Opening Pandora's Box

Charting the ecosystem of Command and Control infrastructures in a terabit-scale network

T.M. Booij

Faculty of Electrical Engineering,
Mathematics & Computer Science

TU Delft

Opening Pandora's Box

Charting the ecosystem of Command and Control infrastructures in a terabit-scale network

by

T.M. Booij

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday September 27, 2019 at 14:00 PM.

Student number:	4221710	
Project duration:	July 1, 2018 – September 27, 2019	
Thesis committee:	Dr. ir. J. C. A. van der Lubbe,	TU Delft, committee chair
	Dr. C. Doerr,	TU Delft, supervisor
	Dr. D. Tax,	TU Delft, committee member

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

TU Delft

Preface

What a journey it has been. Before you lies my thesis, the result of more than a year's of hard work and the concluding part of my master's degree in Computer Science. The final part of my study has proven to be the most enjoyable. During the masters program I have made a couple of close friends and found my passion for cybersecurity.

This research could not have been possible without the help and support of a number of people. First and foremost I would like to thank my thesis supervisor, Christian Doerr, from all the enthusiastic and inspirational meetings to the critical review sessions. Working on three papers together has taught me a great deal of important academical and personal skills. You have been a true inspiration for me the past year. I also would like to thank my other supervisor, Harm Griffioen, for the great number of sparring sessions and laughs we had during this period. Furthermore I would like to express my gratitude to the members of my thesis committee, Jan van der Lubbe and David Tax, for taking the time to attend my presentation and review this work.

Most importantly I would like to thank my friends and family without whom it would not have been possible to finish this thesis. I especially want to thank Hugo Bijmans for all the great moments we have shared during this period, underlining our collaboration by writing two accepted papers. Additionally I want to thank Lars van de Kamp, Wilko Meijer, Rico Tubbing for their brainstorming sessions and providing a listening ear. A special thanks to my girlfriend, Eva Zillen, and my parents, Theo and Marjan, for not only standing by me during every step, but also who have always encouraged me and dragged me through the roughest times. And finally I thank you, the reader, for taking the time to pick up and read my work.

T.M. Booij
Delft, September 2019

Abstract

The amount of people and devices connected through the Internet has been growing at a rapid pace; as of June 2019 58,8% of the world's population and billions of devices are joined by this vast network of information resources and services. Not every Internet user however has benign intentions. Cybercriminals use this technology for their own personal gain, by creating malicious software with the objective to compromise and even control the devices of unsuspecting victims. After devices have been infected with malicious software, they will be controlled by a central networking infrastructure, or Command and Control (C&C). Numerous studies have developed detection methods to find the commanding servers behind these attacks – which is important for locally implemented anti-virus software – but the infrastructures behind these attacks is only uncovered after these malicious servers have been taken down. We have collaborated with one of the largest Tier 1 Internet Service Providers and have collected ~4Tb of global NetFlow traffic consisting of daily connections made worldwide, giving us the possibility to analyze malicious infrastructures from a new perspective. This dataset allowed us to evaluate one of the most promising data sources to uncover and analyze adversarial C&C infrastructures; open source cyber threat intelligence. After having introduced a taxonomy to evaluate this intelligence, and having assessed the defensive advantages users will gain when adopting these information feeds, we came to the conclusion that – even though advertised differently – this intelligence only provides a small fragment of the total picture. For this reason we have decomposed Internet networking infrastructures into three categories in which Internet devices can be divided, computer *clients*, Internet of Things (or *IoT*) and *routers*, and use a machine learning driven methodology to paint the threat landscape of the tactics, techniques and procedures adversaries employ within these three categories. When looking into infections on clients, a numerous amount of different structures can be observed which cybercriminals use to hide their infrastructure like round-robin DNS or interesting server hopping sequences. When learning these patterns, we successfully use a combination of machine learning and statistical functions to complement threat intelligence feeds. Where these feeds during the course of 2018 only show 1,105 malicious servers, we find that there are more than 188,000 malicious servers in our dataset. A more recent addition to the Internet are smart, or IoT, devices like surveillance cameras, which have proven to be incredibly vulnerable. In an analysis we report on the scale and impact of the first IoT based malware, Mirai, and its variants during the start of 2018. 1 in every 2,345 scanned and bruteforced devices is successfully infected with a Mirai variant. This poses a great attack vector, taking into account that there are about 7 billion IoT devices as of 2019. For our last analysis we have focused on a firmware vulnerability towards MikroTik routers which cybercriminals have exploited to rewrite outgoing user traffic and embed cryptomining code in every outgoing connection. Accordingly, for every web page visited they can use the computation power of the victims computer to mine for the cryptocurrency Monero. We report on the tactics, techniques and procedures, and coordinating infrastructure of the adversaries, which had control of up to 1.4M routers over a period of 10 months, which is approximately 70% of all global MikroTik devices. Our work shows that an entire world of possibilities emerge in terms of network security when able to analyze NetFlow data. In the ongoing battle against cybercrime, anti-virus companies try to outsmart adversaries by using novel device based detection techniques. This is an evident rat-race between defenders and attackers. We have shown that ISP providers could play a big role in this, by analyzing NetFlow data flowing through their routers to perform detection of malicious behavior based on previous misuse and cyber threat intelligence.

T.M. Booij
Delft, September 2019

List of Figures

List of Tables

List of Listings

Contents

1 Introduction

Our generation cannot imagine a world without the Internet. With the Internet's growing role in coordinating the structure of our daily lives, we heavily depend on it. People have grown a need for being interconnected and alleviating tasks by automation. Filing for task returns, signing binding legal contracts or simply posting a new photo on a social media website is all within the click of a button. In this day and age, with our reliance on modern technologies such as personal computers and mobile phones, combined with the ease of criminals getting their hands on malicious software, or *malware*, we have surpassed everything society could have envisioned a couple of decades ago. With the global proliferation of cybercrime, we virtually broadcast our digital presence around the world, waiting to be exploited and used by criminals or competing companies in any way they like.

Cybercriminals are individuals or teams of people who use these technologies to commit malicious activities on digital systems or networks with the intention of stealing sensitive information, generating monetary profit or crippling these systems. We can wonder why digital security has become a major issue over the past years - communication and computer networks have been around quite some time. Cybercrime can be dated back to the 1970's [123]. In 1971 John Draper, a phone phreaker, used a toy whistle from a cereal box to break into the AT&T phone system, allowing him to make free long-distance calls. Not much later Ian Murphy, AKA Captain Zap, was the first person to be actually convicted of a cybercrime. Again telecommunications was the target, as Murphy hacked into the AT&T network and changed internal clocks. This allowed him to pay off hour costs at peak times when making phone calls, saving him a significant amount of money. Over the last 15 years, cybercriminals have become incredibly advanced and have created an ecosystem sustaining malicious activities. It used to take great skill to, for example, steal someone's credit card information to perform identity fraud. This has however become extremely simple with the introduction of e-commerce, as this enables every online user to be a potential target to this kind of fraud. The introduction of these kind of technologies incentivized adversaries to adopt and misuse new technologies to perform large and global malicious campaigns. Nowadays, the amount of impact cybercrime has on society is enormous as predictions have shown that the resulting damage will globally cost $6 trillion annually [66]. Writing a sophisticated piece of malware, which takes an incredible amount of skill, rarely proves to be effective for monetary gain. Instead, cybercriminals focus on new technologies as it became much more lucrative to exploit these technologies with emerging vulnerabilities, maximizing profit with minimal effort. They are continuously searching for innovative opportunities to achieve their malicious goals, including developing and utilizing new forms of malware. A wide variety of cybercriminals exist, each with different incentives and malicious goals. From individual, and most of the time inexperienced, criminals who mainly rely on pre-made malware from third parties, to organized cybercrime, which have the possibilities to run elaborated malicious infrastructures and perform highly sophisticated attacks.

1.1. Organized cybercrime

Cybercrime has been on the rise for quite a while now, with no sign of slowing down any time soon. Criminals have been able to explore different exploitation techniques for monetary or political gain. They have also realized that it can be very effective to combine skills of like-minded people. This has even allowed cybercriminals to run their operations like agile and modern businesses. Illicit cyber organizations which employ

people with varying skills, are not unheard of. From managers to programmers, from network administrators to data miners, combining these kinds of expertise to perform harmful activities can be very effective and lucrative. Not only the organizational structure, but also the approach to perform and maintain misuse can be very elaborate. The opportunities of these coordinated structures has led to emerging underground markets, used to offer illegal services or sell new vulnerabilities for high prices.

One of the most innovative ways of performing malicious activities is by exploiting new, or reusing old, vulnerabilities to compromise devices and subsequently gaining control. When performing this on a single computer, not much harm can be done. The owner of the system might have fallen victim to, for example fraud or identity theft, and thus only causing harm to the owner. However, imagine this single controlled device turns into a network of thousands of compromised devices. The options of large scale abuse for the administrator controlling an infrastructure this large immediately explode. Such a network is called a *botnet*, consisting of a lot of compromised devices, or *bots*. These botnets can be used to roll-out extensive malicious campaigns with all sorts of attack vectors in their grasp for monetary gain or for pursuing other goals. The attack vectors can be categorized in a lot of different activities. Spyware, which can send information, like passwords, credit card information or other information, to its administrators about the user controlling the infected device. The information these kinds of software gather can for example be sold in underground markets, enabling individual cybercriminals to use this kind of information without having any kind of technological skills. An old favorite, SPAM e-mails, are designed to trick the users into clicking on links or downloading malicious software, or click fraud, which occurs when the user visits website to create false traffic for commercial gain. Over the past years ransomware, or scareware, has been blooming by trying to extract money from the infected victims, as well something called cryptojacking, using the combined computational power of the infected machines to acquire cryptocurrencies. There are more of these attack vectors, each having its unique niche or method to exploit infected devices, but the most predominant use of botnets currently still is distributed denial-of-service, or *DDoS* attacks. In such an attack, a set of infected machines is used to send a lot of requests towards a service, forcing it to overload and preventing legitimate users to access these services. As can be imagined, the goal of taking something off the grid is not directly used to steal data or money, however, these attacks are not going away any time soon as adversaries can indirectly make a lot of money from them. The power of having such an infrastructure could provide opportunities of extortion, by threatening large companies with DDoS attacks if they do not pay them. This can be lucrative, but technological advantages have provided more options for cybercriminals to utilize their network. After the introduction of Infrastructure-, Platform- and Software as a Service in the cloud, adversaries have adopted this idea and started to rent parts of their botnets, creating DDoS as a Service. By doing this, the incentive to infect more machines grew. These services, also called stressers or booters, lowered the technological barrier to perform DDoS attacks as any user can anonymously sign-up and perform attacks for just a few dollars. It turns out, not much is needed to actually rent such a botnet. Sometimes users can even pay with well-known payment methods like PayPal. All what then is left is ill-will towards someone or a company and the willingness to break the law. It may sound strange, but just about anybody can launch an attack and cripple important infrastructure. Early 2018, an 18-year old Dutchman was arrested on suspicion of being responsible for DDoS attacks on the Rabobank, the Tax Authority and a couple of other websites. All he wanted to do was *"to show that a teenager can crash all banks with a relative simple attack"* as he wrote in an e-mail towards a newspaper. This proved that to locate one of these booters or stressers, you do not have to be skilled with any kind of malicious technologies or search underground marketplaces. As it turns out, a simple Google search is enough.

One component all infected devices have in common; in order to be effective, they have to acquire commands to perform these malicious activities. These commands are spread by means of one or multiple Command and Control, or *C&C*, server(s). The specific function of such a server can vary tremendously, but the goal is always to effectively control the devices which have been infected by some kind of malware the cybercriminals have employed. The topology of these servers has also changed over the years. In the early years of botnets, they were mainly centralized servers controlling all the infected devices from one machine, but as technology has evolved so have the hiding techniques behind these servers. Instead of having one central server to control all the infected machines, multiple servers can be used to perform load balancing or short-lived cloud-based servers can be used to leave no trace of the commands they have sent. Early 2018, the social media platform Twitter has even been used as a channel to provide new commands to infected machines [138]. These compromised devices were programmed to watch the updates of a specific Twitter feed and interpreting the posted information to perform attacks. Ultimately, cybercriminals can be extremely creative in setting up these malicious infrastructures.

1.2. Defensive measures

Cyber security naturally is an adversarial discipline; on each part of the spectrum, people are trying to find vulnerabilities and test limitations of the 'opponents' technology. On one side, attackers are trying to exploit vulnerabilities, find configurational errors and essentially finding loopholes in the security of systems, and on the other side defenders are trying to stay one step ahead of them by continuously improving the flaws in their systems. This division can be seen throughout the entire cyber threat landscape. Most of the time, ordinary computer users are often not aware of this and rely on the services of their antivirus provider. Companies even have multiple levels of defenses in place; intrusion detection systems, demilitarized zones, spam traps or other mechanisms. Again cybercriminals keep trying to adapt and keep hidden from these defenses. Security researchers counter this by focusing on more advanced and novel detection strategies.

Detecting malicious activity most of the time happens on host level. Antivirus companies design software which can detect when bad activity is taking place on the computer it is installed on. This industry has been around and necessary ever since the appearance of the first computer virus called the *Creeper virus*. During the years, as technology has improved, detection strategies have also improved. Traditional customer grade software relies heavily on signature based detection. When a new type of malware is found it is analyzed by antivirus companies, who subsequently create a fingerprint or signature of this specific malware. These signatures can be used to identify malware and stop them before they can cause harm. The moment a system tries to execute a program, its signature is compared with those of malware stored in databases of the antivirus company. This approach can be very effective against sudden outbreaks of new malware, containing the spread. The authors of malware try to stay ahead of this defense mechanism by writing adaptive malicious code which can encrypt parts of itself or modify itself in such a way that it does not compare to its original signature. Besides authors creating adaptive code, the world of cybercrime also has to deal with a lot of copycat-behavior. These types of adversaries might reuse malware and also change it slightly to remain hidden from antivirus software. A blogpost by Zheng Bu and Rob Rachwald, both researchers at FireEye, even shows that most malware remain active for no more than two hours [13]. Their analysis concludes that 82% percent of malware disappears after one hour, and maybe even more shocking, 70% percent of malware only exists once. They say that with the half-life of malware being this short, signature-based antivirus detection has more in common with ghost hunting instead of threat detection and prevention. Defending against malware at this level might feel like a whack-a-mole game. After identifying and stopping a new type of malware, a similar threat pops up somewhere else and after also blocking this one, five adapted versions of the same malware show up. This essentially has become a rat race between 'good' and 'bad'.

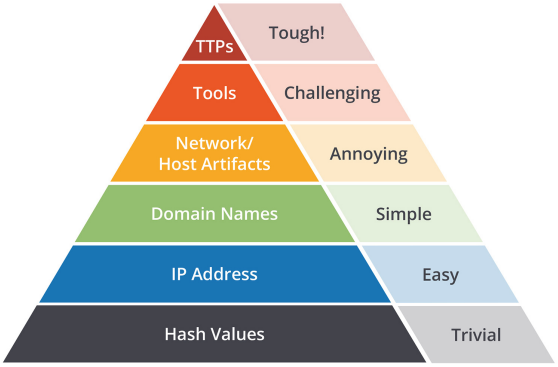

Figure 1.1: The Pyramid of Pain as proposed by David Bianco [10]

In 2013, David Bianco introduced a concept called *the Pyramid of Pain*. An illustration of this concept can be seen in Figure 1.1 and shows the level of difficulty to detect certain types of adversarial activities [10]. Starting at the bottom, the easiest way to detect malicious behavior is by hash values corresponding to malware, and all the way at the top are the tactics, techniques and procedures (or *TTPs*). TTPs refer to the plans adversarial groups conduct to orchestrate and manage attacks. Evidently the goal of antivirus companies, and other related organizations, is to figure out these TTPs and stop cybercriminals at their source. It has become clear that these TTPs will not be found by merely looking for malware signatures. In order to see the global patterns of TTPs, it would be necessary to look at the source of the problem from a global perspective. Analyzing such things would require network traffic directly from the backbone of the Internet, as this kind of data is incredibly rare.

With this in mind, we need to take one step back. Host-based detection will eventually not solve the problem, but postpone it. Malicious traffic passes through global networks, even if it is stopped in its tracks by a host based detection. The traffic however still travels through the Internet backbones. Analysis on actual TTPs is scarce and even non-existing on the scale of global network traffic. New detection methods are predominantly created for detection at the host level. Antivirus software and detection methods exist as front-door safety, stopping the criminal from entering without the right key. It would however be better to stop criminals before they even reach the front-door.

1.3. Cyber Threat Intelligence

Organizations use Cyber Threat Intelligence to better understand, predict and adapt to the behaviors of malicious actors. Cyber Threat Intelligence can take many forms, including information about malware an adversary uses, known C&C servers or specific techniques a group uses to steal information. By far the most common data source is *observable behavior* which indicate if a host or network may be compromised. These observations are called *indicators of compromise*. Organizations naturally collect a certain amount of threat intelligence by fending off attacks or collecting spam emails. However, one organization cannot see every detail of an attack. It has become wildly accepted that third-party companies provide condensed lists containing indicators of compromise, which in turn can be used to tighten defenses. These third-party companies often ask steep prices for their services, which customers often pay without regard. A lot of researchers or companies also provide free threat intelligence, striving for a more secure world. Where antivirus companies mainly use malware signatures to detect adversarial behavior at the host level, Cyber Threat Intelligence can be seen as the 'antivirus' for detecting this behavior on a global scale.

1.4. Research question and approach

The academic world mainly aims for new detection strategies, trying to stay ahead of the cybercriminals and trying to provide protection against emerging threats. Theory about networking infrastructures tells us that these adversaries can apply labor division amongst infected machines, something which has not been shown on a large scale as local administrative data does not provide this information. Global analysis on large malicious C&C infrastructures has not yet been performed in the academic world, as this would require access to a huge amount of Internet backbone network traffic. Where knowledge on the different families of malware, and how to detect them, is wildly populated, analysis on the evolution of adversarial structures or the TTPs behind them does almost not exist. Open source threat intelligence could be the solution for this, as this provides information about current indicators of compromise used in attacks. In the spirit of a more secure society, it would be desirable that these threat intelligence feeds provide enough information to protect against most adversaries.

In this thesis, we will focus on these research gaps by using tier 1 ISP NetFlow data to chart the ecosystem and analysis of global C&C infrastructures. To establish this, we will first analyze the current state of open source threat intelligence. We set out to show that NetFlow data, when handled correctly and thoroughly, can provide a new angle to analyze threats and attack vectors, and ultimately be the next step in fighting cybercrime. This work will aim to answer the following research question:

What are the attack patterns and tactics, techniques and procedures of malicious Command & Control infrastructures, how do these infrastructures evolve and how can we track them on a global scale?

To provide a structured answer to this research question, we have created a research approach to tackle this broad problem. To gain more insights into malicious infrastructures, indicators of compromise are needed as a starting point. Open source threat intelligence would initially be the best source for this information, as researchers and organizations around the globe contribute to this knowledge. Before using this kind of information, it is imperative to know that this information will provide a clear picture global of cyber threats. This leads to the first step in our research approach:

(1) *Determine the coverage of open source threat intelligence feeds and determine if these resources provide enough information to analyze malicious infrastructures.*

Chapter 4 will answer this step, providing a thorough analysis of 24 open source threat intelligence feeds and will evaluate the coverage these feeds provide. This evaluation is necessary to obtain a clear picture of the current state of threat intelligence regarding, among other attack vectors, C&C infrastructures.

After having analyzed and evaluated the publicly available threat intelligence sources, we can use these conclusions and information from these sources to analyze C&C infrastructures. As networking infrastructures are a very broad concept, we will breakdown the analysis of malicious infrastructures into three different components which most people can relate to. Personal computers, laptops, and smartphones have become fundaments of our society, making them one of the most targeted devices of adversaries. Smart devices, referred to as Internet of Things (IoT), have also set foothold into our modern society. Finally, in the end all of these devices have to connect to the Internet, which is always done by some kind of routing device. We will introduce a methodology in Chapter 5 as a next step of our approach, which will be used to answer our main research question. In this step we will elaborate on the choice for these categories and techniques used to help answer our research problem:

(2) *Define a methodology which enables us to utilize NetFlow data, combined with additional datasets, to identify malicious infrastructures.*

To thoroughly analyze malicious infrastructures we will dive into infections based on the three networking infrastructures categories which we have defined; infections on clients, IoT devices and Internet routers, resulting in the final step of our research plan:

(3) *Discover attack patterns and tactics, techniques and procedures of malicious Command & Control infrastructures and their evolutions within infections on clients, IoT devices and routers.*

This final step will be answered in the Chapters 6, 7 and 8 providing extensive analyses on adversarial infrastructures. These chapters will show how the methodologies used in Chapter 5 can be applied on NetFlow data to explore the adversarial tactics, techniques and procedures used by cybercriminals, painting a more clear picture of the cyber threat landscape.

1.5. Contributions

During our work we have analyzed dominant malicious infrastructures and activities on three different Internet infrastructures; infections on clients, IoT devices and Internet routers. To do this, we initially made an analysis of publicly available cyber threat intelligence, evaluating the timelines, sensitivity, originality and impact this intelligence has on society. Following from this, we used a detection method to understand global patterns of Command and Control servers and used clustering techniques to gain more insights into the different behaviors the infected machines controlled by these servers have. In this thesis, we make the following contributions:

Analysis of threat intelligence feeds

- We introduce a taxonomy to evaluate the quality of cyber threat intelligence feeds, aiming to assess the defensive advantage users will gain when adopting these feeds.
- We evaluate the indicators reported on 24 open source threat intelligence feeds across four dimensions, and benchmark using NetFlow data and zone transfers the timeliness, sensitivity, originality and impact of these feeds.
- We empirically analyze the impact an indicator of compromise has, which has been provided by an intelligence feed. This allows us to evaluate the adoption of these feeds in practice and estimate whether a feed is in practice able to protect clients and networks from future harm.

Infections on clients

- We have analyzed adversarial tactics and unveil differences between servers in how they communicate with their infected machines and how this activity evolves over time.
- We are the first to use a combination of learning techniques to find unknown C&C servers in anonymous NetFlow data and complementing data found in open source threat intelligence feeds.

Infections on IoT

- We are the first to analyze IoT infection from the perspective that NetFlow data provides. By additionally using scanning data, targeted at a network telescope, we can estimate the leverage of the infection. According to our data, 1 in every 2345 scanned devices is successfully infected by the Mirai worm.
- We are the first to detect Mirai C&C servers without inspecting the actual malware binary and expand existing knowledge by elaborating on their behavior over time. Additionally we provide two examples of infected devices being utilized for DDoS attacks and correlate this with C&C behavior.
- We estimate the amount of revenue which can possibly be made by setting-up a DDoS booter service based on the amount of malicious traffic flowing through these infected machines. A large botnet could possibly earn up to ~$285,000 per year.

Infections on routers

- We are first to investigate a new type of attack that exploits Internet infrastructure for cryptomining, and show how over a period of 10 months after the initial discovery of the vulnerability groups of criminals launch massive campaigns to control 1.4M routers, with a peak of 460,618 simultaneously infected routers.
- We have analyzed adversarial tactics and unveil the supporting infrastructure used within the campaigns, and are able to show differences between groups in how they locate their victims, compromise routers, and run their infrastructure.
- We demonstrate that previously reported vectors are negligibly small in number of affected users and created revenue, compared to the reported MITM attack. We find that this attack yielded monthly revenues, estimated exceeding $1,200,000 per month for the top 10 grossing accounts, a factor of 30 larger than previously estimated cryptojacking revenues from hacked websites, malicious advertisements and website-owner initiated mining combined.
- We have observed high levels of sophistication in three identified campaigns, of which the largest involved 40 mining accounts linked to one single actor.

The work we have done resulting in our last four contributions will be presented at the 26th ACM Conference on Computer and Communications Security - 11-15 November 2019, London, UK.

1.6. Thesis structure

The analyses presented in this thesis will work towards answering the research question and present the scientific value added to the cyber threat intelligence landscape. This will be done according to the following structure. The next chapter will provide necessary background information on previous work done on the analysis of C&C infrastructures and on open source cyber threat intelligence. Chapter 3 will provide a detailed analysis of the NetFlow dataset used to accomplish the presented goals and Chapter 4 will provide an extensive quality analysis on open source threat intelligence feeds. Based on the conclusions from this analysis, Chapter 5 presents methodologies which have been used for analyzing malicious infrastructures. The following chapters will analyze malicious C&C activities found on three different Internet connectivity infrastructures. Chapter 6 will use open source threat intelligence feeds to analyze the C&C traffic between infected clients and the adversary, followed by an investigation of IoT devices infected by the Mirai virus in Chapter 7. After this, Chapter 8 provides a study on a man-in-the-middle cryptojacking attack targeted on Internet routers. Finally, Chapter 9 will present our conclusions of the results, provide limitations and discuss possible future work.

2

Related work

This chapter will provide an overview of existing academic research on adversarial infrastructures, Command and Control (C&C) detection and clustering, and cyber threat intelligence feeds. We present prior work's techniques and principles and discuss the most important ideas we can learn from this. Section 2.1 will elaborate on known infrastructures and patterns adversaries prevail for misuse, how these structures have emerged throughout the years and what kind of challenges we will be facing in the near future. After this, Section 2.2 will discuss methods on detecting C&C infrastructures and corresponding analyses. As we are using publicly available Cyber Threat Intelligence for the basis of our detection, previous work on this matter will be elaborated upon in Section 2.3. Finally in Section 2.4 we will give an overview on malicious campaign analysis. Besides summarizing the work, we point to potential research gaps, forming the basis for the research questions stated in the introduction.

2.1. Command & Control structures

The definition of C&C infrastructure is quite broad. In essence these infrastructures are controlled by a malicious actor, who directs one or multiple C&C servers. These servers are computers which are capable to issue directives to devices that have been infected with rootkits or other types of malware, such as ransomware [110]. Adversaries can use these C&C servers to establish powerful networks of infected machines, or *botnets*, and utilize these networks to perform malicious activities. It is difficult to accurately define a botnet [105]. Evidently a botnet is controlled by a malicious actor, or *botmaster*, which uses means of C&C to connect to and command each of the machines it has infected, or bots. Communication plays an important role, but the sole ability of malware to connect to other malicious instances is not a sufficient condition to classify an infected computer as a bot. Modern malware is practically always a combination of infection, attack, concealment, adaption, and communication [14], where our focus in this thesis lies with communication. Besides manifestation it is also interesting to know why adversaries pursue these malicious networks. Khattak et al. performed an extensive analysis on the behavior of botnets. They elaborate on various different aspects, including the purpose of these botnets and say that in addition to financial incentives, owners of botnets are also driven by other goals such as stealing intellectual property, espionage and cyberwarfare [68]. In their study they elaborate on seven prime reasons malicious actors establish malicious networks, namely information gathering, distributed computing, cyberfraud, spreading malware, cyberwarfare, unsolicited marketing and network service disruption. Each of these lucrative aspects can incentivize people to create and maintain botnets.

Communication topologies

A lot of research has been done evaluating the type of C&C structures which can be encountered. A report by Trend Micro [40] interestingly elaborates on the history of organized malicious networks. In their report two contenders arise to have been responsible for the malware initiating the botnet waterfall: Sub7, a Trojan, and Pretty Park, a worm. These malware both introduced the concept of connecting to an Internet Relay Chat (IRC) channel to listen for commands. They first surfaced in 1999, which has since then led to constant botnet innovation. Both malware made sure each infected machine would connect via IRC to a C&C server, which entails a specific kind of structure called *centralized*. This centralized structure would quickly be known as

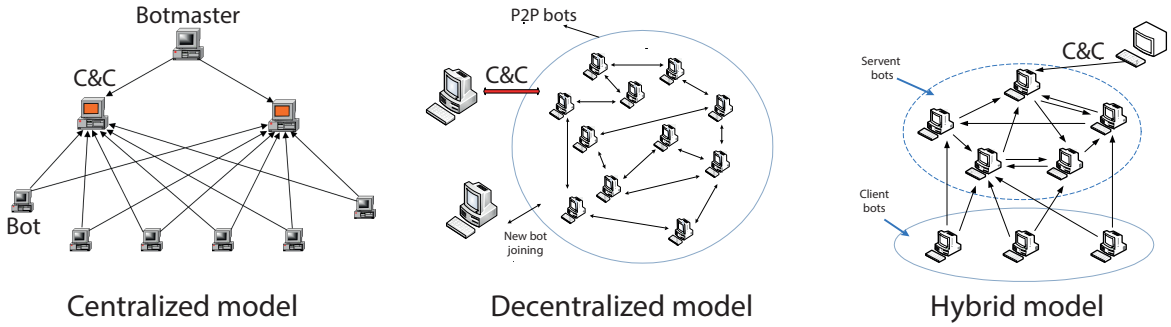

Figure 2.1: High-level overview of centralized, decentralized and hybrid topologies

the fundament on which more advanced structures can be built. As one of the early adopters, Cooke et al. in 2005 investigated the centralized structure and warned that more advanced structures would arise in the near future. In their work they begin with describing how the botnet problem has evolved since 1999 and demonstrate the impact of the botnet problem by using evidence from various data collection sources. Using this information they examined the, at that time, current IRC-based botnet communication strategies. Besides this, they also revolutionized by introducing two other possible C&C topologies; Peer-to-peer (P2P) and random structures [25]. Both topologies have a *decentralized* communication system, which is much harder to disrupt. This means that the compromise of a single bot does not necessarily mean the loss of the entire botnet. Finally they explained that each topology has specific advantages and drawbacks, and the optimal topology for a botnet might be a combination of a centralized and decentralized structure. It is interesting to see that in 2005 they already predicted this course for the future, as they were arguably accurate which we will explain further on.

In the same year, Dagon et al. built upon their research and introduced a set of metrics to utilize this taxonomy even further. They spent attention to key discriminators that could compare important attributes of botnets. In their research they identified three important measures of botnets: size, network diameter, and redundancy. Their analysis shows that decentralized network models give botnets considerable resilience and that these kind of formations resist both targeted and random responses. Finally their analysis suggest that by removing the C&C node, scale-free botnets will diminish after a while [27]. This however seems to contradict the purpose of a decentralized model.

A few years later, Zeidanloo et al. summarized all the work that has previously been done on C&C infrastructure analysis [142]. In their work they have defined a classification for greater understanding of C&C structures which entails centralized, decentralized, and hybrid models as depicted in Figure 2.1. They elaborate upon each of these topologies, starting with the centralized model. All connections happen by means of the C&C server, making this a critical point in this model. In other words, C&C server is the weakest link in this model. If somebody manages to discover and hijack the C&C server, the entire botnet will be worthless and ineffective. This could also happen by discovering a bot belonging to this structure and finding out to where it beckons. Considering this disadvantage, adversaries started to build alternative C&C systems which are harder to discover. Hence, a model was created in which the communication system does not heavily depend on a few selected servers or on one specific bot. They argue that understanding these topologies is essential to precisely identify, detect, and mitigate the increasing botnet threat.

At this point in time it has become clear that these adversarial infrastructures have become increasingly complex and with this harder to uncover, as we have seen them moving away from detectable IRC communication to complex privacy preserving P2P communication. A hybrid model of partially centralized and decentralized communication seems key. Following these researchers, a huge amount of work has been done on botnets and their malicious infrastructures, most of them trying to complement each other's work in the format of a survey of taxonomy. With respect to this, in 2009 Bailey et al. continues the exploration by diving deeper into the intersection between existing botnet research, the evolution of botnets themselves, and the goals and perspectives of various malicious infrastructures [5]. They provide, like others before them, a survey about these infrastructures, however they add novelty by providing a first look into possibilities for understanding these networks and initial techniques to detect various infrastructures. One of their conclusions is that all aspects of the botnet's life cycle, from propagation, to C&C, and attack vectors are all continuously evolving.

It therefore does not make sense to try and frame everything adversarial infrastructure entails in one picture. With this conclusion in mind, in [33] Eslahi et al. again presents an overview of malicious botnet infrastructures, but also provide a first insight into a few detection techniques used in the field. They explain *detection by signature*, which refers to the known patterns or characteristics of threats, *Honeypot and Honeynet*, which are tools that are used as traps to collect malicious activities and analyses for detect, *detection by DNS traffic monitoring*, and finally *behavioral analysis detection* referring to anomaly detection on malicious patterns. The latter is a technique which will endorse a lot of methods in detecting botnet behavior and will be more thoroughly discussed in the next section.

After the work of [33] more surveys and taxonomies arose, with each one elaborating one different adversarial techniques they have encountered. Khattak et al. for example provides a comprehensive framework, based on three taxonomies, which could be utilized to understand the botnet problem and its solution space. These three taxonomies aim to expose the adversary by exploring behavioral features, detection and defense [68]. They also provide insights into botnet trends to come in the next years; botnets based on cloud computing and smartphones. Botnets can utilize cloud services by hosting the C&C servers in the cloud [75] or by initializing bots in the cloud instead of infecting user machines [24]. It is an attractive option to host C&C servers in the cloud because of its on demand and pay-as-you-go nature.

In 2015, Amini et al. tried to summarize previous publications and point out their limitations. They correctly state that most of these surveys each provide different taxonomies and interpretations. However, they also state that there are no previous studies on defense mechanisms [2]. This is incorrect, as we have seen in related work elaborated on in previous paragraphs. More recently Vormayr et al. presented their work on an in-depth analysis of botnet communication, and is by far one of the most comprehensive studies to date [132]. Their main contribution is a novel taxonomy of botnet C&C network communication patterns which was derived from the analysis of a diverse set of botnets. In this taxonomy they present the C&C patterns divided scenarios, propagation, data download, data upload, forward proxy, reverse proxy, and instruction. These communication patterns cover every possible botnet task of the networks they have analyzed.

Diversification can be seen in work being produced over the last couple of years. This is probably due to the fact that the threat landscape of botnets and their C&C infrastructure has been growing out of proportions, and more specified work is needed to understand the inner workings and impacts of these threats. For example, Khan et al. have investigated the most recent solutions proposed regarding detection of botnet generated email spam and tried to categorize these botnets based on methods of defense [67]. They conclude that traditional content based filtering and IP Blacklisting solutions are rendered ineffective, something that is caused by the increasing complexity these networks operate in. Another example of extensive and specific work is done by Hoque et al. who delved into Distributed Denial of Service (DDoS) attacks and how the primary facilitator of modern DDoS attacks are botnets. Currently DDoS attacks are one of the most harmful attack types [94]. They also provide discussion on mobile botnets compared with traditional PC-based botnets, something which was foreseen by [68]. Besides this, the researchers also provide a detailed discussion of defense approaches and methods against botnet-based DDoS, and propose novel solutions to counter these kind of attacks [62].

Communication techniques

In the previous section we have elaborated on the most important topological evolutions regarding C&C infrastructure. Concurrently a few technological means of communication were mentioned. However, to fully understand the increasing complexity of C&C structures it is important to also grasp these popular communication techniques. To give an impression of what techniques have been used and how these have evolved, we have to journey back to the first botnet-like structures in 1999. The first ever malicious infrastructures, Sub7 [35] and Pretty Park [34], based their communication on Internet Relay Chat (**IRC**) as mentioned in the report from Trend Micro by Furgeson [40]. This protocol was designed to allow several forms of communication (one-to-one, one-to-many and many-to-many) and therefore allow command distribution over a large number of clients. With these features in mind, this protocol was initially designed in 1988 for chat purposes across multiple devices, and could be used as an open protocol which normally runs on TCP ports 6660-7000. The inherent flexibility of this protocol enabled third parties to extend it anyway they preferred which made the protocol a good choice for botmasters, simplifying the botnet implementation [104]. One of the drawbacks of this protocol showed up fairly quickly; blocking C&C traffic of IRC was fairly easy, as connections to and from these fairly unusual TCP ports could simply be blocked.

To overcome this problem, adversaries shifted towards a more distributed organization by creating a botnet based on the principles of peer-to-peer (**P2P**) networks. In such a network any node in the topology can

act as both a server and a client. A P2P botnet has the same objectives as a centralized botnet; infecting more devices, dispersing commands and harvesting data. The advantage of a P2P architecture, is the lack of tight C&C infrastructure. The origin of these kind of structures is the most difficult to trace, as the commands could be given to merely one infected machine, which can in its turn disperse it to all peers in its network. Phatbot [36] was arguably one of the first to switch to a P2P based topology. At this point, malware was increasing in complexity as Phatbot was able to change its encryption for each new infection making it close to impossible to detect automatically at the time. P2P seems like an ideal solution for this kind of malicious activity, however maintaining, utilizing and dividing such a network is deemed to be very challenging. Wang et al. in their work even said that the general understanding *"P2P botnet is much more robust against defense"* is misleading [134].

For this reason, adversaries turned to other solutions, trying to fall back to centralized structures as this gives the botmaster more options and freedom. With this in mind, adversaries started to explore the use of command and control communication through the Hypertext Transfer Protocol (**HTTP**). The HTTP protocol operates as a request–response protocol in the client to server model [8] and is currently the most popular protocol for Internet traffic as we will see in Section 3.3. The main advantage of using this protocol for C&C traffic is that it can blend into benign traffic, bypassing basic firewall rules with port-based filtering. Usually firewalls or Intrusion Detection Systems (IDS) block incoming and outgoing traffic to unusual ports, for example IRC ports. However the architecture of the HTTP and IRC protocol is the same, Gu et. al. pointed out that the HTTP protocol uses a *pull* style architecture, where IRC uses a *push* style [50]. Some early adopting botnets using the HTTP protocol include Bobax [122], ClickBot [28] and Rustock [19].

The innovation of using HTTP as base for C&C, opened the gates to more sophisticated structures which would eventually provide more anonymity for adversaries. One method of facilitating resilience to both detection and reverse engineering of the botnet is the use of Domain Generation Algorithms (**DGAs**). The goal of DGA is to dynamically produce and registering a large number of random domain names, and selecting a small subset for actual C&C use [3]. Only a small subset is used, as the C&C domain is randomly generated and only used for a very short period of time, meaning that simple detection methods which rely on static domain lists are left ineffective. Most of these algorithms use the current time as input, requiring every infected machine to have a synchronized time. A simple example of such an algorithm can be examined in Listing 2.1. Vormayr et al said that for most algorithms it was sufficient to be accurate to the nearest hour or day [132]. If this were not the case, solid communication with the C&C server would be close to impossible, seeing that every infected machine would generate a different domain. A lot of clever options can be thought of to overcome this problem, for example the malware can change the systems time on the machines or something more advanced by probing a popular website for its date and time [141]. As can be imagined, the most beneficial characteristic of such an algorithm is the amount of freedom in terms of C&C traffic. As the life cycle of each C&C server is short, it does not matter if one of these servers is detected and taken offline as a new one will pop-up in a short amount of time. A good example of this is the Conficker malware, which generated 250 domain names every three hours [46] [101]. This malware would provide a lookup on each domain which has been generated and attempts to contact each IP address which it finds to download malicious binaries. Taking control over a malicious infrastructure which has been created by a DGA has proven to be very difficult [141]. The only way of dissecting such a network would be to find and reverse the algorithm generating the domains and register all of these to take control of the botnet, which is precisely what researchers have done with the Torpig botnet [102].

A more recent technique used for C&C, often glossed over by literature, is the use of the Domain Name System (**DNS**). DNS is a decentralized and hierarchical system for devices, applications, or other resources

```
def generate_domain(year, month, day):
    domain = ""

    for i in range(16):
        year = ((year ^ 8 * year) >> 11) ^ ((year & 0xFFFFFFF0) << 17)
        month = ((month ^ 4 * month) >> 25) ^ 16 * (month & 0xFFFFFFF8)
        day = ((day ^ (day << 13)) >> 19) ^ ((day & 0xFFFFFFFE) << 12)
        domain += chr(((year ^ month ^ day) % 25) + 97)

    return domain + ".nl"
```

Listing 2.1: Example of a basic domain generation algorithm

connected to the Internet. This system is basically a large address book, as it can direct a machine querying a specific domain name to the IP address it belongs to [85]. Dietrich et al. belong to one of the first researchers uncovering this C&C technique by dismantling a specific type of malware, Feederbot, that showed strange behavior as it seemingly did not use any obvious C&C protocol. They reverse engineered the sample they obtained, and found out that the bot was abusing DNS as a communication channel for receiving commands, by resolving a specific domain name in the malware and using the corresponding IP address to communicate. The primary purpose of DNS is being able to use multiple servers around the world enabling a user access to content closest to them. Dietrich et al. explains that, because of this fact, it is also useful to perform load balancing and provide extra redundancy in case of one C&C server failing. A nifty approach for DNS C&C activity is the use of a **Fast-Flux** network [46]. In such a network the C&C server is protected by a wall of infected machines which are often part of a botnet. These machines will in this case act as a *proxy* to the C&C server, meaning that each request they get for the server will be forwarded to the C&C server and returns the replies to the original requester. The clever thing about these Fast-Flux networks is that they can grow as large as tens of thousands of IP addresses and that each of the infected machines protecting the C&C server has a unique IP address which then all point to the C&C server [59]. A host wishing to make contact with the C&C server makes a request for the domain name and is returned a set of IP addresses, forcing it to connect to one of them. The returned IP addresses are however not the C&C server itself, but on the infected machines protecting the server. Besides this, the IP address which are returned have a very short time-to-live (TTL), forcing a second request to the same domain name to result in a completely different set of IP addresses [46]. Exactly this behavior is what makes Fast-Flux networks extremely difficult to catch.

Besides these evolutions in terms of C&C techniques, botmasters are continuously looking for new means of command and control through infrastructures which preserve privacy even more. An example of this is the option to use ever growing *cloud* solutions for their adversarial behavior. Already in 2009 an article was written explaining the uses of Amazon EC2 as Command and Control structure. On the morning of 10 December, 2009, Dan Morrill elaborated on how the Zeus Trojan had nestled itself in some of these virtual computers, caused by users not updating an application [86]. Another innovation in terms of more complex infrastructures involves publicly available *social networks* like Twitter, Facebook or LinkedIn. Traffic analysis on popular platforms are very challenging if the bots seem like normal everyday users. Nagaraja et al. in their work showed it is trivial to set-up a social media botnet, and using steganographic techniques to hide command and control communication in images shared on social networks [90]. More recently Lenny Zeltner published a blog post about malware using Twitter as command and control infrastructure [143].

For the last couple of years an increasing trend in *smart* devices has been observed. More and more devices are designed to have Internet access; mobile phones, watches, cameras and lots more. For these devices the term Internet Of Things (**IoT**) has been introduced. For consumer markets this may seem like one of the best innovations of the last decade, however in terms of security this has become one of the most worrying attack vectors. Securing these devices has proven to be fairly difficult, as most rely on continues updates by the user [147]. In September 2016 the Mirai malware spread and possibly created the largest botnet ever recorded [71]. Mirai was surprisingly not one of the most complex malware ever found, it basically uses a list of 62 common default usernames and passwords to gain access to private routers, cameras and video recorders [9].

Adversaries can choose from a lot of different topologies and techniques to utilize C&C for infected machines. A lot of different tactics have been elaborated upon in this section, however it must be said that everybody with malicious intentions can possibly figure out new means of command distribution. This thesis will however focus on detecting and analyzing known structures in our dataset and elaborate on the evolution of adversarial behavior.

2.2. Detection and Analysis

Malicious infrastructures using Command and Control to maintain and direct infected machines have been around since the end of the 20^{th} century. Since this time there has been an ongoing battle between researchers and adversaries, each time trying to outsmart each other in terms of innovation. The detection of botnets and their C&C infrastructure therefore also goes back to this time.

Early research

Pioneering the malicious infrastructure detection Spitzner et al. introduced the Honeynet project. This project began in 1999 as a informational mailing list to which researchers could subscribe to learn about

various network attacks [121]. Spitzner et al. described their work as a network of machines strategically placed on the internet, intentionally capturing unwanted and unauthorized behavior on these machines. By utilizing this behavior they could intercept unknown attacks and the malicious software belonging to it. This information gathering nowadays goes by the term *Honeypot* and is wildly implemented.

A few years later, in 2005, Cooke et al. proposed a simple mechanism to monitor activity on TCP port 6667, which was a popular port for IRC used by adversaries for C&C activity. They however acknowledged that these ports can be easily interchanged, and proposed a second system utilizing a honeypot which could capture C&C traffic between the botnet and the machine [25]. The drawback of this system was that botmasters could easily change the behavior of their bot and the C&C server, rendering the system useless. Cooke et al. additionally said that no technique could actively detect this kind of behavior.

Two years later Gu et al. proved them wrong, developing two detection mechanisms, together called BotHunter, to tackle this kind of behavior [48]. The first mechanism, Statistical Scan Anomaly Detection Engine (SCADE), monitored incoming traffic on 24 TCP and 4 UDP ports for C&C traffic associated with known malware. Besides this it also monitored internal traffic for hosts trying to connect to an exorbitant amount of IP Addresses. The second mechanism, Statistical Payload Anomaly Detection Engine (SPADE), inspects if the packets entering the network are malicious by calculating the standard deviation between this traffic and known benign traffic. Not long after this, in 2008, Gu et al. improved their previous BotHunter and built BotSniffer. This is a system which could detect IRC and HTTP C&C traffic in an internal network by using anomaly detection [50]. By recognizing behavior hosts have after communicating with a probably C&C server, they developed algorithms which could detect C&C traffic as well as network scanning. With this system they proved to detect 100% of IRC botnet C&C traffic with only a false positive rate of 0,16%.

In 2007 Karasaridis et al. also designed a detection technique based on anomaly detection. In their work they could estimate the size of a malicious infrastructure and identify the C&C server belonging to it [65]. This looked very promising however, their solution only worked on IRC and with a centralized C&C infrastructure. In their work they also recommended future work to focus on P2P and HTTP based botnets. With this in mind Gu et al. returned with another addition to their defense arsenal, BotMiner, which they claimed to be versatile regarding C&C topologies. They realized that infected machines inevitably have to connect to the C&C server at one time, regardless of the structure. BotMiner focuses on the communication between hosts and clusters them based on their behavior [49]. Using prior knowledge of what activity was malicious it could make a distinction between good and bad traffic.

Wang et al. followed the example of detection based on C&C traffic patterns and developed a technique which could detect this kind of behavior irrelevant of the type of botnet. This would add significant novelty with respect to prior work, as the rest relied on some kind of base for malicious activity. They only used the packet size and interval time of packets entering the internal network to determine whether it was C&C traffic [135]. This showed less promising results, as they could only detect botnets belonging to four different malware types using centralized topologies. Up until now it has proven to be extremely difficult to find a solution which captures all malicious traffic.

With this in mind, Nagaraja et al. focused on developing a detection technique for P2P infrastructures by creating BotGrep. They designed an algorithm which clusters P2P communication between hosts pairing with each other. It uses graph analysis to identify which hosts belong to a botnet. In order to be effective, the system does need a ground truth of what these malicious infrastructures look like. This system would not work well with detecting new emerging botnets, but with the right data it could gain more insights on known structures [89].

All of the previous work has focused on either host or network based detection. In 2010 Zeng et al. combined this and believed combining these two would provide even better detection results. For their monitoring system they kept track of changes on filesystems, registry changes and network stack changes on the device it was monitoring. Next to this they provided network level detection by using NetFlow traffic traces [144]. They extracted features from these traces, and used them to analyze the similarity or dissimilarity of network behavior. Eventually the method they propose is effective against IRC, P2P and HTTP botnets, but they mentioned it is unfortunately is not a scalable solution. This detection system would only work in a controlled environment, where access to the overarching network is available for the NetFlow data. Zeng et al. were to our knowledge the first to use NetFlow data as a detection method for botnet activity.

Advanced detection

A brief history has been given about early research in the field. From this point research diversifies a lot, and for this reason the focus from here on out will be more towards detection close to the network itself. Around

2010 adversaries focused their attention less on IRC and centralized C&C infrastructure, and more towards P2P, HTTP and decentralized topologies. That is why in 2011 Francois et al. used large datasets with NetFlow traffic to detect P2P communications, overcoming forensic limitations [42]. They have a similar approach as [89], utilizing an algorithm which uses graph analysis to cluster behavior of these botnets. They were also the first to adopt big data into detection, making use of Hadoop [117] to increase the speed and efficiency of detection. Zhang et al. went to the drawing board in 2011 and also developed a technique to detect P2P communication, but they applied statistical fingerprinting of this traffic to detect anomalies. Their approach proved successful in local area networks (LAN) and in wide area networks (WAN). The proposed method used a lengthy pipeline of techniques to detect malicious peer-to-peer traffic. First signatures are created for different kinds of applications by using the length of time a P2P programs operates, as malicious programs tend to run as long as possible [146]. After filtering they further differentiated traffic based on IP addresses contacted by each P2P host because infected machines in a P2P botnet will each contact the same IP address. Finally they applied a filtering which analyzed if a high percentage of these hosts ran the same protocol and communicated with a lot of overlapping IP addresses. Zhang et al. shared their results and were able to detect 100% within the network traffic that they captured with only 0,2% false positives.

Adversaries keep innovating towards more versatile and anonymous systems. Barthakur et al. realized this and developed a procedure for detecting encrypted P2P botnet communications. They used Support Vector Machines (SVM) to analyze network traffic, classifying P2P traffic based on statistical differences between P2P and benign traffic. In their work they conclude that botnet P2P traffic use a lot of random ports and keep the packet sizes to a minimum, whereas benign traffic does not do this [7]. On the other hand Han et al. believed that adversaries also tried to avoid detection by keeping the sizes of malicious infrastructures small and enabling multiple C&C servers. In their work they propose a system called Garlic, used to supposedly effectively suppress botnets. As a real garlic, the structure of the system is distributed in cloves or *nodes*. Each of the nodes collects C&C traffic of detected botnets and learns its pattern to be more effective in detecting [54]. Their system was unfortunately only effective on IRC botnets, the amount of which have been decreasing for a while.

Zhang et al. investigated the increase of drive by download attacks, a terminology used for downloads which a person has authorized but without understanding the consequences. These attacks may happen when browsing to a website, clicking on a link or opening an attachment in an email. Zhang et al. proposed a detection technique to classify malware at the infection stage [145]. They realized that many new botnets infected machines by sending malicious emails or using websites. By collecting HTTP traces from various honeypots and using a web crawler to crawl domains, they tried to identify different techniques used for drive by downloading. After this they clustered groups of domain names which shared IP addresses. By doing this they could tackle a couple of botnets using Fast-Flux, a technique described earlier in this section. In the same work, Zhang also proposed a system which would reduce the amount of packets needed for actual packet inspection, by correlating NetFlow data with packet header information. This allowed deployments in larger networks for C&C analysis [145].

In the same year Bilge et al. developed Disclosure, a botnet detection system which used NetFlow data to identify the C&C servers instead of the infected hosts [12]. Their method extracts a large set of features from the NetFlow data corresponding to the pattern of the C&C servers. It uses flow sizes, client access patterns and temporal features together with a Random Forest classifier to distinguish between malicious and benign traffic. By building a benign data set from Alexa Top lists, and mixing this with known malicious data, a machine learning model was trained to successfully detect C&C servers in large networks. The best result they obtained was a 65% detection rate of C&C servers with no false positives. Bilge et al. are in our knowledge the first and only ones using NetFlow data from a large external network to perform detection. This thesis will use and expand on their method for detection on our own dataset, as will be explained in Chapter 5.

Zhang et al. in another work also presented a process based on NetFlows described as a flow-capturing monitor resting at the edge of large networks collecting possible botnet behavior [145]. They achieve this by sampling the data specifically on malicious looking activities. Each packet flowing through an edge router on a network passes through special monitoring software. This software has four components: the Counting-Sketch, which tracks the amount of packets sent from source to destination, the Sampling-Sketch, categorizing the IP addresses flowing through the system, the Synchronized IPs Detector, which keeps track of both external and internal hosts having similar patterns and finally the Priority-based Sampling Probability Calculation which dynamically calculates the sampling probability for each category of IPs.

Adversaries try to encrypt their C&C traffic, increasing difficulty to track it as Garant et al. elaborated on in their work. They claim that existing techniques are ineffective against unknown botnet traffic, as they all rely

on known structures or patterns [45]. To counter this, they crafted a botnet which fully employs encrypted communications to test their proposed detection technique against these kind of threats. This method uses a machine learning based decision tree classification method, using six features which are unique to C&C traffic. The features which they extract are the amount of bytes and packets transferred, duration of the flow, direction of the flow, and TCP flags used. They achieved results of 90% successful detection with false positive rate of 9,9%.

Haddidi et al. also realized that innovation was required to overcome the problem of encrypted traffic. Like Garant et al. they employed machine learning to realize a system which could detect encrypted botnet traffic, but in contrast to Garant et al. they showed that packet analysis is not needed for this [53]. By utilizing C4.5 and Naïve Bayes classifiers on extracted features from NetFlow data, they could detect HTTP botnet traffic with detection rates varying from 7% to 88% and false positive rates between 1% and 16%. The encryption of packets does not matter as they used NetFlow data for their work. It must however be mentioned that they only tested on a small internal dataset and testing only on a few known botnets.

Studies performed in recent years confirm a growth regarding the popularity of flow-based anomaly detection. A lot of researchers focus on network activity to detect specific malicious infrastructures. In 2015 Grill et al. acknowledged this by stating that packet inspection, clustering, and reverse engineering are time consuming approaches for anomaly detection and are unfeasible for large scale networks [47]. They present a technique which can successfully detect malware using DGA for sizable networks. In their work they describe their method, which uses a statistical approach and models the ratio of DNS requests and visited IP addresses for each host in the local network. More recently Homayoun et al. also propose a system which focuses on the same problem, called Botnet Traffic Shark. Their method also avoids dealing with encrypted payloads, and applies more advanced techniques compared to the statistical approach of Grill et al. Instead, they identify correlations between original features and extract new features in every layer of an Autoencoder or Convolutional Neural Network (CNN) [60]. In contrast to Grill et al. their method is capable of detecting IRC or P2P individually. In both their work they describe the specialization of the research as limitations.

More recently, Wang et al. published a very interesting research which also used NetFlow traffic. In contrast to previous work, they employed actual traffic logs from a large scale University campus network totaling 376Gb. In their work they also acknowledge that prior work on detecting malicious traffic is not scalable to large enterprise networks [133]. They however only focus on P2P botnet traffic and cluster by utilizing the MapReduce algorithm for aggregation in their BotCluster system. It merges unidirectional NetFlows to bidirectional and combines similar groups into sessions. They state that the clustered groups can be considered malicious, as only human designed malware would generate similar patterns in network traces. The experiments they ran show an average 97,58% detection rate. This number is achieved by filtering their data with whitelists, which could create a bias and even filter malicious content as these lists might also include malicious behavior. They do not elaborate on this in their discussion or conclusion. Nonetheless their approach will most likely inspire other researchers, as using NetFlow data for research in this field could be the trend for the coming years.

2.3. Cyber Threat Intelligence feeds

Previous sections have elaborated on prior work done on malicious infrastructures and detection solutions presented throughout the years. Cybercriminals attack multiple industries, which at their turn do not always have the resources to institutionalize security management [63]. A lot of companies would most likely collect threat intelligence solely of attacks on their network or spam e-mails they repel, but a single company collecting this data will only provide a small footprint of reality. For this reason, collecting threat intelligence data has become a niche, filtering and curating quality information about current threats. Security companies or researchers with access to information about current attack vectors on a large scale, sometimes offer this for sale or even as open source intelligence. The latter seems like an ideal solution for creating a more protected and safer world. Research on the quality and evaluation of such threat intelligence feeds are sparse, most prior work has been done on interpreting and gathering threat intelligence feeds.

According to Pawliǹski and Kompanek [97] there are multiple dimensions in which the quality of a threat intelligence feed can be measured. In their presentation, they establish eight criteria with which threat intelligence feeds can be evaluated. These criteria are categorized in *quality of information*, which includes the relevance, accuracy, completeness, timeliness and indigestibility of threat intelligence, and the *scope of the information source*, including the detection method, vantage and volume of collected data. In 2014, Kührer

et al. published a paper in which multiple blacklists are empirically analyzed [74]. For a number of domain lists this work identifies differences in active, parked and sinkholed domains. In their analysis they focus on measuring accuracy, completeness and timeliness of these blacklists. The results may be limited however, to our knowledge they are the first to attempt analyzing threat intelligence blacklists. Furthermore a Defcon presentation by Pinto and Maxwell [98] aims to measure the effectiveness of threat intelligence feeds in two dimensions. In this presentation, they show evaluations for the scope and accuracy of these feeds but do not go in depth into any analysis. Metcalf et al. focused on the volume and intersection of various IP and domain blacklists they aggregated, but they do not provide any metrics to evaluate their results [82].

After this, several studies have examined the effectiveness of blacklists for threat intelligence. The biggest limitation of these evaluations is that they are focused on specific types of threat intelligence and merely investigate their operational performance instead of giving an empirical evaluation. Sheng et al. for example looked into the effectiveness of phishing blacklists, finding that they are not resilient to short phishing campaigns [115]. On the other hand, Ramachandran et al. specifically studies threat intelligence regarding spam e-mails. In their results they say that these feeds are both incomplete and slow in responding to new emerging threats [106]. In a follow-up study, Sinha et al. analyzed four large spam feeds and concludes that they have a very high false negative rate [119].

Furthermore the work of Tounsi et al. states that there are still many limitations when it comes to threat intelligence. One of them is that there is too much information available, with over 250 million indicators per day. In their work they also say that threat intelligence available to enterprises is often out of data, and therefore not always useful [128]. In a survey in 2015 [99], authors state that most intelligence was not timely or specific enough. Additionally, 66% of respondents state that the information is not timely. This is further backed by a follow-up survey in 2018 [100]. Ring et al. mentions that real-time threat intelligence is expensive, and that there is a problem with sharing threat intelligence. This is why the author states that open source threat intelligence is not effective enough, or out of date [108].

2.4. Campaign analysis

Analyzing adversaries and accordingly clustering their activities lies within the research field called *campaign analysis*, which is surprisingly comparable with stylometry in the field of linguistic written language to attribute authorship to anonymous or disputed documents [58]. Whilst it is important to detect new malicious campaigns not previously known by the community based on algorithmic detection methods, it is imperative to analyze current threats and thoroughly understanding their nature and behavior. Previous campaign analyses have been done on multiple aspects of malicious behavior. About eleven years ago, Kanich et al. were one of the first to publish work analyzing malicious campaigns. In their work they have analyzed two *spam* campaigns; one designed to propagate the Storm malware and another one to market online pharmaceuticals [64]. To do this they analyzed about 500 million spam e-mails and looked at the number which successfully passed through popular anti-spam filters, how many users visited the advertised sites, and an estimation is given for the amount of sales and infections produced. They conclude that different campaigns use different tactics and marketing methods produces huge variations in malicious spreading. They also give an estimation that these campaigns produces a revenue of $2.731,88 over 26 days in their scope. Extrapolating this worldwide they estimate that the campaigns could have a revenue of about 3,5 million dollar on a yearly basis.

In another study published by McGrath et al. in the same year also analyzed a lot of spam data and specifically the phishing behind it [81]. Using results from detected phishing e-mails and domains, they combined *whois* records and *zone files* to gain insights about the machines hosting the phishing sites. They conclude that phishing domains and their URLs have different lengths compared to benign versions on the Internet. Also most of the time the targeted website is contained in the domain name of the phishing website. One year later, Kreibich et al. just like [64] investigated the Storm botnet and its infrastructure with the goal being able to say something about the sophistication of this botnet and its utilization [73]. They also used spam e-mails for the basis of their work. Interestingly, they developed software to request spamming workload from active Storm C&C servers. In their first conclusion they identified over 90 different campaign types used in the Storm botnet, all combined targeted over 630 million e-mail addresses and infected over 90 thousand machines to send spam. Their second conclusion elaborated on the diversity of the campaigns and how they utilize their infected machines to employ a high level of sophistication.

At this point it is clear that for threat analysis spam e-mails are popular amongst researchers to perform these kind of analyses. Gao et al. innovated and focused their campaign analysis on spamming accounts

on Open Social Networks (OSNs) like Facebook [44]. In their work they analyzed a large dataset of 187 million public wall posts from 3,5 million Facebook users. Using automated grouping techniques they created clusters employing similar behavior in terms of advertisement and text usage. Using these threshold-based techniques, they identified 200 thousand malicious posts spreading 57 thousand accounts. 70% of these posts contained phishing behavior and the majority of these accounts were not fake accounts, but benign users which were hacked.

Besides looking at different attack vectors, researchers now also use more advanced clustering and machine learning models to aggregate and estimate malicious campaigns. Thonnard & Dacier from Symantec showed a good example of this in their work, published in 2011 [127]. By using techniques relying on multi-criteria decision analysis, they show that some tight relationships exist among different botnet families, but also show some huge differences in spam campaigns performed by other infected machines. They do this by providing an analysis of group-level behavior of spam botnets, performing feature analysis using this to correlate infected hosts with each other. They have the advantage that at the time they worked at one of the largest anti-virus companies, providing them with amazing data to perform these analyses. They analyzed e-mail traffic, and the according source information, captured by their spam-trap for a period of 13 months containing about 1 million e-mails. This may seem less than the other researchers, however they also used locale and IP information on all of the occurring spam, something previous work could not analyze. Using the same dataset, other researchers from Symantec explored the visual options for this data [129]. They presented a graph-based interactive visual analytics tool to assist analysts in reasoning about various attack phenomena observed on the Internet.

More recently researchers started to use unsupervised learning, where the goal is to model underlying structures or distributions in the data. The work of Cao et al. employed such techniques to dig through OSNs, performing behavioral analysis on user accounts and clustering fake ones together [16]. They designed a generic and scalable detection system, called SynchroTrap, which is able to cluster large groups of users into specified actors. Coping with the huge amount of data on OSNs, they used Hadoop and Giraph to deal with this problem. During one month of deployment at Instagram and Facebook, SynchroTrap unveiled 1156 large campaigns and more than two million malicious accounts which were involved in these campaigns. In their discussion they stress that by extracting attack signatures from the malicious campaigns and accounts is necessary to be able to use supervised learning to develop fast classifiers that can detect attacks in real time.

Another mentionable campaign analysis was performed by Fachkha et al. in 2015, where they aimed to cluster victims targeted by the same DDoS campaign [37]. They proposed an approach to extract features from various DDoS attacks by creating numbers on time series and fluctuation, analysis techniques, statistical methods, and forecasting approaches. In their evaluation they presented three attack case studies to demonstrate possible extracted insights and inferences. One of these case studies for example showed the amount of unique bots involved in an attack over time. Unfortunately they do not have any interesting insights on behavioral similarities or show numbers of similar campaigns. In the same year, Santanna et al. published interesting work on DDoS Booter services, which are websites offering these DDoS attacks in exchange for money [111]. They were able to see which users accessed these website and used these services to launch attacks. From their analysis they conclude that Booter services have very distinct characteristics. They found that the majority of the users accessing the Booter services are only willing to pay less than $10 for attacks lasting about 5 minutes. This is in line with their hypothesis that Booters are used by adversaries with little skills. On the other hand, they also found malicious users hiding behind VPNs and proxies to hide their identities, which were willing to pay several hundreds of dollars to perform multiple attacks per day.

2.5. Research gaps

This Chapter has provided an overview of work which has previously been done in the field of malicious infrastructures. Both detection and analysis have been wildly adopted in published work, where detection is the dominant subject of the two. Since the first malware utilizing some kind of C&C infrastructure emerged in 1999, researchers have been chasing adversaries and their activities. Where IRC was popular in the early stages [25, 48, 65], more recent threats turn to advanced techniques to hide their activities [47, 53, 60]. However, most of the novel research mainly focuses on the creation of advanced and accurate botnet or C&C detection methods, but not on the actual threat landscape behind these adversaries.

In the 20 different studies performed on the detection and analysis of C&C infrastructure, most of them provided knowledge on more increasing levels of detection spanning multiple threat vectors. For each of these detection methods, not regarding the difficulty level, an evaluation has been made on the performance

Research	Year	Detection	Dataset	Scale	Analysis
Cooke et al. [25]	2005	IRC	Traces of one machine	Local	Found two known infections
Gu et al. [48]	2007	IRC	Traces 130 machines	Local	
Karasaridis et al. [65]	2007	IRC, C&C	Internal traces	Local	Size estimation of a few botnets
Gu et al. [50]	2008	IRC, HTTP	Traces of 130 machines	Possibly scalable	
Gu et al. [49]	2008	IRC, HTTP, P2P	Traces of University network	Possibly scalable	
Wang et al. [134]	2009	C&C traffic	Internal traces	Local	Activity of four botnets were detected
Nagaraja et al. [89]	2010	P2P	Internal traces	Local	Host detection
Zeng et al. [144]	2010	All bot traffic	Internal NetFlow	Local	
Francois et al. [42]	2011	P2P, HTTP	Limited ISP NetFlow	Not mentioned	
Zhang et al. [146]	2011	P2P	A few internal traces	Not scalable	41 P2P hosts detected
Barthakur et al. [7]	2012	P2P	Botnet flows	Local	Detected known P2P botnets
Han et al. [54]	2012	IRC	Traces of five machines	Local	
Zhang et al. [145]	2012	HTTP	Honeypot traces	Local	Drive-by downloads
Bilge et al. [12]	2012	C&C servers	2 weeks of ISP NetFlow	2x real time	
Zhang et al. [145]	2012	All bot traffic	Internal NetFlow	Local	
Garant et al. [45]	2013	All bot traffic	Created own botnet	Not mentioned	
Haddidi et al. [53]	2014	HTTP	Created NetFlow dataset	Scalable	Zeus traffic analysis
Grill et al. [47]	2015	DGA bots	Created NetFlow dataset	Local	DNS usage analysis
Homayoun et al. [60]	2018	IRC, P2P	Internal NetFlow	Not mentioned	
Wang et al. [133]	2018	P2P	376Gb University NetFlow	Scalable	Focused on performance analysis

Table 2.1: Summary of related work on detection and analysis

of the method. In general we can say that some sort of dataset is needed to perform such an evaluation. As can be seen in Table 2.1, more than half of the previous work has evaluated and tested their method only for local setups [7, 25, 47, 48, 54, 65, 134, 144–146]. These methods are mainly focused on detection on the client level, or within a small enclosed network. This is extremely necessary to protect users against a lot of attack vectors, regardless of the infection. These studies also performed evaluations based on the threats detected by their methods and provide specific analyses on this. Only a few publications step out of the local setup and observe from a more global scale [12, 133], where only Wang et al. actually provided some insights into a large flow-based dataset. From Table 2.1 we can conclude that scalable analyses can only be done on datasets containing more high level information like NetFlows. Bilge et al. were the only researchers which focused on global detection of C&C infrastructures, which unfortunately only performs at 2x real time, where Wang et al. actually showed that it is possible to deal with large NetFlow datasets but lacked the depth of analysis. In our analysis we will overcome the problem of detecting global C&C infrastructures in our dataset containing 16 months of global NetFlows of a tier 1 ISP provider.

Besides using advanced detection techniques to shield from attack vectors, public cyber threat intelligence has become an industry trying to bring the detected threats under the attention of users or companies. Companies sell their information on malicious IP addresses or domain names, so people can protect themselves from them. Some companies combine their knowledge and sell it, for example for a monthly fee, however some institutions provide open source threat intelligence in the spirit of creating a safer world. The analysis of these publicly availably open source feeds has only been done on subsets of feeds, as we have seen in [74, 82, 97, 106, 115, 119]. Most of these studies only provide an analysis regarding a snapshot of data. Using a novel method of analyzing NetFlow data for threat intelligence, we can look at the evolution and timeliness of these open source feeds, something which has not been done before.

Clustering malicious activities and identifying related actors is something which has only been done partially, or in specific scenarios. In this thesis we will try to overcome research gaps in global C&C detection by building on the work of Bilge et al. [12] and evaluating open source threat intelligence sources. Previous researchers have done campaign analysis, again more focused on specific topics like spam or phishing [16, 44, 64, 73, 81]. This thesis will also try to provide a complete picture of how cybercriminals use and manage C&C infrastructures throughout the entire Internet by utilizing every available and new forms of malware. To make this comprehendible, we propose campaign analysis on three network segments, each having their own structural possibilities and attack vectors. Infections on respectively clients, IoT devices and routers will be investigated for actors, utilization and quantification.

3

ISP NetFlows

In collaboration with a Tier 1 provider, we have collected an extensive NetFlow dataset covering the internet backbone of this provider for a period from September 2017 up to and including December 2018, spanning 3 million binary files totaling 4 terabyte (TB). Most of the analyses which have been made in this thesis, are based on this dataset. Any other datasets used throughout these thesis will be explained in the relevant chapters. Also an exploration of the dataset will be provided by looking into a subset of the global data to get a feeling of its value. Furthermore, this chapter will elaborate on the nature of this dataset, the tools which are necessary to deal with this dataset and the importance of NetFlow data in the realm of network security and cyber threat intelligence.

3.1. NetFlow background

The published origins of flow export date back to 1991, when the aggregation of packets into flows by means of packet header information was described in [83]. Subsequently, Cisco worked on its flow export technology named NetFlow, which finds its origin in switching. Routers can use packet-based or flow-based switching, where packet-based is the most accurate but flow-based is significantly faster. In flow-based switching, flow information is maintained in a flow cache and forwarding decisions are only made in the control plane of a networking device for the first packet of a flow. Subsequent packets are then switched exclusively in the data plane [18]. The value of the information available in the flow cache was only a secondary discovery and the next step to export this information proved to be relatively small [57]. NetFlow was patented by Cisco in 1996 and with this they aimed to make network traffic analysis easier and less resource intensive. Since its introduction, NetFlow has been widely adopted and implemented by researchers to inspect network traffic.

NetFlow defines a data type that summarizes packets exchanged between applications of two communicating end-points. Traffic from a source IP and port towards each combination of destination IP and port is being aggregated into separate flow records, allowing a high level analysis on all communicating processes mobilizing minimal resources compared to saving each packet. A flow record summarizes a traffic channel into a variety of statistics, of which the following selection is the most common: connection type, timestamp of packets, duration of flows, total number of bytes transferred and total number of packets transferred. Additionally, the NetFlow can contain network layer 3 header and routing information and level 4 TCP or UDP parameters as well as the source and destination Autonomous System (AS) number or the immediate neighbor AS (first AS of AS-Path) depending on the collecting setup. A typical NetFlow architectural setup can be seen in Figure 3.1.

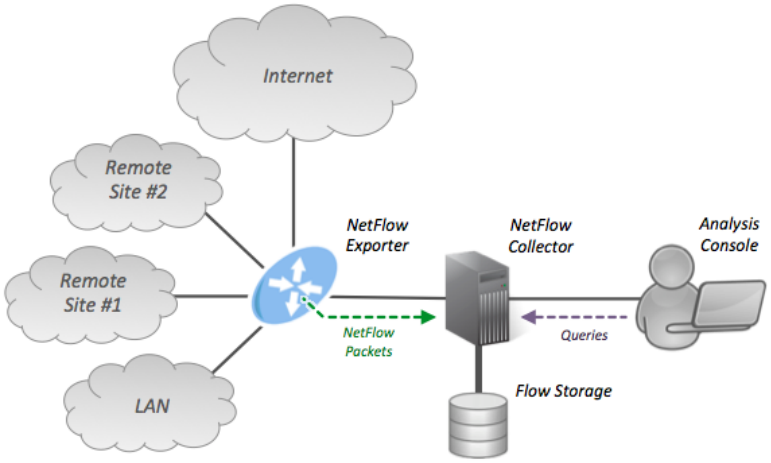

Figure 3.1: Typical network architecture when working with NetFlow technology [137]

This data format can be applied in various different scenarios. It can be used to track network usage and is thus often embraced by Internet Service Providers (ISP). NetFlow data can for example be used to calculate the amount of resources customers or users need or give a broader view on the overall usage of the network to see where improvements could be made. Besides this, NetFlow data can also be used to detect security vulnerabilities or network anomalies. For example, adversaries scanning worldwide for new vulnerabilities can be detected by looking into the amount of traffic towards specific ports. Or a network administrator could monitor when a machine within the internal network has been infected and is trying to contact a (known) malicious host outside the network. Obtaining NetFlows in a network requires two main components, an exporter and a collector. The exporter is a switch or a router which has NetFlow converting capabilities, whereas the collector is a centralized host on the network which receives the NetFlow records from the exporters and combines this. For large networks it is crucial to plan where the collectors are placed inside the network, as the extra traffic of sending the NetFlows over the network will most likely utilize more bandwidth. It is also important for such a setup to minimize packet loss between the exporter and the collector, as this could lead to gaps in the collected data.

3.1.1. NetFlow v9

Since the introduction of NetFlow in 1996, Cisco has released a number of versions. The first version only included IPv4 without IP mask and AS numbers. Version 5 is the most common one, enabled on most standard routers however, only has IPv4 enabled. The most recent version of Cisco's NetFlow is version 9 (v9), which supports summarizing more protocols than the older versions. When configured v9 can summarize additional information like Multiprotocol Label Switching (MPLS) labels, IPSec, and IPv6 addresses and ports. Each packet which is forwarded within a switch or router and has the NetFlow protocol enabled, is examined for a set of IP packet attributes which are used to identity and create a fingerprint of the packet and determine if the packet is unique or similar to other packets. The NetFlow protocol uses these attributes, which are listed in Table 3.1, to group all packets with the same source/destination IP address, source/destination ports, protocol interface and class of service into a flow. Amounts of packets and bytes are tallied into this NetFlow. A large amount of network information is condensed into a database of NetFlow, which makes this method of fingerprinting scalable. A lot of other information is stored within the NetFlow, as for example timestamps, next hop IP addresses, subnet masks and TCP flags. However, once packets are grouped into a flow, this also implies some restrictions for the data. For example, for the duration of the flow, the sizes of the packets are aggregated into one size for the entire flow. This has even more implications when the data is sampled, multiplying this aggregated number with the sampling ratio. A network administrator configuring the NetFlow collector can customize the attributes saved in the NetFlow data. They can, for example, choose for comprehensive information, saving all information flowing through the network, or when in situations with a lot of traffic they can choose for a more modest approach, sampling the flows and only saving specific information needed for analysis.

Attribute	Description
Source address	IP address of the sending host
Destination address	IP address of the receiving host
Source port	Port used by the sending host
Destination port	Port used by the receiving host
Layer 3 protocol	type Protocol used (TCP/UDP/ICMP)
Class of service	Priority value that can be used by QOS
Router or switch interface	Interface used on the device

Table 3.1: IP packet attributes on which NetFlows are grouped

3.1.2. Cisco tools

In order to effectively work with binary NetFlow data, Cisco has released a toolset called *nfdump* [51] [52]. The goal of this toolset was to be able to analyze NetFlow data from the past as well as to track current interesting traffic patterns. Each program in this toolset serves a specific purpose; from capturing raw network traffic and exporting it, to filtering the NetFlows for specific traffic.

nfcapd is the NetFlow capture daemon. This reads network information from the network and stores the data into files of, typically, 5 minutes. Each time-interval it rotates and renames the output file with the timestamp *nfcapd.YYYYMMddhhmm*. For example *nfcapd.201807300650* will contain data from July 30th 2018 06:50 until 06:55.
nfprofile also reads stored NetFlow data and can apply filters according to the specified filter sets and can store the filtered data into separate files. This program is optimized specifically for the visualization software called *NfSen* and can optimize the filtering for visualization purposes.
nfreplay can read NetFlows from the files stored by *nfcapd* and sends it over the network to another host. This can be helpful when analyzing the behavior of a specific application or recreating a (malicious) occurrence seen in the NetFlow data.
ft2nfdump can export flows in other formats, not defined by Cisco, into NetFlow format which can subsequently be processed by nfdump.
nfdump is the daemon which allows the user to read NetFlow data stored by *nfcapd*. It provides the user with a variety of options to print and filter the stored NetFlow data. Smart filters can be crafted to search for specific source or destination address/port. The name can be quite confusing, as the complete toolset is also called `nfdump`. References to `nfdump` in this thesis will from here on refer to the deamon.

As NetFlow was designed to efficiently store huge amounts of data, the tools to use this are required to be equally efficient. [57] discusses the speed of this toolset compared to using a DBMS for data management in terms of network information. Their measurement results indicate that *nfdump* is the best solution to query large network datasets when observing response time. This tool will be the basis of the analysis of our NetFlow dataset. An example of custom filtered NetFlow output is shown in Listing 3.1. Note that all IP addresses are

```
Date first seen          Proto       Src IP Addr:Port           Dst IP Addr:Port     Bytes
2018-09-01 10:02:55.298 TCP      239.169.82.154:80     ->     78.168.232.73:38807    12.2 M
2018-09-01 10:03:00.655 TCP      239.169.82.154:443    ->    78.168.205.110:50750    12.9 M
2018-09-01 10:03:27.420 TCP       86.210.55.89:443     ->    132.231.68.252:52923    425984
2018-09-01 10:03:03.278 TCP      62.156.229.110:443    ->     54.76.127.152:49462    12.2 M
2018-09-01 10:03:27.315 TCP      62.156.229.110:443    ->     54.153.90.165:61738    24.4 M
2018-09-01 10:03:08.966 TCP       162.60.140.81:443    ->      76.49.78.181:33083    425984
2018-09-01 10:03:16.072 UDP       199.0.197.93:45917   ->    207.72.183.154:25604    630784
2018-09-01 10:03:40.961 TCP      187.24.207.162:8080   ->     151.170.72.61:7125     327680
2018-09-01 10:03:22.411 TCP       155.49.46.147:80     ->    78.168.205.170:52494    12.2 M
2018-09-01 10:02:49.466 UDP     193.165.247.153:10488  ->   181.162.120.198:41688    983040
2018-09-01 10:03:23.025 ESP     193.127.189.158:0      ->    151.170.22.249:0         1.0 M
2018-09-01 10:02:48.844 TCP      239.169.82.153:80     ->     78.168.191.97:51100    24.4 M
2018-09-01 10:03:33.531 TCP       187.5.239.101:443    ->   173.126.209.248:46601    12.3 M
```

Listing 3.1: Example of NetFlow traffic

shown in *italic*, showing that these addresses are anonymized as will be elaborated upon in Section 3.2.4. It is interesting to see that the sizes of the flows seem pretty large. Already some interesting information can be observed. Some web traffic can be seen on TCP ports 80 and 443, and a flow using the ESP protocol which is used within the IPSec for providing authentication, integrity and confidentially of packets in IPv4 and IPv6 networks. Section 3.3 will go into more detail on the protocols used, the distribution of ports and filtering tactics for smooth analysis.

3.1.3. Sampling NetFlows

With today's and future high-speed links, capturing every packet and recording statistics of every flow require too much processing capacity, cache memory, and I/O and network bandwidth, in order to update, store, and export flow records. For these Internet backbone links, efficient and effective packet sampling are not only desirable, but also increasingly becoming a necessity. To cope with this issue, Cisco introduced a sampling method to save resources [20]. This sampling method uses only one in every n packets for computing statistics over the flows. From the subset of flows which are captured, the actual statistics are calculated and averaged with the sampling rate. This approach has fundamental limitations due to static sampling frequency. As the accuracy of estimation depends on the characteristics of traffic as well as the number of samples, sampling does not always ensure the accuracy of estimation as *Choi* and *Bhattacharyya* elaborated on in [20].

3.2. Data collection

For this research we have collaborated with a tier 1 ISP, to collect NetFlow data of their backbone. Before we dive into the amount of data that we are dealing with for this research, we first need to bring the topology of this ISPs network into picture to understand their coverage. This section will elaborate on the infrastructure on which our dataset is gathered, how it was collected and what privacy preserving technologies were applied.

3.2.1. Tier 1 Internet Service Provider

When visiting a website or an application, access to the Internet is most likely required, which is provided by an Internet Service Provider or *ISP*. An ISP is an organization that provides services for accessing, using, or participating in the Internet [136]. ISPs are organized in various forms, with the largest being commercial, but they can also be community-owned, non-profit, or otherwise privately owned. There are three levels of ISPs; tier 1, tier 2, and tier 3 providers, which all three pose an important role in providing Internet access. A tier 1 ISP is the largest of the three and can be best defined as a network that can reach every other network on the internet without paying for peering or having to purchasing IP transit [131]. The combination of all tier 1 providers in the world collectively represent the backbone of our interconnected society. All of these providers can freely transmit traffic between their networks, where some tier 2 and all tier 3 networks must pay to transmit traffic through these tier 1 providers.

3.2.2. Origin of our dataset

A tier 1 ISP controls a large number of routers to disperse traffic through its network and to other providers. A few of these routers are located at the network boundary, specialized in exchanging traffic with other operators, these routers are called *edge routers*. In total 35 of these routers are located in 26 different countries scattered mainly throughout Europe and North America, as shown in Figure 3.2. Each of these routers are configured to adopt the NetFlow protocol and save network traffic at a 1 to 8192 sampling rate to one central location. The location of these routers also shows an approximation of the coverage capacity their network has. As can be imagined, this infrastructure provides a lot of possible peering options with other providers. Most of these peering partners belong to the largest in the world.

3.2.3. Data restrictions

As can be imagined, the possibilities for research are huge seeing that this dataset encapsulates a lot of traffic passing through the backbone of the Internet. However, a dataset this large also comes with its restrictions. The first, and possibly the most important restriction regards the sampling of the dataset. As explained in Section 3.1.3, some operators may use the possibility to introduce a sampling rate for the collection. The ISP of our dataset has also employed this tactic, introducing a random sampling ratio of 1:8192. Care must be taken when sampling is introduced into data, however Choi et al. found that sampled NetFlows perform correctly without significant overhead under the load of their experiment [20]. The second restriction of our

Figure 3.2: Locations of the edge routers of the ISPs backbone

dataset implies the infrastructure on which our data is collected, which can be observed in Figure 3.2. In this figure we can see all the edge points in the ISPs network, suggesting a location based bias. This bias is however relative, as each dataset obtained from an ISP will encounter this problem. The only dataset which has the same amount of richness without a bias, would be a snapshot of all traffic passing through all existing ISPs, which is unfeasible to obtain. Another restriction is bound to the NetFlow collectors used by each edge router in the network. Each of these routers were iteratively added to the network at some point in time. Therefore, not all collectors have collected NetFlow data for the extended period of time, which may limit locality information. Finally we can see that the sizes of the flows seem pretty high, as Listing 3.1 shows. This is also due to the sampling ratio. When estimating the packet sizes throughout this thesis, we will divide the seen flow size by 8192.

3.2.4. Anonymization

Real-world traffic traces are crucial to conduct certain research, but only a very few real world traces are made public. The most decisive reason for companies or institutes to not make these traces publicly available, is that privacy sensitive information may be leaked within these traces. This information may be leaked by the IP addresses contained in these traces, which may point to the locations of users or organizations. In order to preserve the privacy, each NetFlow trace was anonymized using the technique described in [41] and obfuscated at the level of autonomous systems. This allowed us to quantify the activity of malicious endpoints without learning anything of about identity of the actual users.

CryptoPAn

While for NetFlow datasets only a deterministic, random one-to-one mapping of original to anonymized IP addresses is necessary to match outgoing requests with returning answers, in such blind randomization the relationship information of networks is lost. Thus, it is not possible to preserve locality information such as a C&C activity realized by several hosts in the same /24 subnet, as these hosts would be scattered across the entire IPv4 space. In this thesis, we use the method proposed by Xu et al., who introduced a random one-to-one mapping while preserving network information. If we represent network addresses in a binary tree which each bit of the IP address when read left to right will result in a transition to the left or right subtree under a node, an IP block under a shared prefix will be expressed as an entire subtree under one specific node. Consider the example in Figure 3.3(a), all IP addresses in the prefix P_1 start with the digits ”00” in their address, while IP addresses in the adjacent address block begin with ”01”. Under each leaf node – which are marked in grey – are then all IP addresses associated with this particular IP allocation.

In Xu et al.’s *Cryptography-based Prefix-preserving Anonymization* (CryptoPAn) [38, 140] scheme, the bit value of every non-leaf node is flipped randomly. This means that if two IP addresses shared a k-bit prefix, also the anonymized IP addresses will share an identical, but now randomized k-bit prefix. Within each netblock, IP addresses can now be scrambled without losing information about the logical coherence of the addresses to one provider, and prefix-preserving anonymization comes in handy for the evaluation of threat intelligence feeds as related activity is often located in adjacent IP addresses or subnet blocks as we will show later. The randomness in CryptoPAn is drawn from the AES block cipher, and a short encryption key is thus sufficient

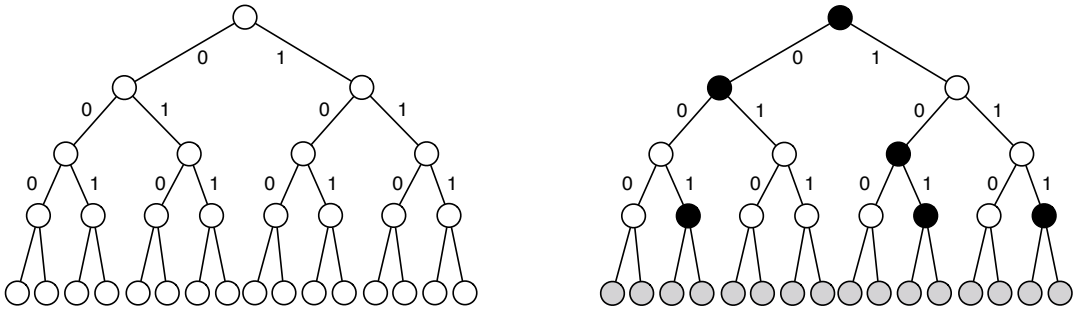

Figure 3.3: Prefix-preserving randomization after Xu et al. [140][38]

to provide an effective IP randomization function. This also allows multiple traces to be sanitized in a consistent way, over time and across locations. For example, the same IP address in different traces converges to the same anonymized address, even though the files might be anonymized separately at different times or locations. The authors prove in [140] and in [38] that this scheme delivers semantic security.

The anonymization of NetFlows was done on site at the ISP using a secret key chosen by the operator, in such a way that only obfuscated data was analyzed within the context of this thesis, thus preserving the identity of Internet users. In order to match the information on malicious activity from threat intelligence feeds, or other sources, to the traffic patterns in the NetFlow dataset, we were able to anonymize IP addresses on-site at the Tier 1 operator to enable the analyses presented in this thesis. Furthermore it is important to notice a difference in emphasis of anonymized and actual IP addresses. Anonymized IP addresses will always be written with *italic* emphasis. For example *32.56.112.32* refers to an anonymized IP addresses, where 131.180.77.82 refers to an actual IP address.

3.3. Data Exploration

The amount of traffic flowing through the backbone of an ISP provider can grow considerably, which proves to be a problem when performing data analysis even when the data is sampled. For this research it is imperative to gain understanding about what kind of data we are dealing with, such as the different Internet protocols observed and distribution of traffic towards ports in this dataset. This section will first elaborate on how to handle such a massive dataset. Filtering must be done efficiently, otherwise it could take up to months to perform simple analyses. On the other hand exploration of the data is provided to give insights into the richness of the dataset. For simplicity these analyses and examples will be focused on the NetFlows from one day; the 1^{st} of September, 2018. The traffic which passed through the network on this day will be closely examined.

3.3.1. Filtering approach

As the total NetFlow dataset contains information about everything flowing through the backbone, it will be necessary to be able to filter for specific parameters. It is also important to again mention that the NetFlow files are not text-based files, but are binary files designed for efficient storage. In order to filter these binary files, Cisco provides a toolset called *nfdump* as described in Section 3.1.2. The syntax of this program is powerful, comparable to tcpdump. It gives the user the ability to filter NetFlow files for any element provided in the v9 format and output the filtered result to either a binary file or as text, as shown in Figure 3.4.

The filter syntax is straightforward; each filter contains one or multiple expressions, which can be combined by the logical operators *and*, *or* and *not*. The expressions consist of elements from the v9 format, for example *destination port*, a value and a comparison operator to combine them. Following from this, '`dst port = 80`' is an example of a simple filter which filters for the destination port 80, whereas a more complex example '`inet6 and proto tcp and (src port > 1024 and dst port 80)`' will provide all IPv6 connections to any web server. Next to rudimentary filtering, nfdump also has the possibility to aggregate traffic based on any elements and produce statistics. This is especially helpful when dealing with large datasets. Listing 3.2 shows an example of the top five IP addresses on the 1^{st} of September, 2018, ordered by amount of bytes sent and received. From the summary we can also learn that on this day, a total of 4129.9TB has flowed

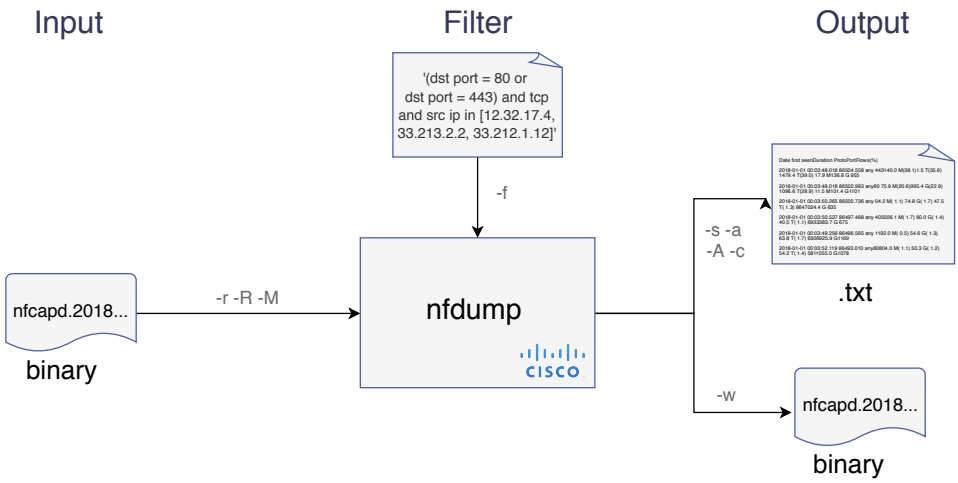

Figure 3.4: Flowchart for filtering NetFlow files with *nfdump*

through the network.

```
Top 5 IP Addr ordered by bytes:
Date first seen          Duration Proto        IP Addr   Flows(%)      Packets(%)        Bytes(%)      pps      bps
2018-09-01 00:03:48.283 86499.073 any      61.68.160.114  6.0 M( 1.6)  67.7 G( 1.6)  77.8 T( 2.1)   783227  7.2 G
2018-09-01 00:03:55.266 86481.341 any      62.66.191.246  2.4 M( 0.6)  54.0 G( 1.2)  54.5 T( 1.4)   624781  5.0 G
2018-09-01 00:03:55.266 86482.490 any      62.66.191.245  2.0 M( 0.6)  48.8 G( 1.1)  49.1 T( 1.3)   564590  4.5 G
2018-09-01 00:04:15.418 86472.473 any     103.188.110.118 2.2 M( 0.6)  26.7 G( 0.6)  26.7 T( 0.7)   308541  2.5 G
2018-09-01 00:03:50.527 86495.562 any     104.134.155.9   3.5 M( 0.9)  34.8 G( 0.8)  26.7 T( 0.7)   402431  2.5 G
IP addresses anonymised
Summary: total flows: 349182574, total bytes: 4129.9 T, total packets: 4.5 T, avg bps: 350.8 G, avg pps: 50.3 M
Time window: 2018-09-01 00:03:48 - 2018-09-02 00:05:32
Total flows processed: 349182574, Blocks skipped: 0, Bytes read: 20590605736
Sys: 618.664s flows/second: 594317.1   Wall: 629.152s flows/second: 584409.1
```

Listing 3.2: Example nfdump output providing the top 5 IP addresses by bytes

3.3.2. Protocol distribution

After grasping the syntax needed for dealing with the data, we can start to dig into the data to get a feeling of its usage and size. We will start off by looking at the network usage per protocol. These protocols are used in the Internet Protocol (**IP**), which is the fundamental communications protocol and is able to send datagrams across network boundaries. IP transfers packets from the source host to the destination using the IP addresses in the packet headers. This is why different packet structures, or protocols, are designed which enclose the data. When a router receives a packet destined for itself, the protocol field is used to identify what kind of data has been sent and how the router should interpret it. The top ten protocols observed for this exploration are shown in Table 3.2.

The first observation we make is that the TCP and UDP protocols are responsible for almost all of the traffic. TCP, or Transmission Control Protocol, is the most commonly used protocol on the Internet. When requesting a web page or using an application, the client sends TCP packets to the web server responsible for that web page or application. The web server then answers with TCP packets to form the page or application. This protocol is so popular and globally used because it is a structure based on reliability. Each of these TCP packets if tracked in such a way that no traffic is lost. File downloads are a good example of this reliability. When downloading a file, these documents do not get corrupted directly when a small hiccup in the network occurs. The second most popular protocol used is UDP, or User Datagram Protocol. This protocol is basically designed in a similar manner as TCP, but without the error checking which makes TCP reliable. UDP traffic can be sent a lot faster across the network as there is no checking back-and-forth if a packet is received without problems. When an application is using UDP, the traffic is just sent to the receiver. The sender will not check if the packets have arrived at the destination, but instead will just send the next packets. This protocol will therefore not be a good solution for mail traffic, but find its application more in streaming services like online broadcasts and gaming. When packets are lost whilst watching a live video stream, it can be preferred

Protocol	Flows(%)	Packets(%)	Bytes(%)
TCP	73.4	75.3	85.1
UDP	25.8	23.3	13.9
ESP	0.5	1.0	0.8
GRE	0.1	0.3	0.1
ICMP	0.3	0.2	< 0.1
IPIP	< 0.1	< 0.1	< 0.1
L2TP	< 0.1	< 0.1	< 0.1
IPv6	< 0.1	< 0.1	< 0.1
AH	< 0.1	< 0.1	< 0.1
VRRP	< 0.1	< 0.1	< 0.1

Table 3.2: Top 10 protocols ordered by packets

to continue with the live broadcast instead of waiting for those lost packets.

Two other protocols which have considerable amounts of data are the ESP and GRE protocols. ESP is an important protocol for security in our society as this, also known as IPsec, encrypts all packets sent through this protocol. Both TCP and UDP can use this protocol, but does create significant overhead when also encrypting packets on other levels. GRE, or Generic Routing Encapsulation, on the other hand was developed by Cisco as a tunneling tool to carry packets over an IP network. In essence, this protocol creates a private point-to-point connection often used in virtual private networks (VPN). GRE often encapsulates the payload of a packet inside another packet before sending it, where routers will not be able to read the intermediate packet. GRE is often combined with IPsec to also make sure the inner packet is encrypted. Furthermore it is interesting to see the adoption of IPv6 in Table 3.2. IPv6 is the newest design of the Internet Protocol, more secure and with more options than its predecessor IPv4, waiting for embracement since its introduction in 1990. From this data we can say that is it not yet being adopted as much as the community would hope.

3.3.3. Port distribution

Each connection, or flow, passing through the NetFlow data has a source and destination pair constructed by a pair consisting of an IP address and a port. Ports are logical constructs which identify a specific process or a type of network service. Such a port is always combined with an IP address sending or receiving on that port, as well as with a protocol. Combining this information gives the network address of a flow. Table 3.3 and Table 3.4 provide the top 15 ports receiving and sending information in the NetFlow dataset for September 1, 2019. A lot of interesting information can be deducted from these tables however, we will only elaborate on a few interesting points.

Immediately we can see that port 80 and 443 on both source and destination ports come out on top. Together they contribute to 49% of all packets and 70,3% of all bytes sent through the backbone. This makes sense, as these ports employ the HTTP and HTTPS protocols and are used to send web traffic which most websites and applications use. Port 443 can be seen as the more secure version of port 80, as this facilitates the HTTP protocol but over TLS/SSL and securing the traffic between source and destination. Ultimately it would be good to see a decrease of traffic on port 80 and an increasing amount of traffic on port 443. The ports 8080 and 8081 are mostly used as alternative ports for port 80, which explains the amount of traffic they send. At a first glance it is odd to see that traffic originating from HTTP and HTTPS is higher compared to the destination statistics. However, when requesting a website the packet size can be quite small, just a message indicating the browser would like to download the necessary files to view the website. The response on the other side would be larger, because the server would need to send the files to the user. Where port 443 provides more cryptographic security, port 4500 is mostly used for IPSec commands, needed to successfully create a VPN-tunnel and provide a certain level of anonymity for the user. Port 1194 is another example of a VPN protocol, OpenVPN. Port 0 is an interesting case, tailing port 80 and 443 on a third place. The interesting part about port 0 is that it officially does not exist [26]. The designers of the original port interfaces, upon which much of the technology and practice we use today is based, created port 0 to be used as a sort of a wild card port. When designing the port interfaces, port zero is mostly used to "let the system choose one for me". So port 0 was set aside and never defined or used and up until now traffic on port 0 remains a kind of no man's land. Finally it is interesting to mention ports 6881, 8999 and 51413, used mainly for the BitTorrent enabling users to efficiently share documents or programs.

Src Port	Flows(%)	Packets(%)	Bytes(%)
443	29.7	27.6	36.9
80	15.6	21.4	33.4
0	1.0	1.6	1.1
2200	1.0	0.7	1.1
25461	0.5	0.6	1.0
8000	0.3	0.4	0.6
8080	0.3	0.4	0.5
4500	0.4	0.5	0.5
8999	0.5	0.5	0.4
51413	0.3	0.3	0.4
119	0.2	0.3	0.4
563	0.1	0.2	0.3
1935	0.3	0.3	0.2
8081	0.2	0.1	0.2
1194	0.1	0.2	0.2

Table 3.3: Top 15 source ports ordered by packets

Dst Port	Flows(%)	Packets(%)	Bytes(%)
443	9.4	7.3	2.2
80	5.5	6.2	0.5
0	0.8	1.5	1.1
119	0.0	0.7	1.2
8999	0.4	0.5	0.4
4500	0.3	0.4	0.3
6881	0.4	0.3	0.2
40500	0.4	0.3	0.2
3074	0.4	0.3	0.1
51413	0.3	0.3	0.2
9306	0.4	0.2	0.0
25461	0.2	0.2	0.0
16393	0.1	0.2	0.2
50321	0.2	0.2	0.1
1194	0.1	0.1	0.1

Table 3.4: Top 15 destination ports ordered by packets

3.3.4. Visualizing activity

For any campaign analysis it is important to provide the reader with a wide variety of information. Visualizations are a great way to compress a lot of information. In our case, we want to be able to visualize adversarial activity in an orderly fashion. As it takes a long time to analyze the NetFlow data, it is useful to be able to make simple representations of data without having to iterate through each of the files each time. Precisely for this reason we decided to created aggregated rudimental information about IP addresses and ports. For both the IP addresses and ports we summarized flow activity passing through on a daily basis. Besides this we also aggregated the unique connections and amount of different packet sizes used per day, which will be elaborated on in Chapter 5. The aggregated flow information proved extremely helpful understanding different patterns adversaries employ. Figure 3.5 uses the summarized flow information to provide a distribution of connections each IP address has per day, providing a visualization not seen in any previous work. As expected, most IP addresses have little connections, suggesting clients, and few IP addresses have a lot of connections, which are probably large applications used by lots of users. Figure 3.6 on the other hand provides a simple scatterplot, showing the amount of activity an IP address has over time. As we will see in the next sections, this will prove to be very useful to understand adversarial patterns.

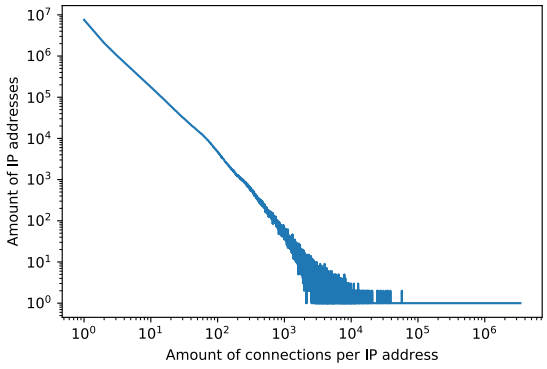

Figure 3.5: Distribution of connections per IP address

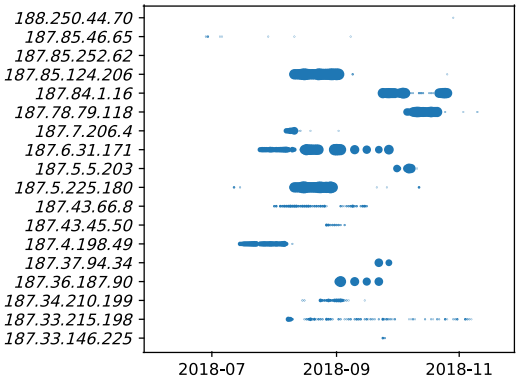

Figure 3.6: Example visualization of IP activity

3.4. Value for research

Network traffic is a data source which is relatively easy to collect. In theory, everybody with a router at home can capture each packet flowing through their network, or set-up the NetFlow protocol to perform superficial traffic analysis. In terms of research, the analysis of network traffic is also widely adopted. This is also the case for analysis using NetFlow data, as this can be used to observe high level patterns with relative ease. Since the release of NetFlow, over 1300 publications have been made using NetFlow data in the field [30]. However,

most of these publications have used traffic originating from internal networks. Since 2000 only 30 publications have been made with NetFlow data on an ISP or tier 1 scale. A lot of these publications also merely have a limited amount of NetFlow data. For example [12] has two weeks of sampled NetFlow data, which they obtained from a tier 1 ISP. We have not found a publication which has been able to perform research on more than two weeks of unabridged NetFlow traffic of the internet backbone of a tier 1 ISP provider. This allows us to follow traffic patterns for a long period of time and dive into the underlying patterns of adversarial structures.

3.4.1. Analysis possibilities

As this thesis focuses on malicious campaigns, it is interesting to see what kind of adversarial activity can be seen. Section 3.3 has provided a brief overview of basic activity in the NetFlow dataset, but it would be good to dive into examples of malicious activity to evaluate how much we see. To illustrate this, we looked into two attacks in our dataset to view the activities. First, we looked into an attack on the Android Debug Bridge active on the vulnerable port 5555, as can be seen in Figure 3.7. An enormous spike can be seen in early July, the same dates as [69] mentions attacks happening on their network. The article did not mention a few smaller spikes, which could indicate zero day vulnerabilities being explored by adversaries. Secondly, Figure 3.8 on the other hand provides insights into the port which caused the largest DDoS attack recorded until now [71]. Port 11211 was rendered vulnerable and as we can see in the Figure was exploited lots of times.

These two examples scratch the surface of the possibilities this NetFlow dataset can bring us. This Section has provided an overview of NetFlow data, how the dataset is structured and a brief overview of the opportunities which arise.

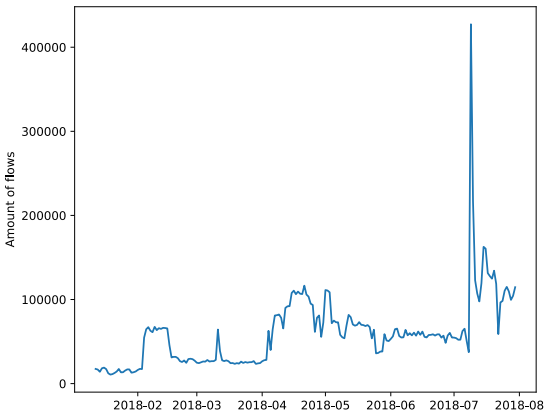

Figure 3.7: Attack on Android Debug Bridge using port 5555

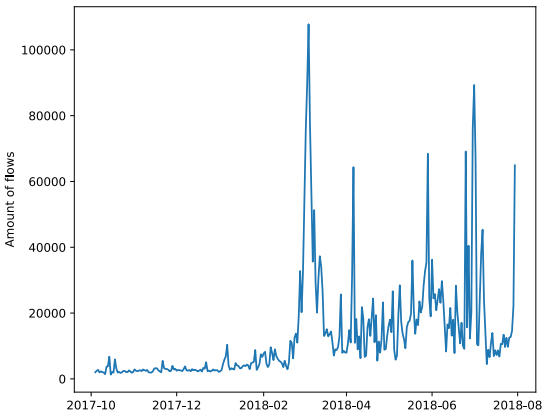

Figure 3.8: Memcache activity on port 11211

4

Exploration of Cyber Threat Intelligence Feeds

In the introduction of this thesis, in Section 1.4, we have posed a research question towards the tracking of malicious C&C infrastructures. In order to answer this research question we must be able to analyze specific *indicators of compromise* helping us to gain a foothold of these infrastructures in our data. The most obvious choice for this information would be specific intelligence sources which have these indicators of compromise, for example IP addresses or domain names. These sources could point to specific IP addresses in our data, enabling us to gain more insight into the underlying patterns and structures. Open source threat intelligence feeds are the only publicly available resources which contain these indicators of compromise. Companies also sell this kind of intelligence for large profits, however we will only focus on publicly available sources.

These open source intelligence feeds have multiple origins, as a variety of people and organizations contribute to this knowledge. Before using these feeds as a basis for our analysis, it is imperative to evaluate to what extent this information can be used. This chapter will provide an extensive evaluation on 24 of these *cyber threat intelligence* (CTI) feeds and determine the coverage of this combined intelligence, answering the question whether these intelligence feeds will prove to be a sufficient foothold for analyzing malicious infrastructures.

In Section 4.2, we have developed evaluation criteria for a quality assessment of cyber threat intelligence that will be used for this exploration. Section 4.1 describes the collected open source intelligence feeds, while Section 4.3 describes their utility in terms of relevance, timeliness, completeness and accuracy. Section 4.4 evaluates the adoption of these sources in practice and the benefit they bring to networks, as well as summarizing the findings and elaborating on the next steps.

4.1. Datasets

Goal of this exploration is to evaluate the quality of cyber threat intelligence feeds. For this purpose, we have monitored a total of 24 open source feeds which blacklist domain names as well as IP addresses based on detected malicious activities, annotated into major categories such as C&C server activity and usage as a phishing domains. These feeds have been continuously monitored over a period of 7 months from August 1, 2018 until February 28, 2019, and when available also all historical records back until January 1, 2018. This resulted in a total of 1,383,040 indicators which we have used. For our evaluation, we monitored 17 threat intelligence feeds over a period of 14 months, and 7 feeds over a period of 7 months. In table 4.1, we have briefly enumerated each of the feeds included in this analysis.

Active domain crawls Based on zone transfers on registered domains from ICANN and national domain registries, we have crawled approximately 277 million unique domains across 1151 generic and country code top level domains on a daily basis. This data shows which IP address was connected to which domain at any day, giving our original NetFlow dataset additional depth.

TI Feed	Automated	Months	# of IPs	TI Feed	Automated	Months	# of IPs
Badips	Yes	14	95	Emerging Threats	No	14	10,464
Bambenek	Yes	14	1,796	Greensnow	No	14	116,748
Blocklist.de	Hybrid	14	944,622	MalwareConfig	Yes	14	19
BotScout Bot List	No	14	1,564	Malwaredomainlist	Yes	14	1,011
Botvrij	No	14	95	Myip	No	7	55,936
BruteForceBlocker	No	14	4,663	Nothink	Yes	7	42
CI Army IP	Hybrid	14	181,439	Phishtank	Yes	14	2,708
CINSscore	Hybrid	14	250,153	Ransomwaretracker	Hybrid	14	383
Charles the Haleys	No	7	38,999	Rutgers	Yes	14	112,898
Cruzit	No	7	49,911	Talos	Hybrid	7	2683
Danger.rulez	No	7	3,099	Tech. Blogs and Reports	Yes	14	6,151
Dshield	No	14	106	Zeustracker	Yes	7	112

Table 4.1: List of evaluated open source feeds

4.2. Criteria for Threat Intelligence

In this section, we will describe four different criteria, which we will use in to evaluate the quality of open source threat intelligence feeds. As discussed in the related work, Pawliǹski and Kompanek [97] have proposed at an industry forum a taxonomy to benchmark threat intelligence along the dimensions of (a) relevance, (b) accuracy, (c) completeness, (d) timeliness, and (e) ingestibility. We find this classification however problematic, as several of the criteria are entangled. For example, in machine learning and pattern recognition domains, relevance is usually measured by precision and recall. In other words, how many of the selected items in a dataset are correctly identified, and how many of the relevant items are found in the dataset, respectively. Recall however also partially assesses similar aspects as completeness, so quantification results would contain some degree of correlation. Along the same lines, accuracy is also widely used concept in machine learning, and in binary classification measures the ratio of true results to all examined data, or $\frac{TP+TN}{TP+TN+FP+FN}$. While threat intelligence is a classification task, classifying activity as either malicious or non-malicious, threat intelligence *feeds* are not classification tasks, but should mainly contain information from one label. Therefore, a binary accuracy characterization does not work well due to imbalance of the data present in the feeds. For these reasons, we propose different metrics to measure the quality of these feeds.

Our Taxonomy for CTI Quality

In order to evaluate the quality of cyber threat intelligence, we propose a set of four metrics: *timeliness*, *sensitivity*, *originality* and *impact*.

1. *Timeliness.* The goal of subscribing to a threat intelligence feed is to obtain early warning of some emergent malicious activity, so that infections in the local area network can be stopped in time before significant losses are incurred. Hence, the earlier indicators such as IP addresses or domain names are flagged, the higher the utility of the feed is to the subscriber, and in turn we can also conclude the better the quality of the provided information. One essential quality criteria of a threat intelligence feed is thus the timeliness of the information posted, in other words how soon a domain or IP address is included in such lists after it has started malicious activities. A high timeliness will minimize the amount of damage that could be incurred as part of a compromise, as it shortens the time window during which hosts may be under adversarial control and the time an adversary may for example exfiltrate data or abuse the infected client.

2. *Sensitivity.* In order to be included into a feed, the threat intelligence provider has to observe some malicious activity in the first place. This is typically done using a variety of sensors, recording network traffic patterns, DNS lookups, as well as for example based on the forensic analysis of malware samples. If a particular malware instance, C&C server, or maliciously acting host shows only low, sporadic activity, there is a high likelihood that it would not be seen by a provider and thereby go by unmitigated until the problem grows above a certain threshold.

   With sensitivity, we therefore assess what volume is necessary for the intelligence provider to take notice of a malicious activity, in other words what is the average and typical minimum threshold at which detection will take place. In addition to quantifying the overall per-feed threshold, we can also measure the sensitivity of a threat intelligence feed with respect to a geographical focus: if a provider predominantly has sensors in a specific region, detection will be biased against threats emerging or deployed

in this particular area, while comparatively insensitive towards threats originating outside of the measurement coverage. As Internet threats by definition operate worldwide, heavy geographical biases therefore introduce a significant risk of getting hit unprepared.

3. *Originality.* In practice, an organization would likely subscribe to several threat intelligence feeds, as CTI providers often specialize towards a particular type of threats. We also see this behavior in threat intelligence providers themselves, who – as we have said earlier – are also often aggregating, curating and repacking other sources to be marketed as their own service. An essential metric of a cyber threat intelligence feed is therefore originality, in other words the amount of information that is unique to this particular source and that could not be obtained otherwise.

   While originality measure the contribution made by one specific feed, it can also be used as a metric to quantify an ecosystem of intelligence feeds as a whole. Consider a number of k feeds which all report malware C&C servers. If all indicators provided by these feeds are highly unique, in other words there is no or only limited overlap between them, this also means that even their union provides only an insufficient peak at the population of C&C servers. We can thus say that in case of high ecosystem-wide originality each feed only draws for samples from a large problem space, and in these cases the set of intelligence feeds is unsuited to provide sufficient defense against this particular type of threat.

4. *Impact.* When an organization applies the information obtained from the threat intelligence feeds, this should lead to a mitigation of a particular threat, as connections to and from a malicious host are suppressed and no command & control activity or an initial infection should happen anymore. Based this positive impact, an application of the threat information can also have negative consequences, especially if the information is not specific enough or contains false positives.

   The former is particularly of concern if feeds only provide IP address information, such as the IP address a command & control server is currently hosted at. While in times of DGAs indicators such as domain names have an extremely short lifetime, in many circumstances an actor will not host malicious infrastructure on a dedicated machine, but rather employ the services of commercial vendors as this offer much higher flexibility and incurs no loss (expect for the forfeiture of prepaid service) such as the seizure of own hardware. This however also means that at particular IP address that is flagged as malicious other services may be present which are then also blocked as collateral damage.

   Our metric impact measures the consequences to an organization if the information from a threat intelligence feed is applied, for example by blocking IP addresses in the firewall. This can have both positive and negative consequences, and we care whether all of the malicious activity will be suppressed given the feed's data, and whether it *only* covers malicious activity or the application will also cause harm to benign services. For example, if a malware communicates with its C&C server using 10 IP addresses, the blockage is only really successful and useful if all 10 addresses are included in the feed as otherwise the activity simply continues using an alternative channel, and only these 10 addresses are blocked.

4.3. Evaluating the feeds

Based on the criteria introduced in Section 4.2, in this section we discuss the results of the quality evaluation of the 24 tested cyber threat intelligence feeds. The following subsections will first review their performance in terms of timeliness, sensitivity, originality and impact, before in Section 4.4 we will in further detail analyze the question of their overall utility and adoption in practice.

4.3.1. Timeliness

The goal of subscribing to a threat intelligence feed is to obtain early warning of some emergent malicious activity, so that infections in the local area network can be stopped in time before significant losses are incurred. Hence, the earlier indicators such as IP addresses or domain names are flagged, the higher the utility of the feed is to the subscriber, and in turn we can also conclude the better the quality of the provided information.

In this section, we are assessing the timeliness of cyber threat intelligence feeds based on the amount of traffic a particular destination has received, prior and after it was included in the analyzed feeds. Figure 4.1 depicts connections within the Tier 1 network to seven exemplary destination IP addresses between July 2018 and January 2019 that were in the second half of 2018 flagged as malicious. For each day, we aggregated flows from distinct clients towards each destination, the size of each circle shows in logarithmic scale the total number of recorded flows. Note that the IP addresses are anonymized as discussed in Chapter 3: while

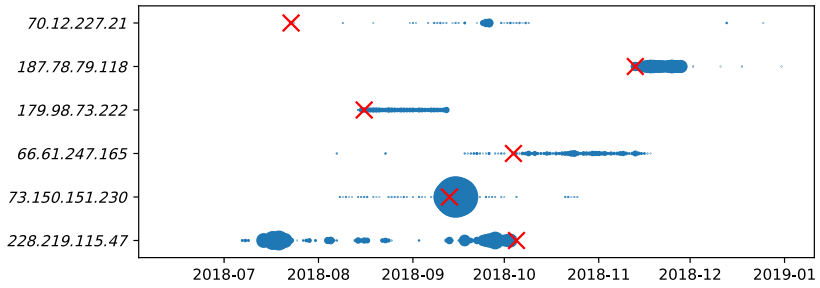

Figure 4.1: Scatter plot of netflow activity. The size of the line shows the amount of traffic observed. Crosses denote when the IP address was blacklisted.

the anonymization protocol matched the feed indicators to the IP addresses, the shown IP addresses are randomized at the level of prefixes. Thus, no conclusion can be taken about the concrete IP addresses at hand or their location in the world.

As we see in the graph, we find that activity on IP addresses and their appearance in intelligence feeds frequently diverges significantly in practice. The first three IP addresses in the figure are examples of a very timely detection – the IP addresses are reported as soon as the first activity arises, and in the first case even months before significant botnet traffic appears towards this C&C server. Not every intelligence report is however as successful. In the fourth and fifth case, the IP addresses are active for several weeks prior to reporting, and in case of *73.150.151.230* it is only marked as malicious after a significant traffic volume emerges. An even worse outcome is shown just below in case of *228.219.115.47*: while after the including of the IP address in the threat feeds activity abruptly stops, the IP address had been active for almost 3 months prior, and been engaged in thousands of connections with clients.

We conducted this analysis for all 1,383,040 indicators across the 24 threat intelligence feeds and counted the number of consecutive days IP addresses have received activity from clients before they were included in a particular list. Based on this analysis, we find that the first examples of successful indication in Figure 4.1 are the exception rather than the norm, surprisingly we find that it takes on average 21 days before indicators are included in a list.

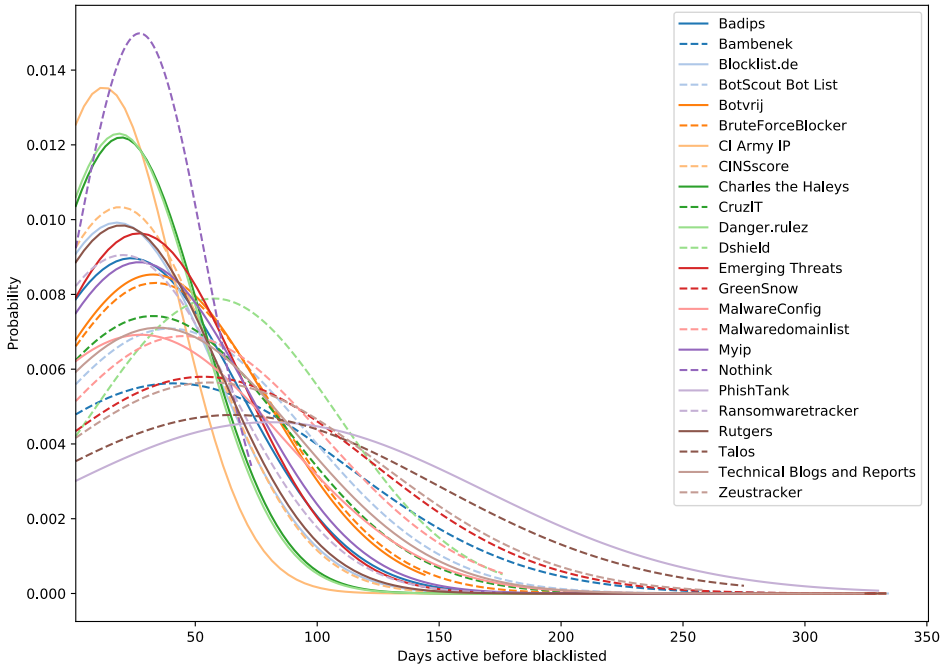

Figure 4.2: Hosts are routinely active for multiple weeks before a destination is marked as malicious by threat intelligence feeds.

Figure 4.2 splits this analysis, where each curve shows in a probability density function the number of active days until listed by an individual provider. As can be seen in the graph, a handful of feeds are clearly leading the pack, with the response time of CI Army being about 50% better than the overall average. In the collection of feeds, we surprisingly find high homogeneity and overall slow inclusion of malicious sources into the feeds. This while having turnaround times of approximately one month on average. Even lists commonly praised by practitioners as “high quality” or “industry standards”, such as the widely used *Emerging Threats* score surprisingly average in this respect. At the lower end of the scale, we have already observed activity for on average 65 to 80 days, before the slowest to respond feeds – Talos and PhishTank – include these IP address in their reports as malicious. This lag between the emergence of malicious activity and the inclusion in the threat intelligence feeds might be due to a slow update frequency. To investigate this hypothesis, we analyzed the inter-arrival time when information was included across the 24 feeds, Figure 4.3 shows a cumulative density function of the time in between list updates. As we can see from the graph, information is pushed at a very high frequency to the portfolio of lists, in one third of the cases updates occur at least hourly, while approximately two thirds of items are updated at a granularity of at least once per day. Thus, it is not the processing of the lists where reporting latency occurs, but during the selection and preparation of indicators.

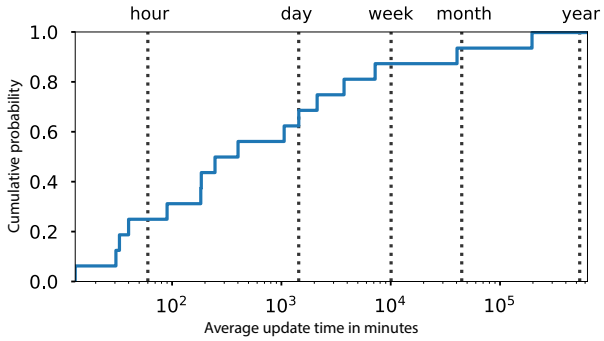

Figure 4.3: CDF of update frequency of the evaluated lists. Two thirds of the threat intelligence feeds are updated a least once a day, one thirds even includes new indicators in hourly intervals.

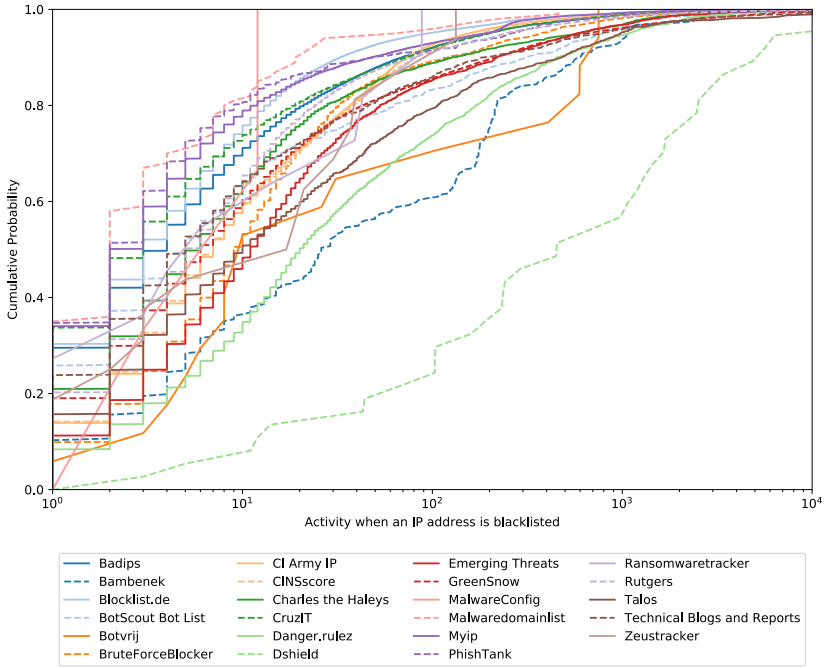

Figure 4.4: Cumulation density function of the minimum activity before an IP addresses is included in a threat intelligence feed.

4.3.2. Sensitivity

As we have already seen above, there seems to be a significant deviation between feeds in how soon indicators are included after the first sign of network activity. Figure 4.4 lists a cumulative density function of the number of connections we observe before an address is included in a particular intelligence feed as malicious. While the majority of lists is surprisingly homogeneous in their sensitivity – we see that the bulk of them triggers with 50% likelihood if at least 6*8192 - 10*8192 flows are recorded –, also here many drastic outliers emerge. Dshield's sensitivity is across the board 1.5 - 2 orders of magnitude lower than the rest, here indicators are almost exclusively listed only when they show major activity. When we refer back to Figure 4.2, we notice this feed to also perform sub-average with respect to timeliness. Other feeds that are also not very sensitive, like Danger.rulez, have better timeliness.

Sensitivity however does not only depend on the types of sensors a threat intelligence provider utilizes, but also where these sensors are located. Threats emerging in specific geographic areas might be under- or overestimated, leading to an overall bias in sensitivity. As there is no ground truth on where threats are actually located, we can only do a relative evaluation on the position of the listed indicators – based on their IP prefix information – for each individual intelligence feed. Figure 4.5 shows this relative geographical distribution of indicator per feed, which clearly reveal major differences in reporting between the providers and likely the location of their sensor infrastructure. For instance, more than 40% of all reports made by Bambenek are located in the United States, whereas more than 40% of reports on MalwareConfig originate from Turkey and around 40% of the data provided by GreenSnow relates to China.

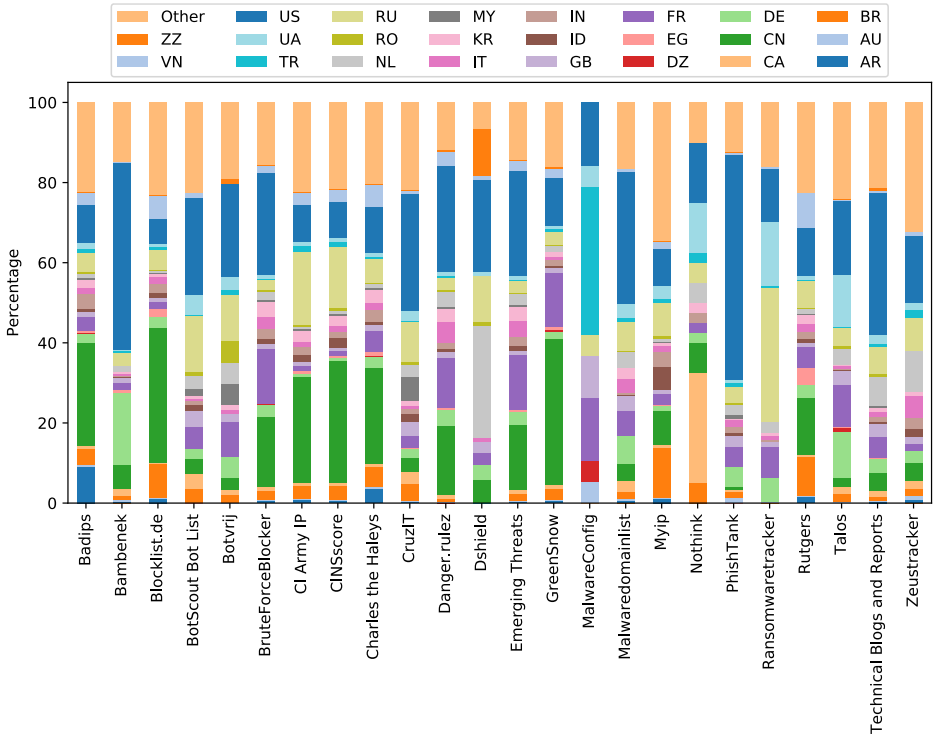

Figure 4.5: Geographical distribution of indicators per list.

These biases become even more apparent when we normalize the IP addresses reported as malicious by the number of IP addresses allocated within that region. Assuming that malicious activity is not strongly concentrated within individual countries, we thus obtain a normalized geographical reporting as shown in Figure 4.6, this shows that for example CI Army and CINSscore are heavily leaning towards reports from Turkmenistan, which is nearly entirely absent in the reports from all the other threat intelligence providers.

On a positive note, the distribution of reports by Emerging Threats, Badips, Blocklist.de, BruteForceBlocker, CruzIT and Myip show no clear geographical preference, and which seems to lend to the conclusion that their measurement infrastructure is sufficiently diverse.

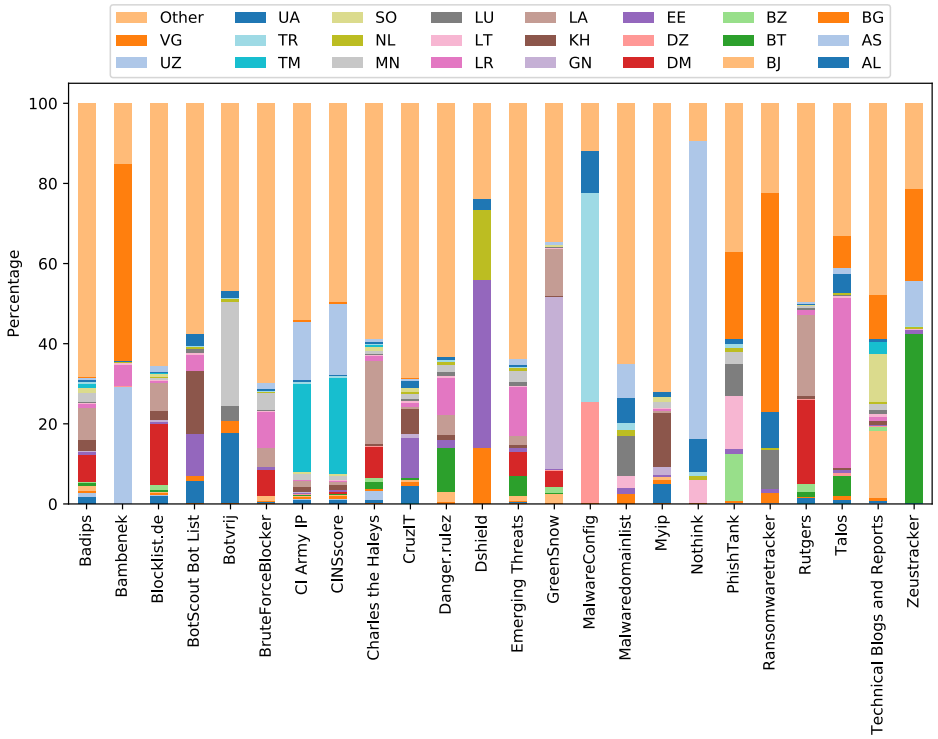

Figure 4.6: Relative geographical distribution of indicators, normalized by total number of IP addresses in a country.

4.3.3. Originality

To investigate the uniqueness of the provided information, we traced for each of the 1.38 million indicators when it first emerged on a particular list and whether individual indicators were afterwards also included on other lists. Besides the result of independent original research, such reuse might also indicate that a particular feed would import the data provided by others. Figure 4.7 shows the later reuse of indicator information across lists, where an arrow indicates that information first originated at the source of the arrow and was later included in the list its points to. The thickness of the arrow and the corresponding label corresponds with the percentage of information on the receiving list, that earlier appeared somewhere else. For readability, only flows where more than 5% potentially originated from a different list are shown.

While commercial threat intelligence providers often only consolidate and curate information as discussed above, we see also some repackaging – although at highly varying degrees – in case of open source feeds. The indicators first appearing on the feed Blocklist.de routinely appear on other lists, and in two cases a fifth of all indicators are shared with Blocklist.de where they appear earlier. Similarly, a quarter of the indicators on Danger.rulez previously appeared on Emerging Threats, however at a global scale such repacking is comparatively seldom and only 11 out of the 24 lists showed such relationships at all. Across the entire dataset, indicators reappear comparatively seldom, in total only 85,906 out of the total 1.38 million entries were also listed on another feed (6.2%).

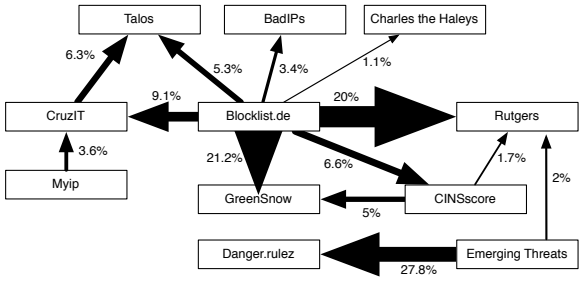

Figure 4.7: Indicator reuse across feeds occurs only sporadically, with the exception of two threat intelligence feeds.

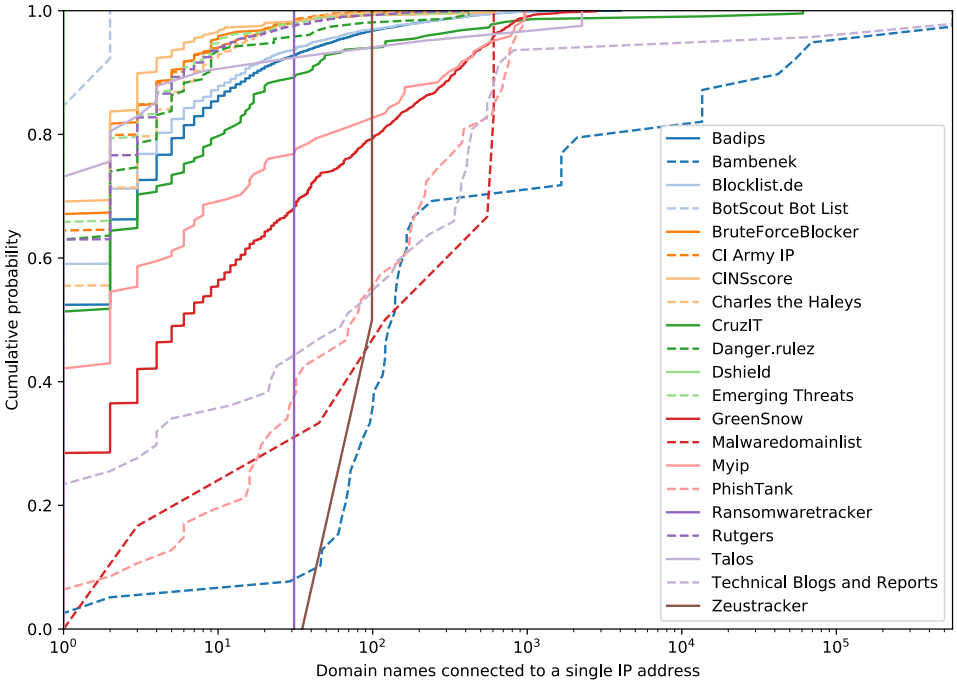

Figure 4.8: Cumulative probability function of the domain names associated with the IP addresses indicated as malicious per threat intelligence feed.

4.3.4. Impact

In order to evaluate the level of potential collateral damage, we resolved the IPv4 and IPv6 A records of 277 million domain name across 1151 top-level domain zones that we received from the TLD operators on a daily basis. For every threat intelligence feed, we analyzed how many domain names were pointing to a particular IP address on the day it was marked as malicious, as all of these domain names would no longer be resolvable if a customer would apply the ruleset provided by the threat intelligence provider in, for example, a firewall. Figure 4.8 shows the cumulative density function of the number of domain names resolving to the indicated IP addresses by threat feed.

As we can see in the graph, there are drastic differences in the amount of collateral damage between feeds. A homogeneous set of feeds – among them BruteForceBlocker, Talos, CruzIT, CI Army IP, and Rutgers – are comparatively targeted, more than half of their entries are not affected by any other domain names, while the 80% most targeted indicators affect less than 6 other domain names if applying an IP-based block. This is somewhat logical for a list that focuses on brute forcing, which is typically not happening from servers that host websites or are operated by a shared web hoster, this is however not the case for the information included on CruzIT, CI Army IP or Rutgers, which include IP addresses used to attack or probe certain networks. For Talos we know that it is curated by Cisco, which makes it likely that this is the reason for the low amount of collateral damage.

This is however not true for all of the feeds. In case of Bambenek, only 16% of the best performing indicators will block less than 50 live domains hosted at these websites, where the 50% worst performing indicators even affect 100 or more domains as collateral damage. While some of the blocked domains may certainly also contain malicious activity, some of the instances included large shared hosters, in one case with more than 900,000 domains pointing to the blacklisted IP address. While such issues could be explained due to automatic collection of indicators, we also found a surprisingly poor track record in case of “Technical Blogs and Reports”, a curated list of human analyst reports, which indicates that human-made feeds are not necessarily better, or at least suggests that feeds do not filter records that could potentially be harmful to normal system operation.

4.4. Discussion

After an evaluation of each of the 24 feeds across the four dimensions timeliness, sensitivity, originality and impact, we will take a step back in this section and evaluate the ecosystem of intelligence feeds as a whole.

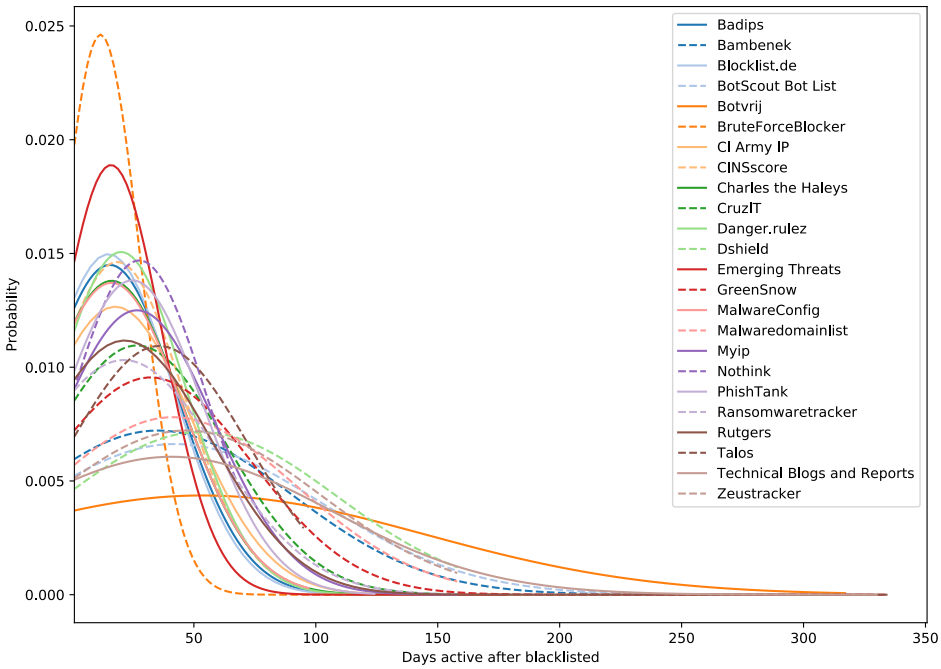

Figure 4.9: Continuation of activity to flagged destinations after listing in a feed.

Specifically we will investigate how widely these feeds are adopted by network owners and operators in practice, and review the issue of the surprisingly high level of originality for the ecosystem as a whole.

4.4.1. Adoption of Intelligence Feeds

As cyber threat intelligence feeds are meant to alert and empower network owners to block malicious activity, their application should lead to a reduction in network traffic towards the hosts flagged as malicious. This observation provides us with an angle to investigate the adoption of threat intelligence feeds across networks worldwide, after all as soon as an indicator has appeared on a list we would expect a significant drop of activity – if not the absence of requests – from a subscribing network.

When networks apply the information provided in cyber threat intelligence feeds at scale, we should ideally see the activity from infected clients to malicious destinations drop and eventually die out. As we have however seen above, intelligence feeds are not universally adopted but have a specific regional footprint and there exists only a marginal overlap between lists; thus, even if a network would subscribe to and apply the information from every single intelligence feed we cannot expect all activity to immediately seize.

Figure 4.9 shows a probability density function of how long activity towards a particular destination continued after an indicator was listed on a particular intelligence feed. As we see from the graph, the inclusion of indicators on for example BruteForceBlocker seems to be an effective deterrent, as on average activity stops within 2 weeks' time. Other lists such as Botvrij are less successful: more than 50% of all hosts reported as malicious by this feed continue their activity for at least 79 days, with an extremely long trail, thus a listing on this block list seems to have almost no impact on the criminal activity itself. Like in case of timeliness until detection (see Section 4.3.1), the threat intelligence feeds are also surprisingly homogeneous with respect to the continuation of activities, and we can clearly see in Figure 4.9 two main clusters, with activity termination peaking around 20 days after listing and 60 days after reporting.

4.4.2. Coverage

There remain however questions about the quality of the ecosystem of cyber threat intelligence providers as a whole. Although it is desirable for a customer that CTI feeds have a large degree of originality as otherwise a customer would subscribe – and pay for – redundant information, we have seen that the amount of overlap between the entire spectrum of analyzed feeds was actually remarkably low. This on the one hand is commendable as it maximizes value of CTI users, on the other hand it also raises questions whether the cyber threat intelligence feeds really provide sufficient information to stop malicious activity in their tracks.

As discussed in Section 4.1, the 24 evaluated open source feeds which span the entire ecosystem of malicious activity from brute forcing activity, ransomware and other malware, to botnets. As each type of malicious activity was covered by multiple feeds, we actually would expect *some* overlap in reported indicators. The fact that there is almost no overlap between lists of similar scope could be the result of two reasons: first, all individual lists for example targeting botnets or ransomware rely on orthogonal detection methods and are therefore providing complementary information. Second, the lists monitor malicious activity in comparable ways, but the overall volume of malicious activity is so large that effectively each lists only obtains a tiny sample, and such low rate sampling from a large universe would statistically lead to a very low chance for duplicates.

Conceptually, we can see that we are probably dealing with the latter than the former reason, after all methods to for example detect ransomware C&C servers are limited, and most likely all providers would employ off-the-shelf tools such as an analysis of network activity across malware samples or a forensic analysis thereof. This unfortunately drives us to the conclusion that cyber threat intelligence feeds cover much less of malicious activity than we would expect and require, to apply intelligence feeds and confidently expect that with a very high degree of certainty malicious activity will be stopped through these indicators.

In this chapter we have introduced a taxonomy to evaluate the quality of cyber threat intelligence feeds aiming to assess the utility a user may gain from such a feed and benchmarked the timeliness, sensitivity, originality and impact of these feeds using our NetFlow data and zone transfers. After having empirically analyzed the impact an indicator of compromise has on such an intelligence feed, we have been able to evaluate the adoption of them in practice. We came to the conclusion that utilizing these feeds for defensive purposes lacks the depth we would have hoped. Reflecting back on our research goal, we are going to need another approach to effectively paint a picture of the C&C infrastructural landscape as the total coverage of the existing feeds is insufficient. Apparently these CTI feeds only show a part of the picture, and as we want to show how large this problem is we ”simply” need to find more malicious servers. A way to achieve this is by combining auxiliary datasets and using tools applicable to NetFlow data. To approach this methodologically, the next chapter will explain what steps we need to take from this point on to answer out research question.

5 Methodology

The previous chapter has evaluated various aspects of open source threat intelligence, concluding that the combination of these feeds do not have a complete picture of malicious indicators of compromise on the Internet. The evaluation towards the timeliness, sensitivity, originality and impact of these feeds have proven that, in order to answer our research question, we must obtain additional data to explore these malicious infrastructures in more detail. Besides this, it is a good idea to use more a structured approach in answering our research question, seeing as publicly available CTI feeds will not provide this structure. In the second step of our research approach, as mentioned in Section 1.4, we set out to form a methodology which will use our NetFlow data to identify malicious infrastructures. This chapter explains our method for this by providing structure to our analysis in Section 5.1, how we will attempt to make information provided by CTI feeds more complete in Section 5.2, and how we can validate and cluster malicious behavior in Section 5.3.

5.1. Dissecting the problem

If we take a step back and recall our research question, we want to gain insights into how we can investigate the evolution, attack patterns and tactics, techniques and procedures of malicious C&C infrastructures. Our initial consideration was that indicators of compromise provided by CTI feeds would give us a good foothold for our NetFlow dataset to dig into the structures which they and their owners have. We hypothesized that these intelligence feeds would provide enough evidence for this, but unfortunately they have far less coverage than we would have hoped. To make up for this we will have to combine the knowledge of these feeds with auxiliary information. We have only been looking at feeds which provide solely lists of IP addresses and domain names, but if we dissect the problem into smaller, more specific problems, we might be able to acquire more information. Keeping this in mind, we chose to bring the problem closer to the perspective of an everyday Internet user. These users most likely use a lot of devices which are connected to the Internet, think of personal computers, laptops, smartphones, smartwatches, camera's, cars and lots of others. Each of these devices can be vulnerable for cyber-attacks due to malware or careless users, meaning that they can eventually be controlled and used for malicious intents. We categorized these devices as follows: *clients*, amongst others personal computers, laptops and smartphones, will be one category and smart devices connected to the Internet, or *IoT* devices, is another category. Both these categories are susceptible for misuse in their own way. Finally we also categorize *routers*, as this is the medium for every household to access the internet with their devices and can be vulnerable without knowledge of the owner.

- Computer **clients** is a category which has been the target of classical malware since computers first emerged. Over the years the possibilities and computational power of devices like personal computers and laptops has grown enormously, making them attractive targets for cyber criminals. Chapter 6 will discuss this category in more detail by using CTI feeds, NetFlow data and the methodologies elaborated upon in the rest of this chapter.
- The Internet of Things, or **IoT**, is a collection of simple devices connected to the Internet which is growing rapidly. New kinds of malware have been emerging over the years specifically targeting these kind of devices to gain control over them. Chapter 7 will go into more detail on the vulnerabilities of these IoT devices and the risk it poses for our society. Besides NetFlow data we make use of a network telescope

able of measuring scanning and brute force attempts of IoT malware, which we will further elaborate on during this analysis.

- **Routers** are generally the more unknown devices in households, as they basically need to do their job; connecting people to the Internet. All data flows through these devices, and for this reason they are also a target for cyber criminals. In Chapter 8 we will discuss a new attack vector on routers of the brand MikroTik and the impact this has. Again our NetFlow data and telescope data will be used for this analysis, complemented by active scans made by the services provided by Censys and Shodan.

Having divided the problem into these categories, we will for each of them dive into the tactics, techniques and procedures, and the impact these have on society. We recognize that by dividing them, each category will have its own structure, but the goal will be the same; to what extent can we track the evolution, attack patterns and tactics, techniques and procedures of the C&C infrastructure. Each chapter will give background information for the use case and provide an analysis on its tactics, techniques and procedures. The rest of this chapter will go into depth on mathematical and statistical tools which we will use in the analyses to answer our research question.

5.2. Detection of malicious servers

Section 2.2 has described all novel methods used for detecting adversarial patterns and infrastructures based on a specific set of data. One of the lessons which can be taken from this summary, is that no perfect solution exists for the overarching problem of cyber criminals commanding networks of infected machines. Each of these criminals can think of different solutions to reach these machines as concealed as possible. Most of these solutions involve deep packet analysis or something similar, but mostly aimed to protect individuals in closed network setups. The NetFlow dataset which is used in this thesis does not have any local networking information, but provides a more global picture. A few publications have been made regarding detecting malicious or C&C infrastructures based on NetFlow data [12, 42, 45, 47, 53, 60, 133, 144, 145] however, only two actually focus on a relative large and global dataset [12, 133].

The work of Bilge et al. is the only one which has published work on a C&C detection method using a sampled NetFlow dataset retrieved from an ISP [12]. They developed an algorithm using machine learning attempting to detect unknown C&C IP addresses. Their detection rates are promising, however they did not disclose details on the datasets which they used to train their model and the model unfortunately is only able to perform at 2x real time. The latter is the biggest problem, as our dataset covers 16 months of network traffic and analysis of a dataset this large at this speed will take months lead time to analyze.

5.2.1. Disclosure

In 2012, Bilge et al. also realized that previous botnet detection approaches are only effective when applied under the exact circumstances previous researchers have set. This however does not scale beyond local administrations. They argue that due to this limitation, the botnet mitigation is only applied to small endpoints of the Internet whilst the majority remain vulnerable. Adversaries controlling these infected machines are an Internet-wide problem, targeting innocent users who sometimes do not even realize they are compromised. One of the most promising solutions to tackle this problem is using ISP NetFlow data. Due to its global nature, it can observe far more than any singular administration. When network traffic is saved into the compressed NetFlow format, all evidence of the malware is lost. In their publication, they present **Disclosure** which is a large-scale, wide-area botnet detection system, employing various filtering techniques to reduce the challenges NetFlow data has. They extract features from the traffic to reliably detect C&C IP addresses from benign flows. After they train a machine learning model and test on unlabeled data to retrieve maliciously labeled IP addresses. Resulting from this, they claim Disclosure is able to recognize 65% of known C&C servers with only 1% false positives by applying post-filtering. This filtering was done by hand and thus does not suggest this can be done automated. Additionally, unlike our data their data was not anonymized, providing an extra challenge on preserving privacy.

5.2.2. Feature extraction

Disclosure extracts three sets of features for each IP address in the dataset: flow size, client access patterns, temporal features. Each set of features extracts a number of statistics of the analyzed IP address. As Bilge et al. did not disclose any of their data or code, we have implemented this system to be able to add or change

features. Our implementation builds on the work provided by Nicola et al. which implemented the feature extraction used by Disclosure in Python [29].

Flow size features
The first features extracted from the IP addresses are based on the flow size statistics, retrieved from both incoming and outgoing flows. C&C commands predominantly have a lot of the same sized packets and are often relative small packets [12], minimizing the impact they have on a network and trying to stay undetected. On the other hand, traffic of benign servers usually has a wide range of flow sizes. To extract features from this a discrete autocorrelation coefficient $R_{\hat{F}_{i,j}\hat{F}_{i,j}}$ is normalized by the variance σ^2 (5.1).

$$R_{\hat{F}_{i,j}\hat{F}_{i,j}} = \frac{\sum_{i=j}^{n} x_i \overline{x}_{i-j}}{\sigma^2} \tag{5.1}$$

A series of flow sizes $F_{i,j}$ is converted to a time series by ordering sizes by time. For each period this function outputs the correlation results. From this the mean $\mu^{F_{i,j}}$ and standard deviation $\sigma^{F_{i,j}}$ over these values are calculated to get all flow size base features. In addition to these features, the amount of unique flow sizes are also derived and are used to create a statistical occurrence density.

Client Access Features
Infected machines regularly have to establish a connection with the C&C server to retrieve the latest commands. As we have mentioned earlier, these connections are probably short intervals to keep under the radar of intrusion detection systems. One important feature of these infected machines, is that they should perform similar access patterns. Clients contacting benign servers should normally not do this and have more variety in this. For each server, statistics are created regarding each connection pattern $t_{i,j}$ with clients resulting in a sequence of arrival times $I_{i,j}$ (5.2). Subsequently, the minimum, maximum, median, and standard deviation are derived of this time series.

$$I_{i,j} = \bigcup_{k=1}^{n} t_{i,j,k} - t_{i,j,k-1} \tag{5.2}$$

It also can happen that a C&C server is taken down or blacklisted. This renders the infected client useless, but it will probably still try to connect to the C&C server. When a client is trying to frequently query a server which is not alive, it is called a zombie. These zombies will have a significant amount of flows towards the server, but no matching flows in the opposite direction. Disclosure uses a time series of unmatched incoming and outgoing flows to quantify this behavior (5.3). In this function $U_{i,j}$ is the time series of unmatched flows of a server and from this the mean and standard deviation are calculated as features.

$$U_{i,j} = \sum_{j \in C} \text{abs}\left(\left|F_{i,j}\right| - \left|F_{j,i}\right|\right) \tag{5.3}$$

Disclosure also extracted *Temporal features* in their model, segmenting a time series of client and flow volume by hourly intervals. However, even though having sampled data, they claim that introducing these features improved the detection rate of their model. After implementing this, we concluded that this did not provide any additional extra detection. This might have to do with the possible different nature of our dataset compared to the dataset that was used by them.

Extracted features
After implementing the extracting of these features and building a pipeline for our data, the data could be formatted to be used in a machine learning algorithm. Listing 5.1 provides an example of these features once extracted in a JSON format. For data processing convenience, the IP addresses have been converted to numbers as can be seen in the listing.


```
3624208180: {
        "autocorrelation_mean_from": 723.4487194773106,
        "autocorrelation_mean_to": 1.5145964657519804,
        "autocorrelation_std_from": 0.5985024215098864,
        "autocorrelation_std_to": 9.134987761149308,
        "cap_max": 30645567.0,
        "cap_median": 22284679.5,
        "cap_median_normalized": 0.5,
        "cap_min": 13923792.0,
        "cap_std": 0.37518544971669887,
        "label": "0",
        "mean_from": 4149163.8146487293,
        "mean_to": 1625739.6363636365,
        "std_from": 1.2003551738281206,
        "std_to": 1.6352551061940865,
        "unique_flow_sizes_entropy": 3.4473171258580715,
        "unique_flow_sizes_entropy_from": 3.4437111688342608,
        "unique_flow_sizes_entropy_to": 1.7677614722893296,
        "unique_flow_sizes_kurtosis": 1.9273838117374908,
        "unique_flow_sizes_kurtosis_from": 1.9233626506156565,
        "unique_flow_sizes_kurtosis_to": 7.258284797135221,
        "unmatched_flows_mean": 49.657407407407405,
        "unmatched_flows_std": 0.5289755761725886
    }
```


Listing 5.1: Example of features extracted from the activity of a server

5.2.3. Pre-filtering
This detection algorithm is created to detect servers acting as C&C instead of the infected devices. Therefore, only the features of actual servers have to be compared with each other. For this reason Disclosure has implemented a server selection before extracting the features. Especially because multiple services can be used on each IP address. Each server is represented of IP address and port $s_i = \langle a, p \rangle$, where a and p are respectively the IP address and port. Each IP address is marked as a server which has a specific percentage of the same ports, which they do not mention in their work. We have implemented this as well and require an IP address to have at least 25% traffic on the same port. This is however where the limitation of their model arises. This will still leave the user of the model with an extremely large amount of detected servers. We took this a step further by making a high level comparison with more rudimental features of actual C&C servers. For each IP address seen on each day in our dataset we created files per day containing the amount of flows sent or received by the IP address, the amount of unique flow sizes per day and amount of unique IP addresses it has connected that day. Using historical data of the threat intelligence feed created on C&C servers by Bambenek and Zeustracker, as seen in Table 4.1, we used the maximum amounts of the described daily IP counts to introduce an extra filtering. By adding an additional 25% leniency per maximum amount of known C&C servers, we could already filter for a more specific set of suspicious IP addresses. In total we have created a speed up of approximately 36x real time, which significant enough to perform analysis on huge amount of available NetFlow data.

5.2.4. Detection model
The next step after extracting the specified features for the required time frame of NetFlow data, is to build the detection model required to extract malicious labeled IP addresses. The work of Bilge et al. compared a few different applicable machine learning algorithms and chose the random forest classifier as this gave the best result. Random forest, like the name implies, consists of a large number of individual decision trees

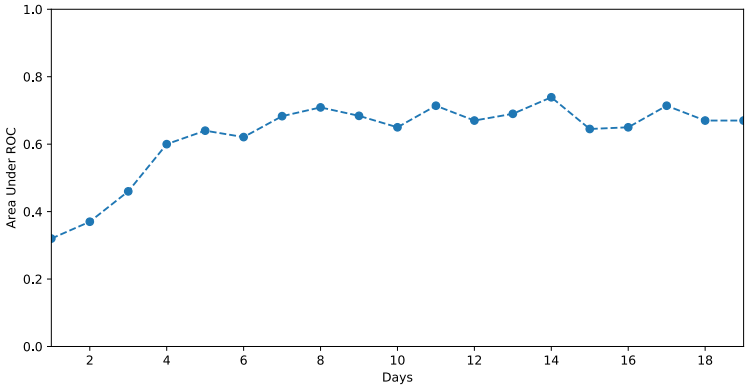

Figure 5.1: Area under ROC for different amounts of days incorporated in the training window

which operate as an ensemble. Each of these trees provides a prediction for the model, and eventually the prediction with the most votes will be the output of the algorithm.

Each machine learning algorithm builds a mathematical model based on training data to make predictions. By doing this, the algorithm does not have to be specifically programmed for each different task it has to perform. Bilge et al. used samples of C&C servers provided to them through a paid provider. Unfortunately, we do not have that opportunity and will use C&C servers provided by the Bambenek and Zeustracker threat intelligence feeds and take this as our ground truth for the training data. Additionally, benign servers have to be included in the training data to create a model which can discriminate benign from malicious. For this, servers belonging to the Alexa top 1000 most visited websites will be labeled as benign. On average, each day has about 100 active C&C servers, but this of course differs each day. To populate our model with more samples, an increasing training time frame is used and tested for the detection rate and false positive rate. To create a balanced model, we evaluated the usefulness of our model performing a 10-fold cross-validation. This evaluation can be seen in Figure 5.1 where the accuracy is measured by computing the area under the Receiver Operating Characteristic (ROC) curve. The ROC-curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR). The TPR is the actual detection rate, or recall, whilst the FPR shows the amount of false alarms.

The best results were obtained by including 14 days of malicious and benign activity around the day to be tested on. Including more days does not increase the detection rate, as can be seen in Figure 5.1. Giving our model exact numbers on how well it performs is incredibly difficult, as this differs for each day. On average we can say that the detection rate, or recall, of C&C servers is 50.4% with an average false positives of 16.4% and accuracy of 87.4%. This recall might not be that good, combined with the amount of false positives, it must however be taken into account that our data is highly sampled. Disclosure has similar results, where our implementation takes about 18 times less to process and thus increasing our opportunities to work with this.

For us it is not a goal to create the optimal machine learning solution to find all existing malicious servers, as this is not feasible with sampled data. Keeping the dataset in mind, we have the do opportunity to create a different model for each day, increasing our chances to find actual malicious C&C servers. This however does mean that the detection rate will vary for each of these days. Disclosure was able to reduce their false positives to about 1% by filtering manually. This is something which is only preferable when working with small datasets. We will have to eliminate these false positives, or at least bring them down to a minimum, by using specific statistical tests for our situation which the next section will elaborate upon.

5.3. Validation and clustering techniques

The model discussed in the previous section is able to predict C&C servers with a particular certainty each day. This is problematic when the goal is to acquire all malicious servers out there, which is unfortunately something which is not feasible as every adversary can in theory configure their system to work in an undetectable manner. The model does however provide a good base of IP addresses with highly suspicious activity.

In some situations, for example when specifically linking malicious activity to a small set of known malicious servers, it might be necessary to have a higher certainty than the one discussed in Section 5.2.4. This section will first cover two techniques which can be applied in this scenario, clustering small sets of IP addresses for similarity. On the other hand, when trying to gain more insights into clusters of activity we will lastly elaborate on a method used to do this. Each of these methods is used in multiple occasions during the following three chapters to gain more insights into adversarial patterns of cyber criminals.

5.3.1. K-Sample Anderson-Darling test

The detection method proposed by Bilge et al. used un-anonymized IP addresses in their detection, meaning they could perform a false positive reduction after the detection. The anonymized nature of our dataset unfortunately does not allow us to do this. Therefore it would be useful to be able to correlate the activity of two or more servers with each other with a statistic certainty. To realize this we have used the Anderson-Darling test to be able to compare the distribution of flow sizes belonging to servers. This is a statistical test which analyzes wether a provided sample is drawn from a specific probability distribution. In its basic implementation, this test provides a set of critical values correlating to significance levels of which a null hypothesis can be rejected.

In the case of comparing whether two distributions of flow sizes are drawn from the same distribution, the K-Sample Anderson-Darling test can be used [112]. This tests if two or more collections of observations can be correlated to come from a single distribution, not having to specify the distribution function. This equation is shown in (5.4). To simplify the explanation and usage of this equation, we will focus on using two samples. In this equation the hypothesis is tested wether that two samples (or vectors in our case), $X_1, \ldots, X_m$ and $Y_1, \ldots, Y_n$, are drawn from the same distribution. $X_1, \ldots, X_m$ is represented by the empirical distribution function $F_m(x)$ and $Y_1, \ldots, Y_n$ is represented by the empirical distribution function $G_n(x)$. Instead of using a pre-set distribution to test both these vectors with, H_N is introduced to simulate a distribution created by the input distributions and is calculated by $H_N(x) = \{m * F_m(x) + n * G_n(x)\} / N$ where $N = m + n$.

$$A^2_{mn} = \frac{mn}{N} \int_{-\infty}^{\infty} \frac{\{F_m(x) - G_n(x)\}^2}{H_N(x)\{1 - H_N(x)\}} d H_N(x) \tag{5.4}$$

The result of this equation, A^2_{mn}, is used to test the null hypothesis that $F = G$ without having to specify the common continuous distribution function. The resulting, so called, critical value can be used to either accept the null hypothesis or reject it, based on the chosen significance level. The significance level is the probability of rejecting the null hypothesis when it is true. For example, a significance level of 0.10 indicates a 10% risk of concluding that a difference exists when there is no actual difference between the distributions. Because of this, it is good to have this value as low as possible. A significance level of 0.05 is normally used, but other values can be chosen to acquire a higher or lower threshold [112]. The critical values and corresponding significance levels of two samples is provided in Table 5.1.

Significance level	0.15	0.10	0.05	0.025	0.01
Critical values	0.325	1.226	1.961	2.718	3.752

Table 5.1: Significance values and corresponding critical values defined for the K-Sample Anderson Darling test [112]

To gain insights on a more intuitive level, we can draw our attention to Figure 5.2. This figure shows four example distribution of server activity in amount of flows observed with a specific size, which can indicate a specific behavior. A similar distribution would mean similar communication patterns. In this example, (a) is known to be malicious and (b), (c) and (d) are servers linked to this behavior by, for example, the decision model explained in 5.2.4. When using the K-Sample Anderson Darling test with these distributions, a critical value of 7.543 can be obtained when using (a) and (b) indicating the null hypothesis that these samples come from the same distribution can be rejected at the 1% level because the returned test value is greater than the critical value for 1%. For (a) and (c) however the null hypothesis that these samples come from the same distribution can be rejected at the 10% level because the returned test value is greater than the critical value for 10%, but lower than the critical value for 5%, as the result is 1.279. Each time this equation is used in this thesis, the critical value has to reject the null hypothesis at least at 2.5% before we can say the samples are drawn from the same distribution.

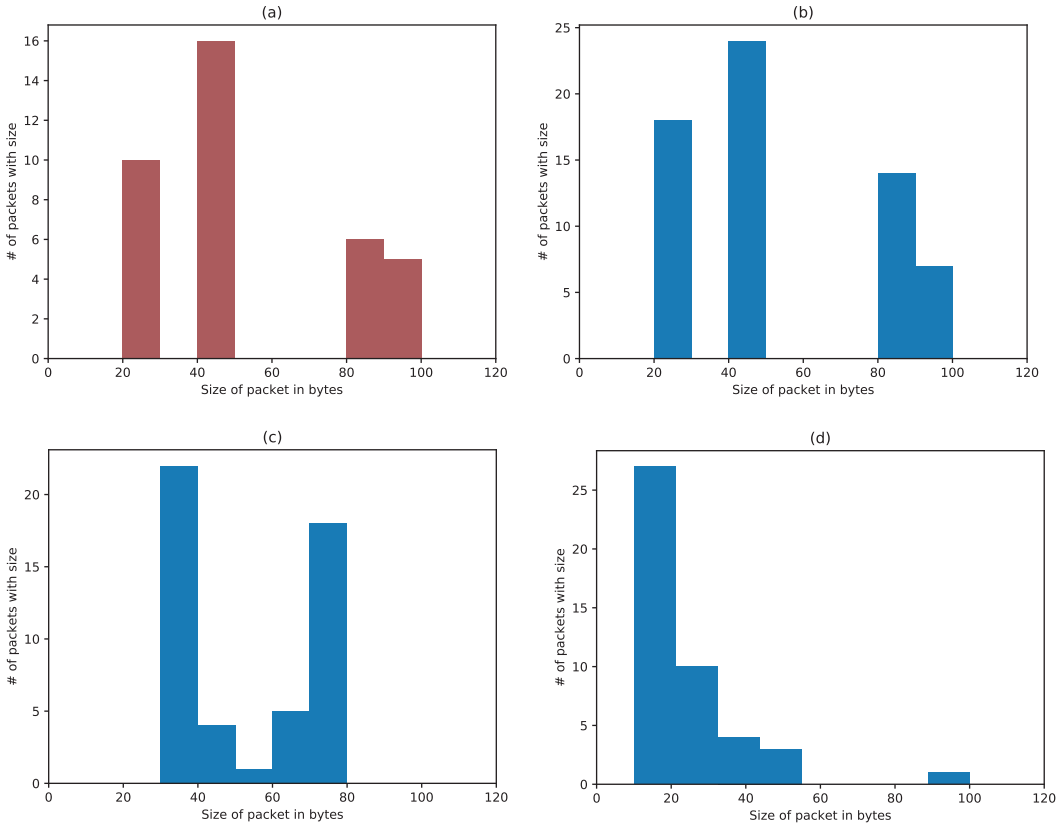

Figure 5.2: Mock flow size distributions of a malicious server with three potential correlated servers

5.3.2. Cosine Similarity

The Anderson-Darling test provides a robust way to validate whether two servers are drawn from the same sample. This can provide the validation necessary to complement the detection model. However, when performing campaign analysis between a large number of servers or infected machines it will be helpful to be able to say something about the similarity of activity seen between IP addresses. To measure this similarity, the cosine similarity will be used. This will specifically be used to measure the similarity between the percentage of traffic per port for a specific time frame. In this case, the magnitude of the vectors containing port distributions does not matter, making the cosine similarity a good choice to calculate the similarity. The cosine is derived by using the Euclidean dot product formula as seen in (5.5).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{5.5}$$

Given two vectors, A and B, this equation measures the similarity between these vectors inner product space that measures the cosine of the angle between them. A_i and B_i are respectively components of the vectors, used to solve the equation and return the cosine value, indicating the similarity between the two vectors with a 0 to 1 score. In our case this has been used to detect the similarity between activity of two servers. This is measured by creating a vector $\vec{v}$ with the length of all TCP ports, filled with the sum of flows each port has been identified in the NetFlow data. See $\vec{v}_1$ for an example of this. As we are dealing with sampled data, we have chosen to deduct the relative percentages for each port and put these values in the final vector which can be used for comparison. For simplicity $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$ only have 6 'ports'. Let's say the server with activity $\vec{v}_1$ is known to be malicious. $\vec{v}_2$ and $\vec{v}_3$ can be accordingly compared with $\vec{v}_1$ with the cosine similarity, respectively resulting in similarity values of 0.9488 for $\vec{v}_1$ and $\vec{v}_2$ and 0.5916 for $\vec{v}_1$ and $\vec{v}_3$. This indicates that the activity of $\vec{v}_2$ is highly similar to $\vec{v}_1$.

$$\vec{v}_1 = \begin{pmatrix} 230 \\ 3 \\ 0 \\ 88 \\ 0 \\ 20 \end{pmatrix} = \begin{pmatrix} 67.4 \\ 0.8 \\ 0 \\ 25.9 \\ 0 \\ 5.9 \end{pmatrix} \qquad \vec{v}_2 = \begin{pmatrix} 280 \\ 50 \\ 0 \\ 30 \\ 10 \\ 54 \end{pmatrix} = \begin{pmatrix} 66.0 \\ 11.8 \\ 0 \\ 7.1 \\ 2.4 \\ 12.7 \end{pmatrix} \qquad \vec{v}_3 = \begin{pmatrix} 30 \\ 0 \\ 0 \\ 88 \\ 0 \\ 54 \end{pmatrix} = \begin{pmatrix} 17.4 \\ 0 \\ 0 \\ 51.2 \\ 0 \\ 31.4 \end{pmatrix}$$

5.3.3. HDBSCAN

In the previous two subsections we have explained two techniques which can be used to gain insights into the similarity of IP addresses. However, one final technique has been used gain more in-depth knowledge regarding the division of activity in a C&C server. In the next chapters we will apply an unsupervised clustering algorithm for a few specific cases to analyze whether there is labor division amongst the infected devices within malicious infrastructures.

Clustering by heart is all about grouping together data similar data points, regardless of the format. Various techniques exist which all have different definitions of what a cluster means and approaches to find these clusters. Naturally clustering algorithms break apart into centroid based or density, and flat or hierarchical based algorithms. Centroid based clustering algorithms will make explicit or implicit assumptions about the shape of the clusters, requiring a priori knowledge about the data. This allows the algorithm to work on small amounts of data, which can be visualized very efficiently. On the other hand, density based clustering is going to allow the data to define the clusters, implicitly requiring a lot more data to be effective but also allowing for noise in the data. Either of these clustering techniques will also employ a flat or hierarchical type of clustering. Flat clustering is going to create clusters based on a priori knowledge of the amount of clusters, also again requiring to be able to visualize the data. Hierarchical clustering is specifically designed to address this parameter selection problem by creating a natural hierarchy over the data and use this to define the correct number of clusters. Examples of clustering methods using the four different combinations of these categories are show in Table 5.2.

	Flat	**Hierarchical**
Centroid	K-means	Complete-linkage
Density	DBSCAN	HDBSCAN

Table 5.2: Categorical overview of clustering approaches

The data which we want to cluster is based on the daily port distribution per IP address to cluster, as we can see in the vectors shown in Section 5.3.2. A $M * N$ matrix is created with M representing all IP addresses to be clustered and where N is the distribution of all 65536 TCP ports. The higher dimensional nature of this data does not provide a clear a priori understanding of possible clusters. These clusters which we want to create essentially are sets of samples which are found to be similar, measured by some distance metric. According to this metric, neighbors of samples are compared with each other and put together. Density-based spatial clustering of applications with noise, or **DBSCAN**, is a clustering algorithm which eliminates the parametric centroid distribution assumption. It accounts for noise in the dataset, something which K-means for example does not facilitate. DBSCAN however still requires some beforehand knowledge about the cluster sizes. This can be solved by using a hierarchical approach and still having the density based clustering. **HDBSCAN**, as introduced by Campello et al. [15], combines both these worlds. It internally creates a dendrogram with decreasing numbers of clusters to fit the purpose and goal of the to-be-clustered dataset. This fits the requirements of the dataset and the intention of our clustering.

This chapter has both elaborated upon the structure in which we will perform analyses and explained mathematical and statistical techniques which will be used in these analyses. It is important to mention that these techniques will only be used if applicable in the situation.

6

Infections on clients

Time frame	January 2018 - December 2018
Datasets	Tier 1 ISP NetFlows and open source threat intelligence feeds
Goal	Understanding the evolution, spreading and impact of infected clients

Nowadays personal computers, laptops and smartphones are devices used by a large portion of the world population. In our world today these devices have become essential as a lot of important infrastructures rely on computers and people operating them. The possibilities of these machines are constantly growing; they are gaining more power, more memory, and more developers and companies are interested in writing applications for monetary or personal gain. Cybercriminals also notice this and innovate in deployment of malicious software and in Command and Control distribution, as we have seen in Chapter 2. Computers these times are incredibly vulnerable for malware infections.

This chapter will analyze infected clients controlled by C&C servers, the first category of networking devices which we have defined in our methodology. We show that by using NetFlow data we can gain valuable insights how adversaries operate their networks. Previous work has not yet provided a global understanding of how these C&C servers operate, what the different tactics are between them and how they utilize their network. We identify a lot of previously unidentified structures being used by adversaries throughout 2018. We also look at the differences in behavior of the infected machines which are controlled by cybercriminals and the evolution of this behavior. By doing this, we gain understanding of a not yet explored perspective on C&C servers, allowing us to find unknown malicious servers; a technique which can possibly be used by ISPs for mitigation of threats.

6.1. Background

A lot of different types of malicious code fall under the global term malware. Every adversary can write malicious code to perform misuse for their personal gain. A user downloading a popular image processing tool for free might think they have found an amazing deal, free software which normally costs hundreds of dollars, but at the same time the infected program requests malicious commands from a remote C&C server administrated by an adversary.

Infected machines, or bots, in this analysis have been infected by a kind of classic malware variant and all connect at some point to a C&C server coordinating their infection. During their infection the bots can contact a C&C server for various reasons. In the most simple infrastructures the bots periodically connect to the C&C server to ask for commands or send collected data (for example collected usernames and passwords). More advanced structures will however disperse this kind of activity to stay undetected; a few different servers for C&C activity and a few other servers to gather data. We hypothesize that current C&C servers will diverge a lot in behavior, tactics and day to day utilization. In this analysis we will make the following contributions:

- We have analyzed adversarial tactics and reveal differences between servers in how they communicate with their infected machines and how this activity evolves over time.
- We are the first to use a combination of learning techniques to find unknown C&C servers in anonymous NetFlow data and complementing data found in open source threat intelligence feeds.

For the remainder of this analysis we will first provide some additional information about the datasets used in this chapter. Section 6.2 will go into the tactics, techniques and procedures observed and what we can learn from this, and Section 6.3 will go into the detection of new servers and how this can aid malware detection.

6.1.1. Datasets

To perform this study we additionally make use of cyber threat intelligence feeds. In Chapter 4 we have analyzed various blacklists related to different malicious IP addresses or domain names on threat intelligence feeds. As we have seen in this exploration, these lists are far from complete and do not provide a complete picture of the entire threat landscape. We can unfortunately not say anything about paid threat intelligence providers, but ultimately the open source community can provide insights into attack vectors which can contribute to a safer society. As a starting point for this chapter, known C&C servers are needed. The open source threat intelligence feeds of *Bambenek* and *Zeustracker*, as can be seen in Table 4.1, provide C&C servers linked to certain types of malware. We have used these blacklisted IP addresses, which are a total of 993 IP addresses listed by the Bambenek feed and 112 IP addresses listed by Zeustracker.

6.2. Adversarial Tactics, Techniques and Procedures

In this section, we will analyze the techniques, tactics and procedures (TTPs) adversaries use concerning C&C servers of infected clients and their subsequent abuse. We will do this in a structure which starts by analyzing global structures, followed by behavioral analysis and the evolution of the behavior of different C&C servers.

Literature tells us that cyber criminals in theory employ different techniques regarding their infrastructure, to hide their misuse from the public eye. The goal of this subsection is to show actual cases of this theory being used in practice. Furthermore we will discuss what kind of behavior the bots connecting to their C&C servers have and what we can learn from this. Ultimately we want to have a better understanding about what adversaries actually do in practice, to be able to stop them.

6.2.1. Malicious infrastructures

To provide insights into the basic activity of these C&C servers, we have aggregated rudimentary statistics each of these servers have made per day during the timeframe of our NetFlow dataset, as mentioned in Chapter 5. Using the amount of connections an IP address makes each day, we can visualize the activity belonging to this IP address. Using this information we can hopefully uncover information about the infrastructures behind these malicious servers.

Load balancing Load balancing is a technique often used for highly popular websites or applications to ensure availability. By engaging more than one server, each having their own IP address and by making them all point towards the same domain, companies can ensure that their service stays online even after a server breaks down. When for example performing a simple lookup, it can be seen that `www.amazon.com` has three IP addresses pointing towards this single domain name. This is called round-robin DNS and is one of the more popular load distribution techniques. Round-robin is a method which does not require one dedicated server. This technique also uses multiple IP addresses pointing to the same domain name, but the operator can choose the amount of time each IP address links to the domain name. DNS delegation on the other hand is more effective, as this makes use of the geographical location of the client. When visiting a website which uses DNS delegation, it specifies zones to accordingly send the request to one of the servers linked to the domain name. This however only works when servers are distributed on several locations world-wide. These techniques, whilst being designed for distributing networking load, are also very attractive for adversaries to hide their structures.

We have found a number of different examples of adversaries employing evasive behavior regarding their infrastructure. For each of these examples we have tested their behavior according to the Anderson-Darling, test as explained in Section 5.3.1, and can conclude that their behavior is the same. Therefore we can say that these servers belong to the same infrastructure. Figure 6.1 shows three IP addresses which are showing an example of a round-robin structure being used. These IP addresses not only overlap in behavior regarding the size of the flows, but the IP addresses also overlap regularly. This tells us that victims do not always make contact with the same malicious server, but iterate through the first available server.

Another example can be seen in Figure 6.2, where one C&C server has been active since February 2018 and three new servers pop-up after the start of October. After these servers have been activated, all four also start to act in a round-robin fashion.

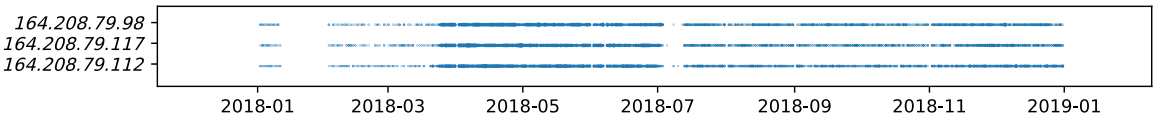

Figure 6.1: Round robin structure used by a malware linked to the distribution of ransomware

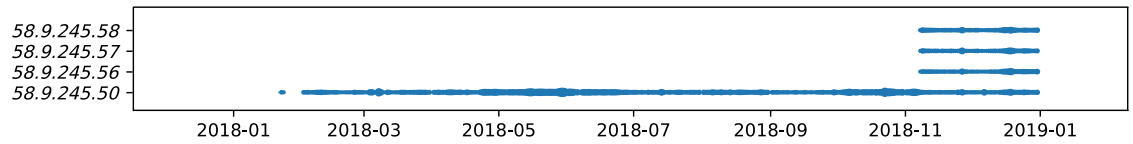

Figure 6.2: New servers emerging contributing to load balancing of the initial server hosting .. malware

Both the these examples have initially been found by visualizing their activity and confirming they belong together. Note that the IP addresses are in close vicinity of each other. However, in Figure 6.3 this is not the case. In our data we found 44 C&C servers active from November until which are all part of AS14618, or Amazon Web Services. This says that adversaries are using cloud provides to mobilize their misuse, something which

we can expect. Most of these IP addresses are active for a short time-span, this is probably due to Amazon monitoring behavior of its cloud services. After testing the behavior of all these servers with each other, 4 servers popped up to have the same behavioral patterns. As we can see in Figure 6.3, the servers are alive for approximately 20-30 days. After a server has stopped, another server with similar behavior emerges right the previous one has concluded its activities. Multiple of the same victim IPs have been observed on all four of these servers. This tells us that the adversary behind this operation, keeps moving it around, trying to keep it alive. It turns out that this kind of 'hopping' sequence is not unique. Figure 6.4 shows another example of this kind of structure, this time used by the *bedep* malware variant.

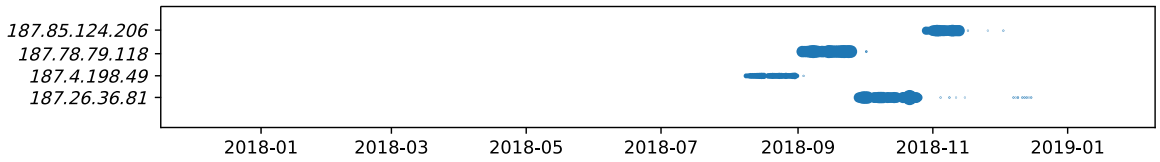

Figure 6.3: Evasive behavior found regarding the *tinba* malware hosted on Amazon Web Services

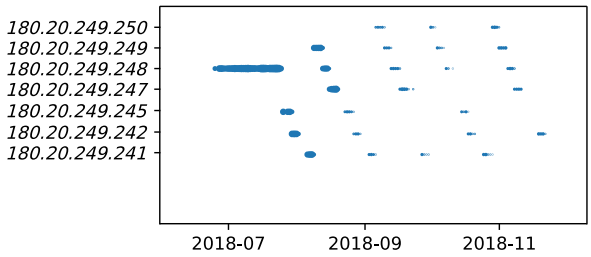

Figure 6.4: Another example of evasive behavior used by the *bedep* malware

6.2.2. Behavior

After having gained insights into structural patterns, which mostly show high level information of behavioral patterns, it is interesting to dive more into the actual activity of the bots controlled by the C&C server. The hypothesis is that C&C servers controlling a meaningful infrastructure will divide the labor of their bots, instead of focusing all power on one task and increasing the sophistication. Global behavioral patterns are only possible with data that actually contains all of this information, instead of testing malware in a concealed environment or collecting data of internally infected machines.

It is difficult and not entirely realistic to compare the activity of each C&C server with each other, which is also not the goal of this Chapter. Our hypothesis is tested by comparing thirteen different C&C servers, listed in Table 6.1, and their activity, evolution and utilization. The only criteria to effectively compare the activity of these C&C servers, was that they have one month of unabridged C&C activity. By limiting our analyses to this criteria ensures that the compared servers are actively used by an adversary and are actively spreading commands to infected machines. Furthermore this list contains both large and small structures, related and unrelated servers, and a mix of related malware. The sizes of these infrastructures are estimated accordingly to the sampling ratio of 1 to 8192 of the NetFlows.

Comparing malicious activity Each infected machine at some point in time requests from or sends information to its C&C server. These infected machines are most likely personal computers or laptops affected by the malware consolidating the infection, this is something the NetFlow data will not be able to tell us. However, the activity of these clients can provide meaningful insights into the utilization of the commands they received. Unfortunately real world data has a lot more noise than synthetically created data, but it can be assumed that machines used by users have a baseline of activity containing mostly web traffic [65]. To differentiate between activity of infected clients or servers we have used cosine similarity measuring the the angle between two vector, as elaborated upon in Section 5.3.2. This is a comparison of orientation resulting in 1 when both vectors have the same direction and 0 when the two vectors are opposite of each other. These vectors are filled with the amount of activity towards each port. This activity is normalized, allowing us to

C&C Server	Estimated size	Linked malware
164.208.79.99	73.728	Generic
164.208.79.98	204.800	Generic
164.208.79.117	114.688	Generic
164.208.79.112	57.344	Generic
172.165.138.217	458.752	Banjori
200.43.79.87	139.264	Ramnit
216.130.162.157	4.931.584	Simda
223.87.191.197	40.960	Virut
73.5.141.200	221.184	Suppobox
73.5.141.201	131.072	Suppobox
73.5.141.198	417.792	Suppobox
58.9.245.50	745.472	Kraken
49.182.4.150	770.048	Ramnit

Table 6.1: C&C servers analyzed with the estimated unsampled amount of infected machines and linked malware

compare activity between larger and smaller botnets. In this situation 1 can be interpreted as the exact same port distribution, whilst 0 corresponds to having an opposite port distribution.

Putting this into context, this methodology has been applied to the list of C&C servers and the combined port distribution of all its infected machines, resulting in a similarity matrix provided in Figure 6.5. This image compares the activity of all combined infected clients of each C&C server with the rest of the servers from Table 6.1 and some interesting observations can be made. When looking at the similar IP combinations, diversity can already be seen. *73.5.141.200*, *73.5.141.201* and *73.5.141.198* are related by malware type and also show great similarity in the activity its infected machines produce, which is evidence similar instruction patterns are provided by the adversary. *164.208.79.99*, *164.208.79.98*, *164.208.79.117* and *164.208.79.112* on the other hand are also related by malware type but show great diversity in activity. Some other interesting observations which can be seen is the similarity between *172.165.138.217*, *49.182.4.150* and *216.130.162.157* which do not seem to have a direct connection in terms of the same malware for example. Furthermore both differences and similarities can be observed in Figure 6.5. From this we can conclude that from a high level perspective, we can make indications of behavioral resemblances. Inspection of the actual packets rather than the flows can further verify this.

Figure 6.6 shows the cosine similarity between infected clients the C&C server *49.182.4.150*. As expected, clusters can be seen on various locations of the image, as well as a few clients correlating only with each other. The goal of this image is to show how some infected machines have potentially similar behavior, hinting towards a decent level of sophistication by grouping activities together. From an adversarial perspective it makes sense to be able to use infected machines for all sorts of different malicious activities. Booter services can for example rent a part of such an infrastructure to perform DDoS attacks with only a subset of clients.

6.2.3. Evolution in behavior

As we have observed, by analyzing the behavior of infected machines controlled by C&C servers overlap and division can be seen. When thinking from the perspective of an adversary managing malicious infrastructures, a lot of different possibilities present itself in terms of utilization. One can for example choose to rent out half of the infected machines to set up DDoS as a service for monetary gain or it might be possible to rotate the usage of these infected clients to minimize suspicious activity. They can even choose to provide each of the machines with an individual task list which they have to perform at certain times, without connecting to the C&C server. Keeping this in mind, we have shown a method to gain foothold on behavioral similarities and differences between IP addresses in the NetFlow data. This allows us to gain more insights into the inner workings and commands provided to the infected machines contacted by a C&C server. Instead of comparing the cumulative daily activity of these commanding servers with each other, the same method can possibly show differences in activity over time by comparing itself with the previous day. Differences over time can possibly show an evolution in commands provided to the infected machines and can provide us with an initial feeling if there is for example labor division.

To evaluate this, the cosine similarity of the port distribution has again been used between the selected

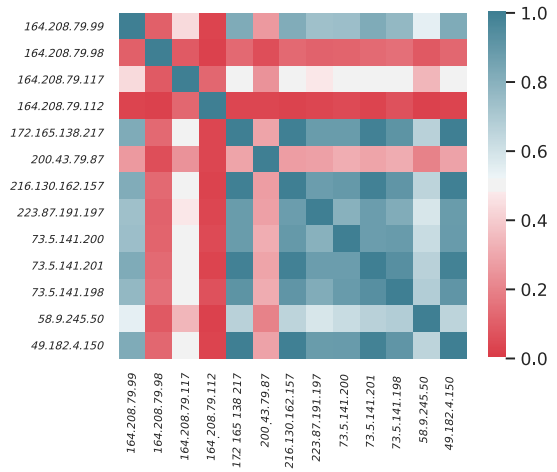

Figure 6.5: Similarity in activity of infected clients of selected C&C servers

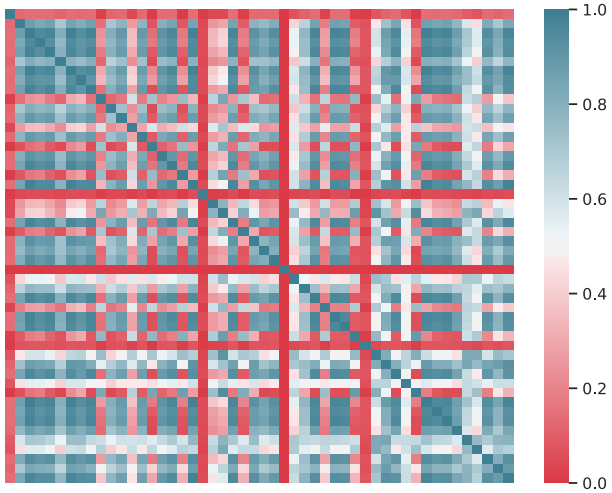

Figure 6.6: Similarity in activity of infected clients of *49.182.4.150*

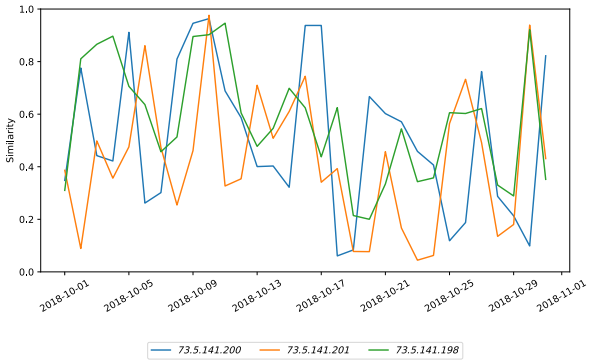

Figure 6.7: High similarity in evolution regarding the behavior of all bots related to three linked C&C servers during October

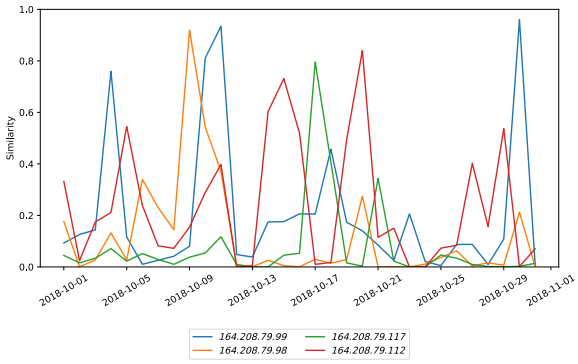

Figure 6.8: Low similarity in evolution regarding the behavior of all bots related to four linked C&C servers during October

C&C servers however, instead of comparing them with each other we have now compared the servers with their own activity of the previous day. Again a vector of the cumulative daily activity is created for each day of October 2018 for these servers, and the cosine similarity is calculated between the current day and the next day. Figures 6.7, 6.8 and 6.9 respectively show these results, depicting the two related clusters of C&C servers and four unrelated servers. To clarify, the vertical axis shows the amount of similarity compared with the previous day.

Looking at an example, in Figure 6.7 *73.5.141.201* on the second day has an approximate cosine similarity of 0,1 with the first day, indicating little behavioral overlap. Following from this we say that the C&C server provides different commands to its infected machines, affecting this change of behavior. The behavioral evolution of this specific IP address shows a lot of ups and downs regarding this similarity. This is a curious pattern to see and hints to new commands being communicated to the infected machines on intervals. Even more interesting, when comparing the evolution of all three IP addresses shown in Figure 6.7 an underlying pattern worth mentioning can be seen, indicating the same set of commands being sent from the C&C to the infected clients. In Figure 6.8, comparing *164.208.79.99*, *164.208.79.98*, *164.208.79.117* and *164.208.79.112*, the same pattern amongst the servers can also be seen. For these servers this is however more interesting, as the similarity between these servers was extremely low which we saw in Figure 6.5. Especially when comparing these servers with the servers shown in Figure 6.7, which did have a high similarity between themselves. This shows that the servers in Figure 6.8 have the same behavior regarding C&C infrastructure, but have a higher level of sophistication as the cumulative daily activity between them is different.

Figure 6.9 on the other hand shows C&C servers which are not directly related based on our analysis. At a first glance different behavioral patterns can be observed and shows a lot more diversity. *49.182.4.150* is a particularly interesting example, as the similarity stays around 1,0 during the course of October. An adversary

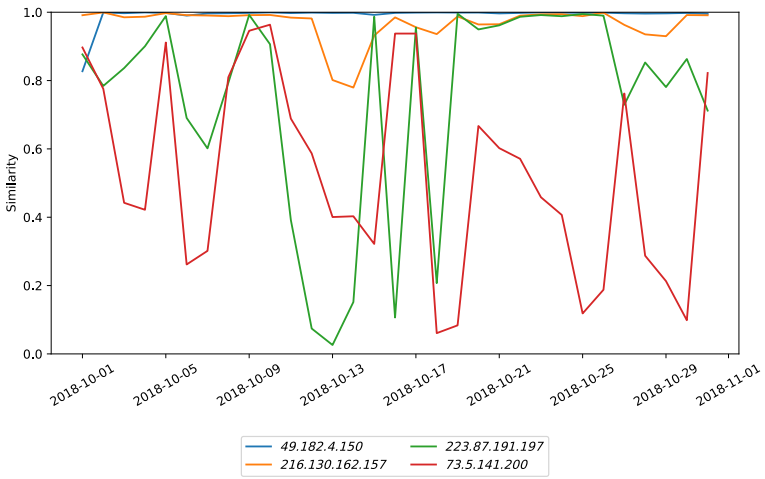

Figure 6.9: Evolution regarding the behavior of all infected clients of four unrelated C&C servers during October

can for example design their botnet infrastructure in such a way that it has the same labor division each day, making the C&C infrastructure probably easier to maintain. *216.130.162.157* also shows the same kind of behavioral similarity over time, except for a small change of infrastructure which can be seen around 13th of October. The activity of the infected machines controlled by the other C&C servers in this image show great contrasts, again indicating different utilization of the managed infrastructure. The other two C&C servers however show a huge diversity in activity each day, which hint to more diversification in commands, also something which we expect to happen. *223.87.191.197* for example shows three peaks of behavioral changes which could be explained by the purpose or goal of the adversary.

6.3. Detection of unknown C&C servers

The previous sections have shown that C&C servers vary a lot with regards to infrastructural choices, and both similar and diverging behavior can be observed. These malicious structures already tell us a lot more about the nature of the operation than literature tells us and helps us understand how they roughly work. We have shown that we can correlate servers to each other.

6.3.1. Using machine learning to our advantage

Our extensive evaluation of open source threat intelligence in Chapter 4 showed that these feeds are not complete. They give a good baseline for detection and threat intelligence, but there is definitely a lot more adversarial activity in 'the wild'. It is close to impossible to know and see all threats, even when knowing malicious patterns. The past 15 years has given us a lot of technological evolution in the world of threat intelligence as we have seen in Section 2; researchers have been creating advanced detection and machine learning algorithms to predict malicious behavior on known patterns. Most work has been done on detection on local machines, which can aid anti-virus software accountably. This is mostly detection based on a few types of malware used in specific scenarios. ISP level NetFlow data can potentially give more insights into globally similar patterns, providing more insights into C&C infrastructures and ultimately stopping them before the operation harms a lot of people. For this reason we have chosen to adopt the machine learning method proposed by Bilge et al. [12] to be able to apply it to our NetFlow dataset.

By using a sliding window model and re-training the classifier for each specific day, a lot more C&C servers could be detected according to the accuracy and precision defined in Section 5.2. After applying our detection system a total of **251,117** possible new C&C servers have been found during the course of 2018, with an average around 3,430 each day with a lot of overlap. We say possible, as each day poses a certain amount of false positives, so even though these servers all have suspicious activity, we cannot with certainty say that these servers are all malicious. However, even when incorporating a lenient amount of false positives of 25%, which is more than the largest amount of false positives found, we can still say that there have been approximately **188,337** servers in 2018, and around 2,572 each day, exhibiting malicious behavior similar to the behavior we have observed from the cyber threat intelligence feeds. This is already a huge amount more than the total of 1,105 servers presented by the feeds, and also an order of amount which we would have

expected as 1,105 is relatively low compared to the 4.3 billion IP addresses available making our estimation more realistic.

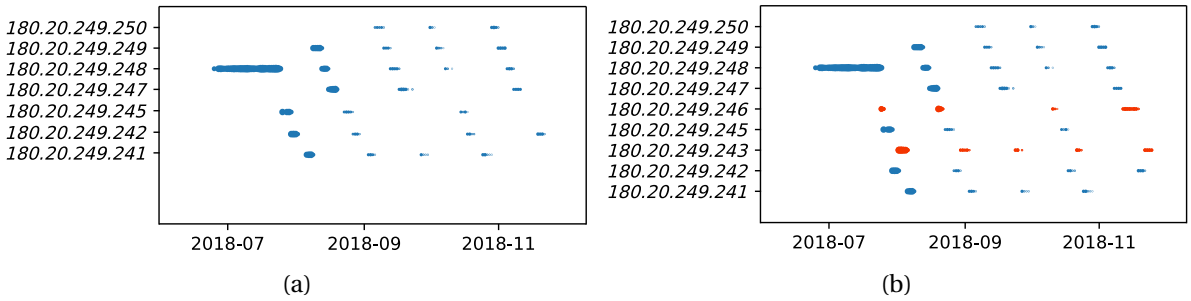

Figure 6.10: Complete infrastructure with missing C&C servers detected and verified with our method

6.3.2. Complementing CTI feeds

As is it difficult to accurately measure how much of these predicted servers are actually malicious, we have defined a clustering method based on the Anderson-Darling test which we have used earlier this chapter and explained in Section 5.3.1. By applying this method we can test if the packet size distribution of all connections in one IP address is taken from the same distribution as another IP address. In our case, this method allows us to test whether predicted IP addresses have the same behavior as known C&C servers. Revisiting Figure 6.4, also shown in Figure 6.10, it can be observed that activity is missing according to the shown structure. To approach this, we initially took all the malicious servers found for a specific day for which it looked like data was missing, for example the 6th of September. By using the distribution of bytes used by one of the other servers, in our case we chose *180.20.249.242*, and used the Anderson-Darling test to compare all other servers detected on this day. Using this approach for two days, we found the two missing IP addresses (*180.20.249.246* and *180.20.249.243*) to complete the malicious infrastructure shown in Figure 6.10 (b).

In our datasets we visually found 4 other similar structures to the one in (b), and in total found 10 'missing' IP addresses, which is almost 1% more than the corresponding cyber threat intelligence feeds show. This percentage might seem low, but this example is done by manually finding and testing this. It would be incredibly valuable to extrapolate this to a magnitude where all known malicious servers are compared to each detected possible malicious servers. If ISPs would incorporate and automate a technique like this, more adversaries can be stopped before actual misuse has been done. This way anti-virus protecting can be brought to a global scale, compared to device-based protection.

7

Infections on IoT devices

Time frame	January 2018 - February 2018
Datasets	Tier 1 ISP NetFlows and TU Delft telescope
Goal	Understanding the evolution, spreading and impact of IoT infections

In the beginning of September 2016 a number of considerable DDoS attacks temporarily took down some large websites. One of these attacks was towards Krebs on Security, exceeding 600 Gbps which is one of the largest attacks on record [71]. Interestingly, these attacks originated from thousands and thousands of the least powerful devices on the Internet - Internet of Things, or IoT Devices - controlled by a botnet called Mirai. This new attack vector is presently reported as one of the most malicious attack vectors of our time. A lot of privately owned devices are being connected to the Internet, providing new technological services to users. Surveillance cameras, TV's, printers, lighting or even smart doorbells connected to the Internet to provide remote access are examples of these kind of devices. Unwillingly, the industry provided cybercriminals with a candy shop filled with vulnerable devices, as most of these devices have standard usernames and passwords which users will most likely not have changed. Mirai targets these devices, using a list of standard usernames and passwords to break into them.

This chapter will analyze our second category of networking infrastructures, IoT devices. By combining NetFlow data with two months of scanning and brute forcing data, retrieved from a network telescope, we can understand IoT infections and C&C structures from a global perspective. Up until now, the only reports of this attack vector have elaborated on the inner workings of the malware. In this chapter we will gain insights on the tactics, techniques and procedures of Mirai and its variants. We report on the average infection rate amongst Mirai variants, based on newly infected machines after having been compromised in these two months. It is imperative to understand how easily IoT devices can be infected and misused. We use the traffic patterns of a known C&C server to uncover additional malicious C&C servers based on their behavior. We have taken a sample of these servers and have analyzed the behavior of their infected machines with each other, which has proven to be rather primitive. Furthermore we have elaborated on the evolution of the infected machines belonging to a C&C server over time. Finally a global revenue estimation, based on the attack capacity of Mirai variants, will be given showing the impact this attack vector can have.

7.1. Background

Internet of Things is a term used to describe the extension of Internet connectivity into physical devices and everyday objects. Electronics are embedded with access to the Internet, enabling users to connect to their devices remotely. Sending a job to the printer whilst commuting, checking the perimeter of the house by remotely checking the security camera or even turning on the heater right before you return home. These are all examples of IoT devices providing new services to users and can ultimately be compromised by an adversary. In this analysis we will make the following contributions:

- We are the first to analyze IoT infection from the perspective that NetFlow data provides. By additionally using scanning data, targeted at a network telescope, we can estimate the leverage of the infection. According to our data, `1` in every `2,345` scanned devices is successfully infected by Mirai variants.
- We are the first to detect Mirai C&C servers without inspecting the actual malware binary and expand existing knowledge by elaborating on their behavior over time. Additionally we provide two examples of infected devices being utilized for DDoS attacks and correlate this with C&C behavior.
- We estimate the amount of revenue which can possibly be made by setting-up a DDoS booter service based on the amount of malicious traffic flowing through these infected machines. A large botnet could possibly earn up to ~$285,000 per year.

The remainder of this analysis will be structured as follows: This section will provide background information on Mirai and the network telescope used in this analysis. Section 7.2.1 will cover basic analysis obtained when combining the datasets and initial explorations, Section 7.2.2 on the other hand provides more in depth information about the infection and its consolidation. Section 7.2.3 will explore C&C opportunities Mirai has and explore our dataset for this kind of traffic and Section 7.2.4 will expand on this by elaborating on monetization options. Finally Section 7.3 will give a revenue estimation based on attack capacities.

7.1.1. Mirai

The Mirai malware is a self-propagating worm, scanning the IPv4 address space for devices that run Telnet or Secure Shell (SSH). When a device running these services is found, the infected machine attempts to log in using a hardcoded dictionary of publicly available IoT credentials. In August of 2016 the first reports of Mirai popped up, where researchers noticed a huge spike of Telnet scanning activity [130]. One month later, in September 2016, Mirai first showed up in the new headlines with large scale DDoS attacks on Krebs on

Security [71] and OVH [95]. Not much later, DNS provider Dyn was targeted by Mirai, affecting a few large companies like Amazon, GitHub and Twitter [55]. Early 2017, the actors behind the Mirai malware were identified [72]. Not much later, the source code of the malware was released on a forum called `hackforums.net`. This provided researchers a foothold on analyzing the code and creating countermeasures. However, also individual cybercriminals could examine this code and use it for their own malicious intents. This led to the introduction of a wide range of Mirai variants, using more advanced password brute-forcing techniques or exploiting other vulnerabilities.

Attack name	Type
GRE Attack	UDP
TSource Query	UDP
DNS Flood	UDP
Random Flooding	UDP
SYN Flood	TCP
ACK Flood	TCP
PSH Flood	TCP
Port 80 Flood	HTTP

Figure 7.1: Pre-set Mirai attack options [6]

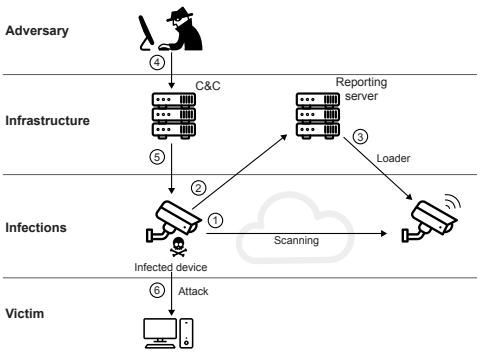

Figure 7.2: Basic infrastructure to maintain and utilize Mirai [4]

Figure 7.2 provides an overview of Mirai's initial operation as provided by the source code [43]. On the initial setup the first IoT device is infected with the malware and will start asynchronously sending TCP SYN packets to pseudorandom IP addresses, ignoring a few hardcoded blacklisted addresses (1). These TCP SYN packets specifically probe Telnet ports 23 and 2323, targeting the software suite called BusyBox, providing UNIX utilities in one single executable. When a potential target has been found, the malware will try to brute-force the username and password using a preset list of 64 factory default credentials. If successful, the initial infected device will report back to a specially designed reporting server (2) which in its turn will load the Mirai worm onto the vulnerable device (3). Alongside with this, Mirai will block TCP ports 23 and 22, preventing other malware to 'take over' the device. At this point the newly infected machine is added to the list of active bots for the C&C server to use. The controlling adversary can, through a simple command-line interface (4), give commands to all of the infected IoT devices in its botnet (5). The Mirai source code comes with a pre-set list of available attacks, as seen in Table 7.1. One of these attacks then can be easily used to target a specific company or other victim (6).

7.1.2. Telescope data

A network telescope, or darknet, is a system allowing the observation of malicious trends taking place on the Internet. This system consists of all unused IP addresses in a network and is mostly used by large organizations or research institutes. In general, these IP addresses are not used for any kind of traffic. Therefore, all traffic to these addresses is suspicious and allowing the observer to gain insights into new or prevalent attack vectors. For this analysis we additionally use insights provided by such a network telescope, located in the Technical University of Delft. Table 7.1 shows the IP address ranges of the TU Delft, totaling 196,602 IP addresses. These addresses are dynamically allocated to users, so not all IP addresses are always used for the telescope.

As by the scanning behavior of Mirai, the network telescope also shows artifacts of IoT devices infected with Mirai scanning the network telescope for vulnerable devices. Each of these addresses have been logged by the telescope, and for this analysis we have focused on the scanning activity during the period of January 2018 up to and including February 2018. The acquired dataset from the telescope, besides from the scanning source address, also provided information about the Mirai variant of the infected device. In total, a number of `698` unique infected devices and `14` different Mirai variants have scanned the network telescope. As a starting point for this analysis, these IP addresses have been anonymized by the provider and extracted from the NetFlow dataset, which we will elaborate on in the next Section.

IP Address range	Subnet
130.161.0.0 to 130.161.255.255	16
131.180.0.0 to 131.180.255.255	16
145.94.0.0 - 145.94.255.255	16

Table 7.1: IP Address ranges of the Technical University of Delft

7.2. Adversarial Tactics, Techniques and Procedures

Some previous work has been done exploring the Mirai infections however, this work has mainly focused on both the inner workings of the malware itself and attacks performed by infected devices. This section will discuss the techniques, tactics and procedures (TTPs) from a more global perspective by combining the data extracted from a network telescope with our NetFlow dataset. Following the infection steps of the Mirai malware, the identification of new vulnerable devices will be discussed. After a device is compromised, this new infected machine will start scanning by itself, consolidating the infection. An adversary can provide commands to these infected devices by means of a C&C server. Each of these steps will be discussed in this section, providing a more complete picture of infections on IoT devices.

7.2.1. Identification & exploitation

In order to gain a foothold on possible vulnerable IoT devices, TCP ports 23 and 2323 are aggressively scanned by already infected devices. At the beginning of 2019, 7 billion of the 17.8 billion devices connected to the Internet are IoT devices, and this number is predicted to grow to 21.5 billion IoT devices in 2025 [76]. This provides a huge and increasing attack vector for individual cyber criminals. For this reason, pseudo randomly scanning the IPv4 space would likely result in a large botnet to control. Before this can be analyzed, we examined the total activity of scanning behavior as shown in Listing 7.1 in our NetFlow data. This example shows scanning activity from a few of the infected devices retrieved from the telescope. To achieve this, IP addresses attacking the network telescope were collected and used to extract world-wide information about the infected devices. Furthermore it can be seen that the compromised devices use the same source port for their scanning activity, inherit to horizontal scanning.

```
date       timestamp    proto          source:port          destination:port    size
2018-01-10 00:10:53.212 TCP     237.200.155.91:27683 ->    222.120.88.17:23      360448
2018-01-10 00:11:50.775 TCP       55.8.174.159:22561 ->   110.111.10.148:23      327680
2018-01-10 00:19:18.610 TCP      201.16.71.152:4127  ->     73.29.67.241:23      491520
2018-01-10 00:49:50.074 TCP      228.12.254.58:18953 ->    207.249.63.65:23      327680
2018-01-10 00:52:11.620 TCP       55.8.174.159:22561 ->    181.233.17.28:23      327680
2018-01-10 00:53:10.124 TCP     237.200.155.91:27683 ->      60.31.25.42:23      360448
2018-01-10 00:54:10.676 TCP       55.8.174.159:22561 ->  173.223.227.212:23      327680
2018-01-10 01:18:16.770 TCP    144.115.234.106:60288 ->   219.80.157.142:23      327680
2018-01-10 01:19:17.059 TCP      45.192.247.31:1025  ->    219.85.188.56:23      327680
2018-01-10 01:25:09.260 TCP    144.115.234.106:60288 ->   47.251.107.193:23      327680
2018-01-10 01:30:08.201 TCP      45.192.247.31:1025  ->    209.186.2.247:23      327680
2018-01-10 01:44:35.774 TCP    203.228.154.242:34058 ->    181.48.11.184:23      344064
2018-01-10 01:49:58.699 TCP       55.8.174.159:22561 ->    60.30.224.234:23      327680
2018-01-10 01:54:41.220 TCP      45.192.247.31:1025  ->    111.13.38.108:23      327680
2018-01-10 02:01:12.566 TCP      45.192.247.31:1025  ->  221.162.100.157:23      327680
```

Listing 7.1: Scanning activity on port 23 originating from infected devices

After having extracted the flow traffic from the NetFlows, Figure 7.3 and Figure 7.4 provide insights into this data. Figure 7.3 depicts the total amount of scanning activity of the 698 infected devices over time. Additionally this shows the amount of scanning activity targeted at the network telescope of the university, illustrating the tremendous scale of which we can analyze this. As the dataset from the telescope also provided each malicious device with a timestamp and Mirai variant, this evolution could be plotted over time in Figure 7.4. The name Mirai was given to a Mirai bot because of the strings `/bin/busybox MIRAI` and `MIRAI: applet not found` which can be found in the source code. These commands determine if the bot has successfully brute forced a targeted IoT device. It is clear that there are four prominent versions of Mirai active: *daddyl33t*, *dwickedgod*, *JOSHO* and *MIRAI*.

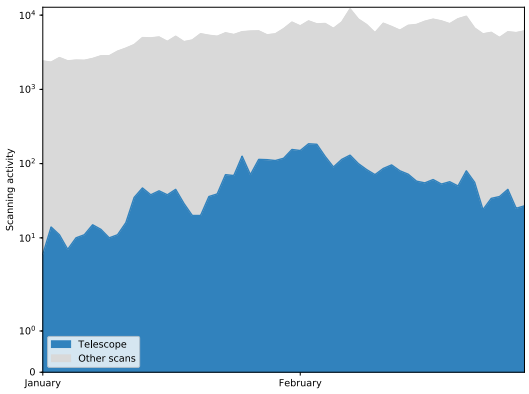

Figure 7.3: Total scanning activity towards port 23 of devices infected with Mirai found in the NetFlow data in log scale

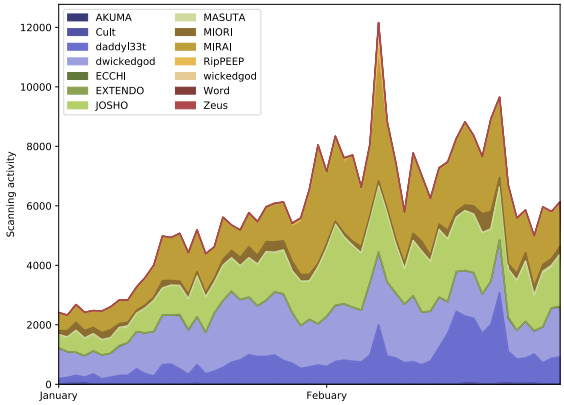

Figure 7.4: Port 23 scanning activity divided by Mirai variant found in the NetFlow data

7.2.2. Infection consolidation

After an IoT device has been infected with Mirai, the malware closes access to ports 23 and 22, and the newly infected device asynchronously starts scanning for other vulnerable devices. The file `scanner.c` from the source code ensures the infected bot will start scanning. The source code also unveils some IP ranges which are skipped purposely, like the internal network or invalid address spaces. Interestingly, the NetFlow data provides a new perspective on the infection as it can be seen when a new device has been infected. When, for example, a known infected IoT device can be observed scanning in the NetFlow data, the subsequent activities of this targeted IP address can also be seen flowing through our NetFlow data. When an IP address has never scanned for ports 23 and 2323 before, and it suddenly starts scanning as aggressively as Mirai infections, we can conclude that this device has been infected. An example of such activity can be seen in Figure 7.5. This example device has never scanned for ports 23 and 2323 before, but starts scanning heavily at 5am on January 23^{rd} after have being scanned by one of the known infected devices derived from our telescope dataset. Interestingly, two days later at 6pm January 25^{th} a sudden stop in scanning activity can be seen, indicating Mirai could have been removed from the device.

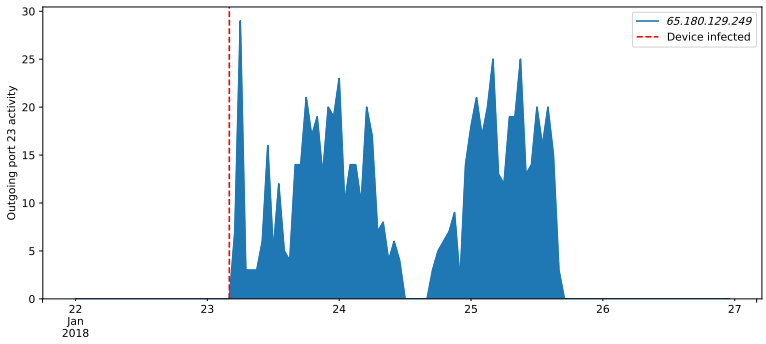

Figure 7.5: Example of an IoT device being infected and scanning for other vulnerable devices

This provides an interesting angle for this analysis, as this can be extrapolated for the observed timeframe and an estimation can be made of the amount of devices which have been infected with Mirai. All infected machines scanning the IP ranges of the telescope have scanned a total of `326,039` unique IP addresses in the NetFlow data. After analyzing traffic from and to all targeted IP addresses by the `698` infected devices, we see **139** of the targeted IP addresses are aggressively scanning after being found by one of the infected IoT devices. From this we conclude that these devices have been compromised. Following this, we can state that 0.0426% of all scanned devices, or 1 in every 2345, get infected with Mirai. Table 7.2 shows the distribution of originating countries of the compromised devices.

Attack name	Type
Brazil	72.7%
The Netherlands	12.9%
Turkey	11.5%
Israel	2.2%
Venezuela	0.7%

Table 7.2: Country distribution of new infections

7.2.3. Command & Control

A successful scanning and infection campaign will provide the adversary with a number of compromised machines under its command, which can then be utilized for misuse. As Figure 7.2 shows, the C&C is an important step for this. After an IoT device has been compromised and registered to the reporting server, it will periodically make contact to the C&C server. In the original Mirai code, the C&C server was interestingly preconfigured to solely talk to the infected bots from port 23 as can be seen in Listing 7.2 [43, 79]. This was probably due to the simplicity of the Telnet protocol. The NetFlow dataset provides the opportunity to investigate the C&C servers and the command propagation. By using the 698 known scanners from the telescope dataset, all traffic from and to these scanners in the NetFlow data and the detection method discussed in Section 5.2, we attempted to find C&C servers in the flow traffic. Initially an example C&C server was found by manually looking for these conditions, shown in Listing 7.3. In this example *63.141.24.225* is the C&C server talking to a few of the infected devices as seen in the telescope. All other connecting IP addresses were verified using AbuseIPDB [77] to be malicious addresses scanning for port 23.

```
func main() {
    tel, err := net.Listen("tcp", "0.0.0.0:23")
    if err != nil {
        fmt.Println(err)
        return
    }
```

Listing 7.2: Source code of Mirai establishing a connection to the C&C server as found in main.go [43, 79]

```
date       timestamp    proto         source:port          destination:port    size
2018-01-10 00:19:49.819 TCP    63.141.24.225:23    ->    219.76.16.146:55997   581632
2018-01-10 00:21:03.132 TCP    63.141.24.225:23    ->   245.181.162.29:34829   942080
2018-01-10 00:21:25.098 TCP    63.141.24.225:23    ->   179.91.106.133:54528   581632
2018-01-10 00:36:36.637 TCP    63.141.24.225:23    ->   245.181.162.29:34829   581632
2018-01-10 01:19:53.000 TCP    63.141.24.225:23    ->     60.19.149.12:38781   360448
2018-01-10 01:20:07.910 TCP    63.141.24.225:23    ->   179.91.106.133:54528   581632
2018-01-10 01:24:36.783 TCP    63.141.24.225:23    ->    219.76.16.146:55997   581632
2018-01-10 01:25:17.891 TCP    63.141.24.225:23    ->     60.19.149.12:38781   581632
```

Listing 7.3: Example of Command & Control traffic to infected IoT devices

Detection Manually looking for probable C&C servers is not doable, therefore we have used the same method which has been used in Chapter 6 to detect C&C servers. Applying the sliding window detection model on this dataset for the timeframe of January up and until February 2018, resulted in 130 probable C&C servers. Analysis showed that a few of these servers were also talking from ports 80 and 443, indicating either a false positive or a more sophisticated infrastructure. For this reason, these servers were removed from the set, leaving 87 highly probably C&C servers. All connecting addresses for five of these servers were manually checked on AbuseIPDB, and all of them showed signs of either brute-forcing or port scanning.

Behavior After acquiring a set of C&C servers, we can look at the same behavior of these servers as shown in Chapter 6. We can hypothesize that the behavior of C&C servers related to IoT infections overlap a lot,

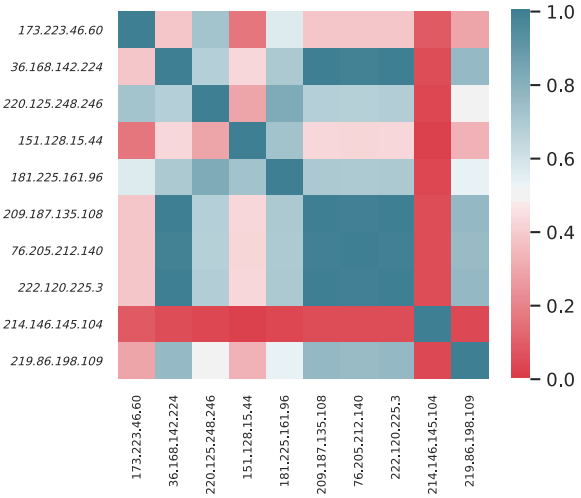

Figure 7.6: Similarity between the activity of the infected devices of ten random detected C&C servers on January 13, 2018

as each infected device most likely does not have a lot of traffic compared to a personal computer. Complementing this, most of the activity will probably employ scanning for other vulnerable targets. To compare this activity, again vectors have been constructed containing percentages of activity for each port as explained in Section 6.2.2. Then these vectors are compared using *cosine similarity*, measuring the cosine of the angle between these two vectors. This methodology has been applied to the list of C&C servers and the combined port distribution of all its infected machines in a similar way as the clients have been compared, resulting in a similarity matrix provided in Figure 7.6.

Some interesting observations can be made when examining this figure. As expected, the similarity between these servers is overall high as most of the activity of the infected devices is scanning for ports 23 and 2323. *214.146.145.104* notably has a significant low similarity with the other servers, indicating interesting activity that will be analyzed further in the next section. *173.223.46.60* and *151.128.15.44* also have a low, but less significant, correlation with the other servers. Following from these results, it is interesting to compare the activity between a few of these C&C servers over time which will be discussed in the next Sections.

Evolution Contrasting to malware infections on clients where compromised devices can be used for multiple attack vectors, botnets sustaining IoT devices have primarily been used in DDoS attacks. This is due to the fact that IoT devices simply do not have much technological options. Having observed the differences in activity between a few C&C servers for one day, it would be interesting to take this a step further and look at the differences in activity over time. For this analysis, three C&Cs representing different similarities were chosen from Figure 7.6. *214.146.145.104* because it has no similarity with the compared servers, *219.86.198.109* as is has diverse range of similarity and finally *220.125.52.78* as it compares well with a lot of servers in contrast to the other two.

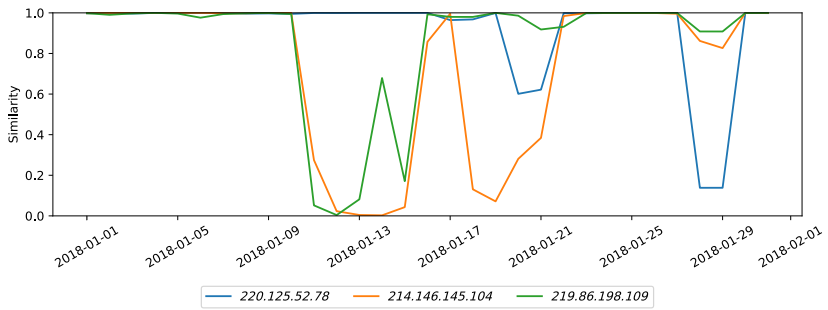

Figure 7.7: Daily behavioral evolution between the infected devices of three detected C&C servers

Instead of comparing the cumulative daily activity of these servers with each other, the same method can show differences in activity over time. These differences over time can possibly explain the contrast in activity of the infected machines. Again by applying the cosine similarity, but this time comparing the C&C servers with themselves, the evolution of the infected devices can be seen in Figure 7.7. For each day the cosine similarity of the port distribution is calculated between the current and previous day. As can be seen in this figure, a similarity of 1 is very common. This can be explained by the fact that IoT devices most often do not have a lot of activity by itself and when infected only talk to ports 23 or port 2323. *220.125.52.78* is a good example of this. The interesting parts of this figure are in the huge negative spikes in similarity, which indicate a sudden change of behavior of all of the compromised devices linked to this C&C. The similarity of *214.146.145.104* in this figure explains the difference with the other servers of the 13^{th} January in Figure 7.6. *219.86.198.109* on the other hand did not show a lot of discrepancy in the heatmap, but does show significant activity changes in this figure.

7.2.4. Monetization

In the previous sections we have discussed various different perspectives of the Mirai malware and how it moves throughout the landscape of IoT devices. During the course of Mirai's lifecycle, DDoS attacks have been its main application. Previous work has mostly been done on the largest attacks with limited data. The results of the previous sections have shown possibilities to dive into the utilization of these infected devices even further. As explained in Section 7.1.1 there are a limited number of DDoS attacks pre-set in the source code of Mirai. Besides DDoS attacks, IoT botnets can also be used for other kinds of malicious purposes, amongst which are sending spam, identify theft and mining for cryptocurrencies. It is interesting to try and divulge how these Mirai variants utilize their network for monetary gain.

Labour division Figure 7.7 illustrated structural behavioral evolution of the infected devices linked to a few C&C servers. These examples show a sudden change in utilization of their resources. Especially *214.146.145.104* and *219.86.198.109* show interesting behavior. The only explanation of such a drastic change of behavior must be some kind of attack taking place. To uncover this, we took these two C&C servers and performed two different analysis on both servers, resulting in Figures 7.8 and 7.9 for *214.146.145.104*, and Figures 7.10 and 7.11 for *219.86.198.109*. Both these analyses are performed for a timeframe in which interesting activity took place according to Figure 7.7.

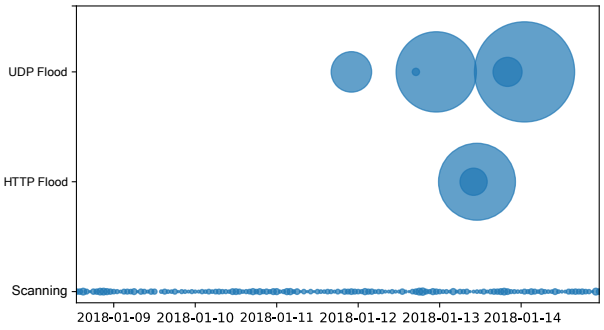

Figure 7.8: Malicious activity of all infected devices of the C&C server *214.146.145.104*

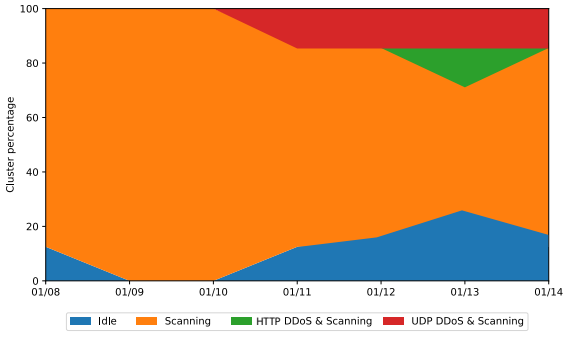

Figure 7.9: Detected clusters all infected devices of the C&C server *214.146.145.104*

First we plotted the destination port distribution over time. If the infected devices would only be scanning, a steady activity would be visible towards port 23. In both Figures 7.8 and 7.10 this steady trend can be seen, indicating active scanning for new devices to infect. Besides the scanning activity, both C&C servers also show huge spikes of activity towards TCP port 80 and towards a lot of high UDP ports indicating DDoS attacks. Figure 7.10 for example has peaks of 250 Gbps. This behavior explains the differences of activity in the evolution as seen in Figure 7.7 and is also expected behavior according to the theory.

Secondly, we have employed the HDBSCAN clustering method, as explained in Section 5.3.3 to uncover the division of labor of the compromised devices controlled by *214.146.145.104* and *219.86.198.109*. Figures 7.9 and 7.11 show the resulting cluster movement over time. In both figures it is clear that most infected machines are used to actively scan for new vulnerabilities. It is also interesting to see not all infected machines are being utilized for DDoS purposes, which could be a tactic from the adversary to keep a part of its

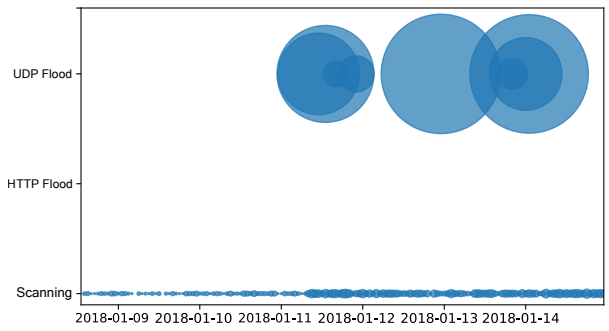

Figure 7.10: Malicious activity of all infected devices of the C&C server *219.86.198.109*

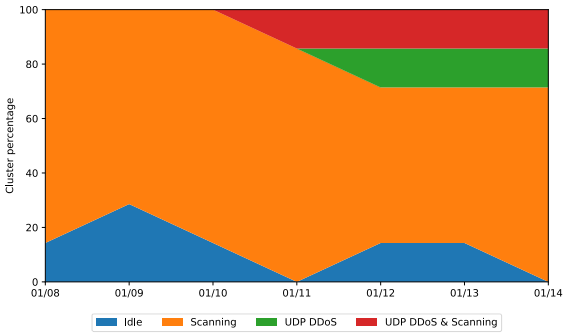

Figure 7.11: Detected clusters all infected devices of the C&C server *219.86.198.109*

infected machines not blacklisted and thus consolidating these devices.

DDoS booter service Services can monetize their networks through various means, however these are very difficult to estimate. A malicious actor could hold a victim ransom for a certain amount of (virtual) money in exchange for not taking the victims' website or service down. Measuring this is not doable, as there will likely be no historical data of this besides the occasional news item. Identity theft, although less likely, can also be possible with IoT botnets. This information can then be sold on underground markets for profit. What we however can estimate, is how the malicious infrastructure can be used a service to perform DDoS attacks. The first Mirai version was initially created to specifically perform DDoS attacks. Over the years actors have tried to make this more lucrative by offering DDoS as a service, or so-called DDoS booter services. These services allow clients to pay, usually with cryptocurrencies to ensure anonymity, for bandwidth to attack specific websites or services. This bandwidth can be created by a network of infected (IoT) devices.

Mirai variant	Scanning	DDoS	Other
AKUMA	97,06%	0,00%	2,94%
Cult	12,84%	84,32%	2,84%
daddyl33t	31,62%	59,19%	9,19%
dwickedgod	20,75%	68,89%	10,36%
ECCHI	34,85%	0,00%	65,15%
EXTENDO	44,17%	47,34%	8,49%
JOSHO	16,76%	69,71%	13,53%
MASUTA	99,14%	0,00%	0,86%
MATOS	100,00%	0,00%	0,00%
MIORI	19,14%	66,55%	14,31%
MIRAI	22,75%	64,43%	12,82%
MMIKKI	12,62%	86,39%	0,99%
NGRLS	44,92%	0,00%	55,08%
OWARI	99,00%	0,00%	1,00%
RipPEEP	23,39%	0,00%	76,61%
SORA	90,53%	0,00%	9,47%
wickedgod	34,92%	63,12%	1,96%
Word	22,67%	0,00%	77,33%
Zeus	98,60%	0,00%	1,40%

Table 7.3: Division of scanning, attack and other traffic approximated for each Mirai variant.

Compared to normal web clients like personal computers, IoT devices normally do not send or receive an abundant amount of traffic. Therefore, activity out of the ordinary can be spotted with ease. When an IP address sends a lot of traffic to various different targets at a specific port, hosting a known vulnerability, we can say this entails scanning or brute force activity. On the other hand a DDoS attack can be seen by observing extreme amounts of traffic sent to one target for a specific amount of time. We have made an estimation of the diversification in activity amongst the different variations found after the `/bin/busybox` string of the Mirai code scanning or brute forcing the TU Delft telescope, which can be found in Table 7.3. We have bound DDoS activity when amount of traffic in bytes towards one victim passes a baseline for a consecutive time of 15 minutes. This baseline is defined as the average bytes of the total amount of traffic sent from one infected

Mirai variant	DDoS traffic (Gb) January 2018	Monthly revenue small botnet (5,000)	Yearly revenue small botnet (5,000)	Yearly revenue large botnet (100,000)
Cult	3,501.63	$1.197.14	$14,365.65	$287,312.98
MIRAI	16,212.85	$967.21	$11,606.52	$232,130.34
JOSHO	6,220.08	$694.01	$8,328.15	$166,562.94
dwickedgod	4,805.77	$604.50	$7,253.99	$145,079.74
MIORI	5,038.30	$546.16	$6,553.88	$131,077.65
daddyl33t	2,533.05	$404.48	$4,853.74	$97,074.84
MMIKKI	80.78	$307.75	$3,692.97	$73,859.45
EXTENDO	153.58	$102.39	$1,228.67	$24,573.38
Total		**$4,832.64**	**$57,883.57**	**$87,323,146.98**

Table 7.4: Estimated revenue which can be made by the top 8 most active attacking Mirai variants

device doubled. After investigating the numbers in Table 7.3, some interesting observations can be made. About half of the traffic in the DDoS column goes, as expected, to ports 80 and 443. Otherwise a lot of UDP floods can be seen when looking more closely at the NetFlow data, which are most popular with *MIRAI* and *JOSHO*. *wickedgod* is an interesting case, as this variant shows a few DDoS attacks on port 123, similar to activity what a blog post from Netab describes [39]. Just as in this blog post, this variant also tries to infect new devices on port 81, something which is not used in the traditional code of Mirai. This port leads us to a vulnerability regarding GoAhead and OEM security cameras, estimating that more than 185,000 devices have potential problems. *dwickedgod* and *Cult* show interesting behavior, as their attacks are towards ports 31504 and 58526 respectively, something which is less sophisticated then randomizing these ports. Besides these differentiating attack vectors, different infecting tactics can be seen between the variants, possibly to further optimize infection rates. *Cult* and *MIORI* for example show a lot of scanning activity, besides the usual ports 23 and 2323, towards ports 4672 and 7547 inhabiting vulnerabilities which allow remote shell execution.

7.3. Quantification of revenue

The previous sections have shown how this attack vector can spread easily through landscape of poorly protected smart devices connected to the Internet. To get a feeling of the impact this attack vector we will try to make a quantification of earnings malicious actors may make based on the differences we have amongst the Mirai variants, something which has not yet been done before. As we cannot estimate anything about earnings from holding victims ransom, or other angles which cannot be measured with NetFlow data, we will focus on the amount of revenue which can be made when using the malicious network for DDoS as a service. In the previous section we have elaborated on this attack vector.

Quantifying how much money an actor can make by offering the bandwidth of the malicious network is tedious, as every actor can sell for any price they offer. Clients can buy an amount of bandwidth for a certain price and attack a host of their choosing. Prices range from as low as 100 Gb/s spanning 300 seconds for $5, to 1 Gb/s spanning 150 seconds for $30. This all depends on both what the service offers and how good the protection is of the service/website which is to be attacked. To calculate this, we have formed a simple equation

$$((\text{traffic [bytes]} \div \#\text{ of infected devices}) \times \text{botnet size} \times \text{cost for bandwidth [\$]}) \div \text{bandwidth [bytes]} = \text{revenue in dollars}$$

and we have taken a lower bound of the average of 10 different IP stress / DDoS booter services, entailing 5 Gb/s spanning 300 seconds for $8. This will ensure that our estimation will be conservative. Prices can even be up to four times as high when the victim has DDoS protection in place. The # of infected devices is the amount of devices we have observed with the different variants of Mirai in the TU Delft telescope.

The estimations can be found in Table 7.4 and shows two example revenue's, for a small botnet and for a larger one. These numbers show that a decent amount of revenue can be made by criminals, just by renting the bandwidth of their network of infected devices. This is merely a lower bound of the earnings adversaries can make with such malicious networks, as other methods like holding victims ransom, can possibly create much more revenue.

8

Infections on routers

Time frame	July 2018 - April 2019
Datasets	Tier 1 ISP NetFlows, Shodan, Censys and TU Delft telescope
Goal	Understanding the evolution, spreading and impact of large scale cryptojacking on Internet routers

The final part of our analysis will address routers, networking devices which send data between computer networks. Routers can be seen as the actual backbone of the Internet, as each packet traverses through a substantial amount of routers from source to destination. Each time somebody checks their e-mail or performs a web search, requests flow from router to router to obtain the answer to this request. Routers are positioned at multiple points in computer networks, ensuring that each of these requests get answered. The request for checking a new e-mail will traverse from a home-based router to ISP grade routers, eventually arriving at the router which connects to the server responsible for the e-mail service. It can be imagined that these routers have an incredible responsibility in terms of security, as vulnerabilities in these routers can possibly lead to huge problems.

In this chapter we report on a new attack vector in which cybercriminals used a firmware vulnerability in routers of the brand MikroTik to gain control of the router and rewrite outgoing user traffic. By embedding special code in every outgoing web connection, they can use the computing power of all connected computers for adversarial tasks, in this case to mine so-called cryptocurrencies. By combining our NetFlow dataset with weekly crawls and telescope traffic, we have been able to follow their activities over a period of 10 months. We report on the tactics, techniques and procedures, and coordinating infrastructure of the adversaries. We found that during this period these adversaries were in control of up to 1.4M routers, or approximately 70% of all MikroTik devices deployed globally. Besides this we track the evolution of the used mining services and keys used by adversaries, and report on C&C servers we have found and observed. Additionally, we found different levels of sophistication among adversaries, ranging from individual installations to campaigns involving large numbers of routers.

8.1. Background

Cryptocurrencies, which started with the release of Bitcoin in 2009 [91], represent monetary value secured by a blockchain. Transactions are permanently stored in an ever growing list of records, where transaction data can be added by solving a cryptographic challenge. This puzzle is dependent on the last block, the current transactions and their recipients, and once solved a new record gets inserted into the chain consolidating the record of previously conducted activities. Users are incentivized to participate and donate computational resources to the system, as the one solving the puzzle gets a (fraction of a) cryptocurrency unit as reward.

As the Bitcoin blockchain was designed to continuously increase the difficulty of these challenges, Bitcoin mining is no longer profitable on regular PCs, now requiring specialized hardware such as ASICs. As a result, thousands of other cryptocurrencies, so-called alt-coins, have emerged that replace the proof-of-work algorithm of Bitcoin with alternative mechanisms to validate transactions. The Monero cryptocurrency [125], which uses a private blockchain with transactions not publicly visible, relies on the CryptoNight algorithm, a memory-intensive computation of subsequent reads and writes that can be efficiently run using the processor-level cache found in typical consumer-grade CPUs.

The reward that can be gained from these alt-coins has however also attracted the attention of cyber criminals, who have distributed cryptomining code through malware or as part of botnet installations [96]. With the recent introduction of a JavaScript miner by Coinhive in 2017, cryptomining code can now be shipped as part of a web page and be efficiently executed by a web browser, thereby providing an easy, scalable and low-effort method to roll out cryptomining to a large user population. This has led to new business and revenue models, for example replacing advertisements by letting website visitors donate computational resources [126].

The relative ease with which website visitors can be recruited for cryptomining has also led to a major surge in illicit cryptomining, so-called cryptojacking or drive-by mining, in which the visitor's resources are hijacked without knowledge and consent. Aside from cryptojacking that is initiated by the website owner without their visitors' consent, criminals also seek to increase their revenue by compromising websites to install mining code [21, 80], as well as hiding miners in third party software used by web masters and thus inadvertently being deployed [11, 23, 139]. Cryptojacking is also spread through exploitable vulnerabilities in content management systems [88], through the distribution of advertisements including malicious code [87] or through malware [96]. Previous work by Konoth et al. estimate that cryptojacking possibly yields monthly revenues of $41,000 for the 10 most successful perpetrators across the Alexa Top 1M [70].

In this Chapter, we will analyze a previously unseen attack vector for cryptojacking, namely man-in-the-middle attacks launched through compromised consumer and edge routers that inject mining code into every web page requested by their users. This was made possible by a firmware vulnerability in MikroTik routers discovered in early 2018 [93], which allowed adversaries to change the device configuration and create an out-

going HTTP proxy, and that remained widely unpatched until a year later. By following reconnaissance scans of the perpetrators through a large network telescope, the detection of compromised routers through semi-weekly crawls, and the tracing of connection patterns of adversaries, their supporting infrastructure and the compromised routers based on NetFlows from a Tier 1 operator, we are able to provide a comprehensive insight into how this vulnerability scaled out into massive cryptojacking campaigns that drastically overshadow previous mining activities. In this work, we make the following four contributions:

- We are first to investigate a new type of attack that exploits Internet infrastructure for cryptomining, and show how over a period of 10 months after the initial discovery of the vulnerability groups of criminals launch massive campaigns to control 1.4M routers, with a peak of 460,618 simultaneously infected routers.
- We have analyzed adversarial tactics and unveil the supporting infrastructure used within the campaigns, and are able to show differences between groups in how they locate their victims, compromise routers, and run their infrastructure.
- We demonstrate that previously reported vectors are negligibly small in number of affected users and created revenue, compared to the reported MITM attack. We find that this attack yielded monthly revenues, estimated exceeding $1,200,000 per month for the top 10 grossing accounts, a factor of 30 larger than previously estimated cryptojacking revenues from hacked websites, malicious advertisements and website-owner initiated mining combined.
- We have observed high levels of sophistication in three identified campaigns, of which the largest involved 40 mining accounts linked to one single actor.

The remainder of this analysis will be structured as follows: Section 8.1.1 introduces the concept of cryptojacking and previously used modus operandi, and describes the vulnerability and its exploitation used for a MITM-based cryptojacking on routers. Section 8.1.2 describes the datasets used in this study. Section 8.2 presents the techniques, tactics and procedures in use during the identification, exploitation, monetization and maintenance of the compromised systems. Sections 8.3 and 8.4 puts the techniques and sophistication levels of the ecosystem into perspective and quantifies adversarial revenues.

8.1.1. Cryptojacking

The introduction of memory-bound cryptocurrencies like Monero allowed for new methods of cryptomining, one of them being browser-based cryptomining. These cryptocurrencies, together with new web technologies such as WebAssembly (native speed code execution within the browser sandbox), WebWorkers (separate JavaScript instances), HTML5 WebSockets (simple multiplex TCP connection) and the Stratum Mining Protocol (JSON-RPC formatted mining pool communications) paved the way for the creation of an efficient browser-based cryptominer by Coinhive in 2017. Their miner, and most other mining applications, work as follows: the user visits a cryptojacking website which includes (a reference to) a cryptojacking script. This script explores the host system, downloads a highly optimized WebAssembly module for mining and spawns a number of WebWorkers to run this module. Consequently, it sets up a connection with a mining pool through a proxy server operated by the service, authenticating using a *siteKey* or (Monero) wallet address, which is essentially the account of the adversary. For readability when we mention *siteKeys*, we will refer to them by the first six characters of the key. The mining pool distributes a job to work on, the WebWorkers start mining and found hashes are submitted to the mining pool. When the browser window is closed, all mining activity stops.

Past modus operandi of cryptojacking

As stated by a New Jersey Attorney General in 2015 [56], mining cryptocurrencies with the computing power of others is not considered illegal when a clear notification of such activities is shown and the possibility of opting-out exist. However, most cryptojacking cases lack these and are therefore considered illegal. There have been cryptojacking scripts found on malware infected PCs [96], but since the release of the Coinhive miner, cryptojacking in the form of browser-based mining gained enormous popularity. There is a large number of websites running a cryptominer to increase their revenues, such as The Pirate Bay [126], but cryptojacking has also occurred on websites where the owner did not initiate it. Website compromises, such as government pages of the Indian government [21] in 2018, have led to cryptojacking infections, but cyber criminals are constantly searching for more efficient methods to deploy their miners. To spread an infection over

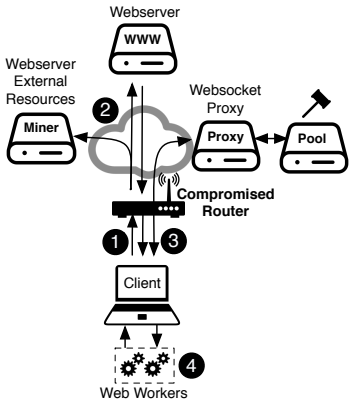

Figure 8.1: Through a MITM attack on routers, adversaries performed cryptojacking on HTTP websites visited by users.

a large number of websites, attackers abused third party software (such as infecting WordPress plugins [139] or exploiting Drupal CMS vulnerabilities [114]) with cryptojacking scripts as well as injected advertisements with mining code and served them through ad networks to websites unaware of any infection [87].

Pervasive Cryptojacking through Man-In-The-Middle Attacks

As mentioned in the previous section, cryptomining code is included as part of the served HTML page, which requires the website owner to explicitly install a cryptominer or inadvertently embed it due to a compromised component. It is however also possible to modify the request in transit, by modifying the HTML as a man-in-the-middle.

In the attack reported in this analysis, adversaries compromised the routers' operating system, and reconfigured the system causing requests from clients to any website to be rewritten and channeled to an internal HTTP proxy server running on the device. With the compromise of the router, the perpetrator installs a script to change the firewall rules of the device, opening telnet and SSH to the Internet if not already exposed, and introduces a firewall rule to redirect outgoing requests on port 80 to a proxy port. Finally, it deploys an HTML page sent by the proxy to each outgoing connection.

While different groups of actors followed slightly different techniques, tactics and procedures as we will show in Section 8.2, it meant as shown in Figure 8.1 from the perspective of the user that any outgoing connection to port 80 was redirected to the proxy on port 8080 (1). This served a web page based on a common template, shown in Listing 8.1 for a connection to `facebook.com`. This led the client's browser to fetch two web resources: the outer frame containing a JavaScript that loaded cryptomining code (2), and within the frame the actual website the user intended to visit was displayed (2). The client's web browser would setup a WebSocket connection to a WebSocket proxy or to a mining pool in order to retrieve instructions (3), and spin up WebWorkers to mine for a specific *siteKey* (4).

From the perspective of the perpetrator, this design has a number of advantages. First, as the iframe opens

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html;charset=windows-1251">
 <title>"http://www.facebook.com/"</title>
 <script src="https://coinhive.com/lib/coinhive.min.js"></script>
 <script>
   var miner = new CoinHive.Anonymous('<mining key>', {throttle: 0.1});
   miner.start(CoinHive.FORCE_EXCLUSIVE_TAB);
 </script>
</head>
<frameset>
 <frame src="http://www.facebook.com/"></frame>
</frameset>
</html>
```

Listing 8.1: HTML returned by the proxy of an infected router, with a Coinhive miner and the actual page in an iframe

the original page, the user will at first sight not notice anything wrong, as the requested web page loads within the borderless iframe. Second, as the interaction with the loaded website functions normally, the victim will remain on the Web page for an extended period of time, thus increasing the time the miner will run in the background. Third, as clicks on the embedded page do not reload the outer frame, the cryptominer keeps running during navigation on the visited web page, thus maximizing mining cycles.

Susceptibility of HTTP(S) connections While the browser address would show a connection to the router instead of the requested URL, the hijack from a usability perspective is both comparatively frictionless and effective. The original URL is displayed as the title of the page, and experimentation on recent versions of both mobile and desktop browsers showed that websites can even be loaded via HTTPS within the iframe without triggering a warning by the browser. In this case, the HTTP proxy loads an unencrypted HTTP page with an iframe showing the secured HTTPS contents. Thus, unless the rewritten URL raises suspicion with the user, we can expect the activity to go by relatively unnoticed.

Vulnerability CVE-2018-14847

The exploited vulnerability in this attack is CVE-2018-14847 and affected MikroTik RouterOS through version 6.42, allowing *"unauthenticated remote attackers to read arbitrary files and remote authenticated attackers to write arbitrary files due to a directory traversal vulnerability in the WinBox interface."* [93]. Of special significance to the attack is that MikroTik uses RouterOS across their entire product line, making the vulnerability applicable to a large number of both consumer and carrier-grade routers. As we will see later, the vulnerability of carrier-grade devices explains the magnitude of cryptomining activity that could be realized in this attack.

WinBox is a small Win32 binary that allows for the administration of RouterOS using a graphical user interface. The functionalities of the WinBox interface are almost identical to the console functions, but some advanced and critical system configurations changes, like changing the MAC address, cannot be made from the WinBox GUI. Several WinBox commands did not require authentication, e.g., an attacker could open files for reading while being unauthenticated, while another allows an attacker to write files to disk given some authentication [124]. By sending a carefully crafted package to the WinBox service on port 8291 exploiting one of these commands, the attacker would retrieve the user credential store `user.dat`, and using these credentials drop files to disk to enable a developer backdoor [124]. Triggered if a specific file, `/pckg/option` or `flash/nova/etc/devel-login`, is present on the system, the developer mode sets up a root BusyBox shell accessible over port 22 (SSH) or 23 (Telnet) giving complete control over the device.

8.1.2. Additional datasets

The study was made possible through a combination of three datasets each covering a different angle of the reported malicious activity: first, we use the traces from a large network telescope to trace adversarial scanning activity. Second, we rely on a periodic crawl for the proxy status page by Censys [31] and Shodan [116] to discover which routers were infected. And third, we use our NetFlow dataset to visualize the communication patterns between the infected routers and the remaining Internet to identify their staging hosts and quantify the volume and revenue of this large scale exploitation.

Network Telescope

In order to exploit routers using the WinBox vulnerability, the attacker must first know where vulnerable routers are located. This identification and localization could be done in one of two ways: either the adversary scans the Internet for open ports or banners that would identify the devices, or obtains a list of devices.

To discover which adversaries are actively scanning the Internet for devices with the WinBox vulnerability, we rely on a large network telescope of three partially populated /16 networks, through which a total of approximately 130K IP addresses are monitored. In order to discover whether TCP port 8291 is open and to send a payload triggering CVE-2018-14847, adversaries first need to complete a TCP handshake. This ensures that perpetrators cannot spoof their source IP as otherwise the handshake couldn't complete, and reveals the location of the adversary or a potential proxy.

Active Scans of Censys and Shodan

As discussed in Section 8.1.1, the exploitation through the rewriting proxy was unusual as it unnecessarily exposed the webpage to the Internet instead of just presenting it to just the users on the inside. Since RouterOS

allows both port 80 and port 8080 to be used by a HTTP proxy, an Internet-wide survey of these ports made it possible to discover which MikroTik routers are currently infected as they are serving the proxy page, and based on the embedded key track who currently "owns" the device.

Censys To trace infections and their evolution, we thus additionally rely on Censys [31], which scans and archives the responses of all IPv4 addresses on a number of common ports, among them 8080 and 80. We retrieved these Internet surveys twice a week between the first wide-scale exploitations in July 2018 until April 2019, and identified a router as a MikroTik system if the proxy header was set to `MikroTik HttpProxy` and as infected if it contained scripts or code for crytomining. This yielded a total of 1,452,550 unique IPs belonging to an infected router at some point during the study.

Shodan A second service that scans devices for open ports is Shodan [116]. Besides listing ports, the service additionally extracts banners to link it with known vulnerabilities, and makes it possible to conveniently search specific devices and credentials. Given the Internet surveys of Censys, we queried the databases of Shodan and recorded when a particular IP that could be identified as compromised due to the mining proxy page appeared in Shodan's database.

8.2. Adversarial Tactics, Techniques and Procedures

In this section, we analyze the techniques, tactics and procedures (TTPs) adversaries use in the exploitation of 1.4M MikroTik routers and their subsequent abuse. We will split this discussion based on the stages in the life cycle of a router infection as shown in Figure 8.2. This life cycle begins with the identification of candidate victims, the exploitation of the vulnerability, and methods used to gain a foothold and consolidate the infection. After a device is compromised, actors will install tools to monetize the exploited routers and perform maintenance, until the infected system is removed from the pool due to decommissioning or patching. As we will see in this section, each of the individual steps can be accomplished in a variety of ways, and we find adversaries using different techniques and tooling in each of the life cycle phases. In Section 8.3, these findings on the individual stages will be combined into an overview of the actor landscape.

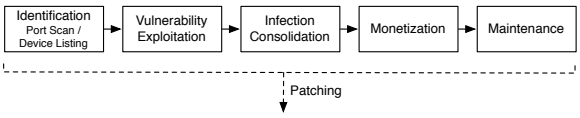

Figure 8.2: Life cycle of the vulnerable routers.

As we will see in this section, each of the individual steps can be accomplished in a variety of ways, and we find adversaries use different techniques and tooling in each of the life cycle phases. In Section 8.4, these findings on the individual stages will be combined into an overview of the actor landscape.

8.2.1. Identification

In order to gain a foothold on a machine, adversaries first need to know where exploitable devices are located. This also holds for vulnerable MikroTik routers, of which according to market surveys approximately 2M units were installed worldwide [113]. Routers are usually deployed in one of three ways on the Internet: (a) they are either provided by the Internet Service Provider (ISP) to the customer who uses the device to connect to the ISP's network, (b) they are bought, deployed and operated by the customer to connect to the Internet, or (c) they are part of the network infrastructure of the ISP. As RouterOS was used across the entire MikroTik product line, we see vulnerable devices of all three types in practice.

Figure 8.3 shows a heatmap of all MikroTik routers that were exploited at least once during the study period, mapped to a geographic location by using the MaxMind GeoIP database [78]. The devices are very prevalent in select parts of the world, especially Brazil or Indonesia, where such a device responded at 29%, and 35% of all publicly accessible IP addresses of the largest operators in these countries, thereby indicating that these devices were provided by the ISP to the customers. Table 8.1 lists the number of compromised MikroTik routers for the 10 most affected autonomous systems and their share of the overall infected population. We can see that 136,659 exploited MikroTik routers could be linked back to the 5 most compromised ISPs. The heatmap however also shows sparse deployments throughout the world, with clusters appearing

AS	Count (%)	AS	Count (%)
Telekomunikasi Indonesia	55,082 (3.8%)	Cat Telecom	12,883 (0.9%)
Telefonica Brasil S.A	33,589 (2.3%)	Rostelecom-AS	11,352 (0.8%)
TCI	21,357 (1.5%)	TOT-NET	11,136 (0.8%)
PTC-Yemennet	13,585 (0.9%)	UKRTELNET	10,993 (0.8%)
BSNL-NIB	13,046 (0.9%)	IR-THR-PTE	9,248 (0.6%)

Table 8.1: Top 10 most affected Autonomous Systems (ASN)

in densely populated areas, proportionally to the number of IP addresses located in an area, suggesting that these routers were owned and operated by end customers.

Discovery using Port Scanning To localize potential victims, adversaries could make use of port scanning to test remote IPs whether they have TCP port 8291, the port associated with the WinBox vulnerability, open. This reconnaissance could be done at different levels of granularity and sophistication: on the low end, attackers could blindly trawl through the entire Internet in a horizontal port scan to discover any potential victim, albeit at the disadvantage of creating much noise and potentially being identified, blocked and blacklisted. A sophisticated scanner could however do some prior background research, and determine in which networks large MikroTik installations exist, as a result of these devices being used within an ISP's network or being given out to its customers.

We can differentiate between these type of strategies using the data provided by the network telescope and general flow statistics of the Tier 1 operator. Figure 8.4 shows the absolute number of packets directed against port 8291 in our telescope as well as traffic carried by the operator during 2018 aggregated by day. The vertical lines show important milestones in the lifespan and news coverage of the WinBox vulnerability. On March the 24^{th}, the average daily traffic towards TCP 8291 exploded by 6 orders of magnitude, as the Hajime botnet executed a short, but concentrated horizontal scan for the port across the Internet [92]. On April 23rd, the vulnerability was discovered and patched by MikroTik, and the resulting news coverage only lead to a very minor continuous increase in scan traffic. Starting mid-July, the first cryptojacking installations started to appear in the wild, followed by a public proof-of-concept for the exploit and beginning of August the CVE report was published in the National Vulnerability Database [93].

As we can see from the graph, the general characteristics of telescope and NetFlow traffic resemble each other. Both record the same sudden increase in network traffic due to the Hajime botnet at the same moment and with a similar magnitude, demonstrating that the botnet initiated an unspecific worldwide trawl for the vulnerability. While after this burst the telescope traffic returns to business-as-usual, aside from selected worldwide scans, we see in the NetFlow data that geographically targeted scans – not targeting our network telescope – immediately followed, and continued to run until the end of the observation period. As the number of infections started to rise in December 2019, we observe increased worldwide scanning activity as both

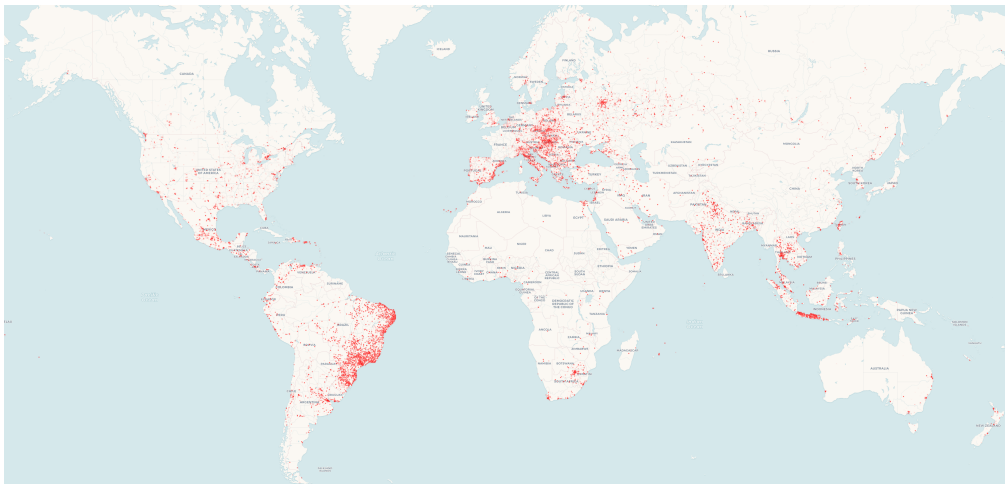

Figure 8.3: Geographical location of the MikroTik routers compromised during the study period.

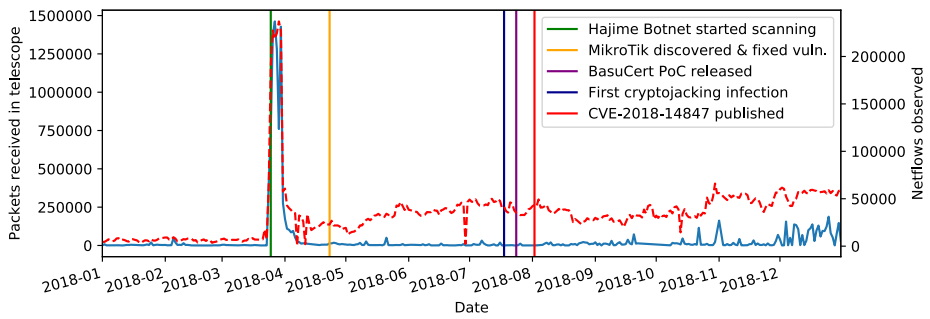

Figure 8.4: Packets received on port 8291 in our network telescope (in solid blue) and NetFlows observed (in dashed red).

our telescope and NetFlow data report more connections towards port 8291.

Out of a total of 1.7M IP addresses that probed the three /16 network ranges in our telescope as well as the rest of the Internet during the late March burst, only 124K IPs continued to probe specific parts of the Internet for router vulnerabilities. This seems to indicate that the scanners used the data collected from previous tests (as our passive monitors would not respond to 8291), or that additional knowledge – such as the popularity of MikroTik in specific parts of the world – is used to steer the search. In order to determine the specificity of these scanners, we compared the traffic distributions of the Tier 1 operator towards all autonomous systems (AS) with the traffic distribution for the anonymized scanning source IP addresses. This relative comparison accounted for the fact that the operator would not be part of an exact random sampling of all worldwide traffic flows, but that due to BGP policies and specific IXP and PoP presences certain autonomous systems would be preferred. From this relative comparison we can determine whether sources showed specific preferences for select networks, or scanned the Internet non-discriminantly. Figure 8.5 shows a summary of all scanners as a histogram of the scanners' deviation from the expected non-discriminatory baseline. As we can see in the graph, there exist three basic behaviors: the bulk – which is also visible in our telescope – targets the entire Internet unspecifically, a smaller but significantly sized group that specializes and concentrates the scan on a specific AS, while a small portion of adversaries scan a large but apparently curated list of destinations.

Localization using Public Datasets In addition to actively scan and probe IPs on the Internet to test whether they are running RouterOS and are potentially exploitable, attackers could try to get a pre-made list of device IPs to connect to potential targets directly, for example by searching on Shodan. To determine whether the attacker uses such services to locate vulnerable routers, we consider the moments Censys retrieved a proxy page from a router with a mining key on port 8080 or 80, which means that at this moment the device was compromised. If at the moment of publication on Censys the router was not listed yet in Shodan, the perpetrator must have found the vulnerable router through independently scanning for it. If prior to the Censys publication, there already existed a record in Shodan, the attacker could have obtained knowledge from this service.

When we track this relationship for when keys first appeared on the 1.4M routers, we find that 54% of

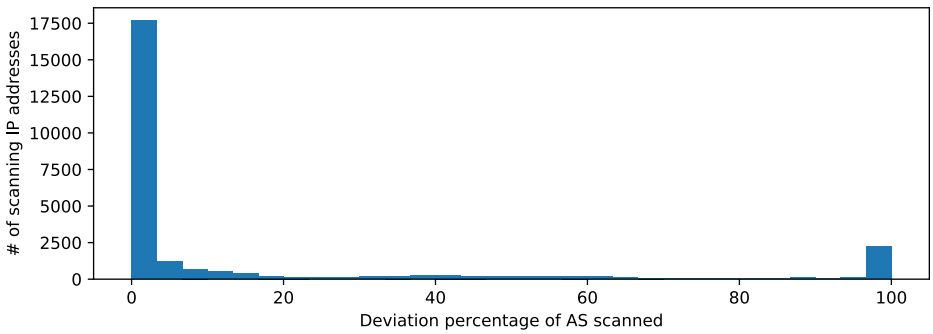

Figure 8.5: Histogram of the specificity of scans for port 8291.

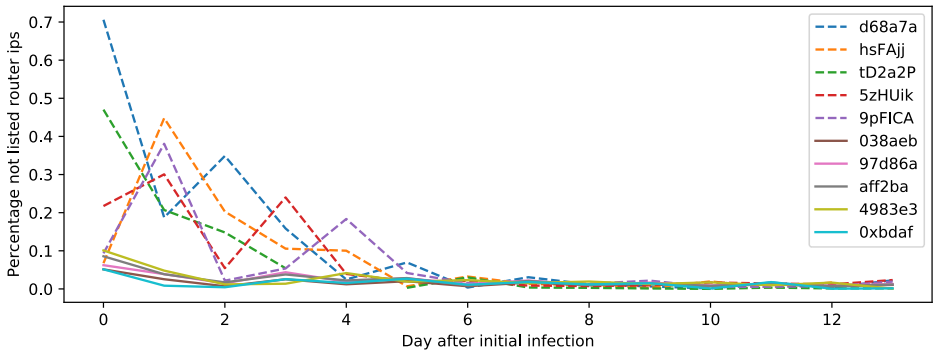

Figure 8.6: Percentage of infected unlisted routers per key.

the cases when an actor first found a device it was the result of it being listed in Shodan, whereas 29% of the initial infections were derived from independent scanning. In the rest of the cases, too few routers were compromised to significantly categorize. While public vulnerability databases such as Shodan seem to give a head start to perpetrators, the situation is more nuanced when we look at individual actors. Figure 8.6 shows the percentage of unlisted routers used by actors within the first 14 days of their activity. We clearly see two regimes. For readability when we mention *siteKeys* during the analysis, we will refer to them by the first six characters of the key. Innovators and early adopters such as *d68a7a* and *hsFAjj* which are shown as dashed lines (for key emergence see Figure 8.10) all perform their own discovery, and start off with a high number of new, unlisted nodes. This percentage drops over time, as the compromised boxes are then included in Shodan. The long lasting campaign *tD2a2P* starts out with 42% unlisted routers on its first, and kept adding unknown devices to its installation base for the months to come. On the other hand, we find a large number of campaigns which primarily feed off public lists to populate their setups. The largest and most profitable campaign *6a9929* had at its peak 13,815 routers infected, almost exclusively drawn from public lists. As we will see in Section 8.3, the degree of innovation is not a proxy for the amount of revenue these campaigns make – innovation does not always seem to pay off.

8.2.2. Vulnerability Exploitation

With vulnerable routers identified, adversaries can trigger the vulnerability by sending a simple payload. While the activities of the perpetrators on the devices cannot be inferred using our datasets, we can and will investigate in this section patterns on how adversaries infect devices, and how infected devices are taken over.

Infections and Re-infections From previous discussion, we have seen that a large number of IPs in our telescope and in the NetFlow data scanned for port 8291, and that actors additionally used records such as Shodan to find exploitable targets. Once a device however appears in Shodan, it could already be infected, as it lists a proxy running on port 80 or 8080. This naturally raises the question whether and how re-infections occur, in other words whether actors are grabbing compromised devices from others.

Figure 8.7 depicts the transition behavior of the 1.4M routers between keys, filtered to only include edges if more than 500 devices are taken over from the original "owner" by a particular new actor. Most visible are the large transitions on the right between initially widely used key *hsFAjj* towards *SK_LCx* and *oDcuak*, which could indicate key rotation by the same actor. There are however significant flows between *SK_LCx* and for example *J3rjnv*, where a little over 15k routers shift back and forth. The left side of the graph shows smaller and more nuanced interactions between the different installations. First, a chain of key transitions from *WDUFD* to *ByMzv3* to *aff2ba* to *ef18c8* shows a sequence of keys actors rotate through. Second, we see a key such as *4983e3*, which draws its installation base from many other keys, using lists of already infected devices and then reinfecting them with a new key. This confirms the trajectory of the line belonging to this key in Figure 8.6. Finally, the graph also shows obfuscation techniques applied by the adversaries. When decoded, *0xbdaf('0x0'* is equal to *4983e3*, and at some point the actor behind *4983e3* updates the installation base to mine using a masked key. While the figure only displays the largest transitions for readability, there are a lot of transitions happening, especially in the long tail of the distribution. Overall, 55% of all routers are infected with more than one key, 15% of all MikroTik devices even rotate through 5 or more keys in 2018.

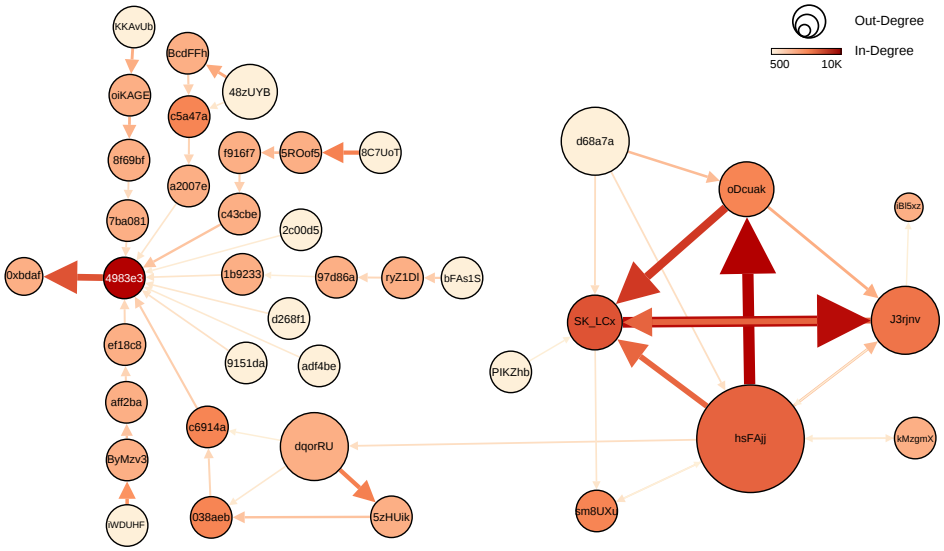

Figure 8.7: Re-infections of compromised devices with different keys with >500 overlapping IPs.

8.2.3. Infection Consolidation

After the adversary has been able to obtain the system credentials and activate a developer backdoor, root access is used to establish a foothold on the device. As described in Section 8.1.1, the firewall configuration is changed, the proxy activated, and additional files downloaded to the system. We defer a discussion on the monetization, the cryptomining, to the next section, and discuss the infrastructure used to perform the scanning, logins and loading of additional components.

Node to node reconnaissance Based on Censys and Shodan data we obtained a list of infected devices over time, and could in the NetFlows thus trace which anonymized IP addresses would connect to the WinBox service on vulnerable and infected routers. While the bulk of these connections came from a variety of anonymized IPs, 6.5% of the flows towards port 8291 were sent from infected MikroTik routers to other MikroTik routers. We observed 948 infected routers which were systematically scanning their local subnet for additional vulnerable routers on port 8291. While based on NetFlows it is not clear whether these infected routers only enumerate vulnerable hosts or also perform the compromise itself, we find this additional structural component noteworthy as interestingly this behavior was only implemented in geographic regions where MikroTik routers seemed to be rolled out structurally by ISPs. We observed this behavior specifically in Brazil.

Infrastructure In 2018, a malware infection spread throughout Brazil, probing to exploit the vulnerability elaborated on in Section 8.1.1 injecting both a miner and a script called *script3_* which would fetch new updates and commands from a *staging server* on port 2008 every 30 seconds. In the NetFlows we have identified six staging servers in the subnet of *211.164.222.**, which confirms the research of [120].

In their research they initially found the C&C server *211.164.222.159*. By combining our machine learning method and the resulting probable C&C servers, and the Anderson-Darling test as explained in Chapter 5 and in Section 6.2.1. These resulting 6 other servers were the only servers which had a significant similarity to the original staging server. We have identified that these staging servers are active from 26 July to 21 September 2018, and these servers have connected to 220 distinct infected routers during this period. The most prominent key exhibiting this was *hsFAjj* as confirmed by [120], however our data also shows that *SK_LCx* and *oDcuak* appear towards the end of this period, suggesting a link between keys.

Based on the connection patterns of the compromised routers and the maintenance activities (which we will discuss in Section 8.2.5), we can deduct the system architecture as depicted in Figure 8.8. While a handful of infected routers are performing scanning and infections within the same prefix, compromised routers remain unconnected among themselves. They only have two flows in common: the connection on port 2008 to a handful of staging or C&C servers, as well as SSH flows from a shared origin.

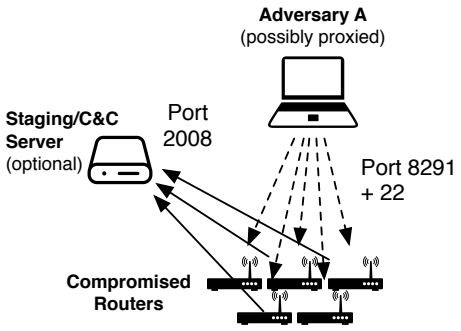

Figure 8.8: Schematic overview of the system architecture.

When a router is taken over, the new perpetrator does not seem to always aim to eradicate a previous infection after having replaced the proxy template and mining key. In fact, we find numerous examples where the routers are taken over by a different key keep beaconing to the staging servers associated with an unrelated actor, who shows no other commonalities or features with the new owner.

8.2.4. Monetization

With the vulnerability triggered and a foothold on the routers established, the adversaries moved to the exploitation of the routers for monetary gain. Over the course of the study period we observed the evolution of two monetization strategies. First, the use of the routers as a proxy service, and second, the injection of cryptomining code into users' web browsing sessions.

HTTP Proxies The first use case of the compromised MikroTik routers was the establishment of HTTP proxies. Here traffic from a web browser to a web server is tunneled through the HTTP proxy, thus masking the IP address of the client towards the server. HTTP proxies are used as a basic variant of a VPN service, although application-protocol specific and with limited authentication options if at all implemented. Starting from July 9, 2018, the first MikroTik routers were repurposed as HTTP proxies, which we identified from the emergence of large incoming traffic towards specific high TCP ports, namely 36551, 53281 and 58833. This use case remained however relatively rare, with only 3,216 of the total 1.4M infected routers being abused in this way. Interestingly, the usage as an HTTP proxy did not seem to serve a monetary gain, as within 3 days 95% of the routers for which these unusual spikes appeared were posted to free public proxy lists [103], and allowed a connection without user credentials. This usage was only relatively short-lived, as most were disabled within 40 days, at which point SOCKS proxies were spun up at TCP 4145.

SOCKS Proxies In contrast to HTTP proxies, SOCKS proxies work at the transport layer and forward traffic transparently with regard to the application layer protocol. This allows this proxy type to be used in combination with any application and thus extending the monetization potential. Shortly after the emergence of this new use case, the HTTP proxies on the MikroTik routers are replaced by SOCKS proxies, and 1,530 MikroTik routers remained in use as SOCKS proxies even until the end of the study. Further characterization of the NetFlows is not possible, as the application traffic itself would be forwarded inside the tunnel and the router would rewrite the outgoing flow to an ephemeral source port. However, we do find that the exploitation as SOCKS proxy was under the control of a few and not deployed pervasively.

This is possible to conclude, as the use of SOCKS proxies was never encountered alone, but only in combination with a cryptomining infection. As we discussed in the previous section, adversaries were routinely reinfecting devices and by changing the cryptomining *siteKeys* effectively snatching the devices away from their competitors. With the infection script reconfiguring the device including firewall and proxy settings, we can thus assess that the "ownership" with respect to an active cryptomining would also indicate who had control over the SOCKS proxy at that point in time. As we discuss in the next section, we identified a total of 140 cryptomining keys on the 1.4M MikroTik routers, but as shown in Table 8.2, only five *siteKeys* were in use on a router whenever the device was proxying traffic. Their impact is however huge: more than 95% of all SOCKS activity that originates from MikroTik routers is the result of proxies operated by these five *siteKeys*, with *hsFAjj* being one of the early adopters of MITM-based cryptomining. The small number of *siteKeys* related to SOCKS proxy activity suggests a relation between those *siteKeys*, as others do not exhibit this behavior.

Key	*hsFAjj*	*J3rjnv*	*SK_LCx*	*oDcuak*	*d68a7a*
% of all SOCKS traffic	53.6%	29.1%	7.8%	3.4%	1.2%

Table 8.2: Router "ownership" based on cryptomining key and corresponding proxy activity

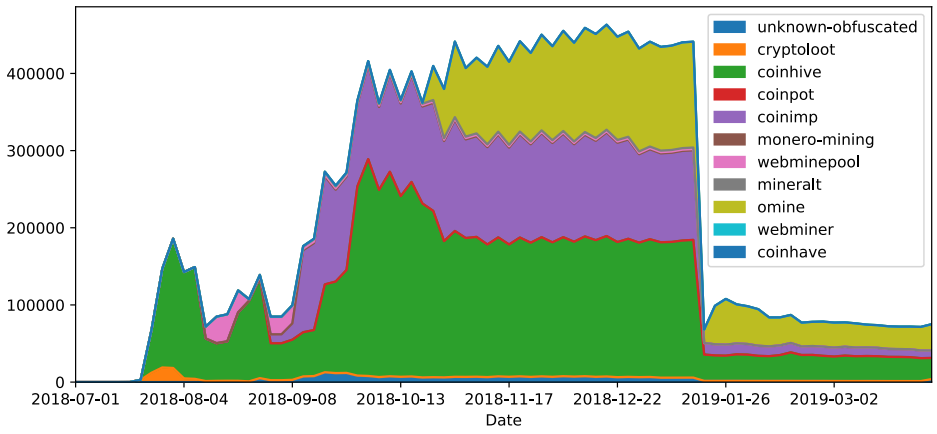

Figure 8.9: Evolution of cryptominers over time per service.

Cryptomining Proxies While the usage as HTTP proxies was not commercialized and only few actors repurposed a limited number of devices as SOCKS proxies, a large number of actors engaged in cryptojacking user connections, with a total of 140 different cryptomining *siteKeys* being installed on the routers during the study, and a maximum of 106 different *siteKeys* being active at the same time.

Figure 8.9 depicts the number of infected routers over time, categorized by the mining service provider used. As we see in the figure, the MITM-based mining started out based on Coinhive, which was at that time the obvious choice to be introduced in the MITM vector as it was the first service for cryptomining and already widely deployed in website-based mining [11, 70, 107]. Starting in middle of September, this homogeneity shattered with first the emergence of CoinImp, and later on Omine, all taking on approximately equal market shares which led to a peak of cryptojacking activity on December 19, 2018, as 460,618 routers were infected concurrently. This activity continues relatively unchanged until January 26, 2019, when suddenly mining activity disappeared from the bulk of infected routers. The distribution of miner applications between Coinhive, CoinImp and Omine remained similar and relatively unchanged.

Interestingly, related keys as found in previous analysis, do not necessarily use the same mining application, possibly to defract risks from accounts becoming frozen by cryptomining services. Despite this risk sharing across accounts, several actors also spread out their activities across multiple mining keys, as can be inferred when the same maintenance hosts connects to routers with multiple keys (as we will show in Section 8.2.5). These movements – and also the strong emergence of CoinImp and Omine – can probably be explained based on fees: while Omine charges a 2% fee and CoinImp is entirely free, Coinhive takes a 30% cut.

Figure 8.10 shows the evolution of *siteKeys* installed on MikroTik routers between July 2018 and April 2019, ordered by the time they were first encountered on a router. The size of circle indicates on how many routers this *siteKey* was installed on a given day. We can see that MITM-based cryptomining was pioneered by three *siteKeys*: first, *d68a7a* emerged first but beside a small peak remained only a minor player. Second, *hsFAjj* who followed one week after, temporarily controlled 70% of all infected routers, and introduced new strategies for controlling and otherwise monetizing the routers, remaining a steady force until the general decline. And third, *oDcuak*, like the first mover *d68a7a* experiencing a small surge followed by steady but comparatively low-volume activity.

Approximately one week after these first movers, a large number of new *siteKeys* started to appear, frequently co-emerging in groups that stay relatively similar in size. Four sequential blocks of 10 *siteKeys* can be clearly observed, two of them using Coinhive, the other two are using Omine as their mining service. While all other *siteKeys* never reach the same size as *hsFajj*'s initial deployment (167,182 infections), each of them is able to hold control over up to 64,539 routers at a time. While we find a total of 1.4M routers to be vul-

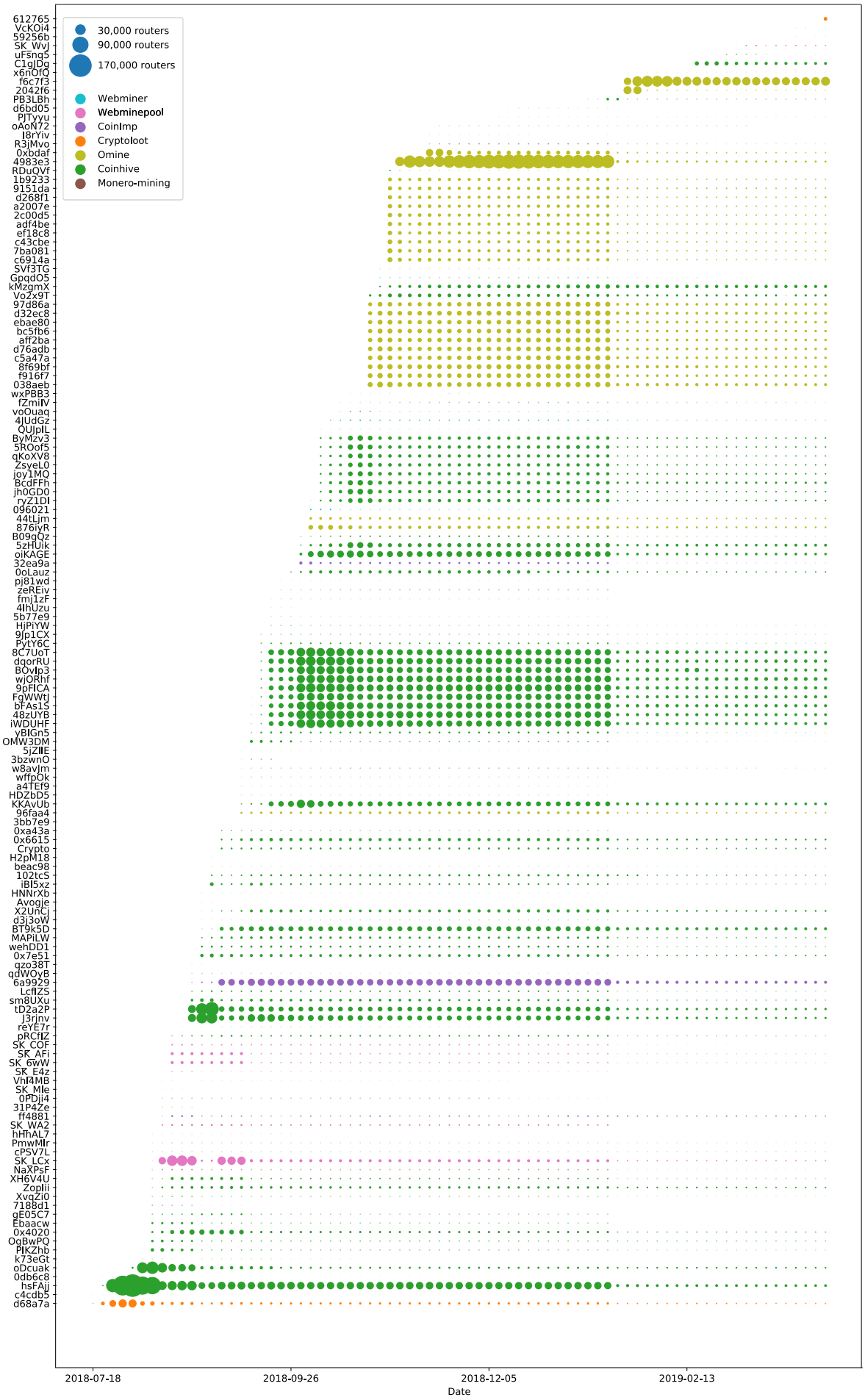

Figure 8.10: Evolution of detected siteKeys over time.

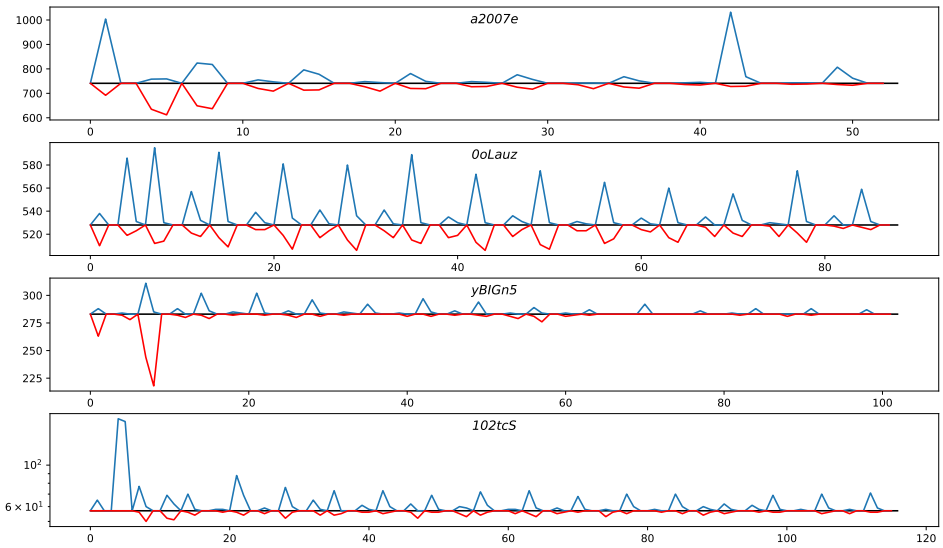

Figure 8.11: Additions/deletions over time by selected keys.

nerable and at some point infected, the perpetrators are never rolling their cryptojacking infections out to all potential victims simultaneously. Instead, we see a constant flux, with new routers being infected so that the mining deployments stay consistent in size. This is necessary, because once infected, most of the routers are patched quickly, as shown in Figure 8.12. This figure shows the cumulative density function (CDF) of the number of days a router is infected on a logarithmic scale. We see that 50% of the devices are patched within 18 days after compromise, whereas only 30% of the devices remain active for more than 50 days, urging actors to constantly replace disappearing routers to maintain their installation base.

This is best observed when we look at the keys in Figure 8.10 that remain relatively constant over time. Four of these keys are depicted in Figure 8.11 with respect to daily additions and removals from the pool, indicated in blue and red respectively, starting from the day the key first became active. This behavior, as well as the sets of keys that appear together, might indicate a strategy to offset risk. If a particular key gets blocked by a miner service, others will still generate profits. The same might hold for the deployment size in general, where an all-out operation from becoming too greedy might lead to increased press coverage and faster cleanup of the vulnerability than maintaining a smaller infection size and thus lower profile. This diversification however stops from December 2018 onwards, where we see that most actors no longer replenish routers lost. This might be explained by Monero's significant drop in value, down by 60% from early November until a month later.

Indeed, we observe a steady decline of new devices that are added to the pool from November 2018 onwards, as shown in the bottom of Figure 8.13, which leads to a flattening out of the overall installation base. As we have already seen in Figure 8.9, the ecosystem of router-based cryptomining drastically changes in late January. Most apparent is the major drop in participating devices, approximately 87% of all infected routers disappear, leading to a drop in the installation base of all keys. While such a large and universal movement would indicate some external trigger, we could not find any evidence for a coordinated cleanup action, for example by an ISP or a grey hat hacker (aside from one who has taken credit for patching 100,000 routers

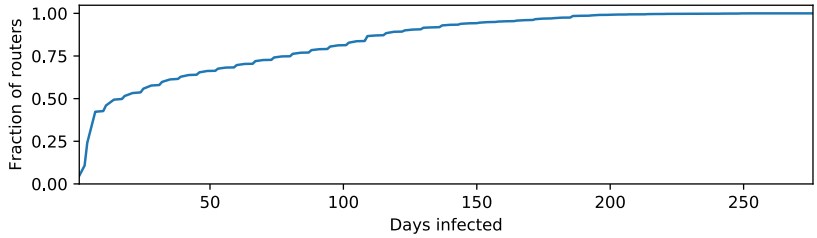

Figure 8.12: CDF of the infection duration.

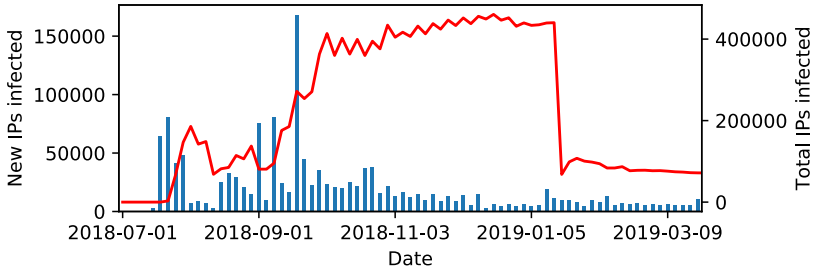

Figure 8.13: New and total of IP addresses infected per day.

in November 2018 [22]). After this cut, we also see a rotation of remaining actors towards new keys, where the new key *f6c73* partially takes over the efforts of *4983e3*, however only a few continue to re-establish their activities and forego previous practices, whereas *f6c73* is responsible for most new infections.

Geographical Focus Based on the heatmap in Figure 8.3 and the large deployment of MikroTik devices in certain autonomous systems as shown in Table 8.1, we have seen that a number of countries seemed prime candidates when looking for MikroTik devices, which would logically mean that advanced adversaries should focus their activities there. As RouterOS is used in both consumer devices and carrier-grade routers, we would naturally expect some devices to be more lucrative than others, immediately posing the question whether reinfection of devices – in other words "stealing" routers – would primarily occur in popular areas and target those devices where a lot of money could be made.

Figure 8.14 shows the number of *siteKeys* as a function of the amount of NetFlows on port 80 this router processed during its infection. Counterintuitively, there is no trend that high-value targets are more fought over than low-value ones. Especially routers with much traffic tend to stick with just a low number of *siteKeys*. This is surprising, as a cryptomining operation on a large router would clearly affect more people, lead to more complaints and thus logically faster patching. The lack of a fight for high-grossing routers can however partially be explained based on the location of the routers, indicated by the color of the data point. While routers in Indonesia and Brazil – the hotspots of the infection – cover the entire spectrum and are changing keys considerably, the most stable infections – and the highest grossing ones for that matter as we will see in Section 8.3 – are in countries that do not appear anywhere near the top in MikroTik installation counts, for instance, 6 out of the 10 most grossing routers are located in Iraq. This means that actors targeting niche markets accomplished much more valuable deployments, as these routers mined longer for them.

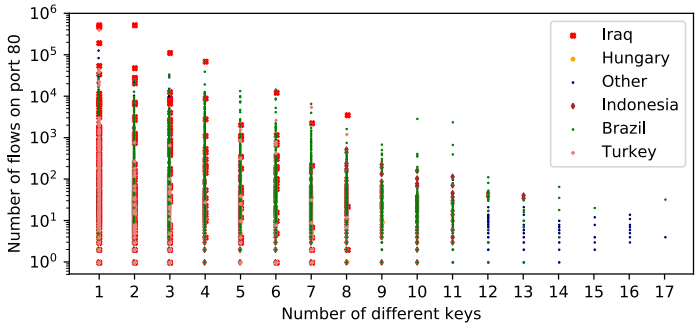

Figure 8.14: Relation between the number of flows to port 80 and the number of keys per router.

8.2.5. Maintenance

When we look at the life cycle of a malware infection as for example a botnet, after the initial exploitation the compromised device remains in contact with the perpetrator or a C&C server to download additional components or receive a new configuration. While we would expect a similar behavior for malware targeting routers, we saw little evidence for post-compromise maintenance operations.

Configuration Access and Periodic Updates As a *siteKey* is directly linked to a particular actor, we analyzed whether any connections were made between an end point and the group of routers that were at a certain moment compromised by the same key. Using the association rules methodology described by Agrawal & Srikant [1], we have searched for maintenance patterns where specific keys have a large probability to coincide with a specific anonymized IP address or port number, as maintenance would likely be performed from a set of C&C servers or the attacker's PC. As connections to port 22 (SSH) and 23 (Telnet) in NetFlows are also caused by prevalent port scanning, we differentiate between port scanning and active SSH sessions in NetFlows based on the packet size and only include connections with a confidence c and support s of at least 40% among our router/key set, in other words we require that at least 40% of infected routers had been contacted by a common origin, while being significantly present in the data.

We have observed maintenance connections on port 22 (SSH), which was only pursued by the actor(s) responsible for routers infected with one of three keys *oDcuak*, *SK_LCx* and *hsFAjj*, while other strains and actors do not seem to deploy such coordinated access. Surprisingly, routers with any of these keys were in contact with the same remote host at a given moment in time, strongly suggesting that the keys were actually related to the same persona. In addition, when a new IP address appeared to make contact with the compromised devices, routers with all three keys were always contacted by the same source. For example, routers with these keys made SSH connections to *236.197.108.8* between 3 and 20 August 2018, while between 11 and 14 August 2018 these routers were contacted by *236.247.130.64*. Each of these IPs seemed to employ automation, contacting routers either at midnight or during the timeframe 16–19h. Besides these IPs, almost no evidence of scripted interactions between a controlling source and the infected routers has been found, which would be evident from a large number of connections being made at the same time, or sequentially within a short time period. In total we observed 5 IPs making such common connections over time, matching our earlier observation about the link between the aforementioned three keys as discussed in Section 8.2.4.

8.3. Quantification of revenue

The results from the previous sections already suggested that MITM-based cryptomining operates at an entirely different scale than previously reported attack vectors. This is due to three reasons:

1. The volume of compromised entities is much higher. Instead of a few thousand websites [17, 61, 107, 109], here a total of 1.4M infected routers is involved. A MITM through routers would also amplify earnings, as *all* users to mine on *any* site.

2. MikroTik uses the vulnerable RouterOS on consumer grade and carrier-grade devices. A carrier-grade router will likely serve significant user populations, and thus within a short time amass large volumes of revenue.

3. While 30% of all website-based cryptomining is removed in 15 days [61], we find that 30% of the MITM-based mining remains active for more than 50 days. Although also routers are often patched quickly, the pool of vulnerable devices is so large that it barely affects the installation base.

In this section we will extend the previous results towards a quantification of adversarial revenue per key using this new attack vector. We will conduct this quantification according to the method established by Konoth et al. [70] for a direct comparison with website-based mining, but make some adjustments for the shifted attack vector. For their analysis, they built a three step estimation model:

- *Estimation of monthly visitors and visit duration:* They estimate visitor count and the average time spent for 1,705 cryptomining sites using data from SimilarWeb [118].

- *Average computing power of visitors in hash rate per second:* Cryptocurrency is mined during the visit on the website. They measure the hash rate of two desktop CPUs, and 16 mobile devices, yielding an average rate per second of 40.5 and 14.56, respectively. Afterwards, information of MineCryptoNight [84] is used to convert that to XMR/s.

- *Current value of cryptocurrency:* The overall mining power of the visitors is then mapped to and monetized in Monero cryptocurrency, which was valued at $\$253/XMR$ at that time, yielding an overall revenue of up to $30,000 per month.

Parameter	Methodology in [70]	This study
Number of visitors	SimilarWeb estimations	# of NetFlows on port 80
Average hashing rate	SimilarWeb estimations	desktop / mobile: 25 H/s
Monero market value	$ 253 as of May '18	$253 for equal comparison
Time on website	SimilarWeb estimations	Average, 1st / 3rd quartile

Table 8.3: Revenue estimation parameters

In the following analysis, we are following the same equation of

$$\text{traffic} \times \text{avg. time [s]} \times \text{mining rate [XMR/s]} \times \text{value [\$/XMR]} = \text{profit in dollars}$$

but adjust them for the specific attack vector observed. First, our NetFlow traces allow for an extrapolation of the actual number of HTTP connections, and we attribute the count of flows to the revenues of a key installed on the proxy page at that time. While the embedded miners also work for iframed-HTTPS connections, we did not find evidence that this attack was pursued in the wild. This will thus be a lower bound on the amount of traffic.

Second, Konoth et al. estimated average visiting times for each of their 1,705 detected websites using SimilarWeb data, but the MITM attack works across all pages of the Internet. As the actual end point of the outgoing connection has been anonymized for privacy, we can approximate the average visiting time as we query the average visiting duration of websites listed in the Alexa Top 10k – the 10,000 most popular websites – on SimilarWeb. The average visiting time for these websites is 293 seconds. We will for our calculation work with three values for visit duration to provide a range of the revenues made by the attackers. We will use the average visiting time, as well as the first and third quartile of visit durations. Yet, already the highly conservative estimation based on the first quartile highlights the magnitude of this new attack vector. Table 8.3 compares the parameters used in [70] to our study.

Third, Konoth et al. also used SimilarWeb data to estimate the hashing rate for both mobile and desktop visitors, being 14.56 and 40.5 respectively. We estimated the hashing rate based on the desktop/mobile device ratio found across the Internet as a whole, which is listed in [32] as 0.58, resulting in a weighted hash rate of 25 H/s. Since we want to compare the profitability of MITM-based to website-based cryptojacking, we could either compare the amount of Monero mined, or translate the Monero amount into more intuitive currency such as USD. Currency exchange rates are however volatile and in between Konoth's May 2018 study and our study, the average Monero price had dropped during August and December 2018 to $92.2/XMR. To compare both attack vectors side by side, we thus use the same exchange rate as in [70], which still makes a fair comparison, as the decline would have equally scaled down the revenues attackers could have generated using website-based mining during our observation period. Even if we scale the revenue down with the declined value of Monero, the MITM-based revenues would still be a factor of 10 higher than the website-based earnings made half a year earlier.

SiteKey	Total # routers	First quartile 2'27" stay	Median stay 4'53" stay	Third quartile 6'19" stay
48zUYB	52,181	$111,447.18	$222,136.22	$287,336.61
6a9929	30,135	$97,626.82	$194,589.52	$251,704.53
8C7UoT	47,981	$90,532.54	$180,449.21	$233,413.82
BOvlp3	49,640	$82,573.82	$164,585.92	$212,894.42
4983e3	117,502	$70,017.28	$139,558.26	$180,520.75
FgWWtJ	39,384	$50,719.01	$101,092.99	$130,765.33
J3rjnv	45,934	$40,551.39	$80,826.92	$104,550.86
hsFAjj	223,844	$35,396.11	$70,551.44	$91,259.37
BT9k5D	8,459	$31,494.11	$62,773.97	$81,199.10
wjORhf	42,342	$27,671.96	$55,155.67	$71,344.70
Total top 10		**$638,030.22**	**$1,271,720.11**	**$1,644,989.49**

Table 8.4: Estimated monthly revenue of top 10 grossing actors based on the average visit duration on the Alexa Top 10K, as well as the first and third quartile according to SimilarWeb.

8.4. Charting the ecosystem of actors

While looking at the life cycle of router infections, we observe different levels of sophistication in every stage. In the identification stage, we discover a clear distinction between *siteKeys* installed as a result of scanning and infection based on public sources, such as Shodan. In the exploitation of the routers afterwards we observe a constantly changing landscape in which actors are regularly infecting new devices and stealing from each other. After infection, only a limited number of actors demonstrate a high level of sophistication by setting up an infrastructure. To monetize the hijacked routers, actors initially set up HTTP proxies, but subsequently increased their revenues by installing SOCKS proxies with cryptojacking scripts. The used cryptomining scripts diverge to multiple services, and we have noticed a continuous flow of router infections and removals. Clear geographical differences in mining characteristics are identified, where Brazil and Indonesia are the most infected, while Iraq seems to have the most lucrative infrastructure to infect. Observed maintenance patterns show that specific anonymized IPs can be linked by behavior to *siteKeys*.

8.4.1. Relating actors and keys

Based on the results of the different independent components analyzed in the previous sections, we are able to link certain *siteKey*s to each other and/or to IPs. To start with, three *siteKeys hsFAjj, SK_LCx, oDcuak* show similar behavior as the same infrastructural patterns can be found on routers infected with these *siteKeys*, as well as regular contacts with the same set of attacker IPs for maintenance over SSH. Figure 8.7 confirms this hypothesis by showing numerous routers transitioning between those *siteKeys*. Interestingly, the analysis of SOCKS traffic also links *J3rjnv* to this set. Additionally, this figure depicts the sophistication level of the actor behind *siteKey 4983e3*, as this actor hijacks vulnerable routers infected with numerous other *siteKeys*, but subsequently changes his own *siteKey* to a masked variant. Revisiting Figure 8.10, which shows 4 clear sequential blocks of 10 *siteKeys* having similar installation sizes and evolutional patterns. This in combination with the aforementioned figure, which shows 5 clear *siteKey* transition chains, an even larger number of *siteKeys* can be linked to one single adversary. By following each *siteKey* within these transition chains in Figure 8.7, we noticed that these transitions resemble transitions between the sequential blocks in Figure 8.10. All the *siteKeys* in the transition chains are located inside these blocks in the same sequence. For each of the *siteKeys* inside these four blocks, the first two blocks (highlighted in green in Figure 8.10) use a Coinhive miner, with the uncommon option `CoinHive.FORCE_EXCLUSIVE_TAB` enabled, and the latter two (highlighted in yellow) use Omine as a mining service. Additionally, all 40 mining scripts within these blocks were set to the same throttle value of 0.1. As a result, this common behavior across multiple *siteKeys* strongly suggests that we can thus link these 40 *siteKeys* to one single actor.

In this analysis, we have reported on a new attack vector for cryptojacking, which does not infect websites but compromises the Internet infrastructure itself. This vector greatly overshadows any cryptojacking campaigns known to date by orders of magnitude, and we find groups of actors compromising a total of 1.4M vulnerable routers, approximately 70% of all deployed MikroTik routers, with various degrees of sophistication. As the injection of miners into network traffic affects any user visiting any website, we find this attack vector to be highly profitable, based on conservative estimates exceeding $1M per month. Curiously, the highest grossing actors are not the ones innovating with new techniques or creating the largest deployment, but those finding the most productive niche where they can operate relatively undisturbed.

9

Conclusion and Discussion

Nowadays cybercrime has nestled itself into the daily usage of modern technologies. Anybody using a device to connect to the Internet at some point has to think about protection against digital criminals. Antivirus software provides services to detect the activity these cybercriminals have on our devices, primarily based on signatures created by malware. This host-based detection has led to a rat race between adversaries and researchers, as new malware on average remains active for no longer than two hours. The underlying concept however remains similar; infected devices connect to a central commanding server, which operate under certain tactics, techniques and procedures. Not much is known about these tactics, techniques and procedures and their evolution, which is why the main research question of this work has been defined as:

What are the attack patterns and tactics, techniques and procedures of malicious Command & Control infrastructures, how do these infrastructures evolve and how can we track them on a global scale?

This chapter will summarize and discuss the results of this thesis. Initially we will present our main results and how this contributes to the academic world. Subsequently we will present the results of the three steps of our research approach and evidently in the conclusion answer our main research question. Our work also encountered a few limitations and drawbacks which will be discussed. Finally our work will be concluded with ideas for future work on this research.

9.1. Main contributions

To be able to answer the research question, we will first discuss our findings and contributions to Cyber Threat Intelligence. Initially we have performed an analysis on open source threat intelligence, evaluating whether the coverage of these feeds is sufficient to answer our research question. Following this we have analyzed the various tactics, techniques and procedures which adversaries employ regarding their C&C infrastructures.

Analysis of threat intelligence feeds

We have analyzed 24 different open source threat intelligence feeds based on four evaluation criteria; timeliness, sensitivity, originality and impact. By doing this we have investigated how widely these feeds are adopted in practice by network owners and operators. To analyze how and which of these feeds are utilized by networks around the world, we traced activity from each IP prefix before and after an indicator was included in an intelligence feed by using Tier 1 ISP NetFlow traffic and zone transfers. When we saw requests emanating from a network before a listing but after its inclusion no further flows were ever recorded, we concluded that the indicator information was applied within a network based on, for example firewall rules or sinkholing. Repeating this process for all indicators provided an assessment which networks subscribed to which intelligence feeds, and thus what the approximate geographical market shares of threat intelligence providers were.

These 24 evaluated open source feeds chart the entire ecosystem of malicious activity; from brute forcing activity, to ransomware and from spyware, to botnets. As each type of malicious activity was covered by

multiple feeds, we would have expected *some* overlap in reported indicators, which however did not turn out to be the case. The fact that there is almost no overlap between lists of similar scope could be the result of two reasons: first, all individual lists targeting for example botnets or ransomware rely on orthogonal detection methods and are therefore providing complementary information. Secondly, the lists monitor malicious activity in comparable ways, but the overall volume of malicious activity is so large that effectively each lists only obtains a tiny sample, and such low rate sampling from a large universe would statistically lead to a very low chance for duplicates. Conceptually, we can see that we are probably dealing with the latter reason, as methods and datasets to for example detect ransomware C&C servers are limited. This unfortunately drives us to the conclusion that open source cyber threat intelligence feeds cover much less malicious activity than we would expect.

Adversarial tactics, techniques, and procedures

After having evaluated the publicly available threat intelligence sources, we used the conclusions drawn from this study to analyze the tactics, techniques, and procedures adversaries employ to utilize their malicious C&C infrastructures. To provide structure in the analysis of these infrastructures, we have looked at three different Internet network fundaments which are encountered daily; computer clients, IoT devices and routers. To gain more insights into the tactics, techniques and procedures of malicious C&C infrastructures, indicators of compromise were needed. As the coverage of publicly available threat intelligence is fairly low, we have used tier 1 ISP NetFlow data to track connection patterns of these indicators of compromise to learn about their structure and use them to detect more suspicious activity on a global networking scale. By adopting a known detection methodology proposed by Bilge et al. [12], we have been able to learn malicious C&C patterns and use this technique to validate more malicious activity than previously known.

We have shown that by combining the proposed C&C server detection technique and various clustering algorithms new malicious servers, performing similar malicious patterns, can be detected. This has allowed us to perform more in-depth analyses and create a more coherent picture of attack vectors. We have been able to track the global activity and connection patterns of C&C infrastructures, uncovering multiple different tactics they employ to hide from the public. Different types of malicious structures have been encountered by for example constantly changing the C&C server to which infected devices connect. Also other load balancing techniques have been encountered, showing high levels of adversarial sophistication. We have also shown that C&C infrastructures employ division techniques in their infected devices, creating clusters of labor division. Additionally we show that we can complement existing cyber threat intelligence by predicting malicious servers and correlate them with known adversarial servers. By combining NetFlow data with information retrieved from the TU Delft telescope we have also analyzed malicious behavior regarding IoT devices. Adversarial scanning towards the network telescope revealed information about devices infected with Mirai, probing the network for vulnerable devices. By extrapolating these indicators of compromise, we have been able to estimate the leverage and the evolution of these infections. We have found that `1` in every `2,345` scanned devices is successfully infected by Mirai variants. Additionally we have been able to detect servers, highly suspicious of performing command and control towards infected IoT devices, and have again seen labor division between the infected devices related to these C&C servers. Finally we have been the first to investigate a new type of attack that exploits infrastructural Internet routers by performing cryptomining. We have showed how, over a period of 10 months after the initial discovery of the vulnerability, groups of criminals launch massive campaigns to control 1.4 million MikroTik routers to perform malicious activities. We have analyzed the different adversarial tactics and have reported on the supporting infrastructure used within these cryptojacking infrastructures. By doing this we have been able to show different levels of sophistication between groups in how they locate their victims, compromise routers, and run their infrastructure.

9.2. Research approach

To accomplish the main goal of our work, we have set out a research approach which will help in answering our main research question. Here we will list the steps in our approach accordingly with our findings.

(1) *Determine the coverage of open source threat intelligence feeds and determine if these resources provide enough information to analyze malicious infrastructures.*

Cyber threat intelligence feeds are designed to alert and empower network owners to block malicious activity. Incorporating these feeds should lead to a reduction in network traffic towards the hosts flagged as malicious. After performing an analysis on 24 distinct open source threat intelligence feeds, we can conclude that the

current state of these feeds is not where it should be to be effective in defense. The originality of these feeds is higher than we would have expected, creating a huge amount of data to process before being able to apply it in, for example, firewall rules. Besides the coverage these combined feeds provide, we have also seen that the overall measured quality of each feed differs greatly. Using these feeds to track malicious C&C infrastructures has proven to not be sufficient.

(2) *Define a methodology which enables us to utilize NetFlow data, combined with additional datasets, to identify malicious infrastructures.*

Resulting from our first step, we understand that open source threat intelligence will not provide us the coverage we had hoped for. For this reason we have created a methodology which first breaks down Internet networking infrastructures into three different categories, making it better to cope with the problem. Secondly we have adopted an existing machine learning method to detect possible new malicious C&C servers by using NetFlow data. This method however brings a level of uncertainty with its prediction. For this reason we introduce methods to elevate the level of confidence when looking for similar malicious indicators.

(3) *Discover attack patterns and tactics, techniques and procedures of malicious Command & Control infrastructures and their evolutions within infections on clients, IoT devices and routers.*

Cybercriminals use various levels of sophistication to accomplish their malicious intents. We have distinguished a wide variance of refinement based on their infection strategies, either as a result of scanning or based on public datasets. We have found evidence of a diverse landscape of Command & Control infrastructures, where adversaries employ all kinds of different strategies to be evasive. Hopping structures and other load balancing techniques have been observed. Each of the analyzed Internet infrastructural components show different techniques to consolidate infections on devices. Where simple IoT devices have been used to spread at a rapid rate of 1 in every 2,345 scanned devices, routers have been used to employ more advanced malicious structures by creating proxies on infected devices to earn money based on cryptojacking.

9.3. Conclusion

We have set out to discover what attack patterns and tactics, techniques and procedures of malicious Command & Control infrastructures, how these evolve and how they can be tracked on a global scale. Antivirus software uses malware signatures to perform detection on adversarial behavior at the 'front door' of a device. Most research has worked on new detection methods which try to realize this. However, whilst these signatures change and disappear at rapid pace, the connection patterns from adversary to this front door stay the same. By taking a step back, we point to cyber threat intelligence as the 'antivirus' which will be able to tackle adversaries at this level of networking. This open source threat intelligence is not yet at the level to be as effective towards these malicious infrastructures as we would have hoped.

To discover the tactics, techniques and procedures these cybercriminals pursue, we have combined the knowledge from these feeds with global detection and clustering mechanisms to paint a more complete picture of the cyber threat landscape. All of this has been possible by combining a rich dataset of tier 1 ISP NetFlow data combined with various other threat intelligence sources. In this research we have just scratched the surface of possibilities NetFlow data presents to the research on threat intelligence. Pandora's box is not empty yet.

9.4. Limitations

During the analysis of our results, three main limitations have been observed which we will discuss in this section. Working with NetFlow data provides a lot of new research angles, not yet seen in the academic world of cyber threat intelligence. This also comes with a few challenges which we have encountered during our research.

Sampling rate

The work presented in this thesis has been made possible by using NetFlow data of a tier 1 ISP operator extensively and combining this with other threat intelligence sources. This rich data source provides all traffic flowing through the Internet backbone of this ISP provider. One challenge we observed whilst working with this data, is the sampling rate at which the NetFlow data has been limited. For each 1 flow seen in the Net-

Flow files, 8192 flows have passed through the network of the ISP provider. This proved especially challenging when estimating the amount of proxy traffic observed in Chapter 8. This also proved challenging when being able to say whether traffic is malicious, as we do not see the whole sequence. We have only made this assumption when this could be validated by multiple sources, for example in Section 7.2.3 where all traffic to the detected C&C servers is sent only by infected devices to a port which should initially not be used for periodic communication.

Location based bias
As we have mentioned, NetFlow data at this scale has never been used before to study malicious infrastructures. Instead of one-way traffic, which can predominantly be seen in host-based analysis, NetFlow gives the opportunity to see more interconnections and more levels of networking depth from source to destination. However, since these flows only contain traffic sent through the network backbone of one ISP provider, it is limited to traffic passing through the routers it controls. Due to the location of these routers (Point-of-Presence, or PoP), Border Gateway Protocol (BGP) policies, and a specific Internet Exchange Point (IXP) footprint could lead to a location based bias of certain autonomous systems being controlled by this ISP operator.

Machine learning
The results provided by the analyses from Chapter 4, pointed out that the current state of open source threat intelligence does not provide enough coverage. We tackled this problem by adopting a machine learning model presented in previous work, and making it suitable for the amount of data we have dealt with in this thesis. Machine learning however relies on statistical models to perform a specific task relying on specific patterns. The sampling rate of the NetFlow data introduces an additional problem, as these patterns are not always the same. We have overcome this problem by introducing a set of clustering techniques, validating assumptions before drawing our conclusions.

9.5. Future work

The results derived from our analyses presented in this thesis have shown the incredible possibilities NetFlow data presents for the academic world. NetFlow data can reveal various, malicious and benign, infrastructures and connection patterns between source and destination. We have shown that by combining this rich data with various threat intelligence sources, a whole new analytical perspective regarding adversarial behavior can be explored. One limitation of our dataset is that the NetFlows are sampled. Unsampled data would eliminate this problem and would show the exact patterns of malicious infrastructures. Analysis on unsampled data would therefore provide more opportunities to observe adversarial behavior. However, our dataset is already incredibly large, containing 4Tb of binary files. An unsampled version of this data would be 8192 times as large: 33,914Tb or 33Pb. This introduces a whole new set of challenges by itself.

Our results are very promising; we have been able to track and analyze malicious infrastructures regarding three different Internet components, and their tactics, techniques and procedures. However, we still feel like we have only just scratched the surface regarding the added value ISP grade NetFlow data can provide for cyber threat intelligence and adversarial detection. Antivirus companies have been performing host-based malware detection for a very long time, however to our knowledge detection on the backbone of the Internet is not something which has been done to date. Cutting off adversaries before reaching the front door of the Internet user, would be the ideal situation. ISP providers could play a big role in this, by analyzing NetFlow data flowing through their routers to perform detection of malicious behavior based on previous misuse and cyber threat intelligence. To overcome a location based bias any ISP would have, these providers could even join hands and work on international collaboration forming the 'anti-virus' of our Internet based society.

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994. URL http://www.vldb.org/conf/1994/P487.PDF.

[2] Pedram Amini, Muhammad Amin Araghizadeh, and Reza Azmi. A survey on botnet: classification, detection and defense. In *2015 International Electronics Symposium (IES)*, pages 233–238.

[3] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 491–506, 2012. URL https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/antonakakis.

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, pages 1093–1110, 2017. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[5] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu, and Manish Karir. A survey of botnet technology and defenses. In *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, pages 299–304. IEEE, 2009.

[6] C.J. Barker. Mirai (ddos) source code review, October 2016. https://medium.com/@cjbarker/mirai-ddos-source-code-review-57269c4a68f/ (June 2019).

[7] Pijush Barthakur, Manoj Dahal, and Mrinal Kanti Ghose. A framework for P2P botnet detection using SVM. In *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2012, Sanya, China, October 10-12, 2012*, pages 195–200, 2012. doi: 10.1109/CyberC.2012.40. URL https://doi.org/10.1109/CyberC.2012.40.

[8] Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen. Hypertext transfer protocol - HTTP/1.0. Technical report, 1996. URL https://doi.org/10.17487/RFC1945.

[9] Elisa Bertino and Nayeem Islam. Botnets and internet of things security. *IEEE Computer*, 50(2):76–79, 2017. doi: 10.1109/MC.2017.62. URL https://doi.org/10.1109/MC.2017.62.

[10] David J. Bianco. The pyramid of pain. January 2014. URL https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[11] Hugo L.J. Bijmans, Tim M. Booij, and Christian Doerr. Inadvertently making cyber criminals rich: A comprehensive study of cryptojacking campaigns at internet scale. In *Usenix Security Symposium*, 2019.

[12] Leyla Bilge, Davide Balzarotti, William K. Robertson, Engin Kirda, and Christopher Kruegel. Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*, pages 129–138, 2012. doi: 10.1145/2420950.2420969. URL https://doi.org/10.1145/2420950.2420969.

[13] Zheng Bu and Rob Rachwald. Ghost-hunting with anti-virus. May 2014. URL https://www.fireeye.com/blog/executive-perspective/2014/05/ghost-hunting-with-anti-virus.html.

[14] P. Burghouwt. Detection of Botnet Command and Control Traffic in Enterprise Networks, June 2015. https://doi.org/10.4233/uuid:1640ad86-fc07-4e54-9f05-1629296738c7/ (June 2019).

[15] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, pages 160–172, 2013. URL https://doi.org/10.1007/978-3-642-37456-2_14.

[16] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. Uncovering large groups of active malicious accounts in online social networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 477–488, 2014. doi: 10.1145/2660267.2660269. URL https://doi.org/10.1145/2660267.2660269.

[17] Domhnall Carlin, Philip O'Kane, Sakir Sezer, and Jonah Burgess. Detecting cryptomining using dynamic analysis. In *16th Annual Conference on Privacy, Security and Trust, PST 2018, Belfast, Northern Ireland, Uk, August 28-30, 2018*, pages 1–6, 2018. doi: 10.1109/PST.2018.8514167. URL https://doi.org/10.1109/PST.2018.8514167.

[18] Cisco Catalyst. 6500 architecture. *Cisco white paper*, 2013. http://www.cisco.com/c/en/us/products/switches/catalyst-6500-series-switches/white-paper-listing.html/.

[19] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. 2007. URL https://www.usenix.org/conference/hotbots-07/case-study-rustock-rootkit-and-spam-bot.

[20] Baek-Young Choi and Supratik Bhattacharyya. On the accuracy and overhead of cisco sampled netflow. In *Proceedings of ACM SIGMETRICS Workshop on Large Scale Network Inference (LSNI)*, pages 1–6, 2005.

[21] Nilesh Christopher. Hackers mined a fortune from Indian websites. *The India Times*, Sep 2018. https://economictimes.indiatimes.com/small-biz/startups/newsbuzz/hackers-mined-a-fortune-from-indian-websites/articleshow/65836088.cms (December 2018).

[22] Catalin Cimpanu. A mysterious grey-hat is patching people's outdated MikroTik routers, Oct 2018. https://www.zdnet.com/article/a-mysterious-grey-hat-is-patching-peoples-outdated-mikrotik-routers/ (February 2019).

[23] Thomas Claburn. Crypto-jackers enlist Google Tag Manager to smuggle alt-coin miners. *The Register*, Jan 2018. https://www.theregister.co.uk/2017/11/22/cryptojackers_google_tag_manager_coin_hive/ (December 2018).

[24] Kassidy P. Clark, Martijn Warnier, and Frances M. T. Brazier. Botclouds - the future of cloud-based botnets? In *CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science, Noordwijkerhout, Netherlands, 7-9 May, 2011*, pages 597–603, 2011.

[25] Evan Cooke and Farnam Jahanian. The zombie roundup: Understanding, detecting, and disrupting botnets. 2005. URL https://www.usenix.org/conference/sruti-05/zombie-roundup-understanding-detecting-and-disrupting-botnets.

[26] Gibson Research Corporation. Port 0 - port authority database, 2016. https://www.grc.com/port_0.htm/ (June 2019).

[27] David Dagon, Guofei Gu, Cliff Zou, Julian Grizzard, Sanjeev Dwivedi, Wenke Lee, and Richard Lipton. A taxonomy of botnets. *Unpublished paper, c*, 2005.

[28] Neil Daswani and Michael Stoppelman. The anatomy of clickbot.a. In *First Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, MA, USA, April 10, 2007*, 2007. URL https://www.usenix.org/conference/hotbots-07/anatomy-clickbota.

[29] Nicola de Cao. A modular framework for botnet detectors prototyping and evaluation, January 2018. https://github.com/nicola-decao/Botnet-Finder-Framework/ (June 2019).

[30] Dimentions. Visualizing publications throughout the years, June 2019. https://app.dimensions.ai/discover/publication/ (June 2019).

[31] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 542–553, 2015. doi: 10.1145/2810103.2813703. URL https://doi.org/10.1145/2810103.2813703.

[32] Eric Enge. Mobile vs Desktop Traffic in 2019. *Stone Temple*, Apr 2019. https://www.stonetemple.com/mobile-vs-desktop-usage-study/ (May 2019).

[33] Meisam Eslahi, Rosli Salleh, and Nor Badrul Anuar. Bots and botnets: An overview of characteristics, detection and challenges. In *2012 IEEE International Conference on Control System, Computing and Engineering*, pages 349–354.

[34] F-Secure. Prettypark - technical details and summary, June 1999. https://www.f-secure.com/v-descs/prettyp.shtml/ (June 2019).

[35] F-Secure. Backdoor:w32/subseven - technical details and summary, May 1999. https://www.f-secure.com/v-descs/subseven.shtml/ (June 2019).

[36] F-Secure. Backdoor:w32/agobot - technical details and summary, June 2004. https://www.f-secure.com/v-descs/agobot.shtml/ (June 2019).

[37] Claude Fachkha, Elias Bou-Harb, and Mourad Debbabi. On the inference and prediction of ddos campaigns. *Wireless Communications and Mobile Computing*, 15(6):1066–1078, 2015. doi: 10.1002/wcm.2510. URL https://doi.org/10.1002/wcm.2510.

[38] Jinliang Fan, Jun (Jim) Xu, Mostafa H. Ammar, and Sue B. Moon. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. volume 46, pages 253–272, 2004. doi: 10.1016/j.comnet.2004.03.033. URL https://doi.org/10.1016/j.comnet.2004.03.033.

[39] Li Fengpei. New threat report: A new iot botnet is spreading over http 81 on a large scale, April 2017. https://blog.netlab.360.com/a-new-threat-an-iot-botnet-scanning-internet-on-port-81-en// (July 2019).

[40] Rik Ferguson. The Botnet Chronicles – A Journey to Infamy, November 2010. https://www.trendmicro.co.uk/media/wp/botnet-chronicles-whitepaper-en.pdf/ (May 2019).

[41] Michalis Foukarakis, Demetres Antoniades, Spyros Antonatos, and Evangelos P. Markatos. Flexible and high-performance anonymization of netflow records using anontool. In *Third International Conference on Security and Privacy in Communication Networks and the Workshops, SecureComm 2007, Nice, France, 17-21 September, 2007*, pages 33–38, 2007. doi: 10.1109/SECCOM.2007.4550304. URL https://doi.org/10.1109/SECCOM.2007.4550304.

[42] Jérôme François, Shaonan Wang, Walter Bronzi, Radu State, and Thomas Engel. Botcloud: Detecting botnets using mapreduce. In *2011 IEEE International Workshop on Information Forensics and Security, WIFS 2011, Iguacu Falls, Brazil, November 29 - December 2, 2011*, pages 1–6, 2011. doi: 10.1109/WIFS.2011.6123125. URL https://doi.org/10.1109/WIFS.2011.6123125.

[43] Jerry Gamblin. Mirai source code, Oktober 2016. https://github.com/rosgos/Mirai-Source-Code/ (July 2019).

[44] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y. Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 681–683, 2010. doi: 10.1145/1866307.1866396. URL https://doi.org/10.1145/1866307.1866396.

[45] Daniel Garant and Wei Lu. Mining botnet behaviors on the large-scale web application community. In *27th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2013, Barcelona, Spain, March 25-28, 2013*, pages 185–190, 2013. doi: 10.1109/WAINA.2013.235. URL https://doi.org/10.1109/WAINA.2013.235.

[46] Joseph Gardiner, Marco Cova, and Shishir Nagaraja. Command & control: Understanding, denying and detecting. *CoRR*, abs/1408.1136, 2014. URL http://arxiv.org/abs/1408.1136.

[47] Martin Grill, Ivan Nikolaev, Veronica Valeros, and Martin Rehák. Detecting DGA malware using netflow. In *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, pages 1304–1309, 2015. doi: 10.1109/INM.2015.7140486. URL https://doi.org/10.1109/INM.2015.7140486.

[48] Guofei Gu, Phillip A. Porras, Vinod Yegneswaran, and Martin W. Fong. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007. URL https://www.usenix.org/conference/16th-usenix-security-symposium/bothunter-detecting-malware-infection-through-ids-driven.

[49] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. pages 139–154, 2008. URL http://www.usenix.org/events/sec08/tech/full_papers/gu/gu.pdf.

[50] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic. 2008. URL http://www.isoc.org/isoc/conferences/ndss/08/papers/17_botsniffer_detecting_botnet.pdf.

[51] Peter Haag. Watch your flows with nfsen and nfdump. In *50th RIPE Meeting*, 2005.

[52] Peter Haag. User documentation nfdump & nfsen. 2006. https://www.first.org/resources/papers/conference2006/haag-peter-papers.pdf (January 2019).

[53] Fariba Haddadi, Jillian Morgan, Eduardo Gomes Filho, and A. Nur Zincir-Heywood. Botnet behaviour analysis using IP flows: With HTTP filters using classifiers. In *28th International Conference on Advanced Information Networking and Applications Workshops, AINA 2014 Workshops, Victoria, BC, Canada, May 13-16, 2014*, pages 7–12, 2014. doi: 10.1109/WAINA.2014.19. URL https://doi.org/10.1109/WAINA.2014.19.

[54] Fuye Han, Zhen Chen, HongFeng Xu, and Yong Liang. Garlic: A distributed botnets suppression system. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCS 2012 Workshops), Macau, China, June 18-21, 2012*, pages 634–639, 2012. doi: 10.1109/ICDCSW.2012.30. URL https://doi.org/10.1109/ICDCSW.2012.30.

[55] Scott Hilton. Dyn analysis summary of friday october 21 attack, Oktober 2016. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack// (June 2019).

[56] John. J. Hoffman, Steve C. Lee, and Jeffrey S. Jacobson. New jersey division of consumer affairs obtains settlement with developer of bitcoin-mining software found to have accessed new jersey computers without users' knowledge or consent, May 2015. URL https://nj.gov/oag/newsreleases15/pr20150526b.html.

[57] Rick Hofstede, Pavel Celeda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. Flow monitoring explained: From packet capture to data analysis with netflow and IPFIX. *IEEE Communications Surveys and Tutorials*, 16(4):2037–2064, 2014. doi: 10.1109/COMST.2014.2321898. URL https://doi.org/10.1109/COMST.2014.2321898.

[58] David I. Holmes. Authorship attribution. *Computers and the Humanities*, 28(2):87–106, 1994. doi: 10.1007/BF01830689. URL https://doi.org/10.1007/BF01830689.

[59] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*, 2008. URL http://www.isoc.org/isoc/conferences/ndss/08/papers/16_measuring_and_detecting.pdf.

[60] Sajad Homayoun, Marzieh Ahmadzadeh, Sattar Hashemi, Ali Dehghantanha, and Raouf Khayami. Botshark: A deep learning approach for botnet traffic detection. *Cyber Threat Intelligence*, pages 137–153, 2018.

[61] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Hai-Xin Duan. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1701–1713, 2018. doi: 10.1145/3243734.3243840. URL https://doi.org/10.1145/3243734.3243840.

[62] Nazrul Hoque, Dhruba K. Bhattacharyya, and Jugal K. Kalita. Botnet in ddos attacks: Trends and challenges. *IEEE Communications Surveys and Tutorials*, 17(4):2242–2270, 2015. doi: 10.1109/COMST.2015.2457491. URL https://doi.org/10.1109/COMST.2015.2457491.

[63] Infosec. Which industries are the biggest security targets?, 2015. https://resources.infosecinstitute.com/category/enterprise/securityawareness/security-threats-by-industry// (June 2019).

[64] Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: an empirical analysis of spam marketing conversion. volume 52, pages 99–107, 2009. doi: 10.1145/1562164.1562190. URL https://doi.org/10.1145/1562164.1562190.

[65] Anestis Karasaridis, Brian Rexroad, and David A. Hoeflin. Wide-scale botnet detection and characterization. 2007. URL https://www.usenix.org/conference/hotbots-07/wide-scale-botnet-detection-and-characterization.

[66] Ryan LaSalle Kelly Bissell and Paolo Dal Cin. The cost of cybercrime. In *Ninth annual cost of cybercrime study*, 2019. URL https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf#zoom=50.

[67] Wazir Zada Khan, Muhammad Khurram Khan, Fahad T. Bin Muhaya, Mohammed Y. Aalsalem, and Han-Chieh Chao. A comprehensive study of email spam botnet detection. *IEEE Communications Surveys and Tutorials*, 17(4):2271–2295, 2015. doi: 10.1109/COMST.2015.2459015. URL https://doi.org/10.1109/COMST.2015.2459015.

[68] Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A. Syed, and Syed Ali Khayam. A taxonomy of botnet behavior, detection, and defense. *IEEE Communications Surveys and Tutorials*, 16(2):898–924, 2014. doi: 10.1109/SURV.2013.091213.00134. URL https://doi.org/10.1109/SURV.2013.091213.00134.

[69] Paul Kimayong and Alexander Burt. Mirai variant has Android devices in its crosshair, August 2018. https://forums.juniper.net/t5/Threat-Research/Mirai-variant-has-Android-devices-in-its-crosshair/ba-p/338560/ (May 2019).

[70] Radhesh Krishnan Konoth, Emanuele Vineti, Veelasha Moonsamy, Martina Lindorfer, Christopher Kruegel, Herbert Bos, and Giovanni Vigna. Minesweeper: An in-depth look into drive-by cryptocurrency mining and its defense. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1714–1730, 2018. doi: 10.1145/3243734.3243858. URL https://doi.org/10.1145/3243734.3243858.

[71] Brian Krebs. Krebsonsecurity hit with record ddos, September 2016. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/ (June 2019).

[72] Brian Krebs. Who is anna-senpai, the mirai worm author?, January 2017. https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author// (June 2019).

[73] Christian Kreibich, Chris Kanich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamcraft: An inside look at spam campaign orchestration. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '09, Boston, MA, USA, April 21, 2009*, 2009. URL https://www.usenix.org/conference/leet-09/spamcraft-inside-look-spam-campaign-orchestration.

[74] Marc Kührer, Christian Rossow, and Thorsten Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In *International Workshop on Recent Advances in Intrusion Detection*, pages 1–21. Springer, 2014.

[75] Rober Lemos. Zeus botnet finds hold in amazon cloud, December 2009. https://www.securityfocus.com/brief/1046/ (May 2019).

[76] Knud Lasse Lueth. State of the iot 2018: Number of iot devices now at 7b – market accelerating, August 2018. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b// (June 2019).

[77] AbuseIPDB Marathon Studios Inc. Abuseipdb: making the internet safer, one ip at a time, January 2018. https://www.abuseipdb.com// (June 2019).

[78] MaxMind Inc. MaxMind GeoIP2. https://www.maxmind.com/en/geoip2-services-and-databases/ (April 2019).

[79] Chuck McAuley. Mirai: A botnet of things, October 2016. https://www.ixiacom.com/company/blog/mirai-botnet-things (July 2019).

[80] Kieren McCarthy. CBS's Showtime caught mining crypto-coins in viewers' web browsers. *The Register*, Jan 2018. https://www.theregister.co.uk/2017/09/25/showtime_hit_with_coinmining_script/ (December 2018).

[81] D. Kevin McGrath and Minaxi Gupta. Behind phishing: An examination of phisher modi operandi. 2008. URL http://www.usenix.org/events/leet08/tech/full_papers/mcgrath/mcgrath.pdf.

[82] Leigh Metcalf and Jonathan M. Spring. Blacklist ecosystem analysis: Spanning jan 2012 to jun 2014. In *Proceedings of the 2nd ACM Workshop on Information Sharing and Collaborative Security, WISCS 2015, Denver, Colorado, USA, October 12, 2015*, pages 13–22, 2015. doi: 10.1145/2808128.2808129. URL https://doi.org/10.1145/2808128.2808129.

[83] Cyndi Mills, Donald Hirsh, and Gregory R. Ruth. Internet accounting: background. Technical report, 1991. URL https://doi.org/10.17487/RFC1272.

[84] MineCryptoNight. MineCryptoNight - Making mining profits great again! https://minecryptonight.net/ (May 2019).

[85] Paul V. Mockapetris and Kevin J. Dunlap. *Development of the Domain Name System*, volume 25. 1995. doi: 10.1145/205447.205459. URL https://doi.org/10.1145/205447.205459.

[86] Dan Morrill. Malware starts using amazon ec2 as a command and control structure, December 2009. https://www.cloudave.com/1082/malware-starts-using-amazon-ec2-as-a-command-and-control-structure// (June 2019).

[87] Margi Murphy. YouTube shuts down hidden cryptojacking adverts. *The Telegraph*, Jan 2018. https://www.telegraph.co.uk/technology/2018/01/29/youtube-shuts-hidden-crypto-jacking-adverts/ (November 2018).

[88] Troy Mursch. Over 100,000 Drupal websites vulnerable to Drupalgeddon 2 (CVE-2018-7600). *Bad Packets Report*, Jun 2018. https://badpackets.net/over-100000-drupal-websites-vulnerable-to-drupalgeddon-2-cve-2018-7600/ (January 2019).

[89] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. Botgrep: Finding P2P bots with structured graph analysis. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 95–110, 2010. URL http://www.usenix.org/events/sec10/tech/full_papers/Nagaraja.pdf.

[90] Shishir Nagaraja, Amir Houmansadr, Pratch Piyawongwisal, Vijit Singh, Pragya Agarwal, and Nikita Borisov. Stegobot: a covert social network botnet. In *International Workshop on Information Hiding*, pages 299–313. Springer, 2011.

[91] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. https://bitcoin.org/bitcoin.pdf.

[92] NetLab360. Quick summary about the Port 8291 scan, Mar 2018. https://blog.netlab.360.com/quick-summary-port-8291-scan-en/ (April 2019).

[93] NIST National Vulnerability Database. NVD - CVE-2018-14847 Detail, Jul 2018. https://nvd.nist.gov/vuln/detail/CVE-2018-14847 (March 2018).

[94] Ekaterina Badovskaya Oleg Kupreev and Alexander Gutnikov. Ddos attacks in q1 2019, May 2019. https://securelist.com/ddos-report-q1-2019/90792/ (June 2019).

[95] Pierluigi Paganini. Ovh hosting hit by 1tbps ddos attack, the largest one ever seen, September 2016. https://securityaffairs.co/wordpress/51640/cyber-crime/tbps-ddos-attack.html/ (June 2019).

[96] Sergio Pastrana and Guillermo Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. *CoRR*, abs/1901.00846, 2019. URL http://arxiv.org/abs/1901.00846.

[97] Kompanek Pawlinski. Evaluating threat intelligence feeds. *First Technical Colloquium for Threat Intelligence in Munich.* https://www.first.org/resources/papers/munich2016/kompanek-pawlinski-evaluating-threat-ntelligence-feeds.pdf/ (April 2019).

[98] Alexandre Pinto. Measuring the iq of your threat intelligence. *DEFCON*, August 2014. https://www.slideshare.net/AlexandrePinto10/defcon-22-measuring-the/ (June 2019).

[99] Larry Ponemon. The second annual study on exchanging cyber threat intelligence: There has to be a better way. *Ponemon*, . https://www.ponemon.org/blog/the-second-annual-study-on-exchanging-cyber-threat-intelligence-there-has-to-be-a-better-way/ (April 2019).

[100] Larry Ponemon. The third annual study exchanging cyber threat intelligence: There has to be a better way. *Ponemon*, . https://www.infoblox.com/wp-content/uploads/infoblox-white-paper-ponemon-infoblox-2018-final-report.pdf (April 2019).

[101] Phillip A. Porras and Hassen Saïdi. A foray into conficker's logic and rendezvous points. 2009. URL https://www.usenix.org/conference/leet-09/foray-confickers-logic-and-rendezvous-points.

[102] Phillip A. Porras and Hassen Saïdi. A foray into conficker's logic and rendezvous points. In *2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET '09, Boston, MA, USA, April 21, 2009*, 2009. URL https://www.usenix.org/conference/leet-09/foray-confickers-logic-and-rendezvous-points.

[103] Proxy Lists 24. Proxy Lists 24 - Daily Free Proxy Server Lists. http://www.proxyserverlist24.top/ (April 2019).

[104] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*, pages 41–52, 2006. doi: 10.1145/1177080.1177086. URL https://doi.org/10.1145/1177080.1177086.

[105] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging. In *First Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, MA, USA, April 10, 2007*, 2007. URL https://www.usenix.org/conference/hotbots-07/my-botnet-bigger-yours-maybe-better-yours-why-size-estimates-remain.

[106] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 342–351, 2007. doi: 10.1145/1315245.1315288. URL https://doi.org/10.1145/1315245.1315288.

[107] Julian Rauchberger, Sebastian Schrittwieser, Tobias Dam, Robert Luh, Damjan Buhov, Gerhard Pötzelsberger, and Hyoungshick Kim. The other side of the coin: A framework for detecting and analyzing web-based cryptocurrency mining campaigns. In *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*, pages 18:1–18:10, 2018. doi: 10.1145/3230833.3230869. URL https://doi.org/10.1145/3230833.3230869.

[108] Tim Ring. Threat intelligence: why people don't share. *Computer Fraud & Security*, (3):5–9, 2014.

[109] Juan D. Parra Rodriguez and Joachim Posegga. RAPID: resource and api-based detection against in-browser miners. *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, pages 313–326, 2018. doi: 10.1145/3274694.3274735. URL https://doi.org/10.1145/3274694.3274735.

[110] Margaret Rouse. How to attack DDoS threats with a solid defense plan, January 2017. https://whatis.techtarget.com/definition/command-and-control-server-CC-server/ (May 2019).

[111] José Jair Santanna, Romain Durban, Anna Sperotto, and Aiko Pras. Inside booters: An analysis on operational databases. In *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, pages 432–440, 2015. doi: 10.1109/INM.2015.7140320. URL https://doi.org/10.1109/INM.2015.7140320.

[112] Fritz W Scholz and Michael A Stephens. K-sample anderson–darling tests. *Journal of the American Statistical Association*, 82(399):918–924, 1987.

[113] Mathew J. Schwartz. Cryptojackers Keep Hacking Unpatched MikroTik Routers. *Bank Info Security*, Oct 2018. https://www.bankinfosecurity.com/cryptominers-keep-hacking-unpatched-mikrotik-routers-a-11627 (April 2019).

[114] Jerome Segura. A look into Drupalgeddon's client-side attacks. *Malwarebytes*, Jun 2018. https://blog.malwarebytes.com/threat-analysis/2018/05/look-drupalgeddon-client-side-attacks/ (January 2019).

[115] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. 2009.

[116] Shodan. Shodan - The search engine for Internet-connected devices. https://www.shodan.io/ (April 2019).

[117] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 2012, Lake Tahoe, Nevada, USA, May 3-7, 2010*, pages 1–10, 2010. doi: 10.1109/MSST.2010.5496972. URL https://doi.org/10.1109/MSST.2010.5496972.

[118] SimilarWeb. Similar Web. Website Traffic Statistics & Market Intelligence. https://www.similarweb.com (May 2019).

[119] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists". In *3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, Alexandria, Virginia, USA, October 7-8, 2008*, pages 57–64, 2008. doi: 10.1109/MALWARE.2008.4690858. URL https://doi.org/10.1109/MALWARE.2008.4690858.

[120] SonicWall. Massive cryptojacking campaign compromised 200,000 MikroTik routers, Aug 2018. https://securitynews.sonicwall.com/xmlpost/massive-cryptojacking-campaign/ (March 2019).

[121] Lance Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23, 2003. doi: 10.1109/MSECP.2003.1193207. URL https://doi.org/10.1109/MSECP.2003.1193207.

[122] J. Stewart. Bobax trojan analysis, April 2008. http://www.secureworks.com/research/threats/bobax// (June 2019).

[123] Florida Tech. A brief history of cyber crime, January 2018. https://www.floridatechonline.com/blog/information-technology/a-brief-history-of-cyber-crime// (July 2019).

[124] Tenable. Mikrotik routeros vulnerabilities: There's more to cve-2018-14847, Oct 2018. https://www.tenable.com/blog/mikrotik-routeros-vulnerabilities-there-s-more-to-cve-2018-14847/ (March 2019).

[125] The Monero Project. Monero: What is Monero (XMR)? https://www.getmonero.org/get-started/what-is-monero/ (December 2018).

[126] The Pirate Bay. The Pirate Bay - Miner, Sep 2017. https://thepiratebay.org/blog/242 (December 2018).

[127] Olivier Thonnard and Marc Dacier. A strategic analysis of spam botnets operations. In *The 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS 2011, Perth, Australia, September 1-2, 2011, Proceedings*, pages 162–171, 2011. doi: 10.1145/2030376.2030395. URL https://doi.org/10.1145/2030376.2030395.

[128] Wiem Tounsi and Helmi Rais. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Computers & Security*, 72:212–233, 2018. doi: 10.1016/j.cose.2017.09.001. URL https://doi.org/10.1016/j.cose.2017.09.001.

[129] Orestis Tsigkas, Olivier Thonnard, and Dimitrios Tzovaras. Visual spam campaigns analysis using abstract graphs representation. In *2012 Symposium on Visualization for Cyber Security, VizSec '12, Seattle, WA, USA, October 15, 2012*, pages 64–71, 2012. doi: 10.1145/2379690.2379699. URL https://doi.org/10.1145/2379690.2379699.

[130] unixfreaxjp. Mmd-0056-2016 - linux/mirai, how an old elf malcode is recycled.., August 2016. http://blog.malwaremustdie.org/2016/08/mmd-0056-2016-linuxmirai-just.html/ (June 2019).

[131] R. van der Berg. How the 'Net works: an introduction to peering and transit, Febuary 2008. https://arstechnica.com/features/2008/09/peering-and-transit/4/ (May 2019).

[132] Gernot Vormayr, Tanja Zseby, and Joachim Fabini. Botnet communication patterns. *IEEE Communications Surveys and Tutorials*, 19(4):2768–2796, 2017. doi: 10.1109/COMST.2017.2749442. URL https://doi.org/10.1109/COMST.2017.2749442.

[133] Chun-Yu Wang, Chi-Lung Ou, Yu-En Zhang, Feng-Min Cho, Pin-Hao Chen, Jyh-Biau Chang, and Ce-Kuen Shieh. Botcluster: A session-based P2P botnet clustering system on netflow. *Computer Networks*, 145:175–189, 2018. doi: 10.1016/j.comnet.2018.08.014. URL https://doi.org/10.1016/j.comnet.2018.08.014.

[134] Ping Wang, Lei Wu, Baber Aslam, and Cliff Changchun Zou. A systematic study on peer-to-peer botnets. In *Proceedings of the 18th International Conference on Computer Communications and Networks, IEEE ICCCN 2009, San Francisco, California, USA, August 3-6, 2009*, pages 1–8, 2009. doi: 10.1109/ICCCN.2009.5235360. URL https://doi.org/10.1109/ICCCN.2009.5235360.

[135] Tao Wang and Shun-Zheng Yu. Centralized botnet detection by traffic aggregation. In *IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2009, Chengdu, Sichuan, China, 10-12 August 2009*, pages 86–93, 2009. doi: 10.1109/ISPA.2009.74. URL https://doi.org/10.1109/ISPA.2009.74.

[136] Wikipedia, the free encyclopedia. Internet service provider. https://en.wikipedia.org/wiki/Internet_service_provider/ (May 2019).

[137] Wikipedia, the free encyclopedia. Netflow architecture, 2019. URL https://en.wikipedia.org/wiki/NetFlow#/media/File:NetFlow_Architecture_2012.png.

[138] Wojciech. Command and control server in social media (twitter, instagram, youtube + telegram). February 2018. URL https://medium.com/@woj_ciech/command-and-control-server-in-social-media-twitter-instagram-youtube-telegram-5206ce763950.

[139] Wordfence. WordPress Plugin Banned for Crypto Mining, Nov 2017. https://www.wordfence.com/blog/2017/11/wordpress-plugin-banned-crypto-mining/ (January 2019).

[140] Jun (Jim) Xu, Jinliang Fan, Mostafa H. Ammar, and Sue B. Moon. On the design and performance of prefix-preserving IP traffic trace anonymization. In *Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop, IMW 2001, San Francisco, California, USA, November 1-2, 2001*, pages 263–266, 2001. doi: 10.1145/505202.505234. URL https://doi.org/10.1145/505202.505234.

[141] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A. L. Narasimha Reddy, and Supranamaya Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC 2010, Melbourne, Australia - November 1-3, 2010*, pages 48–61, 2010. doi: 10.1145/1879141.1879148. URL https://doi.org/10.1145/1879141.1879148.

[142] Hossein Rouhani Zeidanloo and Azizah Abdul Manaf. Botnet command and control mechanisms. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 1, pages 564–568. IEEE, 2009.

[143] Lenny Zelster. When bots use social media for command and control, Febuary 2015. https://zeltser.com/bots-command-and-control-via-social-media// (June 2019).

[144] Yuanyuan Zeng, Xin Hu, and Kang G. Shin. Detection of botnets using combined host- and network-level information. In *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2010, Chicago, IL, USA, June 28 - July 1 2010*, pages 291–300, 2010. doi: 10.1109/DSN.2010.5544306. URL https://doi.org/10.1109/DSN.2010.5544306.

[145] Junjie Zhang. *Effective and scalable botnet detection in network traffic*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2012. URL http://hdl.handle.net/1853/44837.

[146] Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Detecting stealthy P2P botnets using statistical traffic fingerprints. In *Proceedings of the 2011 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2011, Hong Kong, China, June 27-30 2011*, pages 121–132, 2011. doi: 10.1109/DSN.2011.5958212. URL https://doi.org/10.1109/DSN.2011.5958212.

[147] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong Kuan Chen, and Shiuhpyng Shieh. Iot security: Ongoing challenges and research opportunities. In *7th IEEE International Conference on Service-Oriented Computing and Applications, SOCA 2014, Matsue, Japan, November 17-19, 2014*, pages 230–234, 2014. doi: 10.1109/SOCA.2014.58. URL https://doi.org/10.1109/SOCA.2014.58.

The cover of this thesis has been designed using resources from Freepik.com – https://www.freepik.com/free-photos-vectors/people