FINDING RELATIONS BETWEEN BOTNET C&CS FOR FORENSIC PURPOSES

RALPH BROENINK

Master's Thesis
Services, Cyber Security and Safety Group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

May 2014

Ralph Broenink: *Finding relations between botnet C&Cs for forensic purposes,* Master's Thesis, May 2014

SUPERVISORS:
Dr. Maarten H. Everts
Prof. dr. Pieter H. Hartel
Drs. Floor Jansen (NHTCU)

The question of whether machines can think...
is about as relevant as the question of whether submarines can swim.

Edsger W. Dijkstra

ABSTRACT

Botnets, large international networks of infected computers (so-called bots), play a central part in the digital underground economy, providing the infrastructure required for a multitude of malicious activities. To ensure a botnet keeps running, the botnet owner utilizes specialized technologies to send control messages to his bots, while keeping resilience against take down and stealth against detection from law enforcement agencies and rivals. Parties such as these are developing detection and take down methodologies.

However, botnet owners are in the advantage: even after detection and take down, it is hard to trace the owner, who remains unpunished and can continue his criminal career. This proves to be a significant problem for law enforcement, as a confiscated machine may not provide direct leads. Often, it is not known which machine was managed by which miscreant or was part of which specific botnet infrastructure.

In this research, we propose a novel approach in identifying the infrastructure and miscreant belonging to confiscated machines. We define a set of characteristics that can be applied to confiscated hard disks. These will then be used to extract clusters of machines with commonalities from large datasets. We will validate our approach by applying it to a test dataset of 104 different disk images, showing how experts would use this to gain insight in large datasets.

ACKNOWLEDGEMENTS

This work would not have been possible without the help, guidance and feedback of several individuals. I would like to take the opportunity to express my sincere gratitude towards them.

First of all, I would like to thank Maarten Everts for enthusiastically thinking along and providing me with the feedback I needed. Although meeting face-to-face has not always been easy (though *Google Hangouts* did their job), I could always count on his involvement and this work would not have been possible without his guidance.

Additionally, I'd like to thank Pieter Hartel for being part of my graduation committee. Although we weren't able to meet often, his critical feedback has improved the scientific value of this work.

I also wish to thank Floor Jansen for her involvement in this project. Although she was not an expert on the technical aspects of my work, she was able to bring me back to right track when I was lost in a problem. A thank you also to Peter Wagenaar for complementing Floor's expertise and providing me with the topic of this paper and useful technical insights that have greatly improved some parts of this thesis.

I am also grateful to the members of the National High Tech Crime Unit of the Netherlands Police Agency for providing me a place to complete my final research project and welcoming me to the world of crime fighting. They made me feel at home, for which I am sincerely grateful. A special thank you to the investigators that allowed me to interview them in the final phases of my research and lent me their time, creativity and expertise.

Last of all, I want to thank my friends and family, who have been supportive during my entire study, the members of Inter-*Actief*, for allowing me to evolve myself (beyond becoming a *microwave student* – quickly cooked, but tasteless), and everybody else who has been encouraging with respect to the final part of my study.

Ralph Broenink

CONTENTS

1 INTRODUCTION

Not many people may be realizing it, but *botnets* are an important part of our day-to-day lives. Most of the spam received daily originates from a botnet and important websites are down from time-to-time due to botnet-initiated attacks. Currently, your home computer may even be part of a yet undiscovered botnet, without you knowing it.

Botnets are networks of infected hosts (*zombies* or *bots*, from *robot*) under centralized control by a human operator, known as the *botmaster* or *botherder* [46, 63]. Owners of infected computers are unaware of their participation in a botnet; bot recruitment usually occurs via similar channels as infection by other types of malware. However, botnets distinguish themselves from normal malware by the usage of a communication channel to the botnet operator, allowing these miscreants to issue commands to their bots to perform a multitude of malicious activities while remaining relatively anonymous.

Due to the massive scale of botnets – size estimates of single botnets range from several thousands to thirty million bots [79] – they are effective for financial gain, harming vital infrastructures and for performing other illegal activities [33]. Since a larger network with more capacity has more capabilities, it can be put to use more often for a wider range of purposes, generating more revenue for the botmaster. They are therefore always looking to recruit more bots for their network.

Parallels with regular cloud services can be drawn – services are offered on multiple hosts working towards the same goal – making botnets underground cloud service providers, or even Swiss army knives, specifically purposed for illegal activities. However, one must carefully note that nodes in a botnet do not participate voluntarily and recruitment requires a stealthy set-up via malware. Furthermore, contrasting cloud services, botnets consist of unreliable nodes, as any participant may clean or shutdown their computer at any time.

1.1 PROBLEM

The sheer amount of illegal activities performed by botnets has sparked the interest of academia, security companies and law enforcement agencies. Time and effort is put into investigating botnet operations and developing botnet detection, disruption and take down methodologies [3, 20, 46]. Although such efforts lead to a deep understanding of botnets, the ultimate goal of tracking down the miscreants is a hard and tedious process.

Law enforcement agencies are particularly interested in identifying miscreants, attributing malicious activities to a single person or a criminal organization. Often, disks are seized to allow detailed forensic analysis of the botnet, but mostly this does not lead to the capture of any miscreant [63]. Reasons for this are multitude; contributing factors are a robust service-oriented underground economy and the development of resilience and stealth enhancing technologies. Examples include the use of *stepping stones* to obfuscate the real location of the botmaster (see Section 2.3.3).

However, using and managing their botnet, miscreants are bound to leave traces. Such traces may include the specific architecture of their network, their victims or even personal traces on central machines (so-called *Command & Control servers*, or C&Cs) [66]. Combining these characteristics, we can build a profile of the machine (and possibly the managing miscreant) by finding relations between different seized machines.

When doing this on a substantial dataset, we expect to see similar patterns and networks of related machines, matching multiple machines operated by one common miscreant and part of the same infrastructure. This may aid investigation, possibly resulting in direct leads or a (more) complete picture of the miscreant, but may also provide the justification for the allocation of more resources for a more in-depth investigation.

This supports our main problem to be stated as follows:

How can we reliably determine which C&C machines were managed by the same person or have been part of the same infrastructure?

In the process of building a profile based on infrastructural or user characteristics, there are two main requirements. First and foremost, we must build a profile that is precise enough to retain low false positive and low false negative ratios, ensuring that machines are not being disregarded as being not managed by the same miscreant or part of the same network. Secondly, profiling must complete in reasonable time and with limited resources, and results must be presented in a clear way, ensuring that our method does not burden investigators with additional complicated tasks.

From this, we have gathered the following research questions that need to be answered before we can provide a solution to our problem:

1. Which identifying characteristics can be found inside a C&C?
2. How can these characteristics be used to reliably identify a common miscreant or infrastructure?
3. How can results be presented to the investigator?
4. How efficient is using these characteristics when searching through large datasets?

1.2 CONTRIBUTION

The contribution of this research is twofold. In Chapter 2, we will start by providing an overview of the current state of literature regarding botnet technology, economy and detection techniques. We will find that the underground economy is increasingly being powered by botnets, utilizing specific technologies to enhance their stealth against detection and resilience against take downs. We will also discuss take down methods, and explain why botmaster traceback is often hindered.

The second part of this research will focus on investigating the possibility of finding relations between confiscated machines using user and machine specific characteristics. In Chapter 3, we will discuss the characteristics we will use to relate machines together and how we came from this to a visualisation. Chapter 4 describes our steps in confirming and validating our results, including a quantitative analysis, interviews with experts and in-depth analysis of some of our data.

Finally, we will discuss our successes in finding identifying characteristics, but also our suggestions for further improvement of our methods and answers to the research questions, in Chapter 5.

2 BOTNET OVERVIEW

In this chapter, we will discuss the current state of research regarding botnet economy, technology and detection. Section 2.1 describes the underground economy that has formed around botnets and the services both utilized and provided by botnets. Section 2.2 provides an introduction on the technologies used by botmasters to form and operate their botnet. Finally, in Section 2.3 an overview of detection, take down and botmaster traceback methods is provided.

2.1 ECONOMY

Where cybercriminals used to sell their illegal products individually, the underground economy is now evolving into a market where everything is offered to conduct crime, ranging from malicious code to the infrastructure required to control the spread of malware [90]. Services offered are increasingly diversified and trending towards specialization, ranging from supply chain management, software development and distribution, to customer support, training and post-sales services. Most of these services are characterized by their ease of use and strong customer orientation, offering user-friendly controls and dashboards.

Botmasters are increasingly becoming an intermediary within the underground economy, outsourcing the collection of enough hosts for their botnet and the maintenance of their software. They limit themselves to offering malicious services, renting out their botnet and making money in the process. An overview of the underground economy surrounding the botmaster is provided in Figure 2.1.

2.1.1 *Setup*

To start building a botnet, the botmaster needs to invest in software and hardware to control his botnet. Instead of building and developing this themselves, botmasters can opt to outsource this to other miscreants – at a price. In this subsection, we will discuss two main parts of the setup: the software (i. e. *bot kit*) and the hosting of the software.

2.1.1.1 *Bot Kits*

There are developers designing and building so-called *bot kits*, offering everything required to build a basic botnet that can easily be customised.

Basic features such as malware and basic management mechanisms are often included by default [6, 53]. Additionally, these kits allow a great deal of customization, typically through modules and plug-ins that are sold separately. Such modules may provide specific functionality, such as information stealers, form grabbers or *web injects* for specific websites (see Section 2.1.3.6).

In this paper, all prices are listed in US dollar.

Prices for bot kits vary wildly. Basic configurations with the newest versions may be sold for prices ranging from $2000 to $3000 [52, 95], although prices half of that have also been seen [67]. Additional modules are sold for $50–$1000, depending on the capabilities. A complete VIP package, including all modules, is sold for $3000–$6000 [52, 67].

Updates are also sold, for instance to fix bugs or improve malware that is losing its potential due to vulnerabilities being patched and anti-virus software being updated. Kits may also integrate with other malicious services and for instance automatically apply obfuscation methods when required (see Section 2.2.2.2), typically for rates of $10–$20 per obfuscation or in a subscription model [52].

2.1.1.2 *Bulletproof Services*

After acquiring the necessary software, the botmaster needs a place to put this online (see also Section 2.2.1). *Bulletproof hosting providers* offer hosting services that keep customer information confidential to ensure the anonymity of the miscreant. Other services include DDoS protection, *IP masking* (hiding the real IP address of the server) and similar services, aimed at protecting the servers from takedowns from law enforcement and competitors. Furthermore, providers are often based in jurisdictions where cyber laws are weak or non-existent, making them immune to notice-and-take-down requests [52].

It is therefore unsurprising that demand for bulletproof service providers is high and that they are significant market enablers. Bulletproof services are provided for a monthly fee of $300 to $500, after a one-time setup fee of $50–$100 [52].

2.1.2 *Host Recruitment*

To expand their botnet, botmasters continuously need malware to be installed on new hosts. Infection may occur via a variety of ways: email attachments, USB drives, automated infection or an existing backdoor. Preferably, this infection happens stealthy, as to not cause the victim to become suspicious and to try to remove the malware.

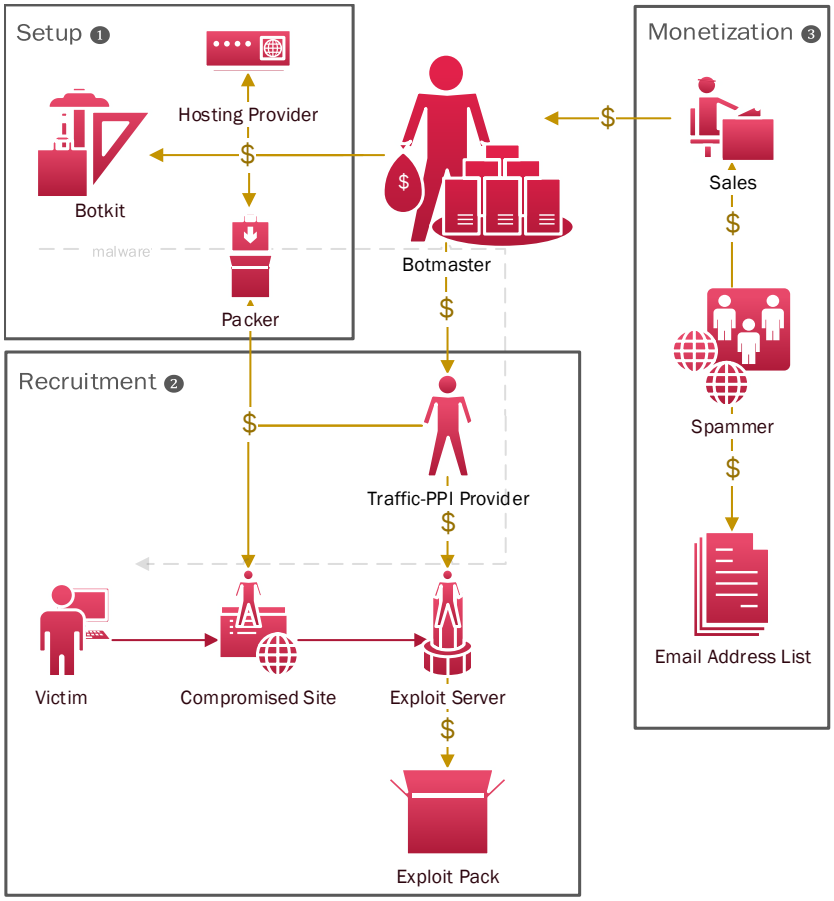


Figure 2.1: Overview of botnet economy and malware flow. The botmaster starts with buying his botkit and optionally a packer to generate his malware, and a server to host his central control server (❶). Malware installs are bought from a (Traffic-)PPI provider (❷), which ensures that an exploit server and compromised site are set up (which may be done by affiliates). The victim visits the compromised site and receives the exploit generated by the botmaster. After setup, the botmaster starts monetization, for instance by renting the botnet out to spammers (❸). Please note that middle-men and affiliates are emerging in every step of the process.

2.1.2.1 *Drive-by Downloads*

A commonly used method to install malicious software, is the utilization of *drive-by downloads*. Such downloads rely on *exploit kits*: software packages bundling multiple exploits targeting vulnerabilities in web browsers and their plugins (e. g. Flash and Java).

When visiting a compromised website or viewing a website using a compromised advertising network [45], victims are redirected to *exploit servers* hosting exploit kits. Upon visit, the browser is analysed and served with the proper exploit for the host's operating system, browser and plugins. If the exploit is successful, malware is downloaded to and executed on the victim's computer [12, 21].

In 2013, a large Dutch news site was compromised through code injection into one of their advertising networks. Users were redirected to an exploit kit abusing vulnerabilities in Java, allowing the download of malware [69].

To successfully perform a drive-by download, four requirements need to be satisfied: a compromised website, an exploit kit, an exploit server and a redirection chain from the compromised website to this server; a user would visit the compromised site and get redirected through the chain to the exploit server hosting the exploit kit. Traditionally, exploit kits have been sold for one-time fees, allowing unlimited use once purchased, leaving the miscreant responsible for the other aspects.

Recently, a software-as-a-service model emerged: the Blackhole exploit kit offers licensing for a pre-installed exploit server that can be configured by clients to drop various malware variants, all aspects of hosting and updating the software are handled by the Blackhole team, starting at $500 per month [77]. Clients are still responsible for getting victims to the exploit server – which is an art yet other miscreants are concerned with.

2.1.2.2 *Pay-per-Install*

Considering the number of targeted victims, their location and the botmaster's budget, the global dissemination of malware can also be entirely outsourced to *pay-per-install* (PPI) service providers, allowing botmasters to focus their efforts on monetizing their botnets. Caballero et al. distinguished three roles in the pay-per-install market [9]:

CLIENTS want to install their malware on a number of hosts, profiting from the malicious activities enabled by malware deployed on the target's hosts. These may for instance be botmasters looking to expand their botnet.

PROVIDERS develop special programs (*downloaders*) that remain under their control. These pieces of software can receive and install the client's programs and track whether the install succeeds. Providers usually install multiple client programs on the same host.

AFFILIATES may optionally act on behalf of the providers when installing client software, specializing in some specific distribution

method, including torrents, drive-by downloads, spam, or even using other PPI services [21]. Instead of victimizing new hosts themselves, providers pass their software down to affiliates and pay per confirmed install.

To keep monetizing PPI, stealth installs are required. Clients are often in charge of providing stealthy programs, while affiliates rely upon the provider to provide them with a stealthy downloader. Stealth can be gained by using *packers* (see Section 2.2.2.2), which are again developed and sold by third parties.

It depends on the PPI provider how exactly installs are generated – clients are generally not concerned with their methods. For instance, *traffic-PPI* providers offer complete drive-by download solutions, requiring their clients only to provide their malware binaries, while taking care of generating traffic to lure victims to malicious-serving hosts, redirecting them toward exploit kits and exploiting them. These solutions may in turn be offered by affiliates, making Traffic-PPI providers middle-men between traffic generators, exploit kit writers and clients [21].

Prices per install depend on the geographic location of the victim. In countries such as the United States, install rates average $100–$150 per thousand installs, while a geographically mixed set of installs sells at prices as low as $7–$15 per thousand hosts [90, 75]. The price differences are caused by the likelihood that a malicious file will be successfully installed, the wealth of the country and the internet speed. For botnets, a mixed set may be sufficient for many purposes, although a specific country may be targeted for information theft purposes, aiming at a specific language, purchasing method or financial institution.

2.1.3 *Monetization*

After having infected hosts with malicious software and having set up their botnet, botmasters have the ability to command their bots to perform a multitude of malicious activities for their customers. These customers have varying motives, ranging from cyber vandalism, cyber crime and hacktivism, to cyber terrorism and cyber warfare [94].

Services may be offered directly by the botmaster, but may also use middle-men handling sales. To build black market marketing and find these customers, miscreants typically utilize *Internet Relay Chat* (IRC) channels, *Jabber* chat networks and web forums [78], but may also buy space for *crimevertisements* in other underground products. Payments are handled using irreversible, unregulated and convenient e-currency. An example of such a provider is WebMoney, which provides a reliable platform and environment to conduct underground business activities

WebMoney (`wmtransfer.com`) allows users to have multiple purses, each for WebMoney units linked to a different currency. For example, WMZ (i. e. the Z purse) is used for carrying out transactions in US dollars. The exchange rate for every purse is fixed by a guarantor based on the local legal environment.

[52]. Bitcoins are another example of a currency that is popular in the underground economy [71].

Malicious activities performed by botnets can be separated in two groups. The first group of activities focuses on systems not part of the botnet, where the botnets have the distinct advantage that they are large networks with great resources, providing anonymity for the botmaster and being flexible enough to stay ahead of the law (e. g. *Denial of Service*, *spam* and *click fraud*). The second group aims at attacking the bots and their users themselves, providing similar services as 'traditional' malware, but taking advantage of the diverse and large group of infected systems to which the botmaster has gained access (e. g. *extortion* and *Bitcoin mining*).

2.1.3.1 *Denial of Service*

One of the most popular uses of botnets involves *distributed denial of service* (DDoS) attacks [16]. Typically, these attacks are executed by setting all bots in the botnet to send large amounts of traffic to the computer under attack, leading to the denial of service of the victim's computer when there are insufficient resources to handle all incoming requests.

Such attacks are only successful if the attacker has accumulated greater resources than the victim; otherwise, the victim could cope with the attack. This implies that larger and better-known websites can only be taken down by coordinated attacks of large botnets.

DDoS attacks are a powerful tool for instances of extortion, unfair competition, hacktivism and (when targeting critical infrastructures) cyberterrorism. A particular example of DDoS attacks on large scale, were the websites of major financial institutions being targeted during the WikiLeaks affair [38], but smaller attacks keep happening every day [97].

2.1.3.2 *Spam*

By commanding large numbers of bots, thousands of unsolicited emails can be transmitted in a matter of minutes [65]. Mails sent typically include a hyperlink to a commercial website or are used for phishing purposes; sending the user to a website mimicking another (e. g. banking) website and persuading the user to enter his credentials or other private details.

Blacklisting is one of the classic methods to divert spam, but due to the transient and distributed nature of botnets, it is hardly effective blacklisting individual bots, as each individual bot will only send low volumes of email by itself and could easily be exchanged for another bot. This allows for great successes of spam runs and explains why the majority of international spam is sent by botnets [65, 96].

Email addresses to send spam to are typically traded in bulk, with rates varying from $25 to $50 for one million email addresses. Prices depend on whether email addresses are validated, whether the bulk list has been used

recently or to which region they belong. The value of an email address is also contingent upon whether it belongs to a free email service (e. g. Gmail or Hotmail), with email addresses belonging to these services selling for half the price of normal email addresses [56].

Prices for a targeted mailing (i. e. *spam-as-a-service*) range from $70 for thousands of addressees to $1000 for tens of millions [86]. Botnets are sometimes also rented out for larger campaigns ($10,000 per month for 100 million emails per day), typically after being offered a free trial to evaluate performance of the botnet [56]. Spam revenue estimates vary wildly, ranging up to two million dollars per day for a single botnet [28].

2.1.3.3 *Click Fraud*

Advertising networks sell advertising space on websites. In turn, these networks pay website owners a portion of their revenue for every click a visitor generates on their website [13, 17]. When bots are set by the botnet owner to perform clicks on his own websites, revenue can quickly be generated, representing fraud, since these bots do not have the intention of buying the product offered.

Well-known and popular advertising networks include DoubleClick and Google AdSense.

2.1.3.4 *Link Building*

To assess the relevancy and popularity of a website, search engines use a number of criteria. One of these is the amount of links to the website; the more links there are, the higher a website appears in search results [17].

Search engine optimization (SEO) is a legitimate business in itself that is concerned with optimizing the factors that impact the position of a website in search results. Botnet operators have automated the optimization process and post lots of links to the same website, on different weblogs, forums and other websites. Such links are stuffed inside randomly generated sentences with specific keywords and placed into legitimate looking comments. Reportedly, SEO spam can be bought for $300 per month [79].

2.1.3.5 *Content Delivery & Proxying*

Other criminals, though especially the botmaster himself, may want to host content that is unwelcome or illegal. Phishing sites, malware or botnet updates must be defended from disruption – hosting such content on a central server makes it relatively easy to identify and take down.

Using *fast-flux networks*, the botnet can be utilized to deliver content and obfuscate the real central location of the malicious content, effectively turning their botnet into a *bulletproof hosting provider*, or may use their botnet as a proxy network providing anonymity services [26]. Every bot in a fast-flux network serves or forwards content independently. Fast-flux networks will be explained in more detail in Section 2.2.2.1.

2.1.3.6 *Information Theft*

Typical botnet malware allows the botnet owner to remotely install additional modules to retrieve confidential information stored on the users' computers. Such modules may include keyloggers, data crawlers, or *man-in-the-browser* phishing attacks [55].

In the latter case, when a victim visits a page specified by the botmaster (so-called *trigger pages*), which may for instance be an online banking environment, a malicious page or additional frame (*web inject*) is injected into the user's browser. Since the injected content carefully mimics the expected page's style, the SSL information appears to be correct and the URL in the address bar does not change, the injection is hard to detect.

The injected page may request the victim to enter his personal or banking details or may request the user to transfer money. When done carefully, the victim may not suspect anything and may even voluntarily bypass any additional security measures set up by the attacked website (e. g. confirmation text messages or security tokens).

Stolen data may also be sold on IRC (chat) networks or underground forums and includes credit card numbers, addresses, names, phone numbers, usernames and passwords, social security numbers and bank account information [15]. Depending on the victim's nationality, prices for full packages of a single user's data range from $5 to $8 [28].

2.1.3.7 *Extortion*

A specific category of malware is called *ransomware*, which uses extortion as a way to gain money from a victim. Ransomware locks an entire computer, showing a message supposedly from a law enforcement agency. Such messages are along the lines of "you have browsed illicit materials and must pay a fine" and are provided with an air of authenticity and can not be dismissed, causing victims to pay up (estimates show that a little less than 3% of victims pay, totalling a worldwide extortion estimate of over $5 million). Other forms of ransomware encrypt files on the hard disk and require payment for the decryption key. However, even after payment, access is often not restored [87].

Popular payment methods include Ukash and Paysafecard voucher codes, which are sold at petrol stations and supermarkets.

2.1.3.8 *Bitcoin Mining*

Bitcoins are a digital currency scheme devised in 2009. Bitcoins can be generated (*mined*) by iteratively performing hashing operations until a certain target hash has been found. Finding a solution takes significant resources; the first to find a solution for a given set of parameters, 'wins' and may claim Bitcoins for their solution. In turn, Bitcoins can be traded for 'physical' currency on special exchanges.

	COMPLEXITY	STEALTH	LATENCY	RESILIENCE
Centralized	low	medium	low	low
Peer-to-Peer	medium	high	medium	medium
Random	low	low	high	high

Table 2.1: Overview of some core properties of botnet control topologies [11]

Due to the significant computing resources required to find solutions to the Bitcoin problem, miners typically join forces in a *mining pool*, where participants share performance and earnings. However, due to the massive amount of computing power available in botnets, they can be used for mining instead, without sharing the earnings with the participants [42]. Additionally, locally stored Bitcoins could also be stolen by the miscreant.

The value of a Bitcoin fluctuates highly; during the writing of this paper, the value of one Bitcoin went from $100 to around $1,000 and back down to $500. More about Bitcoins in general can be found at `bitcoin.org`*. An overview of Bitcoin exchanges is available from e. g.* `bitcoincharts.com`*.*

2.1.3.9 *Pay-per-Install*

Botmasters can also opt to act as a PPI affiliate (see Section 2.1.2.2) and allow other miscreants to install software on bots within their network. This may involve installing a downloader on the compromised host, which automatically contacts a PPI provider for new binaries to install [9, 21]. This service is also sometimes referred to as *Install-by-Install* (IBI) [52].

Installed software may conduct similar activities as mentioned in the previous parts of this section, including click fraud, or the theft of information, but may also sell software to the user under false pretences (e. g. fake anti-virus software) or may even include malware used by other botmasters looking to expand their botnet [9].

2.2 TECHNOLOGY

Botnets use specific technology for infection and recruitment, control, stealth against detection and resilience against take downs by law enforcement or other hostiles. In this section, we will dive into the different technologies used by botmasters.

2.2.1 *Control Topologies & Channels*

To control their botnet, the botmaster issues commands to their bots. These commands may for instance include an address list to send spam emails to or servers to target an attack on. There are three infrastructures to distinguish, each with their own advantages and disadvantages, as summarized in Table 2.1. A botnet owner may opt to combine some of these methods and mitigate some issues each of these topologies have.

2.2.1.1 *Centralized*

The most straight-forward approach to send commands to the bots in a botnet, is to set up a central *Command & Control server*, or *C&C server*, to which all bots connect. The centralized architecture allows for messages to arrive in a predictable manner and with low latency. However, by centralizing botnet communication, all bots connect to the same end point, which is both easier to detect and easier to take down.

Two prevalent technologies to facilitate C&C communications are IRC and HTTP, the latter of which is becoming more and more popular due to its clear advantages [84, 98]. Although there are other technologies, including Jabber or a self-written protocol, they are only used sporadically in comparison to IRC and HTTP.

IRC-BASED One of the first automated bot systems is known as *Eggdrop*, a non-malicious bot that was aimed at helping channel operators with managing channels on the *Internet Relay Chat* (IRC) chat network [84]. Although it did not offer services known to botnets today, it allowed multiple bots to work together under centralized control.

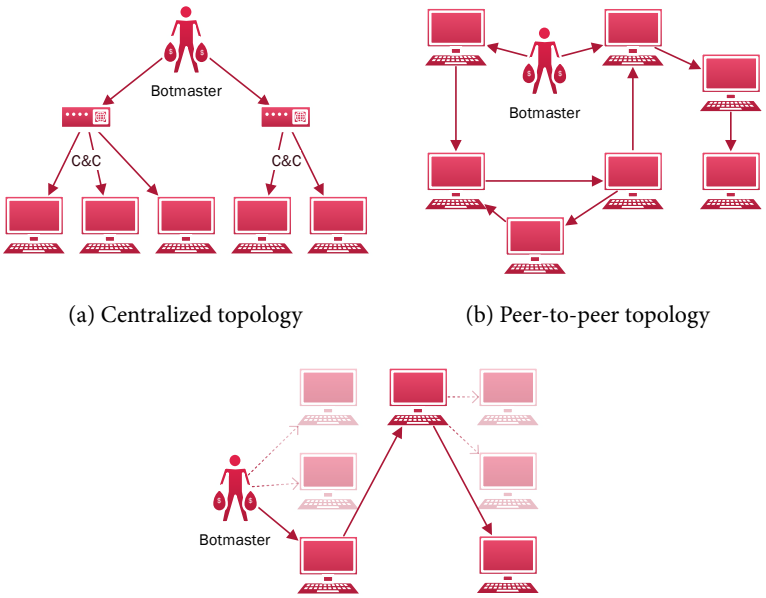
Although the last version of Eggdrop is from 2011, it is still a popular IRC bot. Eggdrop is available from eggheads.org.

Since then, bots with more nefarious purposes emerged, aiming to attack other users or bots on the chat network and hiding the attacker in the process. Bots were increasingly used to take down large targets and more bots were required, but voluntary participants were hard to find. Instead, miscreants resorted to distributing their bot software by hiding in legitimate files, creating the first botnets as we know them today.

IRC is still being used as the backbone of some botnets [25, 84, 85]. These all connect to the same IRC network, where the botnet operator can give them commands. Although usage of IRC is simple and effective for botnet owners, detection and take down methods of IRC-based botnets are widely available. Most firewalls today block IRC connections by default and the usage of these kinds of botnets is therefore declining [98].

HTTP-BASED To mitigate the blocking of IRC, botnets are resorting to using the web (HTTP) as an alternative. Using HTTP, bots regularly pull information from a public-facing website, contrasting IRC-based botnets that get their commands pushed from the central server.

HTTP has the advantage that it can hide in the large amounts of HTTP traffic happening every day and is therefore not easily identified or blocked. Furthermore, compromised web servers can be used to hide communication without these servers to be detected easily [13]. Although recent numbers are lacking, it is clear that the majority of centralized botnets use HTTP as their backbone [98].



(a) Centralized topology

(b) Peer-to-peer topology

(c) Random topology

Figure 2.2: Schematic diagrams of different control topologies

2.2.1.2 *Peer-to-Peer*

In contrast, in a *peer-to-peer* (P2P) communication architecture, there's no centralized C&C server being a single point of failure. Instead, every bot has the ability to communicate with other bots. The botnet operator issues their commands to a few individual bots, which will spread the commands through some of the bots they are aware of, slowly propagating the message through the network [22].

The utilization of peer-to-peer networks increases resilience of the network and is therefore more resistant against take downs, but the design of peer-to-peer systems is more complex and does not provide guarantees on message delivery or latency. Messages could take several days to reach all nodes, which is a significant disadvantage. In 2010, P2P topologies were only used in 2% of the botnets, but their usage is increasing [84].

2.2.1.3 *Random*

A third yet uncommon option is the usage of a random communication topology. This topology has the advantage of not maintaining any administration of the bots in the botnet, as shown in Table 2.2. Each bot knows of at most one other bot, which it randomly detects.

	KNOWN TO OWNER	KNOWN TO EACH BOT
Centralized	all	none
Peer-to-Peer	few	few
Random	zero or one	zero or one

Table 2.2: Overview of number of bots known to participants in botnets with different topologies

It is loosely based on the peer-to-peer architecture; every bot communicates by scanning and searching randomly for other bots on the internet, for instance by trying all IP addresses in a specific range. When active bot malware is detected, the message is sent and the receiving bot continues scanning for the next bot [11].

Although implementation of a random communication model is rather easy, it is hard to make any delivery and latency guarantees. Furthermore, the botnet may easily be detected by its constant scanning and probing for other bots, reducing its stealth.

2.2.2 *Resilience Enhancing Technologies*

Researchers and law enforcement are actively identifying, comprehending and disrupting botnets. Botnet owners have developed several techniques to ensure continuous operation of the botnet and increase their resilience against take downs.

2.2.2.1 *Location*

If a single location of the centralized C&C server has been hard-coded within the bot software, it can't be changed quickly when it is taken down. After disruption, bots will not be able to receive new instructions and although some bots may opt to function in an alternative 'zombie' mode and continue to propagate and harvest information, the botnet has largely become dysfunctional [88]. There are two methods to mitigate this issue:

DOMAIN FLUX Using *domain flux*, the botnet does not utilize a single C&C domain. Instead, every bot receives a *domain generation algorithm* (DGA) that computes a list of domains to be contacted, based on factors such as the current date [55]. Since the domain list changes as time progresses, the botnet owner has to register their domain names in advance – if such a registration fails or a domain name gets taken down, the botnet may be come dysfunctional for a short while, but could be restored by registering a domain that will be used in the future.

Instead of a dynamic domain generation algorithm, a fixed list of domain names could also be utilized.

FAST-FLUX *Fast-flux networks* utilize DNS mappings to obfuscate the real central location of the botnet. In fast-flux networks, one DNS name points to multiple bots in the botnet. These pointers are quickly cycled amongst the participants of the botnet to accommodate for the often occurring population changes within the botnet [39]. Every bot in the network may receive the content to deliver and serve this independently, but may also act as a *reverse proxy*, relaying every request to a central location (*mother ship*) and sending every reply back to the original sender.

Only by shutting down the domain name, the DNS server, or by taking down the command structure, a fast flux network can be brought to a halt. An additional layer of redirection may be gained by hosting the DNS service itself within a fast-flux network (*double flux*), allowing every bot to become a DNS server, accepting and forwarding DNS queries [93].

2.2.2.2 *Obfuscation*

To prevent proper understanding of botnets, malware programmers typically opt to obfuscate their software [51]. Obfuscation involves the hiding of the actual meaning of the malicious code, making the binary code intentionally ambiguous and hindering proper understanding of the program's flow – either by researchers or by anti-virus software – while maintaining the program's functionality and semantics.

One of the solutions for the malware builder is to maintain the client software in encrypted form, (partially) decrypting it as needed during execution [35]. The binary may also be slightly modified to change its signature and size, for example by rearranging instructions or the insertion of junk (e. g. *no-op* commands).

Using *packers*, obfuscation techniques can be applied automatically. It is even possible to implement *anti-debugging traps*, allowing the program to detect whether it runs in a controlled environment or debugger, preventing researchers from investigating its behaviour. As anti-virus vendors keep adding new signatures to their databases and uncloaking programs, executables are often repacked to avoid detection. Repacking occurs on average every 11 days [9], although other malware (e. g. *Citadel*) advises to do this more often. Some distribution software (*Zlob*) was even shown to repack malware for every infected host separately.

Packing can be very effective, to the point that bots stay online for years.

Other obfuscation techniques involve the encryption of communication flows, which may be as straight-forward as utilizing HTTPS instead of unencrypted HTTP, but may also constitute custom encryption software. Additionally, communication may be obfuscated in such a way that it blends in with normal traffic to evade detection by signatures [80].

2.2.2.3 *Interference*

Since botnets are used to make money, competition between botnets is prevalent. As computers get infected with malware from different botnets, botmasters may opt to remove other bots from an infected machine, not having to share profits with other botnets. The *SpyEye* malware, for instance, detected and removed software from the *ZeuS* bot, which is its most prominent competitor [53]. Furthermore, malware may also disrupt and interfere with anti-virus software, to prevent detection and removal of the software.

2.2.3 *Recent Developments*

In addition to the techniques described in the previous section, botmasters are actively exploring new defensive techniques to stay ahead of researchers. In this section, we take a brief look at developments seen in recent years or are expected to emerge in the near future.

2.2.3.1 *Social Media as Control Channel*

Botmasters have started using other communication channels besides the HTTP, IRC and P2P channels described in Section 2.2.1. One of these new channels is the usage of social networks. The global success of social networks in our lives depends on them being online 24/7 and their prevalence ensures traffic does not stand out from normal traffic. This makes them ideal replacements for C&C's of botmasters looking to increase stealth of their botnets [30].

2.2.3.2 *Mobile Botnets*

Miscreants are also looking to expand their networks by expanding to mobile devices. One of the first mobile botnets was formed with the SymbOS/Yxes worm [2], but as mobile operating systems become increasingly more advanced and prevalent, mobile botnet capabilities are also increasing dramatically, both on iPhones [44] and Android devices [41].

Although mobile devices are typically more protected and harder to infect through official application stores, *rooted* or *jail broken* devices, as well as devices using unofficial stores, are more vulnerable to an attack. Second-hand phones may also contain malware installed by the previous user, or in some cases, the seller.

Apart from traditional malware capabilities, phones also provide new capabilities for malware. In 2012, a botnet was uncovered that used fake mobile apps to infect users and send expensive text messages or call premium numbers, generating a revenue of somewhere between $1,600 and $9,000 per day [97].

TECHNIQUE	DESCRIPTION	RESILIENCE	STEALTH	EXAMPLE
Centralized (IRC)	Bots connect to IRC to receive commands	-	-	Agobot [25]
Centralized (HTTP)	Bots pull commands from HTTP sites	-	+	Bobax [92]
Peer-to-Peer	Commands are passed from bot to bot	+	+	Nugache [57]
Random	Commands are sent randomly	++	-	Conficker C [43, 62]
Domain flux	Use of multiple domain names	+	+/-	Torpig [55]
Single fast-flux	Use of multiple IP addresses	+	+/-	Storm [57]
Double fast-flux	Single fast-flux + fast-fluxed DNS	++	+/-	Warezov/Stration [93]
Obfuscation	Hiding application flow	+	+	Zeus [6]
Communication encryption	Hiding contents of messages	+	+	Rustock [10]
Binary encryption	Hiding contents of application	+	+	Mariposa [51]
Instant repacking	Changing application signature on the fly	+	++	Zlob [9]
Interference	Disrupting other botnets or software	+	+	SpyEye [53]
Social media	Using social media as communication channel	-	+	Koobface [58]
Mobile botnets	Using mobile devices as bots	-	+	SymbOS/Yxes [2]
Javascript botnets	Implement botnets using Javascript	-	+	*experimental* [76]

Table 2.3: Overview of different botnet techniques and their effect on resilience (how hard it is to take down the network) and stealth (how detectable the botnet's operations are) of the botnet, based on the text of Section 2.2.

2.2.3.3 *Javascript Botnets*

Instead of using Javascript to start drive-by downloads or otherwise infect a victim's computer, the Javascript capability of a browser can be used to perform malicious activities: as long as the browser stays on an infected page, it has almost complete control over the browser.

Utilizing (for instance) advertising networks, it is possible to create a large, but volatile, botnet of browsers having a web page open with a specifically designed script, allowing commands to be issued [76]. Such botnets may be used to perform DDoS attacks, send email spam, perform distributed computational tasks [83] or other malicious activities and are hard to detect due to the absence of malware and lack of erratic behaviour.

2.3 DETECTION & AFTERMATH

In the previous section, we have discussed methods used to increase the resilience and stealth of botnets (see also Table 2.3). These are just two of many factors characterizing botnets – other factors including *churn* (measuring the rate of growth), effectiveness, population, complexity, intended usage and infection methods [3, 32, 36, 60]. However, resilience and stealth are crucial from the botmaster's perspective for allowing to continue to operate the botnet undisturbed from law enforcement and other parties looking to take down a botnet.

However, researchers are in this same game, looking for methods allowing them to detect and identify the presence of bots and C&C servers in a network. In this section, we will discuss some of the most prevalent methods for botnet detection. We will furthermore discuss methods to take down the botnet and identify its herder.

2.3.1 *Detection Techniques*

Note that some techniques have only proven themselves in a controlled environment. Actual figures on the usage of detection techniques are unknown, as organizations are secretive about their usage of security technologies.

Detecting a botnet may be based on three phenomena: malicious traffic from bots, control traffic from C&Cs or its presence on a file system. The former is typically used to identify a server running bot software within a network, while the latter is used to detect bot software on a specific server.

In recent years, detection methodologies have been successful in detecting botnets with a low false positive ratio. However, most detection techniques focus on a specific characteristic of a botnet and do not focus on protocol and structure independent methods. This makes most detection methods obsolete as soon as a botmaster opts to change his approach. Combining information from multiple phenomena is therefore prudent for successful detection.

2.3.1.1 *Network Traffic*

When identifying bots inside a network, any of the data sources available can be used, including full packet tap data and DNS lookup information, but also *IP flow data*. The latter effectively is a summary of traffic patterns occurring on the network, grouping connections and listing timestamps, number of transferred bytes and protocols used [54].

SIGNATURE-BASED In a *signature-based* detection system, signatures of traffic generated by known malware and botnets are used to identify erratic behaviour in a live network. Although effective for some purposes, using signature-based detection requires the knowledge of the signatures of malicious use in advance. If signatures are lacking for a specific botnet variation, or an entirely new kind of botnet, detection is not possible. This may be mitigated by automatically generating signatures, although extensive training sets are still required [40]. However, the usage of polymorphism by malware authors (Section 2.2.2.2), may make signature-based detection soon obsolete.

ANOMALY-BASED A second technique for identifying botnet traffic is the usage of *anomaly-based* detection systems. Instead of relying on signatures, suspicious behaviour is detected by deviations from normal system operation. Gianvecchio et al. have shown that they were able to distinguish bots from normal users in internet chat networks [19]. Similar principles can be applied to network traffic, possible anomalies including high traffic volumes or suspicious packet sizes.

Researchers have proposed several approaches to developing anomaly-based systems. IRC-based botnets have been identified by the combination of bots operating in the same IRC channel and them generating traffic levels custom to *port scans* [5]. Others have written systems that combine information from automated *intrusion detection systems* (IDS) to recognize typical traffic patterns of hosts trying to infect other hosts [23].

Classifying the malicious traffic generated by bots in a botnet (e. g. DDOS or spam traffic) may also be used for the identification of infected hosts. For instance, a method designed by Brodsky and Brodsky relied on the assumption that botnets tend to send large amounts of spam in a relatively short period of time [8].

2.3.1.2 *Control Traffic*

Although identifying bots may be useful for network administrators, law enforcement is typically looking to locate the central C&C server to effectively take down a botnet. Although analysis of malware-infected hosts may reveal the C&C server, it may be more effective to directly

identify traffic with the C&C, which could directly lead to, but at the very least provide pointers to, the centralized server.

Using IP flow information, Gu et al. have shown to be able to identify control traffic when nodes respond to the same query within the same time frame and similar content [24]. Karasaridis et al. designed a detection scheme to calculate the distances between monitored flow data and a pre-defined flow model [29].

One may also be able to identify control traffic by scanning DNS requests for domain names seemingly generated by domain generation algorithms (Section 2.2.2.1). DGAs will typically generate multiple invalid domains, which are queried by the infected host, resulting in *non-existent domain* (NXDOMAIN) responses from the DNS server. The detection of clusters of NXDOMAIN responses with characteristics of randomly-generated domains, could indicate the presence of bots trying to communicate with their C&C [1].

2.3.1.3 *File System Indicators*

Although most detection methods are geared towards using network data to identify botnets, it may not always be possible to obtain this kind of data: different servers may be sharing the same internet connection, servers may have been confiscated for other reasons or organizations want to identify a C&C by regular scanning their infrastructure. For this, file system indicators (e. g. file names, sizes and contents) may be used to identify a botnet instead [100].

2.3.2 *Analysis & Take Down*

When a botnet has been found, it is prudent to estimate its size, impact and, more importantly, its command structure. This information can then be used to plan a successful botnet take down, or at the very least, provide statistics on botnet usage. There are two ways to shut down a botnet: removing discovered bots or taking down the command and control structure by shutting down the C&C.

The former is a slow and tedious operation. Even relatively small botnets are highly resilient against such a take down and require high amounts of bots to be removed, even in the rare case that the entire network topology is known. This makes such take downs only effective in the early stages of a botnet [60, 61]. Above all, control over every bot is often lacking. A more effective and feasible way is to take down the C&C, or at least to disrupt its communications. However, since there is no C&C structure in a P2P botnet, this is no attack vector in these botnets.

In this section, we will discuss the possibilities to take down a botnet. However, it should be noted that most available techniques for law en-

forcement are limited by law, although it is unclear where the line should be drawn. In 2010, the Dutch National High Tech Crime Unit (NHTCU) has taken down the BredoLab botnet and infiltrated the C&Cs to send a message notifying the victims of the infection (see also Section 4.1). This has been the subject of discussions regarding the legality of such practices [81] and eventually led to the Dutch police receiving a Big Brother Award by digital civil rights movement Bits of Freedom [70].

The NHTCU was formed in 2007 as part of the National Crime Squad. At the time of writing, the unit consisted of approximately 90 investigators (of which 50% digital and 50% tactical).

However, although the Dutch government is currently (2014) in the process of legalising counter-hacking [72], Dutch law enforcement already has competencies for seizing equipment, conducting house searches, arresting and prosecuting suspects, forensic analysis, etcetera [94].

It should also be noted that even a successful take down does not mean it is definitive. Botnets are built to withstand disruption; after a take down, another criminal may step up and continue operations, making the phenomenon of botnets hard to stop [7].

2.3.2.1 *Infiltration*

An obvious way to learn several aspects of a botnet's activity is to infiltrate the botnet by joining the C&C channel [47]. One could mimic the behaviour of an actual bot and record any information it can observe, which may include bot identities and transactional information.

However, for this technique to work, one must be able to join the C&C channel, which can only be achieved in IRC-based botnets. Furthermore, infiltration does typically not allow for the take down of a botnet.

2.3.2.2 *Sinkholing & Takeover*

Since some botnets use domain generation algorithms for their communications, the botmaster has to ensure that the calculated domains are registered in advance. When the algorithm leaks or is deduced from analysis of the malware, it is possible to predict which domains will be used by the botnet and be registered before the botmaster is able to do so [55]. This allows all traffic destined for a C&C server to be forwarded to another server that observes and records all traffic it receives, a process also known as *sinkholing*.

When no backup mechanism is built-in, this may cause the botnet to become dysfunctional. After careful analysis of the communication protocols, the bots may not even recognise a takeover has occurred and follow commands that may go as far as showing messages to the user about their infection or even removing themselves from the infected hosts [89].

2.3.2.3 *P2P Index Poisoning*

Due to the complexity of creating a P2P protocol, some P2P botnets adopt existing P2P file sharing protocols. Such protocols often employ file indexes that are shared amongst participants in the network, allowing participants to search for files in the network.

To receive commands in a for botnet use adapted P2P protocol, a botmaster publishes their commands in this file index, under a specific time-dependent search query. Bots search the file index for this query based on the current date and receive the associated command [61].

However, anyone is able to publish information into this index. Similar to the sinkholing attack, by analysing the source code and behaviour of the malware, queries can be predicted. This can in turn be used to publish bogus information under these queries, severely hindering botnet communications [34].

2.3.2.4 *Sybil Attack*

The name Sybil *is derived from a book with the same title, which is a case study of a woman diagnosed with multiple personality disorder.*

Another method to disrupt communications within a P2P botnet is by inserting large amounts of fake nodes or *sybils* into the network. Instead of forwarding C&C commands, they reroute or block these [14]. In order to be effective, sybil nodes must remain active and participate in underlying P2P protocols (contrasting index poisoning), but do not have to respond to the botmaster's commands to participate in malicious activities.

2.3.2.5 *Blacklisting*

Using blacklists, it is possible to blacklist a C&C server in firewalls, although this will typically not affect the operations on a global scale.

When a query or peer blacklist is maintained within the network, this may also be effective for P2P botnets. Such blacklists may be residual from the underlying P2P networks. Legitimate peers do not process queries in the query blacklist and will not process messages from and to peers in the peer blacklist [60].

2.3.2.6 *Take Down & Seizure*

As mentioned before, when the central command structure has been identified, providers and registrars may be summoned to take down the malicious service. While the former suggested take down methodologies are typically not possible for law enforcement due to legal difficulties, a *notice-and-takedown* (NTD) request is an effective method, that may also be accompanied by seizures of the servers hosting the malicious content, allowing for further analysis [89].

2.3.3 *Botmaster Traceback*

After botnet operations have been brought to a halt, law enforcement is interested in tracing the responsible botmaster. The botmaster is a central player in the botnet economy (see Section 2.1 and especially Figure 2.1), although law enforcement is also interested in the other parties. However, not all parties perform illegal activities (e. g. finding exploits, building botkits and providing bullet-proof hosting may not be illegal [73]), while others are only minor players and no significant enablers of malicious activities (e. g. affiliates or botnet salespeople).

Tracing the botmaster is not always easy, as he typically does not connect directly to his C&C, but uses *stepping stones* to obfuscate his real location [48]. Stepping stones are a chain of SSH servers, proxies, bouncers and any other network redirection services that make manually tracking the traffic a tedious and time-consuming process [63].

Even if all technical challenges would be resolved, all providers in the world would cooperate and the true source IP address would be found, this does not mean that the relevant machine can be physically located. Smartphones, open wireless access points, the TOR anonymization network and public computers may hinder traceback efforts [63].

There are other, non-technological methods to locate miscreants. Traditional detective work may for instance follow the money, e. g. identifying *money mules* (people offering miscreants to use their banking account temporarily) and follow relations from there [20, 37]. However, for complex organizations, even this may not always be possible and the botmaster may never be identified.

2.3.4 *Digital Forensics*

Digital forensics is concerned with analysing disk images, memory dumps, network packet captures and so forth [18]. Digital forensics finds its origins in scientifically proven methods towards the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence [91]. It could provide additional information about the botmaster, which may lead to a successful traceback, may provide (additional) evidence for a possible conviction or may at the very least be relevant for later research on the usage of botnets.

A successful digital forensic investigation constitutes six primary stages [4]. The first is preparation, aimed at maximizing the ability to collect digital evidence and minimizing overhead [49]. The second stage is the initial response to a suspected security incident, aimed at formulating the investigation and collecting initial information about the malicious activity itself. The third stage constitutes data collection in preparation of the fourth phase: data analysis, which is the most complex and time con-

suming phase [31]. The final two stages of a digital forensic investigation constitute the presentation of findings and the closure of the incident, entailing critical review of the forensic investigation and disposal of evidence and preservation of information [4].

Examples of popular forensic tools include Forensic Toolkit (FTK, accessdata.com) and EnCase Forensic (encase.com).

Tools are constantly being developed to support researchers in the third and fourth stages. Several software packages exist for creating and analysing disk images and other data sources. Developing tools for analysis is challenging, as the data is diverse and there are huge amounts of data to be analysed [18].

Though most data will not be relevant to the case, constituting operating system files, libraries and (often) pornographic media, some miscreants leave personal traces, such as their user accounts, traces of login sessions, application traces (e. g. *dotfiles*, see Section 3.1.2.1), emails, browsing cache and chat sessions [66]. Most tools are able to scan the file system and extract the useful bits of information from the filesystem, although further technical analysis, such as file system analysis, memory analysis and inspection of the boot sector could be used to discover even more useful leads for an investigation [66, 74].

2.4 CONCLUSION

Until recently, a botmaster was required to build his own software and recruit his own bots. Nowadays, a service-oriented and diverse underground economy has formed, with miscreants specializing in their expertise of e. g. finding exploits, building software and infecting hosts. This economy is advantageous for botmasters, as they can now primarily focus their efforts on monetizing their botnets.

Botmasters require a communication channel between them and their bots, while remaining resilient against take down and retaining stealth to prevent detection. Different techniques have been developed to keep the botnet active as long as possible, including the usage of obfuscation and interference techniques.

These techniques hinder detection by law enforcement, researchers and other interested parties. Although some detection methods have been successful, finding the botmaster remains a tedious and time-consuming process. This also means that law enforcement is left with evidence it could not link to a miscreant, stalling prosecution.

In the next chapters we will discuss a novel method that will use some of the traces left by a miscreant on a machine to detect commonalities within a large dataset. This will aid law enforcement finding relations between machines, helping them link these to a common miscreant.

3

METHOD

The purpose of this work is to aid law enforcement with their efforts of taking down botnets by trying to find identifying characteristics in confiscated machines. We will use this information to identify a common miscreant and common infrastructure between these machines. In this section, we will discuss the characteristics we use in our approach and how these characteristics lead to a result that can be used by investigators.

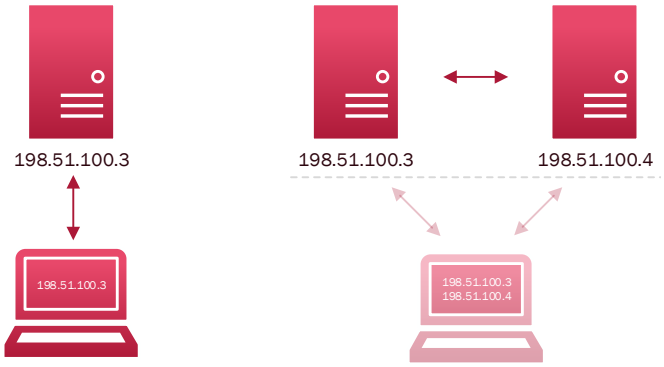
3.1 CHARACTERISTICS

In this section, we will discuss properties we may find within machines that have been used for botnet operations (i. e. C&Cs). However, the same characteristics may also be found in disk images of other machines (unrelated to botnets). Therefore, characteristics will be described in a broad sense, allowing the re-use of our findings in other investigations not specifically tailored for C&C investigations.

The type of characteristics we will include in our approach, depends on the system setup, which primarily entails the operating system used. Many C&C servers (e. g. ZeuS, SpyEye and BredoLab [6, 53]) utilize a Unix-like operating system. Therefore, we limit our scope to these operating systems, discarding other operating systems. Broadening our scope to include e. g. Windows-based disk images would involve accounting for a whole different server architecture, which is left as future work.

In the following discussion, we have created a distinction between characteristics identifying the *networking* operations and characteristics that identify the *operations* of the machine itself. This distinction illustrates a clear difference between the two groups: the first group of characteristics is used to find relations between machines where a direct link already exists (e. g. one machine connected to another), while the second group is used to infer new relations based on unified behaviour, such as similar software or configuration.

Though this distinction may seem subtle, it provides important information on the reliability of the created relations. No interpretation of data is required in the group of networked operations (e. g. machines have either connected or not), while we need to contextually interpret the data before creating a relation with the group of system operations.



(a) Direct relation between two known machines

(b) Indirect relation through an unknown third machine

Figure 3.1: Examples of the two types of relations that can be inferred from the login history. As shown in (a), there is a direct relation between two machines in the dataset, where one has logged in on the other. In (b), a relation is inferred between two machines because a third machine has logged in on both.

Listing 3.1: Example of entries in the `wtmp` file, showing that three different IP addresses logged in to this machine.

```
username pts/3  203.0.113.15  Thu Apr 3 09:01 - 15:01 (06:00)
username pts/3  198.51.100.45 Wed Apr 2 19:51 - 19:53 (00:01)
username pts/3  203.0.113.99  Wed Apr 2 09:05 - 16:33 (07:27)
username pts/3  203.0.113.15  Tue Apr 1 11:45 - 16:35 (04:49)
```

3.1.1 *Network Operations*

All of the characteristics listed as network operations, have been obtained from several log files across the system. In general, the occurrence of the IP address of one machine in the log file of another, indicates a relation between the two machines.

3.1.1.1 *Login History*

The Unix operating system retains several files containing records related to the login history of users. One particular example is the `lastlog` file, which contains the source (i. e. IP address) and date and time of the previous login of every user. Each user has one entry in this file, even if the user has never logged in to the system. This information may be presented upon a subsequent login (e. g. "Last login: … from …"), which may alert the user if he doesn't recognize the previous login attempt.

A second location in which login history is stored, is the `wtmp` file. It contains a full log of all previous logins to the system. This file also contains the IP address and time of login, but also the time of logout. Its primarily purpose is auditing and accounting of system states.

The `wtmp` file can be viewed using the `last` command on most Linux-systems.

Assuming system operators (i. e. botmasters) do not clear these files, we can use these in two ways to create a relation, as shown in Figure 3.1. Firstly, we may have two disk images of which one has logged in to the other (where the client has been used as a stepping stone for the server) and we can match the IP address in the history file with the IP address of the server, resulting in a direct relation. For instance, would we encounter a login history similar to Listing 3.1, we can conclude a direct relation between our machine and `203.0.113.15`, `198.51.100.45` and `203.0.113.99`.

Secondly, we could see an indirect relation between two distinct disk images, where the same IP address has logged in, without actually having the machine belonging to this IP address in the dataset. For instance, would we encounter two similar login histories, e. g. `203.0.113.99` occurring in both machines, we can use this as a relation.

3.1.1.2 *SSH Host History*

Remotely accessing the terminal of a Unix system is usually done using SSH (*Secure Shell*) [68]. It provides a secure channel between the user (client) and the system (server or host), that allows commands to be issued to the machine. The security is partially based on *public-key cryptography* by utilizing a *key pair* that is uniquely generated for every machine.

A key pair consists of two components: a public component that is advertised and a private component that is kept secret. Information encrypted with the public component can only be decrypted with the private component.

The connection between the SSH client and server is secured using this key pair: the public key is advertised by the server to connecting clients, while the private key is kept secret to the server. A client encrypts some random information with the public key, while challenging the server to successfully decrypt it using the private key. To ensure that a future connection to the same address is with the exact same server, the client stores the public component in a *known hosts* file (see Listing 3.2). If the public key changes, the client refuses to connect to the server as this is recognized as a possible *man-in-the-middle attack*. For this to work properly, it is assumed that the public key is unique for the server and no two keypairs with the same public component exist.

Although this feature is essential for securely connecting to SSH servers, it provides useful information for investigative purposes. The known hosts file shows to which servers the client has connected to, which we can use in two ways: either by the IP address stored as identification with the public key or by the public key itself. Note that this information is very similar to the login history of the previous section, although this information is 'reversed', i. e. the login history shows which clients have connected *to* the current host, whereas the SSH host history shows to

Listing 3.2: Example of an entry into the known hosts file [66].

```
192.168.1.103 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA4RjYT7I+f49
    a70MY6S/1YT63tCUDOz5Yt4N1RpeOtiVQ9BBG23YO3/N+byoLs7dLPxuR
    +YR/Gg57YmNiF5xu2EHM0ewoKL2Y3adjFTUoDhJTYaKg7u3fqb1
    TxSERlEeHP2K1i0LaDxl6uDLOvNqy4H/f2XtRuwXMDViysmjWPR0
    dCaTbj+D3jbO1Ucg5MIqRlctcM/AH3kF50xiBfamgINtsiJ/jDz9
    IEDUCXaN9/D+LBT/sw3PS8kWghYlyIyHNLruWQtp3heFekw2P/1
    eKqtaZHZuzf5QzUUyvLQhdQ60fOQmr3JN0jWIruIgRTSDORACHkVy8
    YQUBbw2WbaC/Hw==
```

which hosts was connected *from* the current host. See also Figure A.1 (page 62) for an example.

This also means that we can use this information in a similar manner as with the login history to create a relation, only 'reversed'. We may have two disk images, where one has served as the client and the other as the server, and we can match the public key known to the client with the private key (or IP address) of the server. Additionally, we may also have two disk images both having connected to the same unknown third server.

3.1.1.3 *Authorized Keys*

While normally, users would authenticate themselves to a system using a password, users may also opt to set up a personal key to authorize themselves. Authentication using authorized keys has several advantages, including the removal of the need of sending a password over the network and virtually eliminating the possibility of attackers brute forcing the password as keys are typically much longer than passwords.

Similar to the known hosts mechanism, this also uses public-key cryptography: the user retains his private key, while the public key resides on the remote host. To prove access to the system, the server sends a challenge (i. e. some random information) encrypted using the public key to the client, requiring the client to decrypt it using its private key.

Again, this can be used in two ways to create a link between two machines. In some cases, we may encounter a public key stored on the first machine (e. g. because it was generated there), which is authorized for a second machine. In other cases we may be able to match key authorizations across machines and find two machines that have the same public key authorized. The former may indicate that the user was using the first machine to log in on the second, while the latter probably indicates that a third machine was used to login on both.

3.1.1.4 *IP Addresses*

When a server is used as C&C, it will probably store some information entailing to the IP addresses of victims. These IP addresses may be present in a database, stored in files or present in log files. A second way in which IP addresses could be used on a machine is to connect multiple machines together, e. g. for the sharing of databases.

Based on this information, we can create relations between machines – if one victim's IP address is present on multiple machines, or one machine has the IP address of another machine configured, we can assume that these machines have a relation with each other.

3.1.2 *System Operations*

The characteristics in this section are used to create inferred relations, based on unified behaviour between machines.

3.1.2.1 *Personal Configuration Files*

So-called *dotfiles*, files and directories with names starting with a dot ('.'), are used in Unix-like operating systems to personalise the user experience. These files are located in the user's home directory and may contain settings and usage history for different applications. Even when the user is not aware of the presence of these files (they are hidden by default) they may contain useful information that can be used to create a profile of the user. This information may include, for instance, shell command history, recently opened files or even application passwords [66].

We expect these files to be highly personal, but as described in Appendix B, attempts to use these dotfiles to create a detailed profile have failed due to a multitude of reasons, most importantly because we were unable to obtain a substantive labelled comparison dataset. We have therefore reverted to comparing the exact contents of dotfiles across systems, ignoring any characterizing properties that may be hidden within the contents themselves.

3.1.2.2 *Distinct Usernames*

When users are created on a machine, either during system setup or picked by the system operator in a later stage, they are given a name. This username will typically contain some description of the intended use of the account, e. g. the name of the intended user (`john`), the name of the service (`majordomo`) or the name of a hosted site (`cnn-com`).

Although some usernames will be the same across different unrelated machines and care must be taken to not mix up the creation of incor-

rect relations, we do consider similar distinctive usernames on different machines a relation.

3.1.2.3 *Database Configuration*

When a miscreant installs their (C&C) server, he will probably be using a similar set-up for similar activities. It is therefore also probable that a similar database configuration is used. As inspecting the data in the database itself is rather difficult (as this depends on the specific version of the software used), we will focus on a 'quick win' with the names of the databases: if two systems have similarly named databases, we will assume a relation between the two.

However, we must be careful using this relation. Because the same miscreant will probably use different purposes for different machines in his infrastructure and other miscreants may use a similar set-up for similar purposes, this characteristic is likely to generate false positives, depending on the dataset used.

3.1.3 *Not Implemented Characteristics*

In addition to the characteristics listed above, some other characteristics could be used to link two or more machines together. Unfortunately, there was not enough time to implement them all, leaving them as future work.

3.1.3.1 *Email Addresses*

Similar to matching IP addresses across machines, we could scan for email addresses in configuration files (perhaps from the system operator) or for addresses from victims. Unfortunately, scanning for these takes a long time and many similar email addresses are expected to be found on different unrelated systems, due to e. g. software developers leaving their email address in a copyright or support notice. It is possible to create a proper filter for this, but it would take too much time.

3.1.3.2 *Stolen Login Details*

We could also look for similar stolen login details, which may be obtained from an infected machine. Such details may reside on different machines belonging to the same botnet (e. g. because the botmaster has a redundant set-up), be available in different botnets (e. g. because the victim was infected multiple times) or bought online by different botmasters. Not all of these could be considered a (strong) relation, but making a distinction between these is hard. Even harder is actually detecting the stolen login details, as these are not stored in a consistent format, thus making it a substantive effort to properly match these across systems.

3.1.3.3 *Installed Software*

One could look at the software the system operator has installed; some users prefer to use other software than others. Furthermore, a machine that has for instance been purposed as C&C will probably have other software installed than a simple webserver. Due to differences in system setup, differences between package managers, but above all since one miscreant probably utilizes multiple systems with different purposes, this characteristic did not make it off the drawing board.

3.1.3.4 *File System Structure*

System adminstrators will probably have their own preferences when setting up a system. Although Unix enforces some principles and there are numerous 'good' practices, the contents of a user's directory and the location of custom packages and files may reveal some information about the administrator. Additionally, some parts of the file system may be encrypted and the choice of encryption scheme and vault location are dependent on the administrator's choices. Unfortunately, the creation of such a profile in a system-independent way that is uniquely enough to consider it a relation, is very hard and thus not included.

3.1.3.5 *Domain Names*

Based on the domain name configured to be used by the machine or domain names the machine has connected to, machines could also be related to each other. Although this is similar to the IP address matching, this also allows for a fuzzy approach, where two similar domain names (e. g. `example1.com` and `example2.com`) are also matched. However, creating a match based on this information has not been done, as domain names are harder to detect than IP addresses. Furthermore, creating a reliable fuzzy filter would have been a significant effort.

3.2 CREATING AND VISUALISING RELATIONS

In the previous section, we have defined which characteristics we plan to use to create relations between machines based on the owner and infrastructure of the machines. To perform our analysis, a specialised tool has been developed. In our proof-of-concept implementation we perform our analysis in three steps: accessing the disk images, obtaining all data we would need and finally creating the actual relations between machines based on this data. In this section, we will briefly discuss the proof-of-concept implementation. Appendix A contains a more detailed description and also carefully explains the complexity of this process.

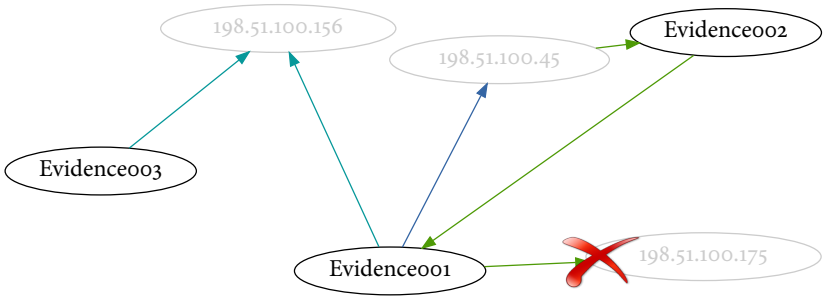


Figure 3.2: Example of a graph before unnecessary nodes have been removed. Most of the relations shown are through an unknown host, but since the same IP address is found on multiple known machines, a relation is still created. However, unknown host `198.51.100.175` does not provide a new relation and is therefore removed.

3.2.1 *Analysis*

The access to the disk images involves the use of different command-line utilities to extract disk images, which are effectively compressed byte-by-byte copies of full disks, to a file system that is accessible from the operating system as a normal disk. Though this may seem straightforward, the process involves customised software accommodating for all different file structures, that was specifically developed as part of this research.

In the second step, we use a combination of file system structure, file contents and other system utilities to obtain the data required for a proper analysis of the characteristics, including data such as usernames, dotfiles, SSH host keys and authorized keys.

The final step involves the actual analysis of the data. We have chosen to represent our relations using a (multi-)graph structure. Every piece of information identified as relevant by a specific characteristic, is added as an edge between a machine and another machine part of our dataset (direct relations) or with an unknown third machine (indirect relations). In the latter case, we consistently identify these unknown third machines with the same node, allowing the reliable creation of indirect relations.

An example of this is shown in Figure 3.2. In this figure, `198.51.100.45` has logged in to `Evidence002`, which was determined by the login history characteristic (a green edge indicates login history and is probably derived from the `wtmp` file). This IP address has also been found by the SSH host history characteristic on `Evidence001`. The combination of these two forms an indirect relation between the two machines. A direct relation was also found: from the login history of `Evidence001`, it was derived that `Evidence002` logged in to this machine.

After this matching has completed, we clean up unknown nodes not used to form new relations. For instance, we remove node `198.51.100.175` from the graph as it only has relations with one known machine. This leaves only nodes useful for finding relations in the graph.

3.2.2 *Limitations*

Though the implementation as illustrated in the previous section seems straightforward, there are certain limitations imposed on the creation of relations to reduce the amount of false positives (i. e. the amount of incorrectly matched machines).

For instance, some of the characteristics rely on IP addresses. However, IP addresses can be reassigned over time and multiple machines may have had the same IP address. We accommodate for this in two ways: we create a list of alias IP addresses to ensure we always refer to the same machine with the same address and we never match IP addresses from multiple investigations to each other, although the latter restriction could be lifted if investigators feel they are affected by this limitation.

Additionally, some of the characteristics require specific alterations to ensure proper and plausible results. These are briefly discussed below and summarized in Table 3.1. The limitations have been explained in more detail in Appendix A.

LOGIN HISTORY may create relations between machines that have no apparent relation to each other. We focus on finding relations between machines within one investigation and it seems plausible that these relations may be intrinsic to our dataset. For instance a hosting provider may have logged in to these systems himself to test their proper working. Although these relations may be of interest for investigators, especially when looking for relations across investigations, we exclude all edges that represent only one login attempt for the purposes of this research.

USERNAMES may be system-created and may therefore not correspond to the miscreant's intentions and preferences. We exclude all system-created users to prevent matching on these accounts. Furthermore, some common usernames, such as `test` and `admin` will be excluded. The list of common usernames we are using is limited and other generic usernames may become apparent over time, but these can easily be added.

DOTFILES may also be system-created (e. g. provided as an example) and not all dotfiles should therefore be used to create relations. We identify and exclude these common dotfiles by manually sifting

CHARACTERISTIC	DESCRIPTION	LIMITATIONS
Login history	Other machine logged in on machine	more than once
SSH host history	Machine used to login on other machine	
Authorized keys	Machine authorized for login on other machine	
Unique dotfiles	Similar configuration files across machines	non-standard only
Distinct usernames	Similar usernames across machines	non-standard only
Database names	Similar database names across machines	non-standard only
IP addresses	Same IP addresses occur on two machines	not used in analysis

Table 3.1: Summary of the implemented characteristics and their limitations.

through all dotfiles found in the dataset, although in the future a heuristic or a database of default dotfiles should be created.

DATABASE NAMES that are too common will be excluded, including system databases and databases created by common software. This is only a limited list that should be extended in the future or improved by using a heuristic.

IP ADDRESSES occur in many files and due to memory constraints, it is impossible to add all found IP addresses to the graph. Therefore, we limit the analysis of IP addresses found in system files to only include relations with already known IP addresses from other characteristics.

Based on a small sample, we have been able to determine that most of the IP addresses found in log files were sometimes created by logging indirectly caused by one of the other characteristics (and thus redundant) and sometimes finding new relations based on connections of (possible) victims of the miscreant. However, regardless of the restriction we imposed, please note that we have excluded IP addresses from further analysis, as will be detailed in Section 4.2.

3.2.3 *Visualisation*

Our implementation relies on a simple in-memory graph structure, which is sufficient for our purposes. The graph uses weighed relations to ensure the visualisation will show nodes with strong relations closer to each other. Simple weights are assigned, using the amount of encountered relations of one type as the weight of the relation with that type.

Although we would like to use a dynamic visualisation to represent our results, due to incompatibilities between our internal graph structure and visualisation libraries, we generate a static image using the *GraphViz* library instead. This is sufficient for our analysis.

Future work could enhance the visualisation of data, for instance by using a network visualisation that would allow investigators to easily and dynamically extract useful information [50, 59]. The use of weights could also be improved by assigning weights based on the reliability of a characteristic instead.

The internal structure could also be enhanced by using a graph database, allowing the storage of our relations, in addition to the raw characteristics data [64]. This would also enable the use of dynamic querying for information by the investigator, e. g. allowing to search for specific IP addresses found in other investigations [27].

4

RESULTS

As described in the previous section, we have devised a method to find relations between confiscated machines. In this section, we will describe how we have validated our results by applying it to a dataset of machines obtained during the BredoLab investigation. We will discuss our findings in three distinct analyses: a quantitative study to determine the reliability of our results, interviews with investigators at the NHTCU on their perception of the method and results and finally a more in-depth analysis of some of the clusters detected using our method.

4.1 DATASET

Our validation was performed at the Dutch National High Tech Crime Unit (NHTCU). Over recent years, the NHTCU has collected a wide range of disk images relating to botnets and other malicious activities. One of their more sensational cases involved the take-down of the *BredoLab* botnet [89]. In this case, the NHTCU seized control over the C&C infrastructure of BredoLab, which consisted of 6 different machines, each with their own specific function. As discussed in Section 2.3.2, commotion was caused as the NHTCU notified victims through the botnet before the servers were taken down.

The takedown severely crippled the BredoLab botnet and led to the arrest of Armenian Georg Avanesov, who was sentenced for 4 years in prison. Unfortunately, not all perpetrators were caught and some parts of the botnet were still managed from Russia [82].

The machines of the botnet were hosted by a bulletproof hosting provider known as John Datri, who rented these machines in turn from LeaseWeb, one of the largest hosting providers in Europe. As there was sufficient evidence suggesting that the other machines rented out by Datri were also involved in illegitimate business, the entire infrastructure of Datri was seized, consisting of 143 machines in total.

As many machines had more than one disk, 206 disk images have been captured and still remain at the NHTCU, measuring 23 terabytes of compressed *EnCase* formatted files. Unfortunately, not all disk images contained a readable volume system. In these cases, the *Master Boot Record* (MBR) was corrupted or in an unknown format, the disk was empty and did not contain a valid volume system or contained only unknown and encrypted volumes. This excluded 56 disk images from our dataset, leaving 150 readable disk images.

Furthermore, we excluded an additional 46 disk images running the Windows operating system, as we have only devised characteristics for Unix-like operating systems (see Section 3.1). The remaining 104 disk

images from 90 different machines have been analysed for characteristics and be used to find relations.

We were unable to obtain metadata of the 53 excluded machines. Note that our results will be incomplete and for instance may be missing machines crucial in finding relations between machines that are part of the dataset. However, the dataset's quality has not degraded significantly, as it is also imaginable that the dataset would have been seized incompletely and our method must accommodate for this situation.

4.2 EXCLUDED CHARACTERISTICS

Despite only matching IP addresses found in files with IP addresses already found by other characteristics (see Section 3.2.2), we have entirely excluded IP addresses as a characteristic from our analysis. As shown in Figure 4.1, it heavily cluttered our results. Though in part caused by our graphics software, this is also caused by the amount of redundant relations, i. e. several machines have multiple similar relations through different IP addresses. Secondly, with other characteristics, we were able to filter out relations inherent to our dataset, but we were unable to perform this filtering with the same level of control with IP addresses. Lastly, it was not possible to verify any of its data based on other characteristics, as the data is already based on information provided by other characteristics.

While this causes the loss of some information, the exclusion of IP addresses made our overall result more reliable and significantly easier to analyse. However, future work should look into making this characteristic more usable. For reference, the total amount of IP addresses and the IP addresses in our filtered approach have been included in Table 4.1.

4.3 QUANTITATIVE ANALYSIS

From the 90 machines in our dataset, we were able to find information in 84 of the machines. After removing machines that were not part of any cluster (i. e. machines with no relations other than with themselves) 45 machines remain, forming 10 clusters (with two clusters of size 2, four clusters of size 3 and for sizes 4, 7, 8 and 10 one cluster each).

In Table 4.1, a breakdown of relations for every characteristic is provided. These numbers account for duplicates, although it should be noted that the amounts of total found distinct usernames and unique dotfiles are probably too high because we have not inspected the uniqueness of all of the encountered usernames and dotfiles. The second column of the table represents the total amount of information found by each characteristic, the third lists the amount of unique relations between different machines.

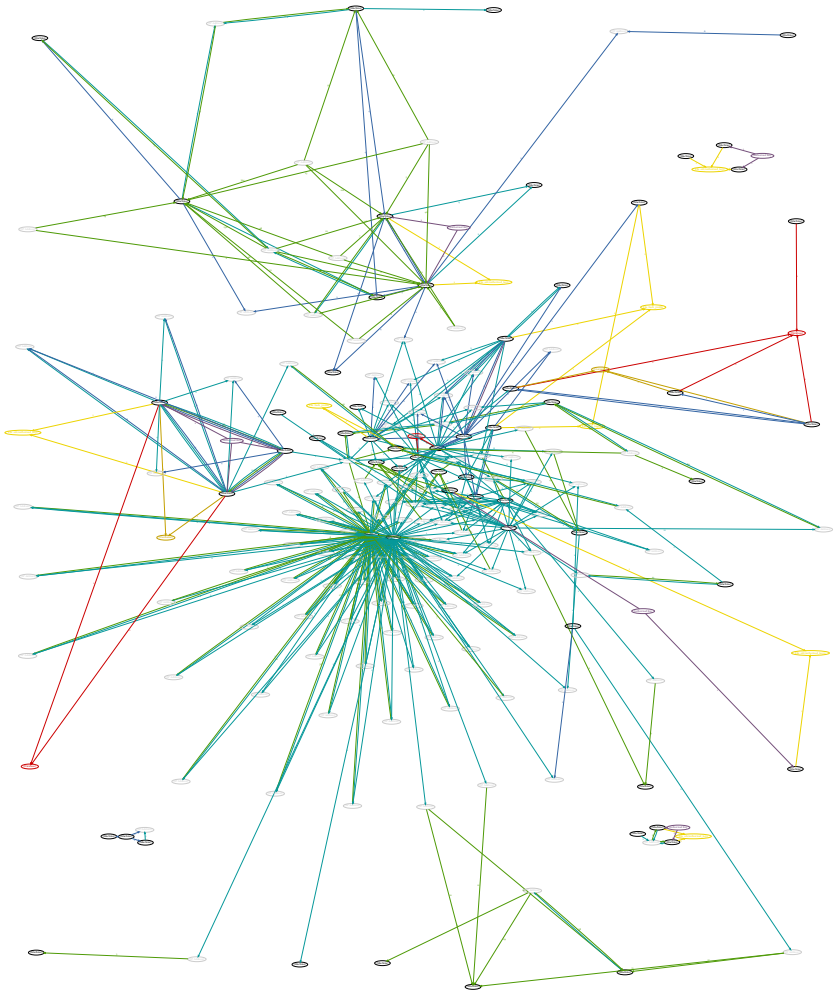


Figure 4.1: Anonymised graph containing results from our IP address analysis, illustrating the clutter created by this characteristic. This figure is only intended as an illustration and is available in full size from `thesis.ralphbroenink.net`.

CHARACTERISTIC	TOTAL FOUND	TOTAL RELATIONS
Login history	190 (32)	51 (22)
SSH host history	240 (44)	44 (27)
Authorized keys	29 (21)	14 (13)
Unique dotfiles	579 (81)	18 (16)
Distinct usernames	49 (20)	5 (5)
Database names	145 (29)	8 (8)
IP addresses (filtered)	390 (70)	290 (41)
IP addresses (total)	6,005,084 (74)	455,920 (72)

Table 4.1: Total number of found pieces of information and the amount of actual relations, both not counting duplicates. In braces, the number of machines concerned is included. The numbers of total found unique dotfiles and distinct usernames are probably too high, as not all default values that were not part of a relation have been filtered out. As described in Section 4.2, the total and filtered amount of IP addresses are only included for reference and have not been used in further analysis.

Although dotfiles have been found on most of the machines in our dataset, we were only able to use these files to link 16 machines together. This can be explained by the presence of temporary and often-changing files across systems, for instance the `bash_history` and other log files, that do not provide any additional information about the actual user that can be matched across systems as the exact usage patterns will change across systems (e. g. dates and times and order of execution). Other characteristics were able to find more relations. The SSH host history, for example, was able to identify 44 relations with 27 machines in total and the login history provided links with 22 machines in total.

4.3.1 *Validation*

Since we have been unable to obtain a labelled comparison dataset, we could not objectively determine the validity of the relations found. However, we can compare relations with each other and establish some level of validity this way.

We are providing two important statistics relating to this analysis in Table 4.2. First, we have counted the amount of relations between two machines that was not confirmed by another characteristic. High values are explained by the total absence of relations of specific characteristics in some of the clusters, e. g. the lack of SSH host history to confirm relations based on login history.

CHARACTERISTIC	DIRECT CONF.	TRANSITIVE CONF.
Login history	23 (45%)	38 (75%)
SSH host history	19 (43%)	38 (86%)
Authorized keys	14 (100%)	14 (100%)
Unique dotfiles	10 (56%)	17 (94%)
Distinct usernames	5 (100%)	5 (100%)
Database names	5 (62%)	7 (87%)

Table 4.2: Overview of confirmed relations. A relation is considered confirmed if at least one other characteristic has indicated the relation. The second column indicates relations on a machine-to-machine basis; the third column shows the numbers when accounting for transitivity of the relations (e. g. when there's a relation between *A* and *B*, and a relation between *A* and *C*, a relation between *B* and *C* is considered confirmed).

Secondly, we have counted the amount of relations that are doubtful by the lack of transitive confirmation. This means that a confirmed relation between two machines (say, *A* and *B*) and a confirmed relation between one of those and a third (*A* and *C*), implies that any relation between the other two (*B* and *C*) is also confirmed. Additionally, all relations between three machines have been considered confirmed if at least two different characteristics are involved in creating this triplet.

From this, we can easily see that distinct usernames and unique database names seem to be highly trustworthy, but their numbers are too small to draw any useful conclusions. Moreover, relations created by these characteristics have been specifically tailored for this dataset and are likely to provide less accurate results for other datasets.

Authorized keys, based on the numbers, have a high confirmation rate, i. e. always occur with one other relation. This is expected, as authorized keys are used to login on other systems and it is likely that other characteristics find traces of these login attempts. This also confirms the suspicion that this characteristic should provide high-quality relations, as the authorization of a key is an explicit action by the system operator and keys are uniquely generated.

Similarly, login history and SSH history should in principle provide reliable results as this characteristic is based on semi-unique data. We can confirm this expectation based on the statistics. Finally, the high confirmation rates of unique dotfiles are surprising and the reliability of this characteristic will probably decline after more machines are added that feature similar dotfiles but are not related to other machines.

4.3.2 *Efficiency*

While indexing the entire dataset of 104 disk images could take up to two hours, with the opening and reading of the disk images taking the most time; the analysis typically completes within twenty seconds.

However, when IP addresses are searched within files, indexing takes a lot longer, up to five days of constant running (and crashing due to memory issues). Would we use unfiltered results from the IP addresses found in files, analysis would require a database preparation that takes anywhere from ten minutes to five hours and the analysis itself would take more than one hour.

4.4 INTERVIEWS WITH INVESTIGATORS

To further validate our results, we have conducted four interviews with both tactical and digital investigators at the NHTCU. All had more than a year working experience at NHTCU and three had a familiarity with the dataset. They have been asked to give their opinion on whether the method could in principle deliver proper results, whether the results seem reasonable from their experience and whether the results could be used in future investigations. We have also discussed the visualisation of our results and how this could be improved in the future.

Questions were asked in an informal setting and answers were not always provided as structured as one would like. In the following sections, a summary of all answers is provided.

4.4.1 *Discussion on the Method*

The investigators were presented with our problem statement and how we used the seven characteristics to find relations between machines in our dataset. They were asked whether this method would deliver high-quality results, how we could improve results and whether they could think of another method that would deliver better results.

Could this method, in principle, deliver qualitative results?

There was general consensus amongst the interviewees that the method itself could deliver useful results, especially for 'cold' data such as sets of disk images. However, it was noted that the tool would inevitably miss the big picture and will not notice machines that look awfully similar (in a general sense, e. g. based on file system structure, system design and usage patterns), but are just different enough for the analysis to not recognize any relations, although an investigator would be able to notice this.

The term 'cold data' is typically used to indicate data that is not commonly accessed (i. e. archived). We use it in a slightly broader sense, denoting the machine is not operating and can not be observed.

CHARACTERISTIC	RELIABILITY	ROBUSTNESS
Login history	++	-
SSH host history	++	-
Authorized keys	++	+/-
Unique dotfiles	+	+
Distinct usernames	-	-
Database names	--	-
IP addresses	+/-	+/-

Table 4.3: Overview of perceived reliability and robustness (i. e. the difficulty for the miscreant to mislead or mitigate this characteristic) of the characteristics based on interviews with experts.

How could (other) characteristics be used to deliver improved results?

Respondents noted that some of our characteristics (e. g. the selection of usernames) are highly dependant on the dataset used and may not apply to other datasets as well, possibly leading to false positives. Not all characteristics where therefore considered reliable – especially the database names characteristic seemed to raise questions. Additionally, the robustness of the characteristics (i. e. the difficulty to hide information by the miscreant) was perceived as being low for most characteristics, as summarized in Table 4.3.

One suggested solution for increasing the reliability would be to create a large comparison dataset, consisting both of 'clean' installs of all operating systems and actual data retrieved from multiple investigations. While the former would be used to exclude common files across operating systems (e. g. default dotfiles and system libraries), the latter would be used to develop an heuristic recognizing common usage patterns across systems (e. g. commonly installed software and common usernames).

Similarly, some of the subjects discussed the use of IP addresses for analysis. Although these have been excluded from our full analysis as they were too omnipresent, it was noted that we could create a list of common IP addresses and exclude these. However, as the exact nature of the IP address would still be unknown, this could lead to confusion about how the relation was realized.

Additionally, since this research focussed on Linux-based machines only, Windows machines were left out of the equation. However, finding relations between and with those machines was deemed to be interesting. Furthermore, our efforts were focussing on machines functioning as server, but even more (and different) interesting details could be found on client machines (e. g. home computers).

CHARACTERISTIC	COLOUR	
Evidence item	black	Evidence item
Unknown machine	grey	Unknown machine
Login history	green	logged in on
SSH host history	blue	logged in on
Authorized keys	purple	authorized for
Unique dotfiles	yellow	contains
Distinct usernames	dark yellow	contains
Database names	red	contains
IP addresses	cyan	contains

Table 4.4: Legend of node and edge colours used in the graphs in this section.

How would another method deliver similar or better results?

Jabber and OpenVPN are client-server applications (as opposed to P2P applications) that are commonly used in some cultures for secure communication.

Suggestions for entirely different approaches were not provided, although we discussed the detection of anomalies in the filesystem structure (e. g. comparison of filenames and file contents) and the use of more case-specific information, such as Jabber and OpenVPN configuration and logs. Information could be enriched by using historical DNS data (if available) to translate hostnames to IP addresses, further enhancing the analysis of machines not part of the dataset.

A final suggestion was to create more relations and more precisely assign weights to each of the relations and allow the investigator to decide which of the relations is useful for himself. For instance, limiting results by only matching IP addresses within the same investigation, would potentially hide useful information.

4.4.2 *Discussion on the Results*

Graphs in this chapter have been anonymised. During interviews, graphs without this anonymisation have been shown.

After been shown an image similar to Figure 4.2 and discussing some of the clusters in more detail (see also Section 4.5), interviewees were asked about their conceived reliability of the graph and whether they recognized some of the usage patterns shown in the graph. Additionally, we discussed whether a similar implementation would be of help during (large) investigations and whether it would be used in the future.

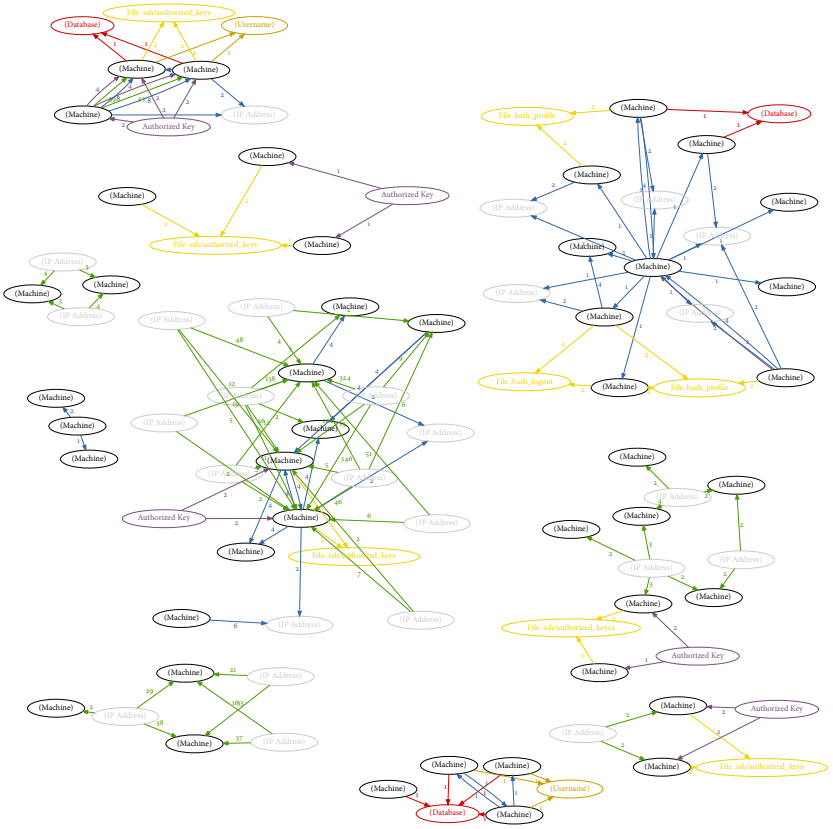


Figure 4.2: Anonymised graph providing an overview of the topology of the full BredoLab dataset residing at the NHTCU. Each edge colour represents a different characteristic (see Table 4.4), black nodes are evidence items, grey nodes are unknown machines and coloured nodes provide information about the specific relation. It is possible that not all machines shown have malicious intent. This figure is provided as an overview; in Section 4.5 we will discuss this graph in more detail, featuring some larger excerpts from this figure in Figure 4.3, Figure 4.4 and Figure 4.5. Nevertheless, this figure is also available in full size and full colour from `thesis.ralphbroenink.net`.

Could you estimate the reliability of the results? Do you recognize some of the patterns emerging from the graph?

Determining absolute reliability was hard for the interviewees, as they were unaware of the exact nature of some of the machines. However, some of the patterns emerging from the graph were recognized and confirmed some of the facts known or expected by the subjects.

How do you think that this approach could help with large investigations?

Most of the investigators were enthusiastic about the potential uses in investigations. Noting that the results would never be used or seen as absolute truths, they would function as the starting point for further investigation, prioritising some machines out of several. One of the interviewees was surprised by the amount of relations found and even went on and look at the BredoLab dataset after the interview, being fascinated by some of the results, although he did not have the time yet to discover new leads based on these results.

The interviewed investigators did note that they would like to see the roles of every machine within a cluster of machines. It was easy to see which of the machines was in the center, but did not indicate the function of every other machine in the network.

Will you use this method in future investigations?

Most interviewees noted that the tool would probably not be used within only one investigation, as chances are slim that a large and unknown dataset would be obtained in the future. However, with some small modifications, the method could be used to compare machines obtained in multiple investigations and possibly enable more in-depth investigations on data otherwise considered as waste. It was added that the results could be valuable for identifying a miscreant, as our method indicates machines that are most likely to have interesting traces on them.

4.4.3 *Discussion on the Visualization*

Some comments on the visualisation were provided during the interview, as it was used as a method to present our results. Our final question addressed this visualisation directly, requesting some comments on the visualisation of our data.

How could the visualisation be improved?

Although interviewees agreed that a network graph was the best way for visualising the data, the amount of information was overwhelming,

especially considering that more machines could be added over time. A suggestion concerned only showing one relation between machines when displaying an overview of the results, while hiding the exact nature of each relation, but still allowing the investigator to 'zoom' in and view the exact nature of the relations.

Adding other dynamic elements to the visualisation, e. g. the ability to re-arrange items, would also greatly improve the usability of the graph and resolve the generation of chaotic images such as shown in Figure 4.1. It could also be used to show additional information (e. g. all related IP addresses) when requested by the investigator. A final suggestion was to improve the use of edge weights and assign weights based on the characteristic rather than one solely based on the amount of relations found of a specific type.

4.4.4 *Conclusion*

Based on our interviews, we have seen that investigators from the NHTCU agree on our method, although not all characteristics were perceived as being reliable. Several enhancements have been suggested, including the use of a file system anomaly detection and characteristics focussed on the Windows operating system. Furthermore, our existing characteristics could be improved by using a heuristic based on a large comparison dataset, consisting both of clean installations and actual data retrieved from multiple investigations.

Investigators immediately saw potential uses, although the method would not be used within one investigation (as large datasets are rare), but to detect patterns across investigations instead (that are too complicated to perform manually). Its use would further be improved by using dynamic visualisation techniques and adding additional information to the figure, such as the actual intended use of the machine.

4.5 IN-DEPTH CLUSTER ANALYSIS

As a final verification step, we have analysed the clusters and checked whether what is shown in the graph corresponds with reality. We have done this by inspecting the disk images of the machines themselves, in an attempt to analyse the nature of the clusters and the different roles of each of the machines. Although all machines part of a cluster have been analysed briefly, we will discuss three of the more interesting clusters in more detail below.

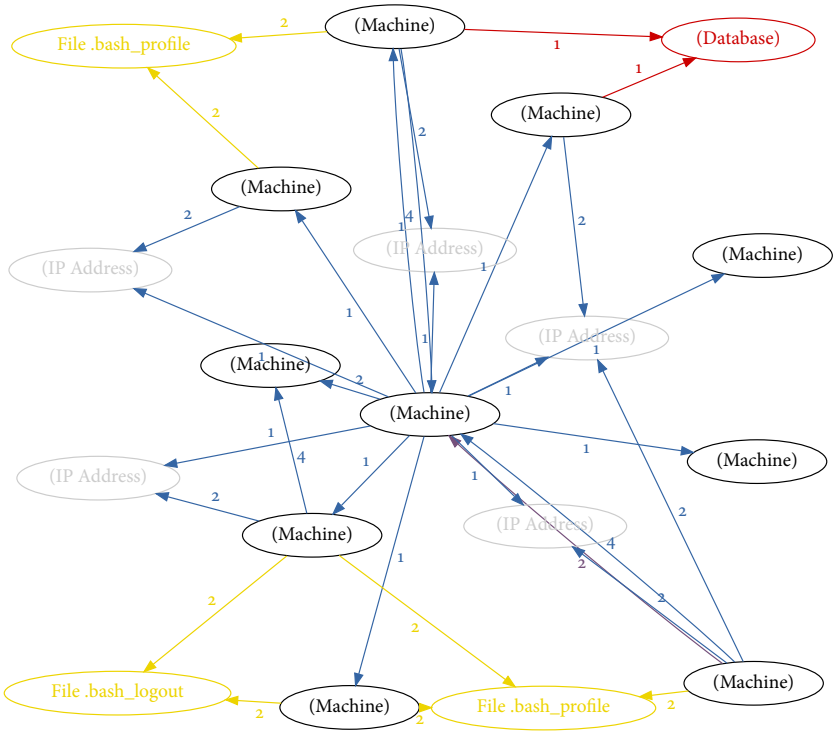


Figure 4.3: Excerpt from Figure 4.2 featuring a cluster with machines belonging to the BredoLab botnet.

4.5.1 *BredoLab Cluster*

Remember from Section 4.1 that our dataset was obtained during the take down of the BredoLab botnet.

In Figure 4.3, all machines part of the BredoLab botnet have been shown as being part of a single cluster. From this graph, it can be derived that there is one central node, which seems to have been used to log in on multiple other machines. This is confirmed by traces of an OpenVPN server installation found on this machine. It is remarkable that one machine was authorized to directly login to the central machine, partially defeating the purpose of the proxy software. The second machine seems to have been used within the botnet for administrative purposes, although the usage pattern may also indicate that it was used as a proxy for the proxy server. Investigators are therefore more likely to find traces of botmaster activity (e. g. the IP address of the miscreant) on these two central machines.

The other machines each contain scripts related to communication (Jabber) or contain other C&C related scripts. We observe a similarly named database in two machines. This is caused by similar software installed on both machines related to harvesting login details. These two machines are even more strongly related than implied by the graph, as

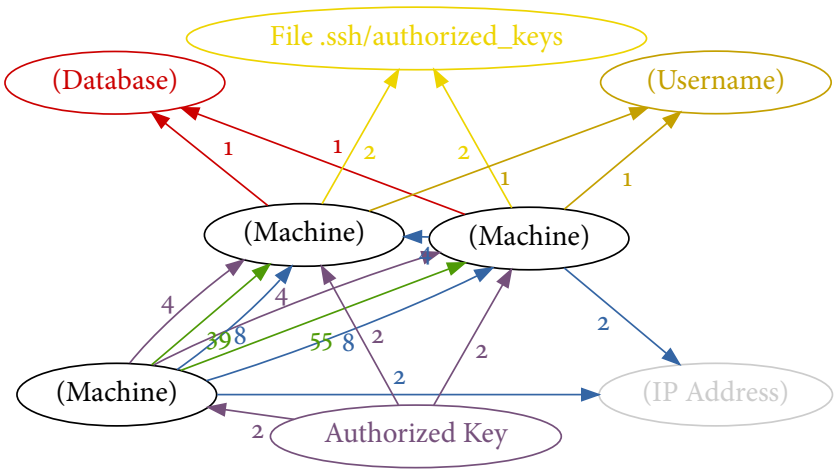


Figure 4.4: Excerpt from Figure 4.2 providing an example of a strongly connected cluster, used as C&C infrastructure.

one machine connects to the second to access its database (this relation would have been found if we included the IP address characteristic).

Another noteworthy observation of this cluster is the absence of any relations regarding login history. Upon further inspection, it becomes clear that on several machines the `.bash_profile` and `.bash_logout` files are set to delete some system files upon login and logout (respectively), including the `lastlog` and `wtmp` files used to store login history. It is therefore surprising that there's any information to be found at all; the miscreant seems to have forgotten to remove the `known_hosts` file. Additionally, it is clear that this attempt to hide his information, has inadvertently created a relation in itself, as some of these `bash` files match across systems.

4.5.2 *Strongly Connected C&C Cluster*

An example of a small and strongly connected C&C cluster is shown in Figure 4.4. There is a relation of every type and it is therefore very likely that the relations found are correct. It is easy to see that there is one central machine that has been used to log in on the two other machines. Upon further inspection, it appears that this machine has proxy software installed, confirming this suspicion.

The two other machines have a similar configuration (e. g. database names and usernames) which emerges from both the graph and the actual disk image. These machines are probably used as C&C server for the same botnet, accommodating a multitude of phishing and other malicious

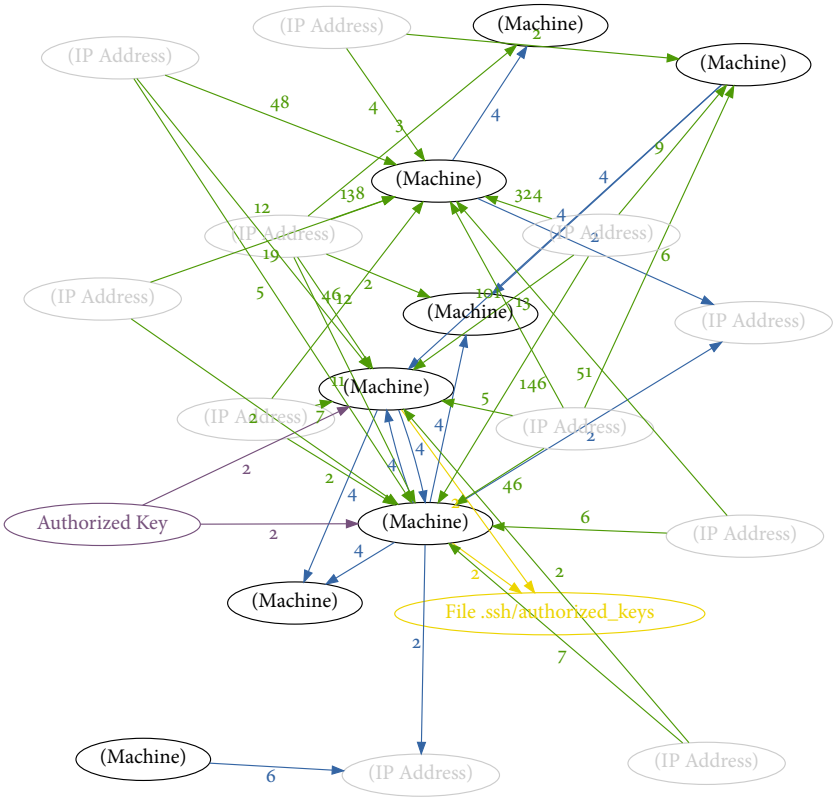


Figure 4.5: Excerpt from Figure 4.2 featuring a cluster with machines containing encrypted vaults.

sites. The unknown machine (of which only an IP address is known), is probably part of the same botnet but hosted with another provider.

4.5.3 *Encrypted Cluster*

Figure 4.5 shows another interesting cluster, the second-largest in our result set. This cluster consists of machines all containing LUKS encrypted vaults, limiting the amount of useful forensic data that could be obtained from these servers. It is therefore remarkable that the operator of this cluster has failed to hide his tracks in other significant ways. For instance, he has logged in on all machines various times from different IP addresses, possibly other machines under his command or perhaps even his own home computer. Interestingly, the same key was set up to be authorized to log in on two machines, indicating the operator deliberately added this information to his cluster.

The almost-orphaned machine (in the lower left corner of the image) does not appear to belong to the cluster as the installed software differs

CLUSTER	MACHINES	FALSE P.	AVG. RELATIONS
BredoLab cluster	15 (10 + 5)	0 (0 + 0)	4.3
Strong C&C cluster	4 (3 + 1)	0 (0 + 0)	8.3
Encrypted cluster	19 (8 + 11)	2 (1 + 1)	6

Table 4.5: Overview of the findings in Section 4.5, listing the total size of the analysed clusters (known + unknown machines), the amount of false positives (i. e. machines that should not have been included) and the average amount of relations of every known machine.

significantly and no encrypted LUKS partition was present. However, both servers have connected to the same IP address, indicating there has to be some relation between the two. Upon further inspection, the orphaned machine seems to have connected to numerous SSH servers, indicating it is probably part of the infrastructure of some bullet-proof hosting provider (perhaps John Datri, but may also be another provider renting servers from John Datri) or from another larger network.

4.5.4 *Conclusion*

The findings of our in-depth analysis of some of the clusters has been summarized in Table 4.5. It shows the size, the false positives and the average number of relations of every cluster we investigated. Although we have been searching for false negatives (i. e. machines that should have been included but are not), we could not identify missing machines.

In our analysis, we have seen that we could identify important nodes within a network based on the nature of some of the relations. Although not all relations have proven to be reliable, we have also seen that most of the relations provide useful insights and information that would otherwise have remained hidden. We should note that (in this case) it is better for our method to include too much machines than missing some machines, as it is relatively easy to manually identify incorrectly included machines, while missing machines may never be found.

Furthermore, most of the data we found, especially in the encrypted cluster, was found by virtue of the carelessness of the machine operator. The robustness, i. e. the difficulty of hiding traces we used, is low and miscreants could easily avoid getting profiled when taking care to leave no traces, which they can accomplish with a few simple steps. For instance, miscreants could remove all the contents of their home directories and never connect directly to their machines.

5

DISCUSSION

Due to their multitude of purposes and ability to stay under the radar, botnets have grown to be the Swiss army knives of the underground economy. Botmasters are using specialized technologies to increase their stealth and resilience against take-down, requiring special attention from law enforcement. In our attempt to aid law enforcement in their investigations, we have created a technique involving the detection of relations based on seven different characteristics commonly found in C&C disk images, although our method has been developed to be applicable to machines used for other purposes.

Although unfortunately, we had trouble establishing a fully objective figure on the reliability of our method, based on interviews with investigators at the NHTCU, we have shown that our method produces plausible results. We have also shown by analysing some clusters created by our method, that our results match with most of the data itself. Furthermore, investigators have indicated that our results provide added value to the current investigative methods and could be of aid in future investigations.

However, our method also has its shortcomings. Although 93% of machines contained information we could have used, only 50% of the machines contained information that could be related to other machines. It is probable that our dataset was not complete and that a more complete dataset would have contained a higher percentage of relatable machines, though full and complete datasets are rare in the forensic world. It is also likely that many machines would not be relatable at all.

Some of the characteristics we have used (i. e. database names and usernames) are highly dependent on the dataset used. Their quality will degrade as the size of the dataset increases or machines with other purposes are added. Our network-based characteristics (i. e. login history, SSH host history and authorized keys) perform much better in this respect. However, these depend on the miscreant leaving crucial information on his machines, but he could just as easily remove or change this information, which would hide his attempts from our approach. Unique dotfiles seem to be the most future-proofed, as their connection between machines is less obvious and could identify relations not intended by the miscreant. Nevertheless, our method in identifying common dotfiles had significant shortcomings and should be enhanced.

While we have attempted to focus our efforts on finding common characteristics identifying the managing miscreant, we were unable to find such identifying information in a consistent manner and instead only

have been able to identify common infrastructures. However, this could also greatly help investigators find this identifying information.

5.1 FUTURE WORK

Our dataset came from one source and we are therefore unable to establish whether it is a proper representation of data found in other places. In addition, our dataset was not properly labelled, preventing us from providing a more elaborate objective validation and we were unable to verify relations not found by our approach. To ensure our method works properly in all scenarios, our method should be tested against other data sources, preferably also from other parties than law enforcement to allow this method to be used in a broader context (e. g. crime prevention).

An important issue with the characteristics we chose is the low level of robustness by relying on the carelessness of the miscreant inadvertently leaving personal traces. If the miscreant removes these traces, our method would not be able to find such relations at all. More robustness could be gained by actually comparing the system layout, which is a characteristic much harder to circumvent. Furthermore, adding more characteristics (including those discussed in Section 3.1.3) would increase detection rates, most notably for machines running the Windows operating system.

Moreover, decreasing false positive rates of individual characteristics could be accomplished by creating a large pool of machines with clean installations of various operating systems to be used as a comparative dataset. Additionally, our method could be further enhanced by modifying the characteristics to include heuristics about the commonality of some of the values encountered in the entire dataset, discarding common information for the creation of new relations.

In our approach, we have put efforts to identify relations within the data of one specific investigation and some of the characteristics have been tailored for this purpose. Much more value could be obtained from comparing machines across investigations, as patterns could emerge that would otherwise have remained hidden, while relations within one investigation are more likely to be found anyway by the investigator.

Lastly, the proof-of-concept implementation generates a static graph with all relations being present in one image. This gets cluttered very easily as more characteristics and more machines are added. Investigators indicated (see Section 4.4) that a dynamic interactive visualisation, hiding and showing information based on the view port requested by the investigator would be of greater help.

As discussed in Section 3.2, our proof-of-concept could be further improved by utilising graph databases, allowing simple querying for information and the storage of additional relations that could aid in the dynamic visualisation of the graphs. Furthermore, additional statistics

could be added to each of the nodes, including the installed software or the expected role of a machine within a cluster.

5.2 RESEARCH QUESTIONS

At the start of our research, we stated the following four research questions that needed to be answered before finding an answer to our problem statement. In this section, we will shortly discuss the answers we found.

Which identifying characteristics can be found inside a C&C?

In Chapter 3 we defined seven characteristics that could be used to identify which C&C machines were managed by the same person or have been part of the same infrastructure: login history, SSH host history, authorized keys, IP addresses found in files, unique dotfiles, distinct usernames and database names. We split our characteristics in two main categories: the first four were based on networking operations, providing direct relations between machines, while the latter three were system operations, used to infer relations that did not already exist.

Based on our quantitative analysis, we have seen that not all characteristics provided data on all machines. For instance, we found dotfiles on most of the machines, while SSH host history was only found on less than half of the systems in our dataset. However, the host history proved to create twice as much relations. Nevertheless, all of our defined characteristics have shown to find information within our dataset.

How can these characteristics be used to reliably identify a common miscreant or infrastructure?

Although we could not fully determine the reliability of our approach, we have shown that by identifying direct and indirect relations between characteristics, we could identify several clusters with commonalities within a large dataset. Investigators at the NHTCU indicated that they could use this to identify critical nodes in an infrastructure and use this to focus efforts on finding traces in these nodes.

How can results be presented to the investigator?

Based on interviews with investigators, we have identified that a graph-based visualisation would be the best way to visualise relations. We did not explore all possibilities, but a dynamic visualisation, for instance adding and showing relations based on the investigator's requested level of detail, would allow the deduction of critical information, while keeping the visualisation clean and simple to use. A further step would be to introduce a query language allowing the investigator to request specific

information from the graph or combining different sources of information within the same visualisation.

How efficient is using these characteristics when searching through large datasets?

A critical requirement was to ensure our method would complete within reasonable time and would not burden investigators with additional tasks. In our proof-of-concept implementation, we have separated responsibilities between different modules, allowing the search to complete within several hours and not requiring any input from investigators, making it a very efficient tool for searching through large datasets and discovering new leads.

5.3 CONCLUSION

In this work, we have been looking for an answer to reliably determine which C&C machines were managed by the same person or have been part of the same infrastructure. Our verification dataset had its difficulties, preventing us from doing the analyses we would like to perform. Our dataset has never been properly labelled before and metadata was lacking. This made it hard to fully fathom the data, hindering us in our research and requiring us to devise other methods than initially envisioned to create and verify results.

Nevertheless, by using seven characteristics, we have been able to find relations between machines part of this large dataset. Our approach has shown to be useful in aiding investigators identifying critical nodes in infrastructures set up by miscreants, which in turn could be used by investigators to identify the responsible miscreant. It has been received enthusiastically by people at the NHTCU, though a modification to perform analyses across investigations, something that is too time-consuming to perform manually, would be welcomed.

However, given the ingenuity of the underground economy, it seems only a matter of time before miscreants increase their ability to hide from and deceive investigators. Continued research on new methods is required to stay ahead of miscreants, our method being a solid start in using an entirely new approach in identifying miscreants.

A

PROOF-OF-CONCEPT IMPLEMENTATION

An important part of this research involved the development of a proof-of-concept implementation that can be used to search for characteristics and find relations based on these characteristics. Our method, that has been described in more detail in Chapter 3, forms the basis of this implementation; the results based on this proof-of-concept have been described in Chapter 4. This appendix describes the complexity of some parts of the implementation and provides the justification for some of the design choices that have been made and limitations that have been imposed in order to obtain a working prototype.

Our implementation consists of three main modules, working together to link machines and present this in an easy to understand way. The utility has been written using the *Python* programming language, with the use of external (command-line) utilities for some specific functionality. Each of these modules will be described separately in this appendix.

A.1 ACCESSING DISK IMAGES

The first module handles the access to the disk image. As mentioned in Section 4.1, all disks are enclosed in *EnCase* disk images; effectively a compressed byte-by-byte copy of a full disk. Opening the disk image is done using `xmount` or `ewfmount`, two Linux tools that expose the uncompressed data contained within the EnCase image. Using *The Sleuth Kit* (TSK), we identify the volume system type and the volumes contained within the image. Lastly, using a call to *mount* a virtual volume is created that can be accessed within the Linux operating system.

However, this process is full of *ifs* and *buts*, justifying the development of a stand-alone tool as part of this project. This tool, which has been open-sourced, enables easy programmatic access while attempting to transparently handle the differences between different disk image formats (including *EnCase*, AFF and `dd`). It is able to reconstruct an entire Linux file system, even if it was spread across multiple volumes, part of a RAID array or enclosed in a virtual LVM volume system.

The tool to easily mount disk images of different types is MIT licensed and available through `thesis.ralphbroenink.net`

Accommodating for all these different types of disk images and disk contents is a complex process, as there are several commands that must be executed in a specific order and their results must be parsed to properly mount the different volumes. This utility has been embraced and is regularly used by several digital investigators at the NHTCU, significantly easing their digital forensic work when using the Linux operating system.

A.2 INDEXING

In the second module, we use a combination of file system structure, file contents and other system utilities to obtain the data required for our third phase. Characteristics are extracted in their 'raw' form as much as possible, allowing a more sophisticated analysis and further parsing in the final step. Data is stored in a local relational MySQL database.

Each characteristic is implemented as its own independent plug-in, each gathering data for one specific characteristic. However, by allowing the order of plug-ins to be specified in a configuration file, dependencies and co-operation between plug-ins have been made possible. For instance, this allows for the collection of the operating system type before any other plug-ins are ran, which enables other plug-ins to skip analysis if it is not applicable to the specific OS.

The following data is collected for the final analysis from all disk images that contain a Unix-like operating system. It should be noted that the determination of where some of this data resides on the file system or how it should be parsed involved numerous iterations in the development process, as it is not always clear how this data is stored.

OPERATING SYSTEM is collected for all other plug-ins to determine whether it should be ran. Currently, all other plug-ins require a Unix-based operating system and are skipped otherwise. The detection of operating system also collects the exact version of the operating system (e. g. Debian or Ubuntu), although this is not used by other plug-ins.

STATIC IP ADDRESS is used for matching IP addresses found in other places with the machines in our dataset. Common places for the configuration of ethernet interfaces are indexed, ignoring local-only IP addresses. The matching must be done carefully, as IP addresses may be assigned to different machines over time.

SSH HOST KEYS are extracted from the `/etc/ssh` directory, which is used for the detection of the SSH host history (Section 3.1.1.2).

USERS are detected and their information is enriched based on three system files. `/etc/passwd` contains a list of all users, their ID numbers and home directories on the system. From `/etc/shadow`, the last password change date is extracted. Finally, the log file at `/var/log/lastlog` contains the last login information of all users (i. e. date and IP address). The user information is used to categorize user-specific data (e. g. their dotfiles and SSH history) with a user and for the distinct username detection (Section 3.1.2.2).

LOGIN HISTORY is found and extracted from the `/var/log/wtmp` file. It contains all previous login times and IP addresses and is used for the login history detection (Section 3.1.1.1).

DOTFILES are retrieved from all user's home directories for matching their contents across systems (Section 3.1.2.1). Filenames and full contents are stored.

SSH KNOWN HOSTS contain the remote IP address (although possibly hashed) and the public key. They are retrieved from a user's `~/.ssh/known_hosts` file, which is also used for the host history detection (Section 3.1.1.2).

SSH KEYS stored in `~/.ssh/` are indexed; the keys authorized to log in are retrieved from `~/.ssh/authorized_keys`, both used for the authorized keys detection (Section 3.1.1.3).

MYSQL DATABASE NAMES are found based on scanning common storage locations for MySQL databases. Each MySQL table has its own `frm` file in a folder named after the database. This information is used to derive the database and table names, while only database names are currently used as a characteristic (Section 3.1.2.3).

IP ADDRESSES are collected by running a regular expression (i. e. `grep`) over all files located in the default web directory and the default log directory (Section 3.1.1.4).

Collected data is stored in a (local) database for further analysis. This has two advantages: firstly, the indexer and analyser are implemented and ran separately, also allowing the separate development of both phases. Secondly, the indexer itself may crash mid-way and recover without losing most of its progress, as its state is persistently stored.

A.3 ANALYSIS & VISUALISATION

The third and final phase of the implementation consists of the analysis and visualisation of the data. As the relations between machines form a graph, the final iteration of the implementation used a graph as the internal data structure. The initial implementation used sets of related machines as its internal data structure, but this meant the loss of too much data. Similar to the indexing phase, the analysis uses plug-ins to handle each of its operations, each type of relation being covered by one plug-in, with the addition of plug-ins for the visualisation of the graph.

A directed multigraph is used internally, to allow the visualisation to show the specific relations in more detail.

Most of the plug-ins are straightforward: they create one or more nodes in the graph and create edges between them. For instance, the login history plugin creates edges from an IP address to a machine for each

Figure A.1: Example of a relation between two machines created by two different characteristics. In this example, the two *Evidence* nodes are servers of which an image is available and the node with IP address `192.168.100.91` is a machine that has not been imaged, but is referenced by the other two machines. Different edge colors represent different characteristics; the edge direction is only used for the visualisation of a specific characteristic and does not indicate a one-way relation (as relations are inherently reflexive).

`lastlog` entry and each `wtmp` entry. Creating relations this way has a great advantage over creating them for each characteristic separately and directly linking machines together. The latter would only find relations between two machines if their relation to each other is indicated by the same characteristic. This would, for instance, not find a relation that combines two different characteristics. An example of such a combination of characteristics is shown in Figure A.1.

Although the nodes are shared, the edges are not: each characteristic is responsible for their own set of edges. The weight of each edge is increased as more similar values are found for a specific characteristic. These weights are shown in the graph and used to increase the *cost* of a longer edge, i. e. the higher the weight, the shorter the drawn edge.

Some of the characteristics rely on IP addresses, although it is possible that the IP address of a single machine may change over time and the same IP address may be assigned to multiple machines. To accommodate for the former, SSH host keys (which are unlikely to change in the lifetime of a machine) found in other machines are used to generate a list of alias IP addresses for the same target machine. The latter is resolved by never matching IP addresses across multiple datasets: only IP addresses from the same investigation are matched to each other (however, this limitation may be lifted if investigators feel that results are severely limited).

Additionally, some of the characteristics required some specific alterations to ensure they provided proper and plausible results. These are listed below and applied to the results described in Chapter 4. Note that most of these limitations have been defined in an iterative development process and may sometimes be tailored to the dataset.

LOGIN HISTORY has proven to be too greedy in creating relations, as machines that have no apparent relation are also linked to each other. As our dataset originates from one bullet-proof hosting provider, it is plausible that the provider has logged in to these systems himself to test their proper working. Depending on the

exact nature of the police investigation, these relations may or may not be interesting. For the purposes of this research, all edges that represent only one login attempt have been excluded, although these could easily be included if necessary.

USERNAMES are only considered distinctive if their associated user ID is higher than 500 and lower than 2,000. By default, Unix-based operating systems reserve users with IDs below 500 (or 1,000) for system users, and users with higher IDs are user-created [99]. High IDs are also used for some system users, e. g. `nobody` is assigned user ID 65534. Additionally, some usernames have been excluded, as they are too generic or used by software packages, e. g. `test`, `admin` and `openvpn`. The list of usernames is limited and over time, other generic usernames may become apparent. These can easily be added when the need arises.

DOTFILES are only used for comparison if they are not standard, such as being provided as part of the OS or a software package. A database of such default dotfiles has been created by manually sifting through all dotfiles found in the dataset and determining whether they should be considered a default. This is obviously not a future-proof method, but was sufficient for this research.

DATABASE NAMES that are too common have been excluded. This includes the MySQL system table (`mysql`), but also common software, such as `jabber` and `roundcube`. Similarly to usernames, this list is likely to change over time and in other datasets, more database names should be excluded, either by manual blacklisting or creating a heuristic that determines the uniqueness of a database name.

IP ADDRESSES have proven to be omnipresent. Due to memory constraints, it is simply not possible to add all found IP addresses to the graph. Even filtering the IP addresses and only including IP addresses that occur on at least two distinct machines was too much for the test setup. Therefore, the plug-in only adds relations to IP addresses already in the graph. Optimizing the plug-in to allow the analysis of the full IP address dataset within reasonable time and memory is left as future work.

After all plug-ins have created their edges, additional plug-ins are responsible for some clean-up tasks and drawing the graph. For instance, one plug-in will remove the nodes that only have relations with one machine, leaving only nodes and edges that created relations between two machines. This ensures the graph is not cluttered with irrelevant information, as we are only interested in relations between known machines.

Finally, the graph is generated using the *GraphViz* library and exported in different formats for visualisation. Time has been spent in investigating the use of dynamic visualisation libraries such as *D3.js*, but due to incompatibilities between our underlying graph structure and the visualisation libraries, this involved too much effort and therefore no interactive graph was created. However, such libraries may improve the usability and understanding of the relations between machines. It would also allow for a dynamic level of detail, where both a general overview and a detailed explanation of each relation could be presented.

An additional improvement would be the use of graph databases such as *Neo4j*, allowing the storage of the relations we have found in a database, instead of only relying on raw data of the characteristics in a relational database. Although results could be roughly the same, it would allow for dynamic querying by the investigator and could prepare more relations that can be shown by the visualisation when requested.

B

ANALYSIS OF DOTFILES

Some users in Unix-like operating systems, use so-called *dotfiles* to personalise their experience. These files, with names starting with a dot (‘.’), hence their name, are located in their personal home directory and may contain settings for different applications. Even when a user is not actively using these dotfiles to tailor their experience to their liking, these files are present in most cases. They may, for instance, contain shell command history, information about previous logins, recently opened files or even application passwords [66].

The initial purpose of this research was to investigate whether a user profile based on these dotfiles could be created. Unfortunately, this has proven to be difficult. As none of the confiscated datasets at NHTCU were sufficiently labelled or substantially large enough to be used as training set, we had to obtain a labelled dataset in another manner. In this appendix, we will describe which attempts have been made to obtain such a dataset and why they were not sufficient to perform our research.

B.1 REQUESTING DOTFILES FROM OTHERS

Our first attempt to obtain a labelled dataset involved requesting these files from known people (i. e. friends, family, people at the NHTCU and people at the University of Twente) in a questionnaire-like setting. Each person was requested to provide multiple sets (from different servers), enabling us to discover relations within these sets and use these to create a profile of the user.

An email has been sent to these people asking them to run a simple script that grabbed dotfiles from their home directory (Listing B.1 contains the three commands that were requested to execute). This script automatically excluded all directories, files containing one of the phrases `pass`, `secret`, `key` or `auth`, and the files `.viminfo`, `.mysql_history` and `.netrc`, files which may be considered private by many people. Additionally, participants were warned when one of their files included phrases such as `key`, `auth` or `pass`. Participants were also invited to review the contents of the remaining files before sending them in, allowing them to remove privacy-sensitive data or use a pseudonym wherever possible.

Furthermore, participants were offered the possibility of encrypting their file using the researcher’s public key, with the assurance that their files would only be opened by the researchers on a machine not connected to the internet. Moreover, participants unsatisfied with running an arbitrary

Listing B.1: Commands that would need to be executed on a machine to collect and submit dotfiles.

```
curl -O http://thesis.ralphbroenink.net/grabber.sh
bash grabber.sh
curl -F nick=PSEUDONYM -F file=@broenink-thesis.tar.gz
    http://thesis.ralphbroenink.net/upload.php
```

script from an untrusted source on their system were allowed to create their own collection of files instead.

The files were collected using a custom-built private website that allowed participants to upload their files. To link multiple uploads together, they had to choose a nickname or pseudonym that would uniquely identify the dataset, but did not need to disclose the identity of the participant.

Despite all these facilities, respondents were very hesitant to participate, stating they were very careful about providing these files to others due to privacy and security concerns. They did not want to risk the chance of any private data being present in these files or the chance of these files being accidentally leaked. They did not trust the automated detection of private information or did not have the time and expertise to manually review their files. Moreover, even a manual review was not deemed sufficient by participants, as private details could still be accidentally overlooked.

The target group consisted of 145 people, with only 6 substantively responding. Dozens have actively (and a few aggressively) refused to participate. The 6 substantive responses included only one set of dotfiles each, not allowing any comparison between files from the same person. We were therefore unable to draw any conclusions from this dataset. Due to the amount and nature of refusals, continuing to collect dotfiles using this method and persuading more people in providing their dotfiles, was not deemed viable.

B.2 DOTFILES FROM GITHUB

Some people publish (some of) their dotfiles at social coding website *GitHub*. They do this partly to be of use to the community by showing their specific configuration and set-up and in part for having a central repository of their configuration that can easily be distributed across all of their machines.

After our first attempt to obtain dotfiles has failed, we have considered using these published dotfiles instead. However, these files have three serious issues limiting their use for our purposes:

1. Published dotfiles typically only contain application configuration and no application use, thus lacking any real data that would also be found on disk images. Publishing application use is considered bad practice because syncing these across machines is useless and will probably even break some applications.

2. People who publish their configuration files will probably have different usage patterns, especially as it is known to them that their files will be published. They will, for instance, refrain from using any private information in their dotfiles.

3. GitHub repositories will contain only one set of dotfiles per person and we are thus unable to compare their dotfiles across machines.

Due to these limitations, we would not able to draw any useful or substantial conclusions from this dataset. This option has therefore also been disregarded as being viable.

B.3 CONCLUSION

Although multiple attempts have been made to obtain a labelled dataset to create a profile based on dotfiles, these have either failed due to privacy concerns or due to significant issues with the dataset.

From this limited research it appears that (some) people are very concious about what they might store in their dotfiles and there seems to be consensus that most of these files, especially files containing application usage data, are to be considered private information. However, we were unable to prove this with any objective data. It is also likely that these files contain useful information for investigative purposes, but we can't say anything sensible on user profiling using dotfiles.

Investigators still have to review these files manually for abnormalities leading to the identification of the miscreant or finding relations between different machines. In this research, we have attempted to match identical files while identifying default files and create relations based on this information (see Chapter 3).

BIBLIOGRAPHY

PEER-REVIEWED LITERATURE

[1] 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 21st USENIX Security Symposium (SEC'12)*. USENIX, 2012.

[2] Axelle Apvrille. Symbian worm Yxes: towards mobile botnets? *Journal in Computer Virology*, 8(4):117–131, May 2012.

[3] Michael Bailey, Evan Cooke, Farnam Jahanian, Yunjing Xu, and Ann Arbor. A survey of botnet technology and defenses. In Lisa O'Conner, editor, *Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security (CATCH'09)*, pages 299–304. IEEE, 2009.

[4] Nicole Lang Beebe and Jan Guynes Clark. A hierarchical, objectives-based framework for the digital investigations process. In *Proceedings of the 4th Digital Forensics Research Workshop (DFRWS'04)*, pages 1–17, 2004.

[5] James R. Binkley and Suresh Singh. An algorithm for anomaly-based botnet detection. In *Proceedings of the 2nd conference on Reducing Unwanted Traffic on the Internet (SRUTI'06)*, page 7. USENIX, 2006.

[6] Hamad Binsalleeh, Thomas Ormerod, Amine Boukhtouta, Prosenjit Sinha, Amr M. Youssef, Mourad Debbabi, and Lingyu Wang. On the analysis of the Zeus botnet crimeware toolkit. In *Proceedings of the 8th Annual International Conference on Privacy Security and Trust (PST)*, pages 31–38. IEEE, 2010.

[7] Dan Bleaken. Botwars: the fight against criminal cyber networks. *Computer Fraud & Security*, 2010(5):17–19, May 2010.

[8] Alex Brodsky and Dmitry Brodsky. A distributed content independent method for spam detection. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, page 3. USENIX, 2007.

[9] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *Proceedings of the 20th USENIX Security Symposium (SEC'11)*, page 13. USENIX, 2011.

[10] Ken Chiang and Levi Lloyd. A case study of the Rustock rootkit and spam bot. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, page 10. USENIX, 2007.

[11] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI'05)*, page 6. USENIX, 2005.

[12] Marco Cova, Christopher Kruegel, and Giovanni Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th World Wide Web Conference (WWW'10)*, pages 281–290. ACM, 2010.

[13] Neil Daswani and Michael Stoppelman. The anatomy of Clickbot.A. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, page 11. USENIX, 2007.

[14] Carlton R. Davis, Jose M. Fernandez, Stephen Neville, and John McHugh. Sybil attacks as a mitigation strategy against the Storm botnet. In *Proceedings of the 3rd International Conference on Malicious and Unwanted Software (MALWARE'08)*, pages 32–40. IEEE, October 2008.

[15] Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proceedings of the 14th ACM conference on Computer and Communications Security (CCS'07)*, pages 375–388. ACM, 2007.

[16] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed Denial-of-Service attacks. In Sabrina De Capitani di Vimercati, Paul Syverson, and Dieter Gollmann, editors, *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS'05)*, pages 319–335. Springer, 2005.

[17] Mona Gandhi, Markus Jakobsson, and Jacob Ratkiewicz. Badvertisements: Stealthy click-fraud with unwitting accessories. *Journal of Digital Forensic Practice*, 1(2):131–142, July 2006.

[18] Simson Garfinkel. Lessons learned writing digital forensics tools and managing a 30TB digital evidence corpus. In Eoghan Casey,

editor, *Proceedings of the 12th Annual Digital Forensics Research Conference (DFRWS'12)*, volume 9, pages S80–S89. Elsevier, 2012.

[19] Steven Gianvecchio, Mengjun Xie, Zhenyu Wu, and Haining Wang. Humans and bots in internet chat: Measurement, analysis, and automated classification. *IEEE/ACM Transactions on Networking*, 19(5):1557–1571, 2011.

[20] Steve Gold. Taking down botnets. *Network Security*, 2011(5):13–15, May 2011.

[21] Chris Grier, Lucas Ballard, Juan Caballero, Neha Chachra, Christian J. Dietrich, Kirill Levchenko, Panayiotis Mavrommatis, Damon McCoy, Antonio Nappa, Andreas Pitsillidis, Niels Provos, M. Zubair Rafique, Moheeb Abu Rajab, Christian Rossow, Kurt Thomas, Vern Paxson, Stefan Savage, and Geoffrey M. Voelker. Manufacturing compromise: The emergence of exploit-as-a-service. In *Proceedings of the 19th ACM conference on Computer and Communications Security (CCS'12)*, pages 821–832. ACM, 2012.

[22] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent Byunghoon Kang, and David Dagon. Peer-to-Peer botnets: Overview and case study. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, page 1. USENIX, 2007.

[23] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee, and Menlo Park. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium (SS'07)*, page 12. USENIX, 2007.

[24] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet Command and Control channels in network traffic. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS'08)*. Internet Society, 2008.

[25] Thorsten Holz. A short visit to the bot zoo. *IEEE Security & Privacy*, 3(3):76–79, 2005.

[26] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*. Internet Society, 2008.

[27] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of Cypher, Gremlin and native access in Neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 workshops (EDBT'13)*, pages 195–204. ACM, 2013.

[28] Chris Kanich, Nicholas Weaver, Damon McCoy, Tristan Halvorson, Christian Kreibich, Kirill Levchenko, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. Show Me the Money: Characterizing Spam-advertised Revenue. In *Proceedings of the 20th USENIX Security Symposium (SEC'11)*, page 15. USENIX, 2011.

[29] Anestis Karasaridis, Brian Rexroad, and David Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, page 7. USENIX, 2007.

[30] Erhan J. Kartaltepe, Jose Andre Morales, Shouhuai Xu, and Ravi Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. In Jianying Zhou and Moti Yung, editors, *Proceedings of the 8th International Conference on Applied Cryptography and Network Security (ACNS'10)*, volume 6123, pages 511–528. Springer, 2010.

[31] Leonard Kwan, Pradeep Ray, and Greg Stephens. Towards a methodology for profiling cyber criminals. In *Proceedings of the 41st Hawaii International Conference on System Sciences*, 2008.

[32] Chao Li, Wei Jiang, and Xin Zou. Botnet: Survey and Case Study. In *Proceedings of the 4th International Conference on Innovative Computing, Information and Control (ICICIC'09)*, pages 1184–1187. IEEE, December 2009.

[33] Zhen Li, Qi Liao, and Aaron Striegel. Botnet economics: Uncertainty matters. In Eric Johnson, editor, *Managing Information Risk and the Economics of Security*, pages 245–267. Springer, 2009.

[34] Jian Liang, Naoum Naoumov, and Keith W. Ross. The index poisoning attack in P2P file sharing systems. *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM'06)*, pages 1–12, 2006.

[35] Cullen Linn and Saumya Debray. Obfuscation of executable code to improve resistance to static disassembly. In *Proceedings of the 10th ACM conference on Computer and Communications Security (CCS'03)*, pages 290–299. ACM, 2003.

[36] Shangdong Liu, Jian Gong, Wang Yang, and Ahmad Jakalan. A survey of botnet size measurement. In Bob Werner, editor, *Proceedings of the 2nd International Conference on Networking and Distributed Computing (ICNDC'11)*, pages 36–40. IEEE, September 2011.

[37] Derek Manky. Cybercrime as a service: a very modern business. *Computer Fraud & Security*, 2013(6):9–13, June 2013.

[38] Steve Mansfield-Devine. Anonymous: serious threat or mere annoyance? *Network Security*, 2011(1):4–10, January 2011.

[39] Jose Nazario and Thorsten Holz. As the net churns: Fast-flux botnet observations. In *Proceedings of the 3rd International Conference of Malicious and Unwanted Software (MALWARE'08)*, pages 24–31. IEEE, 2008.

[40] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX conference on Networked Systems Design and Implementation (NSDI'10)*, page 26. USENIX, 2010.

[41] Heloise Pieterse and Martin S. Olivier. Android botnets on the rise: Trends and characteristics. In H.S. Venter, M. Loock, and M. Coetzee, editors, *Proceedings of the Information Security for South Africa Conference (ISSA)*, pages 1–5. IEEE, August 2012.

[42] Daniel Plohmann and Elmar Gerhards-Padilla. Case study of the Miner botnet. In C. Czosseck, R. Ottis, and K. Ziolkowski, editors, *Proceedings of the 4th International Conference on Cyber Conflict (CYCON'12)*, pages 1–16. IEEE, 2012.

[43] Phillip Porras, Hassen Saïdi, and Vinod Yegneswaran. A foray into Conficker's logic and rendezvous points. In *Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09)*, pages 7–7. USENIX, 2009.

[44] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An analysis of the iKee.B iPhone botnet. In A.U. Schmidt, G. Russello, A. Lioy, N.R. Prasad, and S. Lian, editors, *Proceedings of the 2nd International Conference on Security and Privacy in Mobile Information and Communication Systems (ICST'10)*, volume 47, pages 141–152. Springer, 2010.

[45] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All your IFRAMES point to us. In *Proceedings of the 17th USENIX Security Symposium (SEC'08)*, pages 1–15. USENIX, 2008.

[46] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th Internet Measurement Conference (IMC'06)*, pages 41–52. ACM, 2006.

[47] 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 *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots'07)*. USENIX, 2007.

[48] Daniel Ramsbrock and Xinyuan Wang. A first Step toward live botmaster traceback. In Richard Lippmann, Engin Kirda, and Ari Trachtenberg, editors, *Proceedings of the 11th International Symposium on Recend Advances in Intrusion Detection (RAID'08)*, pages 59–77. Springer, 2008.

[49] Robert Rownlingson. A ten step process for forensic readiness. *International Journal of Digital Evidence*, 2(3):1–28, 2004.

[50] Lei Shi, Chen Wang, and Zhen Wen. Dynamic network visualization in 1.5D. In Giuseppe Di Battista, Jean-Daniel Fekete, and Huamin Qu, editors, *Proceedings of the Pacific Visualization Symposium (PascificViz)*, pages 179–186. IEEE, 2011.

[51] Prosenjit Sinha, Amine Boukhtouta, Victor Heber Belarde, and Mourad Debbabi. Insights from the analysis of the Mariposa botnet. In *Proceedings of the 5th International Conference on Risks and Security of Internet and Systems (CRISIS'10)*, pages 1–9. IEEE, October 2010.

[52] Aditya K. Sood and Richard J. Enbody. Crimeware-as-a-service - A survey of commoditized crimeware in the underground market. *International Journal of Critical Infrastructure Protection*, 6(1):28–38, March 2013.

[53] Aditya K. Sood, Richard J. Enbody, and Rohit Bansal. Dissecting SpyEye - Understanding the design of third generation botnets. *Computer Networks*, 57(2):436–450, February 2013.

[54] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Christian Morariu, Aiko Pras, and Burkhard Stiller. An overview of IP flow-based intrusion detection. *IEEE Communications Surveys & Tutorials*, 12 (3):343–356, 2010.

[55] Brett Stone-Gross, Marco Cova, Bob Gilbert, Richard A. Kemmerer, Christopher Kruegel, and Giovanni Vigna. Analysis of a botnet takeover. *IEEE Security & Privacy*, 9(1):64–72, 2011.

[56] Brett Stone-Gross, Thorsten Holz, Gianluca Stringhini, and Giovanni Vigna. The underground economy of spam: A botmaster's perspective of coordinating large-scale spam campaigns. In *Proceedings of the 4th USENIX workshop on Large-scale Exploits and Emergent Threats (LEET'11)*, page 4. USENIX, 2011.

[57] Sam Stover, Dave Dittrich, John Hernandez, and Sven Dietrich. Analysis of the Storm and Nugache trojans: P2P is here. *USENIX ;login*, 32(6):18–27, 2007.

[58] Kurt Thomas and David M. Nicol. The Koobface botnet and the rise of social malware. In *Proceedings of the 5th International Conference on Malicious and Unwanted Software (MALWARE'10)*, pages 63–70. IEEE, October 2010.

[59] Tatiana von Landesberger, Arjan Kuijper, Tobias Schreck, Joern Kohlhammer, Jack van Wijk, Jean-Daniel Fekete, and Dieter Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719–1749, 2011.

[60] Ping Wang, Baber Aslam, and Cliff C. Zou. Peer-to-peer botnets. In Peter Stavroulakis and Mark Stamp, editors, *Handbook of Information and Communication Security*, pages 335–350. Springer, 2010.

[61] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. *Transactions on Dependable and Secure Computing*, 7(2):113–127, April 2010.

[62] Qian Wang, Zesheng Chen, Chao Chen, and Niki Pissinou. On the robustness of the botnet topology formed by worm infection. In *Proceedings of the 8th IEEE Global Telecommunications Conference (GLOBECOMM'10)*, pages 1–6. IEEE, December 2010.

[63] Xinyuan Wang and Daniel Ramsbrock. The botnet problem. In John R. Vacca, editor, *Computer and Information Security Handbook*, chapter 8, pages 119–132. Morgan Kaufmann, 1 edition, 2009.

[64] Jim Webber. A programmatic introduction to Neo4j. In *Proceedings of the 3rd annual conference on systems, programming, and applications: osftware for humanity (SPLASH'12)*, pages 217–218. ACM, 2012.

[65] Yinglian Xie, Fang Yu, Kannan Achan, and Rina Panigrahy. Spamming botnets: signatures and characteristics. In *Proceedings of the ACM Conference on Data Communication (SIGCOMM'08)*, pages 171–182. ACM, 2008.

OTHER REFERENCES

[66] Cory Altheide and Eoghan Casey. UNIX forensic analysis. In Eoghan Casey, editor, *Handbook of digital forensics and investigation*, pages 201–353. Elsevier, 2010.

[67] Aquabox. Citadel 1.3.45 botnet (extreme edition). Online message (retrieved at October 14, 2013), June 2012. URL http://pastebin.com/gRqQ2693.

[68] Daniel J. Barrett and Richard Silverman. *SSH, the secure shell: the definitive guide*. O'Reilly, 2001.

[69] Joost Bijl. Analysis of malicious advertisements on telegraaf.nl. Online article (retrieved at November 28, 2013), August 2013. URL http://blog.fox-it.com/2013/08/01/analysis-of-malicious-advertisements-on-telegraaf-nl/.

[70] Bits of Freedom. Winnaars big brother awards bekend. Online article (retrieved at October 22, 2013), March 2012. URL https://bba2011.bof.nl/2012/03/winnaars-big-brother-awards-bekend/.

[71] Dancho Danchev. Yet another bitcoin accepting e-shop offering access to thousands of hacked pcs spotted in the wild. Online article (retrieved at October 23, 2013), October 2013. URL http://www.webroot.com/blog/2013/10/16/.

[72] Dutch Ministry of Security and Justice. Wetsvoorstel computercriminaliteit III. Online article (retrieved at October 23, 2013), May 2013. URL http://www.internetconsultatie.nl/computercriminaliteit.

[73] Arnoud Engelfriet. Wettelijke bescherming voor white hat hackers: wenselijk of niet? Online article (retrieved at October 23, 2013), September 2011. URL https://ictrecht.nl/cybercrime/wettelijke-bescherming-voor-white-hat-hackers/.

[74] Dan Farmer and Wietse Venema. *Forensic Discovery*. Addison-Wesley, 1 edition, 2005.

[75] Max Goncharov. Russian underground 101. Technical report, Trend Micro, 2012.

[76] Jeremiah Grossman and Matt Johansen. Million browser botnet. Black Hat USA, 2013. URL https://media.blackhat.com/us-13/us-13-Grossman-Million-Browser-Botnet.pdf.

[77] Fraser Howard. Exploring the Blackhole exploit kit. Technical report, SophosLabs, 2012. URL http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit/.

[78] Zhuge Jianwei, Gu Liang, and Duan Haixin. Investigating China's online underground economy. Technical report, IGCC, 2012.

[79] Alexei Kadiev. End of the line for the Bredolab botnet? Online article (retrieved at August 3, 2013), 2010. URL https://www.securelist.com/en/analysis/204792152/.

[80] Oleg Kolesnikov and Wenke Lee. Advanced polymorphic worms: Evading IDS by blending in with normal traffic. Technical report, Georgia Institute of Technology, 2005. URL https://smartech.gatech.edu/handle/1853/6485.

[81] Merel Koning. *Terug-hacken als opsporingsmethode*. Master's thesis, University of Amsterdam, 2011.

[82] Brian Krebs. Bredolab botmaster 'birdie' still at large. Online article (retrieved at February 20, 2014), March 2012. URL http://krebsonsecurity.com/2012/03/bredolab-botmaster-birdie-still-at-large/.

[83] Lavakumar Kupp. Attacking with HTML5. Black Hat Webcast, 2010. URL https://media.blackhat.com/bh-ad-10/Kuppan/Blackhat-AD-2010-Kuppan-Attacking-with-HTML5-wp.pdf.

[84] Microsoft. Microsoft security intelligence report, botnets today. Online article (retrieved at October 9, 2013), 2010. URL http://www.microsoft.com/security/sir/story/default.aspx#!botnetsection_irc.

[85] Michael Mimoso. IRC botnet leveraging unpatched Plesk vulnerability. Online article (retrieved at October 14, 2013), June 2013. URL http://threatpost.com/irc-botnet.

[86] Yuri Namestnikov. The economics of botnets. Technical report, Kaspersky Lab, 2009.

[87] Gavin O'Gorman and Geoff McDonald. Ransomware: a growing menace. Technical report, Symantec, 2012.

[88] Gunter Ollmann. Botnet communication topologies. Technical report, Damballa Inc., 2009. URL http://www.damballanews.com/downloads/r_pubs/WPBotnetCommunicationsPrimer(2009-06-04).pdf.

[89] Openbaar Ministerie. Dutch national crime squad announces takedown of dangerous botnet. Online article (retrieved at September 23, 2013), October 2010. URL http://www.om.nl/onderwerpen/verkeer/@154338/.

[90] Pierluigi Paganini. Cybercrime as a service. Online article (retrieved at September 4, 2013), August 2013. URL http://resources.infosecinstitute.com/cybercrime-as-a-service/.

[91] Gary Palmer. A road map for digital forensic research. Technical report, Digital Forensic Research Workshop (DFRWS), 2001.

[92] Paul Royal. On the Kraken and Bobax botnets. Technical report, Damballa, Inc., 2008. URL http://bandwidthco.com/whitepapers/compforensics/malware/bots/OntheKrakenandBoBaxBotnets.pdf.

[93] William Salusky and Robert Dunford. Know your enemy: Fast-flux service networks. Technical report, The Honeypot Project, 2007. URL http://www.honeynet.org/book/export/html/130.

[94] Timo Schless. *De organisatie van botnetbestrijding in Nederland*. Master's thesis, Open Universiteit, 2013.

[95] Jerome Segura. Citadel: a cyber-criminal's ultimate weapon? Online article (retrieved at October 1, 2013), November 2012. URL http://blog.malwarebytes.org/intelligence/2012/11/citadel/.

[96] Symantec. Symantec internet security threat report 2011. Online article (retrieved at August 4, 2013), 2012. URL https://www.symantec.com/threatreport/topic.jsp?id=spam_fraud_activity_trends&aid=analysis_of_spam_delivered_by_botnets.

[97] Symantec. Internet security threat report 2012. Technical Report April, 2013.

[98] Team Cymru. Developing botnets…an analysis of recent activity. Technical report, 2010. URL http://www.team-cymru.com/ReadingRoom/Whitepapers/2010/developing-botnets.pdf.

[99] The Linux Information Project. User ID definition. Online article (retrieved at March 6, 2014), July 2005. URL http://www.linfo.org/uid.html.

[100] Peter Wagenaar. *Detecting botnets using file system indicators*. Master's thesis, University of Twente, 2012.

COLOPHON

This document was typeset with LaTeX 2ε using Robert Slimbach's *Minion Pro* type face. The typographic style was inspired by Robert Bringhurst's genius as presented in *The Elements of Typographic Style*, available for LaTeX with André Miede's excellent `classicthesis` package.