan of cluster 16

Figure 6.7: A slow sweeping port scanner. Scanning ports 3000, 3790, 5000, 7443 and 22533 over 3 days.

that the clustering algorithm has found so we can say with certainty that we have clustered the complete group.

6.3.3. Periodic updating port scanner

This category of port scanners will not perform complete scans but will update their table periodically. An example of this type of scanner is shown in figure 6.8. In this figure, it is hard to see how many different ports the cluster scans because of the scale of the x-axis. This cluster in particular scans a handful of different ports but it mostly monitors 443 and 8443. These ports will always be scanned by each source in every active period. To better understand what this cluster does we can look at each active period and we will also only look at the activity on port 443. In table 6.7 we have put the start time, duration, number of destinations and Jaccard index for each active period. Here you see that the cluster will always scan for 30-60 minutes starting at around 4:05.

If we look at the number of destinations we notice that they scan around 22000 destinations each scan, which is only around a third of our telescope of 62883 active observers. We have found that each source per scan in this cluster scans around 1000 destinations. Because the Jaccard index is 0 we know that there is zero overlap in the destinations. We have found 22 sources in this cluster and so they scan in total $22 * 1000 = 22000$ destinations each scan. This indicates that we are probably missing several sources from this scan. We can estimate the number of sources in this scan by dividing the total number of observers by the number of destinations each source scans per scan as indicated by equation 5.10.

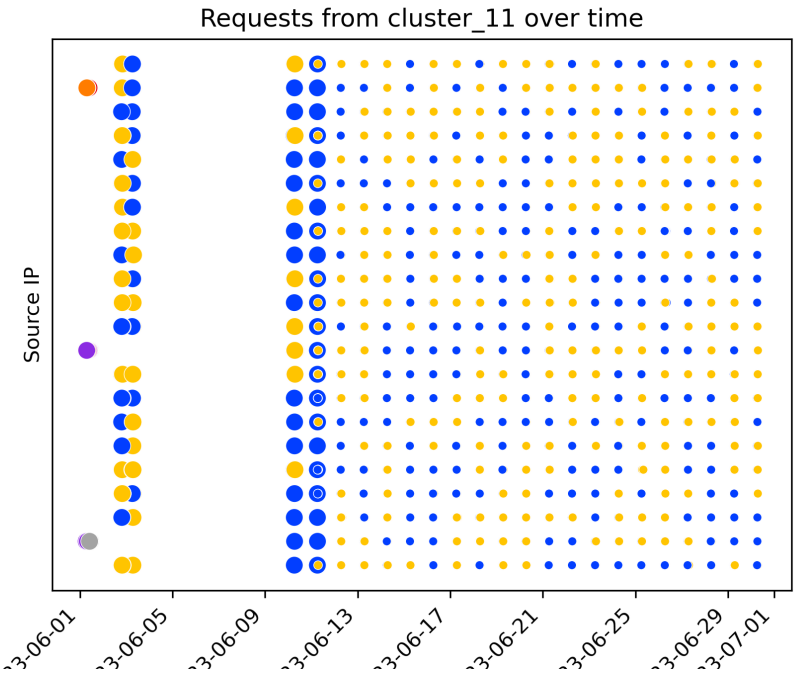

Figure 6.8: A periodic updating port scanner scanning every day for 30 minutes.

	Start Time	Duration	Destinations	Jaccard
0	2023-06-10 04:05:15	0 days 00:31:16	21687	0.0
1	2023-06-11 04:05:17	0 days 01:01:15	21808	0.0
2	2023-06-12 04:05:16	0 days 01:01:53	22161	0.0
3	2023-06-13 04:05:15	0 days 01:01:45	22115	0.0
...	...	...	...	...
18	2023-06-28 04:05:21	0 days 01:02:25	22188	0.0
19	2023-06-29 04:05:16	0 days 00:31:08	21954	0.0
20	2023-06-30 04:05:16	0 days 00:31:11	21786	0.0

Table 6.7: Some active periods of cluster 11.

This will give us an estimation of $62883/1000 \approx 63$ sources belonging to this group. The clustering algorithm has only found 22. This cluster does not belong to the same subnet so we can't use that to find the missing sources that the clustering algorithm missed.

However, because we know the characteristics of all the clusters it will be possible to recover some of these missing sources with filtering. In the case of cluster 11, we know that they will always scan around 1000 destinations each day around 04:00. We queried for this and cross-referenced it with the clusters we already had to confirm that the query could at least return the full cluster. In figure 6.9 we can see some of the recovered sources from the group. We were able to recover 15 more sources bringing the total to 37 which is about half of the expected number of sources. The other missing sources, if they even exist, might have different starting times than the sources we have found. However, upon closer inspection on our network telescope we have not found any additional sources that behave similar to the ones that we have clustered and recovered. We can therefore conclude that this strategy of scanning is used to update their database periodically, without doing a complete scan every period. This would result in an even lower scan rate per host and therefore be stealthy for previous detection systems.

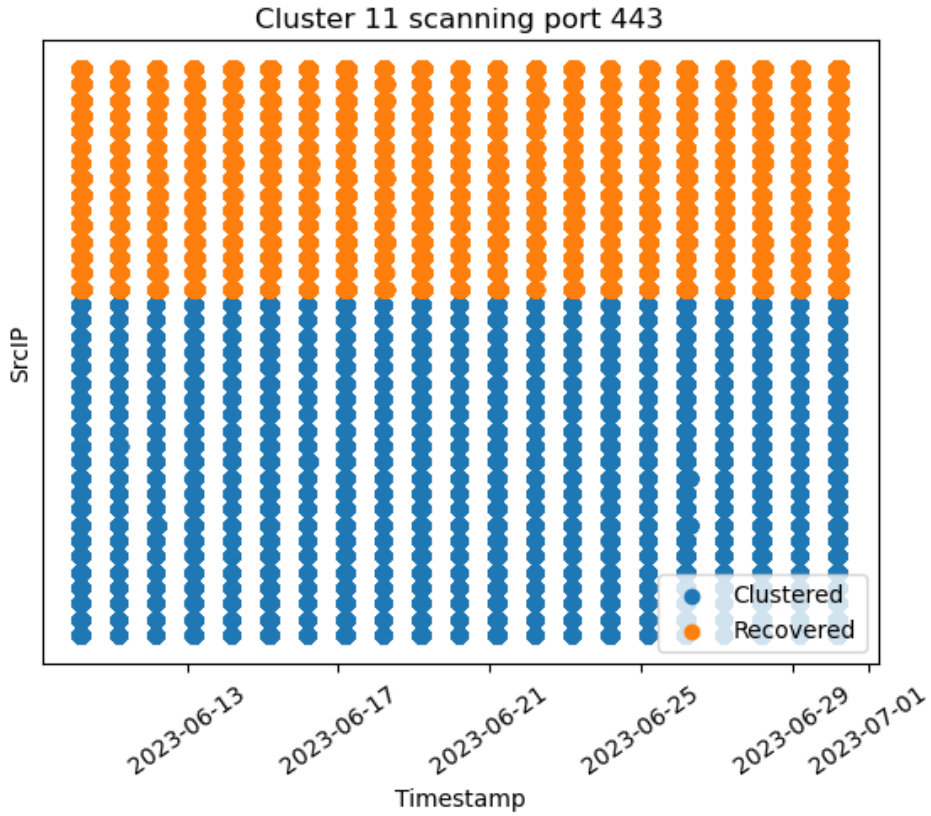

Figure 6.9: Recovered missing sources from cluster 11

6.3.4. Periodic sweeping port scanner

This type of scanner is a combination of the types we have discussed in section 6.3.1 and 6.3.2. Where the scanner will do complete scans of ports and will then be offline for some time before doing another scan. The ones that we have find will focus on a few ports and alternate between them. An example of this type of scanner is shown in figure 6.7. Here we see that this scanner will also focus on a few ports and will do a full port scan every so often. This scans has not a constant period like we have seen in section 6.3.1, making it harder to detect. But even without a composite frequency, the algorithm was still able to detect this cluster. This cluster consists of 252 hosts that each scan on average 249.45 destinations. We can apply equation 5.15 to calculate the estimated number of hosts. This will give us $62883/249.45 \approx 252$ estimated hosts, which corresponds to the actual number of hosts that we have found. The algorithm has therefore found the whole group in the scan. This scan in particular is also on the same class C subnet. We already explained that the total number of scanners in a class C subnet could at maximum be 254. This means that 2 hosts are not actively scanning, these remaining hosts might be used as a controller or master. In table 6.8 we have spliced the different scans and looked at some features of each scan. Here we see that indeed the start time is random but the duration, destinations per host and Jaccard index are always the same. The duration is around 7 hours which means that the telescope will only see a single host with a rate of around 1.7 packets per minute. Furthermore, we have observed that for each scan the unique destinations that each host targets are randomized.

6.3.5. Greedy scanner

The next category of slow scanners that we have detected are greedy scanners. These scanners will do multiple scans at the same time. An example of this category of scanners is shown in figure 6.11. Here we see a scanning group that uses 253 hosts from the same class C subnet to scan the internet. During the month in which we gathered data this cluster performs 151 scans. Some of these scans are shown in table 6.9. We see that a lot of these scans overlap. The hosts in this cluster will use threading to do multiple scans in parallel which would make detection easier as this increases the rate. The seed for the

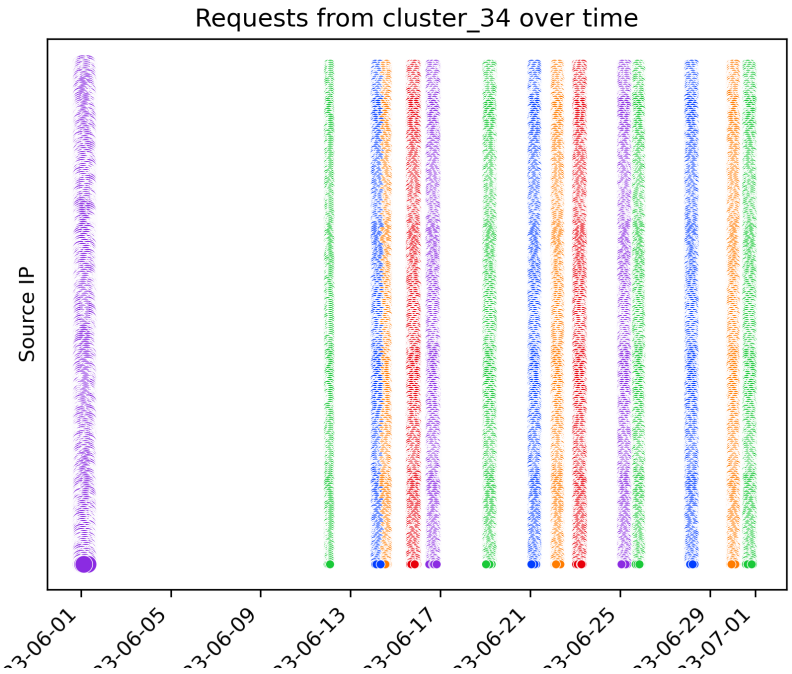

Figure 6.10: Random alternating period scanner

DstPort	Scan	Unique Dest	Start time	Duration	Dest/Host	Jaccard
80	0	62881	2023-06-27 22:16:41	0 days 07:37:05	249.5	0.0
	1	62883	2023-06-13 23:24:05	0 days 07:35:15	249.5	0.0
	2	62877	2023-06-20 22:46:48	0 days 06:58:31	249.5	0.0
443	0	62883	2023-06-21 23:24:23	0 days 06:55:20	249.5	0.0
	1	62883	2023-06-29 19:22:32	0 days 06:35:55	249.5	0.0
	2	62883	2023-06-14 06:59:40	0 days 07:08:47	249.5	0.0
6009	0	62883	2023-06-18 22:37:07	0 days 07:36:22	249.5	0.0
	1	62883	2023-06-30 12:07:15	0 days 07:37:26	249.5	0.0
	2	62883	2023-06-11 21:27:24	0 days 03:27:20	249.5	0.0
	3	62883	2023-06-25 14:18:58	0 days 06:31:31	249.5	0.0
8080	0	62883	2023-06-15 12:59:16	0 days 08:22:32	249.5	0.0
	1	62883	2023-06-22 22:29:13	0 days 08:03:19	249.5	0.0
8443	0	62883	2023-06-24 22:35:37	0 days 07:22:18	249.5	0.0
	1	62569	2023-05-31 22:00:00	0 days 07:53:09	248.3	0.0
	2	62882	2023-06-16 09:58:01	0 days 08:01:18	249.5	0.0

Table 6.8: Features of the different scans of cluster 34.

DstPort	Scan	Unique Dest	Start time	Duration	Dest/Host	Jaccard
80	0	62883	2023-06-23 11:45:04	0 days 01:26:00	248.5	0.0
	1	62854	2023-06-02 18:38:15	0 days 01:13:59	248.4	0.0
	2	62883	2023-06-13 11:24:25	0 days 01:15:02	248.5	0.0
81	0	54027	2023-05-31 22:00:00	0 days 01:03:05	213.5	0.0
	1	62883	2023-06-24 22:29:02	0 days 01:23:00	248.5	0.0
	2	62883	2023-06-09 15:50:44	0 days 01:08:36	248.5	0.0
	3	62883	2023-06-13 16:18:15	0 days 01:13:44	248.5	0.0
...	...	...	...	...	...	...
25463	0	64435	2023-06-24 00:16:03	0 days 01:22:35	254.7	0.0
	1	63139	2023-06-02 19:53:08	0 days 01:09:06	249.6	0.0
25900	0	63138	2023-06-02 10:33:54	0 days 01:09:39	249.6	0.0
	1	62883	2023-06-23 14:36:13	0 days 01:22:35	248.5	0.0
	2	63139	2023-06-03 01:43:37	0 days 01:08:17	249.6	0.0
	3	64435	2023-06-24 07:11:08	0 days 01:22:57	254.7	0.0

Table 6.9: Features of the different scans of cluster 61.

destination assignment of this cluster is also randomized for each scan, so each host would target a greater number of destinations making it less stealthy. That is why we called this type of scanner a greedy scanner, because it uses slow scanning techniques, but because of how greedy the scanner is configured it behaves more like a heavy hitter.

6.3.6. Single Port scanners

We have also noticed a lot of similar activity happening on single ports. Here we will look at all these clusters together because their total activity can be very little. These are all visualized in figure 6.12 where we made a distinction between the clusters. Nearly all these clusters scan port 22 except cluster 18 which scans port 1337.

The features of these clusters are also given in table 6.10. Here we also applied the estimated number of hosts calculation that we got from equation 5.10. We see that for these clusters we also know for sure that cluster 18 is a complete scan because the actual number of hosts corresponds to the number of estimated hosts and the Jaccard index is zero. The total destinations that we have printed here are the total unique destinations that the complete cluster targets. We keep in mind that the total active size of the telescope is 62883. We will also look a bit into cluster 17 which does not have a Jaccard index of 0 but we have noticed in table 6.4 that this cluster has multiple subnets, but upon closer inspection 85% of this cluster is coming from a single organization. If consider the actual group to be this organization and the other 8 founded sources to be coincidences we get a lower Jaccard index of 0.002. We know that this organization's IP range has not targeted our telescope outside of this cluster and every destination is only targeted on average 1.07 times by the whole cluster. We therefore theorize that this organization's total scan is longer than a month and we have not seen the total scan yet. If we isolate the group inside cluster 17 as shown in figure 6.13 we can see that there is a little downtime of 2 hours around the 16th of May. This could be a boundary between the end of one scan and the beginning of another. But the Jaccard index of both halves is still not zero so this line of inactivity might be just downtime of the organization. Still, this line of inactivity is a good indication to differentiate between other sources and to indicate that these sources belong to the same scan.

6.4. Case studies

There are some oddities with the clusters that we have found. In this section, we will go over these oddities.

Figure 6.11: Greedy scanner

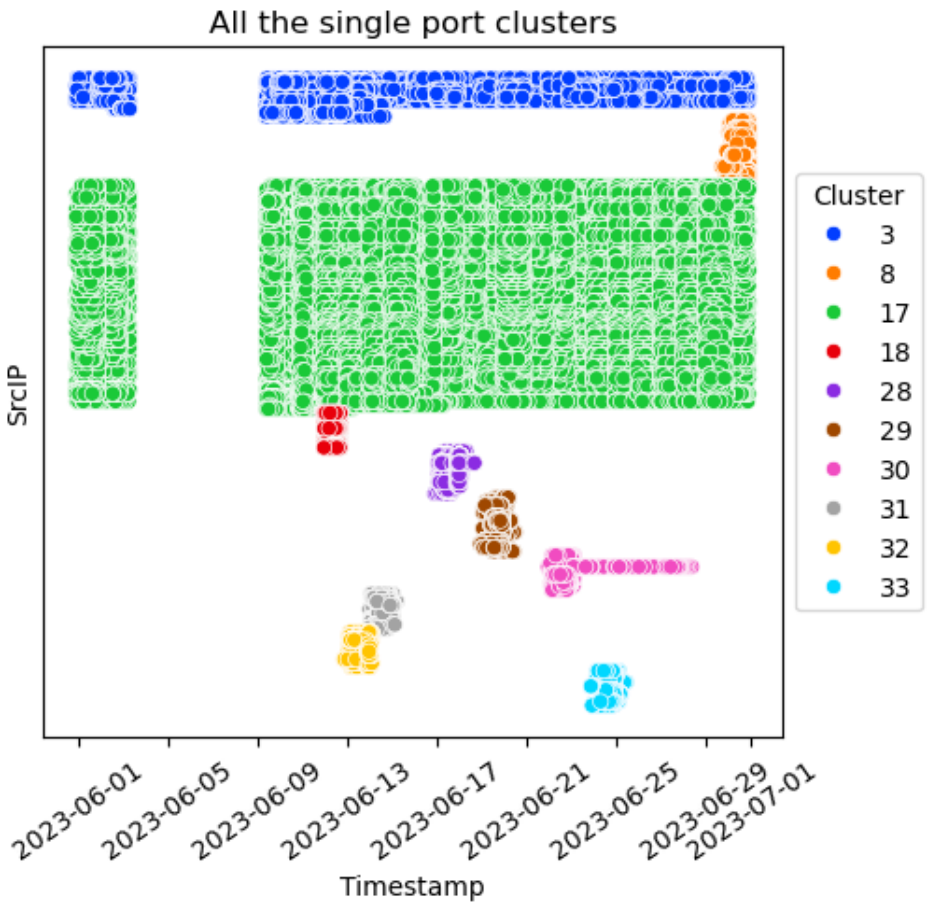

Figure 6.12: All the single port scanners that the clustering algorithm has detected.

Cluster	Total Dst	Hosts	E[Hosts]	Jaccard
3	8475	11	7.62E+01	1.26E-02
8	3018	17	3.46E+02	2.92E-03
17	19081	59	1.74E+02	3.78E-03
18	56722	10	1.11E+01	0.00E+00
28	2186	12	3.39E+02	2.90E-03
29	2570	15	3.59E+02	2.91E-03
30	1765	10	3.52E+02	2.61E-03
31	1969	10	3.14E+02	3.66E-03
32	1989	10	3.11E+02	3.22E-03
33	2335	10	2.65E+02	2.91E-03

Table 6.10: The features of the single port scanners

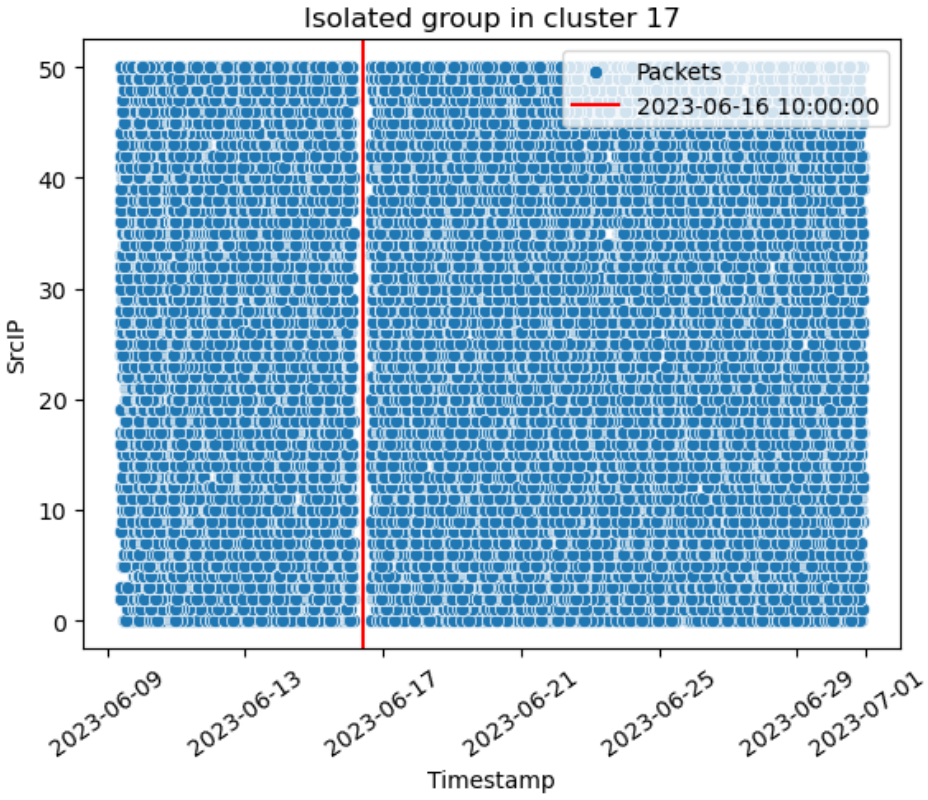

Figure 6.13: Isolated group from cluster 40. The downtime in this group is highlighted.

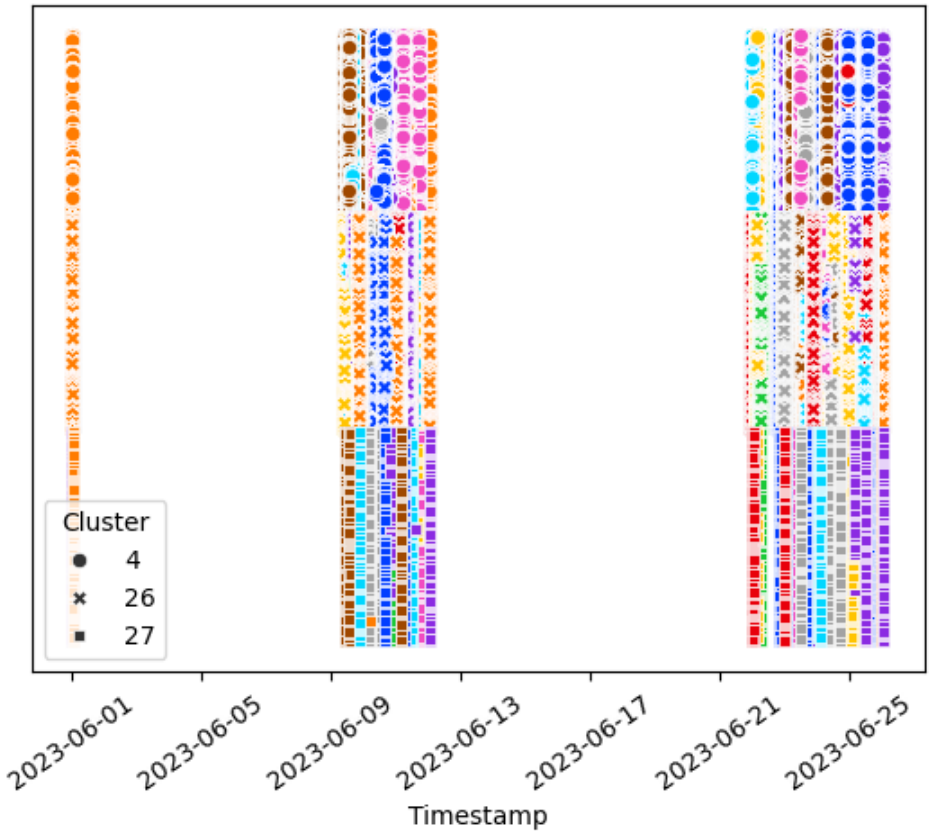

Figure 6.14: Combined clusters 4, 26 and 27 forming a single scanner group.

6.4.1. Split cluster

During our investigations, we have found some clusters that looked similar. Upon further inspection we have found that these different clusters were the same original group. These clusters are 4, 26 and 27. These clusters behave identically to each other as we can roughly see in figure 6.14. Because of the amount of data these combined clusters have it is a little hard to see but the timings here are apparent. The different colors again represent different ports that this group scans. There are also 359 hosts in these combined clusters all scanning on average 175, which again corresponds to our estimation with equation 5.15: $E[H] = \frac{|O|}{\mu}\frac{62883}{175} = 359.3$. So we are pretty sure that these three clusters are in fact originating from the same scans. If we look at table 6.11 we see why the clustering algorithm has grouped these in different clusters. Here we have grouped the sources of each of the clusters and calculated the mean and standard deviation of all the features used in the clustering algorithm. So each value represents the mean or std from all the sources in the cluster. While the start time and number of ports are the same we see that the activity is slightly different for all the clusters and the number of unique destinations is per cluster widely different. The slight differences can be the cause of a few different elements. First, if we look at the hosting providers that this group uses. Cluster 4 uses both "UK-2 Limited" and "Hosting Services Inc", while clusters 26 and 27 use "CARINET". This means that there are some inconsistencies when running a ZMap sharded scan on different types of machines.

6.4.2. Clusters with noise

Because a lot of clusters that we have found are using hosting services some scans are run on machines that have previously been rented to other people. It is in theory therefore possible that machines have been used for other scans before they were used together for the scans that we have detected. This will cause some clusters to have some activity outside of the detected scans. Another reason we could see other activity outside of the clustered scans happens if an adversary who has several hosts runs a test on just a few hosts before he starts scanning. We have gathered a few clusters that have these

DBSCAN	Activity		Period		Start time		#Destinations		#Ports	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
4	1134.63	1.18	1267.82	1.63	43800.00	0.00	38677.92	107.49	105.00	0.00
26	1003.15	2.00	1433.50	2.95	43800.00	0.00	20278.21	81.83	105.00	0.00
27	1114.01	1.87	1291.22	2.19	43800.00	0.00	20680.22	107.77	105.00	0.00

Table 6.11: Assigned features of clusters 4, 26 and 27

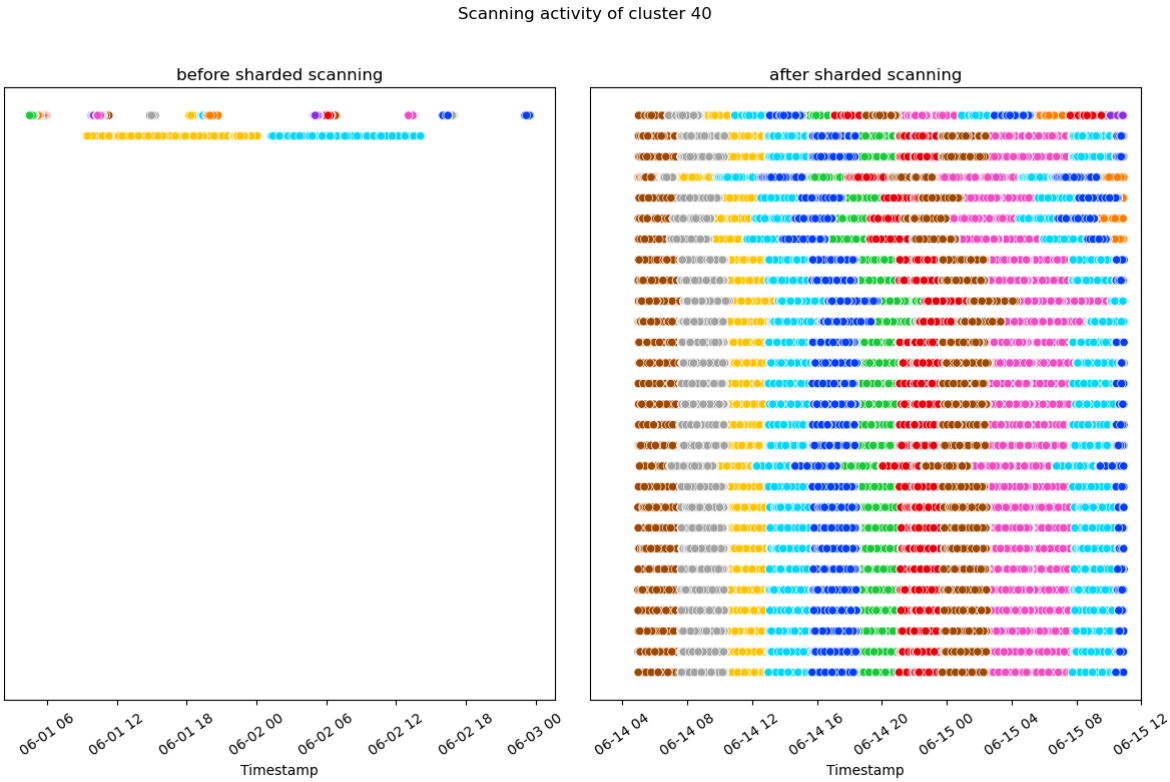

Figure 6.15: A cluster with some noise before distributed scanning.

characteristics. Normally these will fit well inside the different categories we have analyzed in section 6.3. The clustering algorithm is still clustering these types of clusters correctly. One is shown in figure 6.15. We have split the figure over the timestamps before the sharded scanning happens and after. After the sharded scanning we see a sweeping port scan as explained in section 6.3.2. But before this happens 2 hosts in this cluster are already active.

If we look at the first source that is active before the sharded scanning we notice that they have done some scanning over multiple ports over two days. They scan around 259 destinations on our telescope per scan, which means that a complete scan would have an estimated number of hosts of 62883/259 = 242. This host is however clustered by the sharded scanning that happens after this period. There could be two explanations:

1. The host was testing sharded scanning alone before dedicating it to an actual full scan.
2. Because this is a rented server, it could have been part of another group before being part of the clustered one.

In the case of this particular cluster it is because of testing. We can query for other sources that behave similar around the time of testing, searching for other sources that scan the same ports around the same time as this source. This query give us only the source we already had, so we know that the others do not exist and this source was indeed just testing.

Category	Clusters
Noise	3
Single port scanners	10
Sweeping port scanners	34
Periodic single port scanners	3
Periodic sweeping port scanners	4
Periodic updating port scanners	2
Greedy port scanners	6

Table 6.12: The different categories of clusters we have found with our clustering algorithm.

If we look at the sharded scanning part of cluster 40 and use equation 5.15 we can guarantee again that this is close to a complete group as the number of destinations that each host targets is around what we would expect from 28 hosts and the Jaccard Index is zero. So even if some sources would have noise the algorithm is still able to cluster the whole group.

6.5. Categories

In total, the clustering algorithm has found 62 clusters. We have described the different categories of clusters we have in this chapter. The total clusters that we have found for each category can be found in table 6.12. Three of these clusters we have labeled as noise, but even so, we have seen that inside these clusters are multiple scans that we can identify. Furthermore, we see that over half of the clusters we have found belong to the slow-sweeping port scanners category. The other types of scanners are variations on the sweeping port scanner as they also scan multiple ports with the same machines. With this we can categorize every cluster that we have found even the ones that have some noise as discussed in section 6.4.2, which does not hinder the clustering algorithm.

7

Discussion

7.1. Summary

The results indicate that clustering techniques can be applied to classify and identify slow scanners based on their behavior and attributes. By assigning numerical characteristics to different sources that we have observed on our network telescope we can create a dataset to be used by a clustering algorithm. We have used simple known techniques to assign the parameters without using complex calculations for defining an optimal parameter set. With this, the clustering algorithm was able to identify 62 clusters. From these 62 clusters we have identified the correctness per cluster by comparing it with our calculations we have acquired with the help of our simulation and by calculating the Jaccard index for the detected scans. We have seen that most of the clusters are complete scans. There are some clusters that consist of multiple scans or can contain other sources of background radiation, but even inside those clusters, we can identify some distributed scans. Sometimes we have found that the clustering algorithm only clusters a partial scan, but even so, because the characteristics for these scans are known we can still use our network telescope to find the missing sources in those scans. This means that even if the algorithm only finds a part of a scan, it can still be used to help identifying the complete scan. Lastly, we have seen that with or without noise on some sources, the clustering algorithm is still able to correctly identify most of the clusters.

7.2. Interpretations

In our research, we have looked primarily at the question if clustering techniques can be applied to classify and identify slow scanners. We have done some preprocessing and filtering to remove obvious heavy hitters but have not applied other intrusion detection algorithms. In a realistic intrusion detection system where you want to stop all types of scanners, other algorithms are still needed. This is not a replacement for a complete intrusion detection system, it is a method to detect slow scanners that other intrusion detection systems are not able to find. This will help us in understanding the state of the internet better. Furthermore, it will enable us to identify the groups that are scanning instead of just detecting malicious activity per source.

If we cross-reference our findings with free databases we find that we have detected groups that have not been observed by anybody else. Of course, the server hosting IP addresses that we have found are a grey area because we do not have access to the rental information of those providers, we do not know who has rented which server. But if we look at IP addresses not belonging to hosting providers we have some interesting results. For example, we have detected scans coming from government buildings in Argentina that have not been detected by other databases. Other examples include data centers in Mumbai, unknown organizations around the world and Universities, which all scan while there is no trace on databases like AbuseIPDB and Greynoise.

7.3. Implications

The features we have chosen are based on our findings from the simulation, our own experience with using scanning tools and the behavior described by the original developers [12]. These features have proven effective in identifying scanning groups, which was the primary goal of our research. Additionally, all the different features we discussed contribute to the clustering algorithm in various ways. Notably, the offset and the number of ports consistently play a significant role in the clustering algorithm, as these parameters rarely change for each scan. However, sometimes the offset might vary a bit due to jitter or faulty configurations. Additionally, the number of ports can also vary due to configuration errors or some hosts in a clustered scan also being used by other scans. This can happen if an adversary is testing or if a rented server was rented earlier for another scan. Because we have seen that you can not always trust on a single feature, we have assigned multiple. The other features could all help the clustering algorithm in different ways. So we have seen that even if some features are telling if a scan is performed by the same group, we still need multiple features to get a better estimation. That being said, some other features that we have not discussed might also help the clustering algorithm. It might be interesting to see if more features can be assigned to each source. In our research, we have focused on ZMap but other scanning tools can have extra characteristics depending on their configuration that can be used for the clustering algorithm.

The method proposed in our research can be applied to most network telescopes to detect scanning groups. For administrators, this can be used to defend against potential attacks before they happen. However, not every network administrator has access to enough IP addresses to build a network telescope. Then there is still a need to differentiate between user data and scans on their own network. This is possible and has been researched before [35, 5, 6, 32, 21], but this is very hard to do for slow scanners. Still, the insight and detection of slow scanners benefit administrators if they are able to use a network telescope, either through their own network or hired services.

For researchers, our method has many applications. By detecting more scans we can build a bigger dataset and understand the state of the internet better. If adversaries will not use heavy hitters anymore and purely focus on slow scanners that are not able to be detected with current intrusion detection systems, then we will have a harder time researching internet wide events. As they will not be seen completely or even missed. Our method will be able to fill in those gaps. Furthermore, our method can also help in future research because it can help in building a bigger and more complete dataset. Especially in increasing datasets to detect slow scanners. Machine learning for example has been researched in the past to be able to detect DDOS attacks and port scans [1]. But to train a machine learning model you need to have a good dataset.

7.4. Limitations

Our method has some limitations that we will discuss in this section. The first part of the limitations are originating from our design progress. We originally used only ZMap data to design our methodology, this was done so we had a better understanding of our dataset. We knew that most of the packets were part of a scan so there was little to no need for preprocessing or filtering our dataset. There would be little to no other sources of background radiation apart from scans. This made assigning the parameters used in the clustering algorithm easier. We made this decision so we could focus on our research question. We already discussed that there is little to no previous research done on identifying and classifying groups of slow scanners with clustering. We first had to prove that it was possible and now we have proven that it is. Because of this, we have not done any optimizations of the clustering algorithm itself or assigned parameters outside of the knee method. There are probably more slow scans inside our database that have been missed because we did not optimize.

Currently, we tested the clustering algorithm only on obvious ZMap packets by looking at the identification header in all the packets as ZMap sets this to 54321 by default. As people change this in the source code we missed some scans. In theory, this should not be an issue and the method should still be able to identify slow scanner groups. However, some steps need more attention to make this work. The first is preprocessing, where we will filter out as much background radiation that is not associated with scanning as possible to make the clustering easier. When that is done, new parameters could be assessed and assigned. Due to time constraints for our research, we only got to do some clustering and got good results, but we did not have enough time for a full analysis.

Another limitation of our method is that it is not real-time. This is not an issue if our method is used for research purposes, but can be a problem in preventing scans as they happen. It is possible to use the clustering technique on a data stream [7] but more research is needed.

7.5. Future research

We recommend for future research to look at our current limitations. The first thing that should be a follow-up of our research would be the clustering of the complete dataset without the filtering of obvious ZMap scans. Most of the scanning traffic in the current state of the internet is coming from ZMap anyway so this should be an obvious next step.

Another recommendation for future research would be to look at alternative features, or even look at the weights of each of the features. We have seen that the start time and number of ports have been constant most of the time for the different scans and frequency can be a bit more variable. Changing the weights of these features might make the algorithm perform better.

With our method is it possible to identify sharded scans and in particular, multiple sharded scans from the same set of hosts. A lot of these scans are however on rented servers. This means that the same group can potentially use different rented servers every time they want to perform a scan. An interesting follow-up research would be to actually track these groups. They might use the same configuration every time they scan but with different servers. The characteristics of these scans might still be the same every time. For example same number of hosts per scan and same frequency or maybe they always sweep the same 3 ports.

8

Conclusion

In our research, we have shown that clustering techniques can be applied to classify and identify slow scanners based on their behavior and attributes. By looking at the characteristics of ZMap sharded scans and assigning numeric features to them to be used in clustering algorithms, it is possible to identify complete sharded scans without looking at the origin of the source address. This makes it possible to identify slow, distributed and decentralized scans.

We have achieved this by looking at the different sources we have seen in our network telescope and assigning features to all of them. These features were chosen based on observations of simulated scans and calculations on how distributed scans should perform. These features could then be scaled to be used in a spatial clustering algorithm. In our case, we used DBSCAN to cluster the preprocessed data. We used simple known methods for assigning the parameters for the clustering algorithm that does not need calculations and these performed already well enough to cluster 62 groups.

We have found that most of these clusters are indeed sharded (or distributed) scans. We have checked this by referring back to our calculations on how a sharded scan should perform. Mainly by looking at the different ports and the number of destinations each host should have in a single scan. The mean Jaccard Index for each scan was also used to check if the alleged scan was indeed a single scan. For each source in a scan, the Jaccard Index was calculated with the variable being the destinations it targets compared to other scans. When it happened that the clustering algorithm only clustered a partial scan, we showed that because we then knew the characteristics of the observed part of the scan, we could identify the remainder of the sources.

Out of the different scans that we have found we noticed some similar scanning strategies. We have categorized them and described their behavior. This can help in understanding the state of the internet better and see how current scanning strategies are applied.

Our research will help network security administrators to help defend their networks against potential attacks by identifying scans early. Furthermore, our research can be a good basis for future research on slow scanners. Either by optimizing different parts of our method like the features, the clustering parameters or even the clustering algorithm itself. Even without optimization, our method can be used to build datasets for other research. Not only in researching the state of the internet and slow scanners but also in training machine learning models.

References

[1] Muhammad Aamir et al. "Machine Learning Classification of Port Scanning and DDoS Attacks: A Comparative Analysis". In: *Mehran University Research Journal of Engineering and Technology* 40.1 (2021), pp. 215–229. ISSN: 02547821. DOI: 10.22581/muet1982.2101.19.

[2] David Adrian et al. "Zippier ZMap : Internet-Wide Scanning at 10 Gbps". In: ().

[3] Mark Allman, Vern Paxson, and Jeff Terrell. "A brief history of scanning". In: *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC* (2007), pp. 77–82. DOI: 10.1145/1298306.1298316.

[4] Richard J. Barnett and Barry Irwin. "Towards a taxonomy of network scanning techniques". In: *ACM International Conference Proceeding Series* 338 (2008), pp. 1–7. DOI: 10.1145/1456659.1456660.

[5] Ran Ben Basat et al. "Network-wide routing-oblivious heavy hitters". In: *ANCS 2018 - Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems* (2018), pp. 66–73. DOI: 10.1145/3230718.3230729.

[6] Celyn Birkinshaw, Elpida Rouka, and Vassilios G. Vassilakis. "Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks". In: *Journal of Network and Computer Applications* 136.February (2019), pp. 71–85. ISSN: 10958592. DOI: 10.1016/j.jnca.2019.03.005. URL: https://doi.org/10.1016/j.jnca.2019.03.005.

[7] Yixin Chen and Li Tu. "Density-based clustering for real-time stream data". In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007), pp. 133–142. DOI: 10.1145/1281192.1281210.

[8] Mehiar Dabbagh et al. "Slow Port Scanning Detection". In: *2011 7th International Conference on Information Assurance and Security (IAS)* (2011), pp. 228–233. DOI: 10.1109/ISIAS.2011.6122824.

[9] M. Daszykowski and B. Walczak. "2.26 - Density-Based Clustering Methods". In: *Comprehensive Chemometrics: Chemical and Biochemical Data Analysis, Second Edition: Four Volume Set* 2 (1996), pp. 565–580. DOI: 10.1016/B978-0-444-64165-6.03005-6.

[10] Marco De Vivo et al. "A review of port scanning techniques". In: *Computer Communication Review* 29.2 (1999), pp. 41–48. ISSN: 01464833. DOI: 10.1145/505733.505737.

[11] Zakir Durumeric, Michael Bailey, and J. Alex Halderman. "An internet-wide view of internet-wide scanning". In: *Proceedings of the 23rd USENIX Security Symposium* (2014), pp. 65–78.

[12] Zakir Durumeric, Eric Wustrow, and Alex Halderman J. "ZMap: fast internet-wide scanning and its security applications". In: *22nd USENIX Security Symposium (USENIX Security 13)* 13.13 (July 2013), pp. 605–620.

[13] Martin Ester et al. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. Tech. rep. 1996. URL: www.aaai.org.

[14] Carlos R. García-Alonso, Leonor M. Pérez-Naranjo, and Juan C. Fernández-Caballero. "Multi-objective evolutionary algorithms to identify highly autocorrelated areas: The case of spatial distribution in financially compromised farms". In: *Annals of Operations Research* 219.1 (2014), pp. 187–202. ISSN: 15729338. DOI: 10.1007/s10479-011-0841-3.

[15] Vincent Ghiëtte, Norbert Blenn, and Christian Doerr. "Remote identification of port scan toolchains". In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2016* May (2016). DOI: 10.1109/NTMS.2016.7792471.

[16] Harm Griffioen and Christian Doerr. "Discovering Collaboration: Unveiling Slow, Distributed Scanners based on Common Header Field Patterns". In: *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020* (2020). DOI: 10.1109/NOMS47738.2020.9110444.

[17] Félix Iglesias and Tanja Zseby. "Pattern Discovery in Internet Background Radiation". In: *IEEE Transactions on Big Data* 5.4 (2019), pp. 467–480. ISSN: 23327790. DOI: 10.1109/TBDATA.2017.2723893.

[18] Yingqiu Liu, Wei Li, and Yunchun Li. "Network Traffic Classification Using K-means Clustering". In: (2008), pp. 360–365. DOI: 10.1109/imsccs.2007.52.

[19] Johan Mazel and Rémi Strullu. "Identifying and characterizing ZMap scans: a cryptanalytic approach". In: (2019). arXiv: 1908.04193. URL: http://arxiv.org/abs/1908.04193.

[20] David Moore et al. "Network telescopes: Technical report". In: *CAIDA, April* (2004), pp. 1–14. URL: http://www.caida.org/publications/papers/2004/tr-2004-04/tr-2004-04.pdf.

[21] Mehr Nisa and Kashif Kifayat. "Detection of Slow Port Scanning Attacks". In: (2020). DOI: 10.1109/ICCWS48432.2020.9292389.

[22] Stijn Pletinckx and Vincent Ghi. "Classification of Distributed Strategies for Port Scan Reconnaissance". In: ().

[23] *RFC 9293 - transmission control protocol.* https://datatracker.ietf.org/doc/html/rfc9293. 01-08-2022.

[24] Philipp Richter and Arthur Berger. "Scanning the scanners: Sensing the internet from a massively distributed network telescope". In: *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC* (2019), pp. 144–157. DOI: 10.1145/3355369.3355595.

[25] S. Robertson et al. "Surveillance detection in high bandwidth environments". In: *Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2003* 1 (2003), pp. 130–138. DOI: 10.1109/DISCEX.2003.1194879.

[26] Fatin Hazirah Roslan. "A Comparative Performance of Port Scanning Techniques". In: *Journal of Soft Computing and Data Mining* 4.2 (2023), pp. 43–51. ISSN: 2716621X. DOI: 10.30880/jscdm.2023.04.02.004.

[27] E. S. Sagatov et al. "Proactive Detection for Countermeasures on Port Scanning based Attacks". In: *Proceedings of the 2021 17th International Conference on Network and Service Management: Smart Management for Future Networks and Services, CNSM 2021* (2021), pp. 402–406. DOI: 10.23919/CNSM52442.2021.9615577.

[28] Jörg Sander et al. "Density-Based Clustering in Spatial Databases : The Algorithm GDBSCAN and Its Applications". In: *Springer* 2 (1998), pp. 169–194.

[29] Erich Schubert et al. "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN". In: *ACM Transactions on Database Systems* 42.3 (2017). ISSN: 15574644. DOI: 10.1145/3068335.

[30] *Snort intrusion detection and prevention system.* https://www.snort.org/. 2019.

[31] Artur Starczewski, Piotr Goetzen, and Meng Joo Er. "A New Method for Automatic Determining of the DBSCAN Parameters". In: *Journal of Artificial Intelligence and Soft Computing Research* 10.3 (2020), pp. 209–221. ISSN: 24496499. DOI: 10.2478/jaiscr-2020-0014.

[32] Eric D. Vugrin et al. "Cyber threat modeling and validation: Port scanning and detection". In: *ACM International Conference Proceeding Series* (2020), pp. 42–51. DOI: 10.1145/3384217.3385626.

[33] Jie Wang et al. "Clustering analysis for malicious network traffic". In: *IEEE International Conference on Communications* (2017), pp. 1–6. ISSN: 15503607. DOI: 10.1109/ICC.2017.7997375.

[34] Rui Xu and Donald Wunsch. "Survey of clustering algorithms". In: *IEEE Transactions on Neural Networks* 16.3 (2005), pp. 645–678. ISSN: 10459227. DOI: 10.1109/TNN.2005.845141.

[35] Liang Yang, Bryan Ng, and Winston K.G. Seah. "Heavy hitter detection and identification in software defined networking". In: *2016 25th International Conference on Computer Communications and Networks, ICCCN 2016* (2016). DOI: 10.1109/ICCCN.2016.7568527.

[36] Maxim Zolotykh. "Comprehensive Classification of Internet Background Noise". In: *Proceedings - 2020 Global Smart Industry Conference, GloSIC 2020* (2020), pp. 35–41. DOI: 10.1109/GloSIC50886.2020.9267850.