Botnet construction, control and concealment

Looking into the current technology and analysing tendencies and future trends.

Krogoth

Submitted as part of the requirements for the award of the MSc in Information Security.
Special version for the ShadowServer website.

March 2008

Acknowledgements

This thesis is dedicated to **Elsa**. Mooh!

First I would like to thank **fm** for his continuing support. We already worked on that many projects together and by now he knows exactly how I operate. This makes his thoughts and comments invaluable.

I would also like to thank my supervisor, **ja**, for his support and help during my project. His input and comments were very useful and were helping me to stay on track and not losing my focus.

I was given the chance to meet some interesting botnet researchers at the InBot'08 conference in Aachen, Germany. I had many good discussions there and learned quite a bit. It was a great experience thanks to all the participants.

Special thanks to the whole Shadowserver Foundation team and especially to **Semper Securus** and **Freed0**. Without your time, patience, wisdom and resources I would never have been able to write this thesis.

A big thank you to all the reviewers: **rw**, **ch**, **sm** and **sw**. As well as to **rj** for taming my English.

Then I would like to thank my friends and family which showed me their continuing support during my studies and while I was writing this thesis. I really appreciate your backup.

At the end I'd like to add a quote by House M.D. (the TV doctor) which sums up the work on my thesis quite nicely:

"You can think I'm wrong, but that's no reason to quit thinking".

Contents

List of Tables

List of Figures

List of Abbreviations and Acronyms

Acronym	Meaning	Explanation
AV	Anti Virus	Anti Virus software scans computer systems for malware. Malware is the umbrella term for the different types of dangerous software like viruses and worms. The seller of AV software is an AV vendor.
BO	Back Orifice	Controversial remote computer administration software. Most AV software reports it as malware.
C&C	Command and Control	Every botnet has a C&C which is used by the botnet master to manage his botnet.
cDc	Cult of the Dead Cow	cDc is a computer underground group which was founded in 1984 and which is still active today.
DDoS	Distributed Denial of Service	This is a special kind of attack where the attacker consumes all available resources of a system so that legitimate users can't be served anymore.
DUL	Dialup User List	A list of IP addresses assigned to Internet users worldwide. The DUL is used by mail administrators to deny emails sent directly from dialup users.
DynDNS	Dynamic Domain Name System	"Portable" DNS which allows dialup users to use DNS for their IP addresses even when their IP address regularly changes.
HIDS	Host-based Intrusion Detection System	Software which runs on a computer and analyses the file system and other components of the OS for unauthorised changes.
HIPS	Host-based Intrusion Prevention System	Software which analyses the behaviour of programs running on a system preventing potential malicious activities.
IM	Instant Messaging	Special form of Internet chat where the communication is less group oriented as with IRC but participants send private messages to other participants.
IRC	Internet Relay Chat	One of the first Internet chatting protocols. The first IRC server was run in summer 1988 in Oulu, Finland.
ISP	Internet Service Provider	ISPs provide Internet services to customers.

Acronym	**Meaning**	**Explanation**
MBR	Master Boot Record	The first sector on the hard disc. This sector is accessed before booting the operating system. The data within the MBR determines the started OS.
MITM	Man-in-the-Middle	Attack where the attacker sits between the victim and the server the victim wishes to connect to. The attacker mimics the server to the victim and acts as the victim to the server.
OS	Operating System	Windows, OS X and Linux are examples of different Operating Systems. Most bot clients run on Windows based computers.
P2P	Peer to Peer	A P2P network using a network topology where no central server is used for communication between the peers.
VoIP	Voicer over IP	Communication protocol on the Internet used to replace the conventional telephone system (mostly for monetary reasons).

Executive Summary

Viruses and worms exist since for some time. But the increasing scale of the Internet provides new opportunities for malware developers. Computers can be taken over by third parties. Such captured computers, bots, can be virtually rounded up. This allows one person, a herder, to manage a network of bots, known as botnet.

Mikko Hypponen from F-Secure says "We are seeing less of the big virus outbreaks such as Sasser and Blaster, and so some people believe the situation is getting better, when in fact it is getting worse. The bad boys are getting more professional and doing more targeted attacks". Criminals are becoming more professional and attacks are growing more sophisticated.

The quality of a botnet, like the one from the gang behind the famous Storm network, lies in its coordinated and well planned execution. During the Christmas season of 2007 a new wave of spam emails was sent out to lure users onto infected websites to take over their computers. The sending out of the spam mails was timed with the closing of the Russian domain registrar for the Christmas season. Requests to unregister the domain names used to advertise the infected web servers would only be handled after the holidays. This allowed the Storm gang to run the spam wave for a few days without the fear of having to change the solicited domain names.

Although the quality of botnets and malware in general is constantly improving, some aspects remain the same. A botnet has a typical lifecycle. Bots are constructed by a software developer and then released into the public. At some point the bots will get detected by an AV company or an independent malware researcher. The researcher will catch at least one copy of the bot and analyse the architecture and functionality of the bot. When he knows enough about the bot and the botnet he will annihilate the botnet and try to find out about the owner of the bots so he can send that information to a law enforcement agency.

Unfortunately current defence mechanisms won't defend us completely from the new threats. Bots are constantly and automatically changing their signature to successfully avoid the detection by current AV software.

There are indications where the general development of bot software is heading. Authors of bots and herders of botnets are getting more professional, and they are exploring new options and possibilities. They will run the botnets like big corporation run their internal corporate network with monitoring software and intrusion detection. Such professionalisation will make the detection, analysis and annihilation of botnets significantly more complex.

Botnets do have much potential and the criminals realise this. They are testing out different designs and are filling their armoury with new tricks. It is time to gather some intelligence and work on the defence.

The chapter covering trends and future development has been rewritten several times from the time I started this project to submission of the thesis. This is because botnet techniques are evolving faster than the time allowed for my project. Some developments which I predicted were reported in the wild. This is a nice example to show that the whole topic is constantly changing and it is essential that we keep pace with the developments.

Part 1

Background

CHAPTER 1

Preamble

1.1. Fear

There is an old saying that fear is a powerful motivator. Fear is often used in politics and the ordinary life if somebody modifies people's behaviour. Fear sells well for it stimulates the imagination and paves the way for general uncertainty and doubt.

In recent years many countries introduced laws against electronic crimes often termed as hacking. On one hand this helped in protecting infrastructure and convict criminals. On the other hand the research industry was hindered in doing research because their tools became outlawed. The researchers feared their daily work could turn them into criminals.

In most countries it is now illegal to create, possess or distribute viruses and general malware. What was demonstrated with alcohol during the prohibition in the USA, illegal activities can turn into prospering businesses. It should not come as a surprise that organised crime is behind some botnets.

Banks and other organisations spread fear about phishing, pharming and identity theft. The news warns about viruses without supplying concrete information. And worse, only an elitist group really understands the problem. Everybody else is just overstrained with the topic, or how it is communicated.

There is yet another source of fear. When publicly talking about future threats and emerging trends of malware there is evidence that such work is not always received well [**?**], [AM07]. I learnt this myself during the writing of this thesis. While the analysis of botnets is received well, it quickly becomes obvious that publishing findings should only happen within a closed group of specialists.

I do not agree with keeping the research secret. While I agree in keeping intelligence secret to not peril the prosecution of individuals, it is my strong opinion that an open exchange of trend analysis by far outweighs the disadvantages of giving the authors of botnets ideas. Keeping ideas secret won't keep the criminals from having the same ideas. But discussing ideas and thoughts openly at least gives the chance that somebody will find mitigation strategies before an idea is implemented in some form of malware and spread in the wild.

1.2. Uncertainty

According to a recent news article [New07], the number of bots making the storm botnet reached 50 millions of computers. Said article references a press statement [Mes07] by MessageLabs which states that the storm botnet "now estimated to comprise of 1.8 million computers worldwide". Why is there a discrepancy of 48.5 million computers between the news article and the press statement? Which number is correct? What facts were used to build these figures?

The same news.com article also references an interview [ITN07] between Matt Sergeant, chief anti-spam technologist with MessageLabs and the Australian news website itnews.com.au. The itnews article cites Matt Sergeant to have said that he "suspects the botnet could be as large as 50 million computers". While it is not clear how Matt Sergeant determined this number, this is probably the source of "50 million drones in the storm botnet" which was mentioned in the news.com article.

Shadowserver is tracking botnets and their activity on a worldwide scale since quite some time and is an authoritative source for botnet related information. When talking to members of the Shadowserver Foundation it became clear, that the number of 50 million bots is an order of magnitude too large.

According to [RZMT06] "there seems to be little, if any, agreement on what specifically the size of a botnet refers to". The size can consist of many different factors. [Hol08] describes the growing

size of the Storm botnet during the Christmas season 2007/08. It is interesting to see that while the botnet generally grew, there were some diurnal and geographic factors which are responsible for a varying instead of linear growth.

Which leads to the question of how the number of bots in a botnet can be measured and whether inactive or offline computers are counted as well. There are many other factors which need to be addressed. Some computers are running on a link which changes their IP address every time the computer reconnects to the Internet (or once a day, or regularly). Over time such address changes can lead to an incorrect picture since a naive interpretation of the logs can lead to the conclusion that the whole network is infected although it is the same infected computer just reconnecting with different IP addresses.

1.3. Doubt

Some bot developers are developing quite sophisticated software using the latest technology to hide and protect their brood. Because of the networked nature of botnets and their distribution over many jurisdictions it is very difficult to track the criminals behind the botnets. There is also a constant evolution in technology and a constant growth of the quality and subtlety of malware in general which results in the criminals having even more power of control.

There is already quite some material available about botnets. Many companies and groups invest in the research of botnets and the development of defence mechanisms against them. While organisations like the Shadowserver Foundation have existed for some time, they still have much more work to do and measured by the rate of development of new technologies and the sophistication of some of the botnets, the end is not near.

The question is, where are we now and where are we heading to? While the botnets and the technology used by them become more advanced, the current strategy to defeat the botnets is largely the same since day one: Attack the C&C. While still an effective strategy, the question is if that strategy will remain effective for much longer. Or will we need new strategies and threat mitigation alternatives?

Analysing malware is cost intensive, although some automation technologies exist and are constantly developed. The main problem remains. The situation has become a general war of resources where the bot developers raise the bar as high as possible and the botnet hunters counter this trend with more resources and even newer technology. If we can believe in the latest reports on the quality of AV software [hei07], then the wrong side is currently winning the battle.

There is another option to fight botnets. Make the owners of the captured computers aware of what the problem is and why they are part of the problem. But this can only be successful when there is a general discussion of the topic. If it would be successful at all.

1.4. Project Objectives

This project has the following objectives:

- Identify key technologies used for the development, command and control of botnets.
- Identify key players.
- Identify and question current detection mechanisms and incident handling strategies.
- Analyse and research into potential development trends, new detection and monitoring technologies and mitigation strategies.

1.5. Content

Chapter 2 is an introduction to botnets. It contains details about the botnets in general and identifies the key players.

Chapter 3 takes a look at the past. How things evolved and how botnets came into existence.

Chapter 4 is a taxonomy of a botnet. This chapter is a basis onto which the following chapters build.

Chapters 5 to 8 describe the typical lifecycle of a botnet. These chapters discuss how botnets defend themselves and how the annihilation of a botnet could evolve to counter these self-defence mechanisms.

Chapter 9 looks into what could come tomorrow and what we may need to do to defend ourselves.

CHAPTER 2

Introduction to Botnets

2.1. Overview

This chapter introduces botnets. It contains information about naming and the key players. It introduces possible motivations for building and running a botnet. While bot software can be used for legitimate purposes, this thesis is focusing on the illegal uses of bots. Bots in this context are installed without the users consent and for illegal purposes only.

2.2. Naming

2.2.1. Bot. Software which is run on a computer to perform some predefined automated task is called a software robot. According to [EO007] the word robot comes "from Czech robotnik" which means 'slave' and from the word "robota" which means 'forced labor, drudgery'. Robots can have different purposes. They can be used for benign tasks like indexing websites for search engines or regularly updating weather information on a homepage. But Robots can also have malicious purposes like sending out spam or attacking other computers.

An attacker attacks a computer and successfully exploits a software vulnerability or "social engineers" the rightful owner into executing a trojan horse to capture the computer system and gain system privileges. System privileges means the attacker has full control over the victim's computer, with the same privileges the rightful system administrator usually has. In this scenario, the owner of the computer neither agrees in having the computer misused nor does he notice it is misused. At least not immediately. Such a captured system is called a robot or "bot".

It is important to note that the quality of the bots varies greatly based on the skills of the respective developer. While there are bots which immediately terminate themselves because of programming errors, there are other bots which make use of sophisticated P2P architectures.

Zombie. Bots are sometimes called zombies. In literature, zombies are aboulic dead human beings which follow their masters will and their own urge to eat human flesh and brains. This is an appropriate description of bots which do whatever their secret master is asking them to do.

Drone. Bots are sometimes called drones and botnets are called drone armies. This thesis will make use of the word 'bot' for a captured computer, 'bot client' for the malicious software running on the bot and 'botnet' to describe the whole bot network.

2.2.2. Botnet. The name botnet is the composition of the words bot and net. Net is the short version for network. In general, a botnet is a platform for distributed computing [Sav05]. Typically thousands of bots are rallied together to form a botnet [DGZ+05].

The main purpose of botnets is to use hijacked computers for fraudulent online activities [Bar07]. Among these activities are blackmail, fraud and identity theft.

Botnets are managed by single criminals, a group of criminals or an organised crime syndicate. Because of the growing complexity of botnets and their requirements, the various jobs and assignments around the botnet development and maintenance are distributed over different roles.

2.3. Botnet Single Components

2.3.1. Rallying and Command & Control. The botnet gets its power from its interconnectedness and its size. The more bots are in a network, the more dangerous a botnet becomes. Depending on the technique used to control a botnet, the rallying and C&C either scales well with growth of the botnet or there will be a technical limit on the maximum botnet size.

All these factors are determined by the key discriminators of a botnet. Among these discriminators are the size of the botnet and the diameter of the virtual network containing all the bots [DGZ+05]. The diameter describes how far away on the Internet each bot entity is from each other. Where distance is not measured in meters but in network hops between the bots.

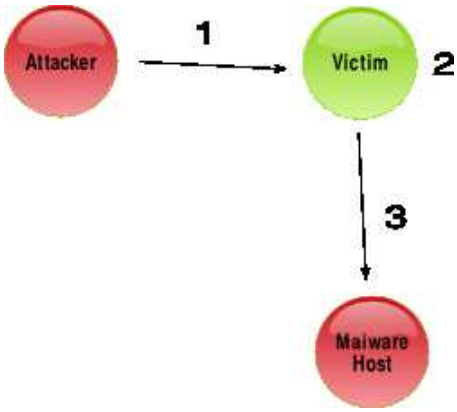

FIGURE 2.1. Dropper

Push vs. Pull. There are two general control mechanisms to manage a botnet. The one is to push the commands from the C&C to the bots. To do so the botnet needs an architecture where the bots are always connected to the C&C infrastructure. The pull mechanism means that the bots are not always connected to the C&C infrastructure and therefore need to regularly contact the C&C and pull the new commands and information on their own.

2.3.2. Dispersion. The dispersion of a botnet typically happens in multiple steps as shown in figure 2.1. First, an attacking system scans network ranges for vulnerable systems. Once a vulnerable system is found the attacking system tries to exploit the detected vulnerability (1). On success a dropper is installed (2). The dropper is usually a very small application which only contains the logic to asynchronously download the actual bot software [PSY07]. Once the dropper is installed and running, it independently downloads the actual bot software (3) from a host serving the bot client. Such a file serving host is usually running on another infected bot.

While there are other variations of the same procedure, this is the most common procedure. The rationale for such an approach is simple. Only the machines doing the scanning are at risk of detection.

Asynchronously downloading the botnet software after a successful exploit has another reason. The exploiting application can be much smaller than it could otherwise be, therefore uses less resources and time to transmit[1]. This leaves more resources for the attacking system. It can attack new systems while the infected computers are installing the bot client on their own. The bot software is only installed on an infected machine which lowers the risk of having the binary analysed by a botnet hunter.

2.3.3. Lifecycle. When looking into botnets it makes sense to describe their typical lifecycle. Doing so helps in explaining the different phases of life and how design decisions during the construction phase are influencing the later phases. The lifecycle as described in figure 2.2 was defined for this thesis and is explained in detail in part two. The following is a short introduction to the different phases.

Construction. In the construction stage somebody designs and develops the botnet software. This typically consists of a C&C infrastructure and a bot client. After construction, the bot is being released in the wild and starts to spread and build up the botnet.

Detection. When a botnet grows to a certain size it becomes visible on the radar of security professionals and especially the botnet hunters. Reasons for being detected can be manifold and are discussed in the respective chapter.

Capturing. Once a botnet is detected, a researcher will get her hands on the bot software for further analysis. There are different strategies to capture bot clients which will be described in a later chapter.

[1]It is interesting to note that bot developers sometimes include utilities for decompressing files in their malware to keep the downloads as small as possible.

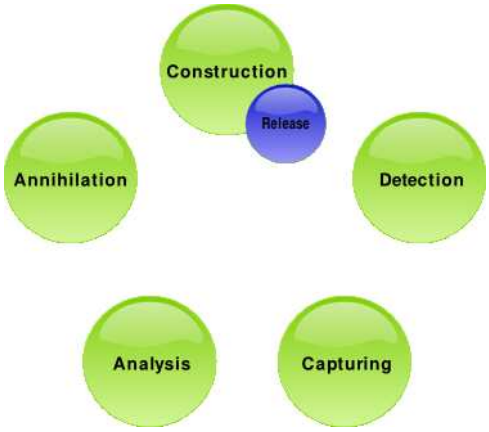

FIGURE 2.2. Botnet Lifecycle

Analysis. Once the bot client is captured, the researcher will study the software and probably watch the different bots and their C&C behaviour to learn more about the nature of that specific botnet and its architecture.

Annihilation. When a botnet is analysed and well understood, botnet hunters will typically try to tear it down. There are different strategies to do so and many motivations why somebody would want to take down a botnet. All of which are discussed in the respective chapter.

2.4. Players

There are several key players involved in the botnet lifecycle. These are described in this section.

2.4.1. Bot Developer. The developer is the person designing and implementing the bot client. The bot developer does not necessarily need to know a programming language very well. It is relatively simple to find source code for bots on the Internet. Besides using free sources, it is also possible to either buy a bot client written by a third party or to use one of the point & click kits which allow the creation of a bot without any programming skills.

Some developers published the complete sources for their bots under an Open Source licence[2] so that others can use what they developed so far. While this provides unique access to bot sources, and therefore makes it simpler for the botnet hunters to detect and annihilate a botnet, it also opens the doors for entire series of clones and enhanced copies of such bots.

According to [Bar07] the criminal organisations behind botnets "employ software developers, they buy and sell infrastructure for their criminal activities and they recruit people (mules) for money laundering to hide their identity". This makes sense since creating and running botnets started to become complex. The same behaviour can be seen in the legal software development field since many years. Software factories often buy and use third party components. These software companies are specialising in some field and for the other needs they buy components from third party developers. Some bot developers do the same; they buy and use components for their bot software. Among these components are infection mechanisms, exploits used to attack computers and others.

2.4.2. Bot herder. The bot herder is described by [ZHH+07] as the human operator controlling the compromised computers. Other sources are using the word botnet master [BS07] or botmaster [Got07], but all these terms mean the same. The bot herder does not necessarily need to be the same person as the bot developer.

[2]An interesting question which I did not further pursue is about the legal aspects of such a move. Publishing malware is illegal in many jurisdictions. This will have an impact on the validity of the licence the malware was released under. But of course, this will not hinder anybody of using such published source as a basis to create new malware.

Some bot herders are buying existing botnets (or taking them by force[3]) or are re-using bot clients from a third party. Other bot herders are using their botnet themselves, or they are renting the whole or parts of their botnet to a third party botnet user.

2.4.3. Botnet user. This can either be the bot herder herself or somebody else. According to [NBL06] botnets are rented by their bot herders to third party botnet users. These users are then using such rented botnets for sending out spam or for committing online crime. Temporarily renting a botnet can be financially interesting. For somebody who wants to blackmail the competitor or wants to send out millions of spam mails it is probably less expensive and risky to rent an existing botnet than to create and maintain a new one.

2.4.4. Victims. Generally there are two groups of victims, the owner of the computer running the bot client and the entity which will be targeted through the botnet.

In a wider sense, we all are victims of botnets. We receive spam sent through bots daily and we all could wake up one day and find out that we are at the wrong end of a DDoS attack controlled by a botnet herder.

2.4.5. Botnet hunter. A botnet hunter is somebody analysing botnets and trying to take botnets and their C&C infrastructure down. Botnet hunters can either be individuals working on their own in a kind of isolated way. Or they are organised in teams sharing intelligence and coordinating the analysis like [Sha05] and [Tea05] do.

2.5. Motivation and Usage

There must be a reason why there are botnets. As in many things humans do, there are different motivations to create and run a botnet. While things started as a game between IRC users, things turned out quickly to become nasty.

Depending on what the botnet is used for, the motivation can be of a monetary nature. A botnet can either be rented to a third party [Mon07] or it can be used to directly generate revenue with installing adware on the bot, running a phishing operation or it can be used for a DDoS attack [FPPS07].

Documents found on a bot can also be sold to companies interested in inside information, for market research, to paparazzi, terrorist cells interested in vulnerable systems, counter terrorism agencies or other parties interested in that data [FA07]. An interesting thought is the possibility for the botnet master to start a bidding war between different parties interested in purchasing the same (stolen) documents.

Another simple reason can be the longing for creating general havoc without any financial interests or motivations. Botnets are often used to find new victims which are then also turned into bots and added to the botnet.

There is the assumption that bots are used as anonymous proxies which can be used by the botnet masters to conceal their real identity. There is also the suggestion [FPPS07] that compromised computers are misused as bots for connecting to black markets.

Another motivation is the recruitment of an army of bots which can be used for cyber warfare, terrorism or political and random protest [WPSC03].

2.5.1. Botnet commerce. Monetary interest is probably the biggest motivator to run a botnet and there are many ways to earn money with bots. Since there are no costs of running a botnet - the unaware owners of the bots are paying all the bills - it is quite simple to earn money from running a botnet. According to [Mon07] a DDoS attack costs between US$10 to US$20 for one hour. The costs to run a DDoS attack for one day are about US$100. The botnet herders even offer 10 minutes of testing time in advance for free, just to proof their botnet's capabilities.

The bots can also be used to send out spam. Looking at the prices for spamming services, 10 millions spam-mails per day were offered for US$500 by one bot herder.

These were only the direct renting offers, other "business cases" apply. For example, using a DDoS attack to extort a company can not only bring in huge sums of dollars very quickly. But such an attack can also be successful when shutting down a competitor's network during a release of a new product or in similar scenarios.

[3]Bots sometimes have backdoors. Especially if they are bought from a third party bot developer. If a rivalling botnet herder knows about such backdoors, he can take over the bots.

The development costs of a botnet are difficult to be estimated. But the same rules apply as in the legal software industry. Depending on how much source code was reused from other bots and how many third party libraries were used, the development costs can be kept quite small. Developing new and sophisticated technologies on the other hand will become quickly expensive.

2.6. Legal liabilities

The problem with the Internet from the perspective of the law is that a botnet can be operated from anywhere in the world. Typically from a country where the risk of being tracked by the local law enforcement agency is low. While it is illegal to operate a botnet in Switzerland[4] and the UK as well as in many other countries, knowing the right person can help in not getting prosecuted [Tun07].

From a persecution of the botnets view, the current situation is worse still. Although the required laws are in place, botnet masters nowadays generally don't risk any legal actions taken against them. This has many reasons. From my own observations, the top reasons are:

- Missing know-how within the law enforcement agencies.
- This specific crime not being on the radar of the general masses.
- Politicians preferring to prosecute child pornography and other cases with a high attention.

Law enforcement agencies do not have infinite resources at their disposal and history shows that whoever decides on which cases to prosecute has no interest in investigating in botnets.

The principal problem with not prosecuting botnet masters is that ignoring them shows an implicit form of accepting what they do. This will lead to even more attacks and even more players joining the game.

[4]Somebody creating and running a botnet from Switzerland would possibly be liable to prosecution because of creating a malware, attacking and entering computer systems, unauthorised obtaining of data and probably the unauthorised change of data.

CHAPTER 3

Botnet History

3.1. Yesterday

3.1.1. The rise of malware. According to [Wikb], "malware is software designed to infiltrate or damage a computer system without the owner's informed consent. It is a portmanteau of the words 'malicious' and 'software'. The expression is a general term used by computer professionals to mean a variety of forms of hostile, intrusive, or annoying software or program code."

Malware, namely in form of worms and viruses[1], is known since quite some time. With the advent of networks, and especially the Internet, the landscape began to change. More and more computers were connected with each other and it became possible to connect from a computer from one end of the world to another computer at the other end.

3.1.2. International real-time communication. In summer 1988, Jarkko Oikarinen (who then worked at the Department of Information Processing Science at the University of Oulu, Finland) programmed and ran the first IRC server. IRC stands for Internet Relay Chat and is a protocol with which users can communicate in written language with each other. IRC was one of the first IM methods on the Internet and is still in use by special user groups. Many other IM products have evolved since then. While they have different names, the basic idea is still the same. Users can send text messages to a group of users or directly to another user.

IRC became quite popular and attracted a mostly young audience. Soon applications were written which did manage the server. But users also started to write software which can be used to play games, which helped manage a channel or which can be used to download and print the latest headlines from a news website. There were many ideas and therefore many different such applications. These helper applications were called bots.

Since IRC servers attracted a young audience, this lead to typical problems where the people evaluated what was possible and what was not. Since the Internet always pretends to give some kind of anonymity, the inhibition threshold was probably lower than in the "real life". One of the famous games of the day consisted of attacking the chat infrastructure so that a legitimate use was not possible anymore. The main tactic was to run the chat client many times on the same computer. This was called to run clones. These clones were connecting to the chat server until the server collapsed. At some point somebody must have had the idea to run clones from different computers to gain even more power. From there the idea of DDoS and using IRC to control the different clones must have evolved. When IRC operators started to exclude misbehaving users from their servers, these users started to fight back with even "bigger guns".

According to [Int06] the first malicious bot, PrettyPark worm, appeared in 1999. PrettyPark was very moderate feature-wise. While this worm was able to connect to an IRC Server and retrieve a set of commands, that set was very basic and only allowed a few commands to be executed on the bot.

This thesis uses IRC as an example for a C&C infrastructure to keep things simple. But of course there are many other chat protocols which can be, and are, used for C&C as well.

[1] Initially viruses and worms were not the same. The difference between those two characteristics of malware was that worms spread on their own while viruses need a spreading mechanism. Worms used to be less lethal to computer systems than viruses. Worms mostly tried to infect as many machines as quickly as possible. While viruses always infected the host and mostly deleted or corrupted files or the whole system on that host.
Things have changed in the last few years and today it does not make much sense to differentiate between viruses and worms [Int06], [WPSC03]. In our connected world worms and viruses joined forces. Worms "logic" is used to infect the machines and after a successful infection the worm part downloads a virus binary which is executed and takes then over from the worm.
Modern viruses either already contain the spreading mechanism usually found in worms or know how to download and run such functionality on their own.

3.1.3. Growing complexity of systems and networks. The Internet is constantly growing. Systems become increasingly dependent of each other and software running on these systems is becoming more complex. This trend has been noticeable some time and history shows that vulnerabilities (in complex systems) will always be exploited sooner or later.

Around the year 2000 the first DDoS tools surfaced. Some of them were called Trinoo, Trinoo Flood Network 2k (TFN2k) and Stacheldraht. While the theory of DDoS was known before, these tools allowed a wider audience to start a DDoS attack on a victim. They lowered the level of know-how which was needed before.

In the same period the Cult of the Dead Cow (cDc) created the Back Orifice (BO) client and server program. BO can be installed on a victim's machine, it is then turning invisible. Invisible means that the software is not visible to the normal user. And the software allows an attacker to completely control the victim's computer [XFO].

There is a famous quote from Edmund Muth, product manager for security from Microsoft. He said that BO "is not a tool we should take seriously, or our customers should take seriously" [CDC07]. As we learned from history, BO was one of the first malware which hides itself on the victim computer and provides a means for another person to control the victim computer remotely. Looking back, it can be said that BO was the beginning of what the botnets are able to do today. Interestingly enough, the same Edmund Muth later said [BW007] that BO "is the kind of software that could produce very substantial damage to someone's computer if it were installed". An interesting reconsideration.

Around 2001 the first "big" worms started to spread. And they did so very quickly. This is probably due to the fact that the Internet was growing rapidly at that time and many systems were running vulnerable software. Code-Red and other worms like Blaster and Sasser all exploited different vulnerabilities within software from Microsoft. Security was not a significant issue then as it is currently. As described in [MSB02] some researchers detected more than 359,000 unique IP addresses infected with the Code-Red worm between midnight UTC on July 19 and midnight UTC on July 20 2001. Other worms reached equal numbers and some even higher infection rates.

The DDoS tools, the BO control software and the spreading of the worms were a sign of where we would be heading to in the next few years. All these different technologies would migrate into one single application and become what we call bot clients.

3.2. Today

3.2.1. New targets, same attack concepts. When looking at attack targets it is possible to see a shifting of targets during the last few years. It is not the idea to look into all the attacks on the Internet in this thesis but to try to find connections between some types of attacks and the evolution of botnets.

The sophistication of attacks evolved with the protective measures and technological conditions. When it became clear that attacks from one single host do not work anymore, the attackers looked around for more powerful attacks and chose to use multiple computers to attack.

It is interesting to see that many attack strategies we see today were already used back in the early days of IRC. Among these strategies were social engineering attacks and distributed denial of services attempts. Users were talked into unintentionally closing their applications or into revealing information about themselves. They were flooded with connection attempts of clones[2] or with hundreds of private messages. Strategies which have survived and evolved till today, in one or the other form. Why change a strategy when it is still successful?

There is currently another shift of technique and target. When the botnet hunters started to automate the take down of simple C&C of botnets, the bot developers started to use technologies which are more difficult to defend against. This can be seen in decentralised C&C infrastructures which already are in use and most probably will see wider adoption.

Taking history into account and comparing it with the current situation, it becomes clear that the current situation is only the beginning of another evolution. The same observation was done by [Bar07]. With the advent of commercial possibilities and the realisation of the new capabilities, the botnet herders we see today are more experienced criminals and less script kiddies.

[2]A clone is a copy of the same IRC application running multiple times on the same host, connecting to the same IRC server.

3.2.2. Evolution.

Evolution of malware. Malware always was as good as the current technology allowed it to become. When networks still were rare, the most common malware were MBR viruses which spread with the use of floppy discs.

With the rise of networking and the Internet, new kinds of malware started to spread. Especially one kind of malware, called worms, started to become an annoyance. Things started significantly with the Morris worm and were brought to perfection with worms like Code Red, Sasser, and Slammer.

An important factor with all these infections is to realise that a software fault or design error is always detected and exploited sooner or later. Although there might be a patch available for a software fault before an exploit is written and released, there is no guarantee that every computer will be patched in time.

Malware following technology. There is a tendency to be noticed regarding the spread of a technology and new forms of malware. While the rise of the Internet provoked worms, the spread of Microsoft Word provoked the appearance of a special type of virus, the macro virus. A macro virus can be embedded within a document and gets executed when a user opens the document on her computer.

The same happened with the email program, Outlook, from the same company. While security researchers warned [Bon99] from such problems ahead of the release of a new and scriptable version of the email client, Microsoft decided to ignore (or downplay) these warnings and promptly provoked new set of email based viruses and trojans.

Mitigation strategies. Some of the described malware techniques were either countered by normal technical advancement or with technical countermeasures. Outlook as example was changed so that some specific attachment types could not be opened by the user anymore. This lowers the risk but does not remove the potential for further exploits. All in all it is interesting to see that some things never change but just evolve with the technology and the "needs" of the attackers.

3.2.3. Growing commercial interests. When looking at the evolution of malware it is interesting to see that commercial interests quickly became a main motivator to write and release malware. Most of the early viruses and worms were written out of curiosity and as pranks by computer professionals.

Over the time the commercial motivation grew and the family of malware written for commercial purposes grew. Viruses encrypted files on the local disc to make access impossible and blackmailed the file's owners to pay a fee to decrypt the files. Diallers emerged which dialled expensive phone numbers without the computer users consent. When the Internet grew in size and businesses and users moved to the Internet, phishing started to become interesting for criminals.

So there is a clear connection between the increased use of computers and the evolution of malware.

3.2.4. Learning from history, or not. As mentioned in earlier chapters, crime in general does not increase; it just changes the time and place where it happens. The Nigeria or 419 scams, which are also known as advance fee fraud, were already successfully used back in the early 1900. Back then that scam was called "the Spanish prisoner". The Spanish prisoner version of the trick was played on victims which were told that some wealthy person was imprisoned somewhere in Spain (hence the name). The victim should pay some sum in advance to free that wealthy person. Of course the name of the wealthy person could not be disclosed for security reasons but it was promised that once set free, that wealthy person would pay a generous sum to the person freeing him.

Exactly the same scam was played with fax machines some years ago. Fax messages were sent to Europeans and Americans. The stories on these faxes were always quite similar to the Spanish prisoner story. Except of being imprisoned, the wealthy person died recently and another person needed the help of the victim to get the sum on the bank account of the wealthy but dead person to a European or American bank account. To transfer the money, the scammer needed some advance payment to pay taxes and fees. Of course neither the promised sum nor the fee paid in advance ever returned back to the victim.

When email became common in Europe and America, fax machines became rare. The criminals adapted to this new situation and started to send out emails with the same story. And again many

victims fell for the scam. Some were held to ransom [BBC08] or lost their money, others have lost their lives [Reg07].

This is a wonderful example to show that where there is a market, there will be somebody to work it. Against this background it should not be surprising that with the growth of the Internet it is said that online crime and fraud also grew to become bigger than the offline crimes. This statement leads to the conclusion that if companies grow bigger, so does the underground economy.

There was always some progress in crime development. But the main factor which changed from the past is the rate of technological progress is increasing. While things took 100 years to evolve, we currently do these turnarounds in 5 years. The previous story of the Spanish prisoner proves that theory.

And worst of all, the vulnerabilities have fundamentally remained the same. Only the way of communication changed. This brings me to the next conclusion. If the only factor which changed is the technology, then the jeopardy which merchants and banks are facing nowadays on the Internet, because of phishing and e-fraud in general, result out of their own making.

Part 2

Botnet Lifecycle

CHAPTER 4

Botnet construction

4.1. Introduction

This chapter contains architectural information concerning botnets and a taxonomy. It contains details about how bots communicate, how they interact, how a botnet herder can command her network. There exists a white paper [Int06] from Trend Micro Incorporated describing the botnet components. While it describes different C&C models, rallying mechanisms, communication protocols and other observable activities, this chapter goes into much more detail.

The other chapters about the Botnet Lifecycle will build on the findings from this chapter.

4.2. To develop, to buy or to steal

Development is only one way of building and starting a botnet. While some bot developers write their own botnet software, others are acquiring bot clients through hijacking, stealing from other botnet masters or by other means [IH05].

4.2.1. Botnet Components. When looking at a botnet it is important to look at its different aspects. Figure 4.1 contains all components which make a botnet. These components will be discussed in the next few sections.

4.3. Architecture

The architectures of bots vary. This section discusses the different software architectures which were chosen by the bot developers in the past.

4.3.1. Single executable. Some bots consist of one single executable. This application manages the infection, the communication with the network and possible attack functionality. This architecture is not widely used anymore and most bot clients consist of many different executables which work together to infect new computers, hide their existence, communicate within the botnet or work on other tasks.

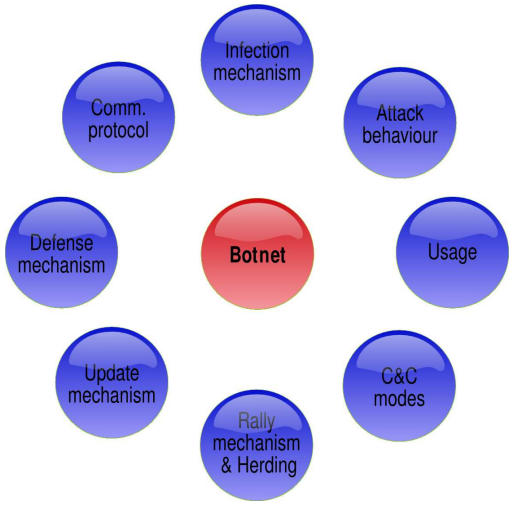

FIGURE 4.1. Botnet components

4.3.2. Multiple executables (Application or library reuse). Some bot clients install multiple executables. Some of the early IRC based bots used the mIRC software to manage the communication with the bot herder. While this strategy removes the need to program these specific features into the own bot, it opens up many risks to the bot developer. Among these risks are the problem of hiding complete installations of third party applications (like mIRC) and that the bot becomes dependent on third party software.

Of course it is possible for one single bot developer to create multiple executables himself. When choosing such architecture the developer creates different tools which work together. A system driver hiding the existence of the malware and an application handling the communication within the botnet.

4.3.3. Plug-in mechanism. Similar to commercial software like Photoshop, some bots contain the functionality to load and execute plug-ins. The idea behind this is that a bot can be updated [FdP07] without the need to rewrite the bot client itself.

To explain this with a scenario: After a new vulnerability is found in the Apache web server, a bot can be commanded to download a new plug-in which contains an attack algorithm written for said vulnerability in that version of the Apache web server. Such a feature allows the bot herder to regularly maintain and update the functionality of her botnet. And it keeps updates as small as possible.

4.3.4. Combinations. It is also possible to combine the above architectures. Having multiple executables working with plug-ins is not that uncommon. The idea behind such architecture is to have specialised and small executables only working on one aspect of the whole botnet but still having the option to quickly update small parts of the bot client.

4.4. Infection mechanisms & attack behaviour

4.4.1. Botnet size and growth. A botnet needs new bots to maintain its size and to grow in size. Computers running the bot client are constantly shutting down for the night, cleaned, taken off the network, e.g. a laptop in a cyber cafe, or something else happens which results in a botnet constantly having to find new members. A bot herder may want to replace bots which are leaving the botnet. Or he simply wants to grow the size of his botnet.

For example, if an IRC based botnet grows above its critical size for a single IRC server, the botnet herder can release an updated bot binary which splits his botnet in halve. One part of the bots staying on the old server, the other halve joining a new C&C server.

4.4.2. Social Engineering: Infections with interaction. Infection mechanisms can be divided into two groups:

- Social Engineering: Directly targets the user.
- Automated: Requires no human interactions.

Users can be tricked into downloading and running software. A user can click on an executable attached to an email or he can be made to go to a website and download and run an application from there. This strategy is tested and proven for some time. Even some of the latest worms trick users into visiting infected websites. There they automatically or manually download and then run the malware.

- Automatically means that there is no need for user interaction when the user's browser has a vulnerability [IH05]. This is also known as drive-by-infection. The Storm worm is known to exploit such vulnerabilities in Internet Explorer and Firefox.
- Manually means, that the user is downloading and running the malware manually.

A popular method to get users to manually download software is to put a message onto a website which says that the user needs some special additional software to be able to see the adult movies on that website. An example of such a website is shown in figure 4.2 on the following page.

Websites and Blogs like the one from F-Secure or others are constantly publishing examples [Jos07] of such attack attempts [Ale07]. Figure 4.3 on the next page is an example of a website trying to talk the visitors into downloading malware. This example appeared early in 2008 and was trying to infect visitors with the Storm worm binary.

FIGURE 4.2. Fake codec

FIGURE 4.3. Infected website

Browser stats (total)	
MSIE	11519 0%
Firefox	1800 0%
Netscape(mozilla)	190 0%
Opera	83 0%
Unknown	12 0%
Konqueror	2 0%

Modules state	
Statistic type	MySQL-based
User blocking	ON
Country blocking	OFF

Country	Traff	Loads	Efficiency
IT - Italy	13501 0%	351 0%	2.6%
FI - Finland	42 0%	0 0%	0%
US - United states	23 0%	0 0%	0%
DE - Germany	7 0%	2 0%	28.57%
CA - Canada	6 0%	1 0%	16.67%
GB - United kingdom	4 0%	0 0%	0%
IN - India	3 0%	4 0%	133.33%
HU - Hungary	3 0%	0 0%	0%
ES - Spain	3 0%	3 0%	100%

FIGURE 4.4. MPack management website

4.4.3. Automated exploit: Infection without interaction. An automated exploit is used when there is no need for user interaction whatsoever.

This type of infection mechanism is divided into two sub-types.

Local exploit. A local exploit is executed if an attacker has legitimate access on a computer system, usually as a user on that system. He can then connect as that user and exploit a local vulnerability. Typical local attacks are kernel exploits or to elevate user privileges to gain the status of an administrator.

Remote exploit. A remote exploit is successful if the attacker can exploit a flaw on the victim's computer over the network without the need to be logged on locally. Popular targets for scans for exploits are web servers (and web applications running on those web servers), Database, SMB and SSH services. If a service has a vulnerability, that vulnerability will be exploited.

MPack is a software package which was written to exploit vulnerabilities in the browsers of the visiting users. PandaLabs has written an interesting whitepaper [Mar07] on MPack. MPack is written in PHP and is updated regularly, approximately every month. Figure 4.4 shows a screenshot of the statistics website of MPack with information about the visiting users browser and country (screenshot inverted for better readability). There are a few source code examples of MPack in Appendix C.

4.4.4. Scanning mechanisms. When a worm tries to find other hosts to infect, it needs a strategy how to find those hosts. Worms try to spread as fast as possible before they get detected and before somebody develops a protective measure against them. When a worm starts to scan whole network segments for other vulnerable machines it will raise suspicion and the possibility for detection will grow. The worm attempts to be as stealthy as possible so that it can spread as far as possible.

As described in [SPW02] and [WPSC03], there are different possible strategies which worms can implement to address the problems of spreading. Although every bot in the botnet can be used to scan, only a defined number of bots will be used to scan for further targets. The main reason to do so is to expose only as many bots to detection as is absolutely necessary to successfully grow. Every scanning bot can potentially trigger an alarm and thereby threaten the whole botnet.

Hit-list scanning. With a hit-list scan, the worm does not try to find infectable computers on its own but uses a pre-compiled list of vulnerable hosts. This list is then worked through and every single computer on that list is attacked. A hit-list is usually made by the worm author. It is then either hard-coded into the worm binary or the worm can retrieve the list from within the botnet, from a web server, the Usenet or from another source.

Topological Scanning. The topological scanning strategy is an alternative to the hit-list scanning. A worm spreads with the help of a P2P system using the list of known peers to spread even farther.

Flash worm. The flash worm is another alternative to the hit-list scanning strategy. Before starting a flash worm the author of the worm does not himself create a list of vulnerable hosts but misuses a pre-existing list for his own purposes. This strategy allows for an even faster spreading of a worm since no scan is required as existing information is used.

For example, the worm author could use the Google search engine to find hosts running a specific version of a web server with a known vulnerability. Another possibility would be to misuse a list like DUL which contains IP addresses of dial up users. Using that list raises the probability to find hosts under control by home users. A similar strategy is described in [IH05] where netblock lists are used for similar results.

Permutation scanning. This strategy solves the problem of reinfection. Reinfection means, that a computer is attacked, although it already is part of the botnet. If a computer was infected previously, there is no need to waste resources reinfecting it. With a permutation scan strategy, a worm is able to detect if a computer was already scanned before.

Permutation scanning requires the existence of some kind of distributed coordination. Such a coordination mechanism can become quite complex and requires a good design and implementation so that it will be effective during spreading of the worm.

Passive scanning. A passive worm does not spread actively but runs on systems and waits for connections or probes being made from a third party. A passive scanning mechanism is mostly used by malware which takes advantage of another worm. CRClean is such a worm which waits on a computer system for a probe by the Code Red II worm [WPSC03]. If a Code Red II infected computer probes the CRClean infected computer, CRClean responds and launches a counterattack, cleaning the Code Red II infected computer and installing itself on the attacking machine.

4.5. Usage

4.5.1. General. Depending on the plans of the bot herder, a botnet can have different purposes and will therefore be used differently. While curiosity historically has been the primary motivator to run a botnet, the motivation has shifted from curiosity to financial gain [IH05].

There seems to be a coherency between the usage and the size of a botnet [RZMT06]. Small botnets exhibit greater portions of C&C communication than medium or large sized botnets. Large botnets are mostly used for resource intense tasks like attacking a third party system or a rivalling botnet.

4.5.2. Scanning. Scanning is necessary for the botnet to maintain its size or grow bigger. See section 4.4.4 for further details on why bots scan networks for infectable machines and how scanning works.

Botnets can also be used to scan one host from multiple bots. While attackers used to have one system scan another, some of the attackers moved to a distributed scanning behaviour. Appendix A contains an example of such a coordinated scan. The signature of that attack looks as if bots within a botnet were commanded to scan a single system. Which resulted in every host doing just a few scans, but taken together it was a normal brute force attack. Such scans are used to trick firewalls. Some firewalls are configured to drop all traffic from a single host after a defined amount of connection and login attempts. Since the bots share the attempts, such a rule will never apply, because the scan is distributed over many hosts.

4.5.3. Installing Adware. Adware is the short name for 'advertising-supported software'. It is important to distinguish two kinds of adware: malicious and benign. For example, Eudora is an email client which can be used in three versions. There is a free but feature restricted version, there is a complete version which has to be paid for, and there is an adware version which is free and complete but which displays ads to the user. Users installing and using this kind of benign adware are aware of the ads and they agree in being shown ads in exchange for not having to pay for the software.

But there is also the kind of adware which runs in the background and displays ads as pop ups in some regular frequency. Such adware is mostly installed by malware without the user's consent.

Companies writing such stealth adware programs are paying money for every installation of their adware. This means a bot herder can install adware on all his bots for financial gain.

It is not as common to install adware on bots as on infected but unmanaged hosts which are not part of a botnet because bot software should be run in stealth mode. A bot should be operated with as few annoyances to the user as possible to not risk the loss of the bot because of the user getting suspicious and cleaning the computer.

4.5.4. Online fraud.

Click Fraud. Some bots are used for click fraud. Click fraud according to [Wika] is "a type of internet crime that occurs in pay per click online advertising when a person, automated script, or computer program imitates a legitimate user of a web browser clicking on an ad, for the purpose of generating a charge per click without having actual interest in the target of the ad's link."

The intention of such clicks can either be to exhaust the marketing budget of a competitor [DSo07]. Or it can be because the bot herder owns the website containing the clicked ads and he thereby would be paid for every click. According to [IH05] click fraud accounts for a market loss of $320 million.

Phishing. Phishing emails are a special kind of emails which try to lure online banking users into revealing their authentication credentials so that the authors of the phishing email can use the victim's online bank account. They use these account to steal the money on it. But they also use accounts for money laundering and funnelling money through different accounts to cover their tracks. There is an excellent introduction into phishing in [Oll04].

According to [FdP07] bots are constantly improved so that the malware can counter the safeguards introduced by the banks. When the banks countered the phishing attacks with screen keyboards, the botnet developers quickly defeated this countermeasure and added screenshot capability to their bots to capture the screen when a user clicked with his mouse. The latest trend with phishing attacks is that the bot software will change the transaction data before it is encrypted and leaves the client computer, which defeats most of today's safeguards.

According to a recent news article [HEP08] damages due to phishing attacks are on the rise. There were 4'200 registered phishing attacks in Germany during 2007, a 20 percent increase from the previous year. Every incident is currently costing about 4'000 to 4'500 Euro, the sum was around 2'500 Euro in 2006.

Pharming. Pharming is an attack where the attacker tries to exploit flaws and vulnerabilities in the domain name system. There exist different techniques but all have the ultimate goal of actively redirecting a users traffic away from the desired location to a computer under the attackers control [Oll05].

A typical example of a pharming attack is when a client computer is secretly re-configured with new DNS servers. These DNS servers are not the legitimate servers managed by the users ISP but these are under the control of the attacker. If a user enters a website into his browser, the browser asks the (false) DNS server where to find the website with that name. The user is then sent the wrong IP address and the browser is contacting the web server controlled by the attacker. Because of the nature of DNS this will not be noticed by the user.

4.5.5. Data theft and packet capture. The bot indexes all the data on the hard disks it has access to. The intention behind this attack is that the bot herder is either interested in documents and data himself or he sells that information to third parties on the black market [FA07].

Software running on a bot scans the hard discs from the system it is running on and searches for the data its master is interested in. Some bot clients scan for contact addresses in IM contact lists, stored emails, the Windows registry and in other places [IH05]. Some bot herders are even interested in software registry keys and harvest these so they can sell genuine licence keys on the black market.

Besides scanning the file system, some malware also scans the network for specific traffic. Password and other traffic sent via the network will be captured and sent to a central storage for further analysis.

4.5.6. Data tampering. After scanning the available hard discs, a bot can modify files. There are many reasons why tampering with the data can be interesting.

Changing reports and numbers in spreadsheets can result in huge problems when such a change goes unnoticed. A companies report can be changed before it is sent to the tax office. Employee's information can be changed, leading to wrong testimonials and certifications. Software can be changed and a back-door can be added to a web server before it is being delivered to customers.

There are many more reasons why somebody could be interested to tamper with files, and the results can be devastating for individuals as well as for companies.

4.5.7. DDoS. Botnets can be used to disrupt services of third parties. This can either be done to fight other bot herders [Gos07] or personal foes. But this can also be used to blackmail businesses as was done during the European Football Cup 2004 as reported by a German newspaper [Bra04]. Perpetrators sent a letter to the online betting company mybet.com. They asked for US$ 15'000 or the website would experience a DDoS attack. When the money was not paid, the website was taken off the net with a targeted DDoS attack.

Another website specialised in online gaming operated by Fluxx AG of Hamburg was blackmailed in 2005. Fluxx AG was asked to pay 40'000 Euros or experiencing a DDoS attack. Instead of giving in on the demand they offered the same amount for information which can lead the police to the blackmailers [Mor05].

Of course there are many other reasons why a DDoS attack could be started. It is interesting to note that commercially oriented DDoS attacks are mostly targeted against high profile websites with special content like porn sites or betting services. The probability that the operator of a porn site will contact the police because of a blackmail attempt is much smaller than the probability that an attacked bank will be contacting the local law enforcement agency. It is all about efficiency and the risk of being caught.

4.5.8. Proxy and anonymous hosts.

Proxy. Bots can be used to send out spam [DSo07]. Either the botnet herder sends his own messages, or he rents his botnet or parts of his botnet to third parties. In such a scenario the bots are used as spam proxies relaying messages from the botnet user to the receivers of the spam.

But bots can also be used as a proxy for other traffic:

- As an anonymiser for the attacker. The attacker routes his connection over such proxies which makes following his trails extremely difficult. Especially when he uses multiple proxies in different countries.
- For different services like DNS and HTTP. Using proxies like that it becomes difficult to trace the original source of information. Fast-flux networks work with a similar concept [All07]. There is a quick introduction to fast-flux networks in section 4.7.4.

Anonymous Host. There are many reasons why somebody may need an anonymous host [IH05]. The important factor is always that the real identity of the person administering the anonymous host is unknown. And that there is no easy way in revealing it.

The following list of reasons is not complete but it gives an idea of the possibilities:

- Web or FTP server to publish malware [Bar07], porn, pirated movies or warez[1].
- Web server to recruit money mules or to be used as a phishing site [KRH07].
- Web servers to run online pharmacies or other scams.

4.6. C&C modes

The different C&C modes can be grouped into three sections: centralised, random and distributed [CJM05].

4.6.1. Centralised. The centralised C&C mechanism as shown in figure 4.5 is probably the classical version. The bot herder commands her botnet from one central location. The problem with this mode is that once the central C&C is taken down, the whole botnet is headless and uncontrollable.

[1]The word 'warez' is used by the software pirating scene to describe illegally distributed software.

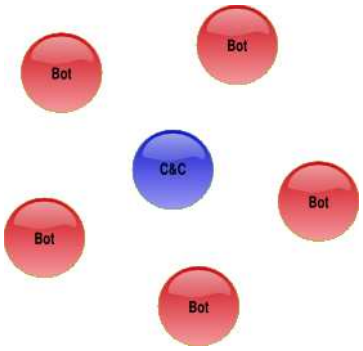

FIGURE 4.5. Centralised mode C&C

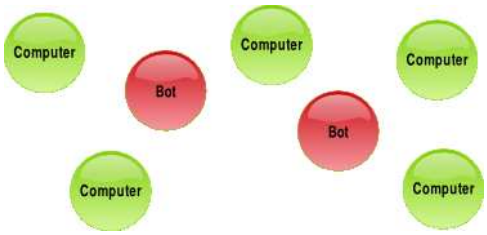

FIGURE 4.6. Random mode C&C

4.6.2. Random. With the random mode the bots do not know each other and there is no control traffic between a central station and the bots within the botnet. This mode is illustrated in figure 4.6. If the botnet master wants to send a command to his botnet he would have to randomly send the command to network segments and hosts within the whole Internet.

The interesting aspect of this mode is the similarity with terrorist cells where members of a cell do not know the members of other cells or their superiors. Since there is no (or spare) communication, the communication can neither be intercepted nor can there be communication injected [NA05].

4.6.3. Distributed. To avoid to lose the whole botnet when the C&C is taken down, bot developers started to design distributed C&C modes as illustrated in figure 4.7. The earlier versions made use of multiple C&C so that once one was taken down the next could be used. This has the problem that such architecture mostly depends on pre-defined IP addresses or domain names. Taking down such a botnet is only a little bit more complex than the centralised model since it is still a fixed small number of servers.

With the advent of distributed C&C based on P2P technology, things are getting very interesting. The idea behind such architecture is that the botnet itself can become independent. Every bot in the botnet can act as the C&C controlling the other bots. There is no need for dedicated controllers.

There is an advantage with this mode that there is the possibility of coordination and load balancing [KL03]. Bots can share the work requested by the botnet master. Some bots can be used to infect new machines, while other bots are used to send spam and another group is used as hosts for a phishing scam.

Fast-flux networks are using the distribution of their nodes to make it extremely complex for botnet hunters to bring down the network.

4.7. Rally mechanisms and Herding

Bot herders need ways to build and advance their network. Newly infected computers need to join the botnet so they can be controlled by the botnet master. This section is about the different rallying mechanisms.

4.7.1. Hard coded IP address. Bots can join the network in contacting a C&C with a predefined IP address. This is an extremely simple rally mechanism to be programmed. There is

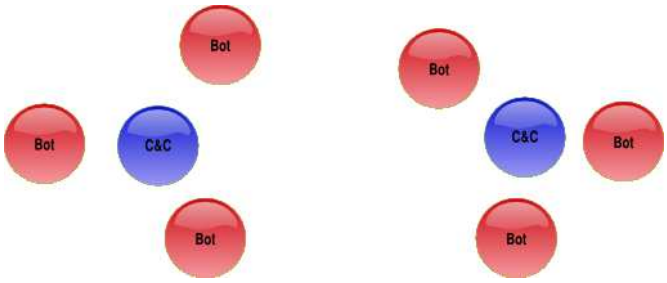

FIGURE 4.7. Distributed mode C&C

no need to program DNS lookups or other support code. But it has a few drawbacks. The hard coded IP address is a single point of failure. When that IP address is unreachable, the rallying mechanism fails. This can happen pretty quickly when there is a network failure or power outage at the server's site. The result is an uncontrolled botnet consisting of infected bots with nobody giving them any commands.

There are possibilities to mitigate these risks. Bullet proof hosting is one of these strategies where the ISP of the IP address in question sells the additional service of not shutting down the IP so that taking down the IP address is not easy. For example, the Russian Business Network is said to offer bullet proof hosting for such purposes [Dan07].

4.7.2. DynDNS. Instead of predefining an IP address, the bot can contact the C&C with the help of dynamic DNS. DynDNS is mostly used by computer users having no fixed IP address. When changing the IP address, typically because their Internet connection was reset and they received a new IP address from their ISP, these users then update the DynDNS entry with their new IP address and their DNS record will be updated accordingly. Resulting in a domain name which "follows" the owner. The same mechanism can be used for a C&C. Once the C&C is successfully taken down, the botnet master just needs to move to a new IP address and change the DynDNS record accordingly [2].

Programming malware which makes use of dynamic DNS is slightly more complex than using hard coded IP addresses. The benefit of this method over the hard coded IP address is that it becomes more difficult to take down a botnet because the C&C can change IP addresses. But generally it is only slightly more difficult to take down the DNS server than to take down some predefined IP address.

4.7.3. P2P. The herding mechanism in P2P networks depends on the P2P technology which is used. Some P2P networks require central servers to log into the network. The central server tells new coming nodes where to find other nodes. This mechanism has the problem of introducing single points of failure. If such a central server can't be connected to, the initialisation mechanism won't work anymore and new nodes can't join the network.

Other P2P networks solve this problem by supplying a list of established nodes to the nodes entering the P2P network. Every new node will get a list of known nodes which the new node will contact to establish its status within the network. This mechanism has no single point of failure, but since a list of known nodes is shared during initialisation, it is possible to learn about other nodes participating within the network. This can be a problem within a botnet where the bot herder may want to hide information about the botnet's size and members.

4.7.4. Fast-Flux Network. In a fast-flux network, a single domain name is assigned to hundreds or thousands of different IP addresses as described in [All07]. But instead of keeping these assignments stable, the DNS records are constantly updated, sometimes every other minute or even quicker. When a user is constantly visiting a website hosted on a fast-flux network, he will always be directed to a new IP address because the DNS record is pointing to a different address since the last visit. This is achieved through a very short Time-To-Live value for the DNS records.

[2]Dynamic DNS providers like dyndns.com are providing services where computer users can rent a domain name which is always pointing to their computer, even when the computer is regularly changing its IP address. Since dynamic DNS names often are used for malicious purposes, the providers of such services introduced acceptable use policies where they state that their service may only be used for legal purposes.

Running a botnet within a fast-flux network has many advantages. Fast-flux networks are sometimes used for domains which are solicited in spam emails. Users are then contacting web servers running within the fast-flux network. As long as there is no possibility to unregister the domain name or getting access to its DNS servers, it becomes very difficult to filter or block access to that domain. Things get worse when the DNS servers are hosted in the fast-flux network as well. This means that the whole network becomes completely dynamic with the idea to minimise the risk of the network failing when parts of it are shut down. In this scenario it becomes nearly impossible to do anything against the fast-flux network. The only option is to contact the domain name registrar which issued the domain name and have him revoke the DNS registration.

4.8. Update Mechanisms

According to [SPW02] the ease and resilience with which an attacker can control and modify his botnet has serious consequences for both how the threat of a deployed botnet can evolve and the potential difficulty in detecting the botnets presence and operation after the initial infection. Bots are precious and need protection, an update mechanism can help in quickly adapting to emerging threats.

4.8.1. Reasons for an Update Mechanism. There are several reasons why a bot could need an update mechanism. Interestingly, many reasons are already known from and implemented in regular commercial software and are now constantly adopted by the makers of malware.

Software Fix. If a bot contains an update mechanism, it becomes possible for the bot herder to have his bots install an update to fix a software error. Some viruses and worms already contained programming errors and implementing an update mechanism allows for a later update to fix errors not noticed in advance.

Another point is that botnets, especially the P2P variants, are very difficult and complex distributed networks which can't be fully tested in advance. So if there emerge some problems during the roll-out, the bot herder has the possibility to update his bots with an updated binary.

Change binary to avoid detection. With an update mechanism it becomes possible to regularly update the bot binaries in very frequent cycles to work around the AV software companies. AV software historically scanned files for known patterns. AV vendors catalogue signatures of known viruses and release so called signature files with their scanners. The virus scanners check the signature of every file on a disk for known signatures. If a known signature is detected, the virus scanner knows that it found a virus and acts accordingly. Changing the bot binary regularly means that it changes its signature, therefore avoiding detection by its signature until the AV vendors catalogue the new variant as well.

AV vendors started to develop and use other techniques to detect malware. But unfortunately the tactic to regularly change the signatures of the binaries is still very successful. Section 5.4.2 contains details about the workings of AV software.

Add features. Some bots use the update mechanism to regularly update their repository of exploits and attacks. If a new vulnerability in an application is detected, the botnet herder can update his bots to start to exploit said vulnerability.

A bot developer can update his bots with new communication channels. He can start his botnet as a typical IRC based botnet and then migrate to a more advanced technology like P2P when he starts to understand the technology and the workings of a distributed system.

4.9. Bot Defence Mechanisms

Bots contain information about the botnet and about how they work. To protect that information bot developers implement defence mechanisms into their bots.

4.9.1. Unload programs. Some bots have a mechanism where the bot detects the launch of an application which is on the bots own black list of unwanted applications [PSY07]. The bot then terminates the starting application. Appendix B showcases such a mechanism. While this example represents a simple version, there are more complex variants which are less obvious and more difficult to detect.

4.9.2. Stealth. Bots can make use of rootkit functionality. Using rootkit techniques will hide the bot process from the list of running programs on a computer. The bot client can also hide its own files and folders from the user. Browsing the file system the user would get wrong information.

Some audio CD vendors sold audio discs with rootkit technology on it. The software installed itself on a customer's computer upon inserting the CD into the computer. The software then made sure that users could not use the CD-ROM drive to make (legitimate) copies of said audio discs. Such rootkit software was detected on audio CDs from Sony. Some weeks later, malware authors already misused the same (already installed) rootkit software to hide their own malware as well.

4.9.3. Attacking the researcher. Some botnets started to attack hosts which were too inquiring. When researchers tried to understand the workings of the Storm botnet DDoS attacks were made on the hosts which they worked on. While such a strategy is not a bullet proof defence it will scare away some researchers and companies which can't risk being the victims of a DDoS attack.

4.10. Communication protocols

4.10.1. Push vs. Pull. This section is about how bots do communicate with the botnet's C&C and with other bots (P2P inter-botnet communication). There are two methods in which the communication between bot and C&C can happen: push and pull. Push means that the C&C decides when it is time to push a new command to the clients. This requires a persistent connection between bot and C&C. Pull means that the client regularly connects to the C&C and asks for new commands. For this method it suffices to only have an occasional connection between the bot and the C&C.

Table 4.1 shows which protocol supports what communication method. These protocols where chosen because they are the representation of the most used protocols. IRC and IM could be combined since they share many characteristics. But they are separated in this list since IRC is the traditional method and is used very often. VoIP was added as a separate entry to show how concepts for heavily used protocols can be applied to less used protocols as well.

Protocol	Push	Pull
HTTP		x
IM	x	
IRC	x	
P2P	x	x
VoIP	x	x
others	x	x

TABLE 4.1. Overview of protocol behaviour

4.10.2. HTTP. Within an HTTP botnet all the bots regularly connect to the C&C and ask for new commands. The server identifies the bots and sends all new commands to the bots.

Since web traffic is very common, this can be a very stealthy method to communicate. This is an interesting communication protocol and gaining in popularity mostly because it is quite simple to be implemented and nowadays it is very difficult to block web traffic. Many companies need to allow web traffic for their employees as well as for their own web servers. Due to this fact, bot clients running on employees computers can often communicate with an external C&C and an in-house web server can be misused as a C&C.

Instead of HTTP the botnet can also make use of HTTPS. The mechanisms will stay the same, except for the communication which is encrypted and thereby more difficult to analyse.

4.10.3. IM (Messenger, ICQ and others). When using an IM protocol, the bot connects to the IM network and contacts the C&C within that network. The C&C can then either send new commands to the bot as a private message, or the bot could join discussion rooms where the C&C can send one command to a group of bots.

It is important to note that IRC as a communication channel has similar features and functionality. It is also important to note that some IM protocols, like the one used in the Jabber[3]

[3]Jabber is an XML based IM protocol and software released under an open source licence.

server, are making use of HTTP for transportation and are miming HTML content. This is making detection complicated.

4.10.4. IRC. IRC based communication is the classic technique. Bots connect to a server and sometimes join a channel. Once a bot is connected to the server it can receive the commands through private messages from a central authority (or the server directly) or through the channels it joined. Using IRC for C&C is quite common and quite easy to be implemented since there are many freely available libraries and sources which can be used to speed up the development.

There is one problem, IRC communication is quite unique and there exist many tools which can detect and analyse IRC traffic automatically. To mitigate this threat some botnet developers started to slightly modify the IRC protocol and the IRC servers to keep snooping eyes out of their servers and consequently making an analysis of the traffic significantly more difficult.

4.10.5. P2P. P2P is the umbrella term for several different technologies. The term P2P describes that there is no need for a central server coordinating the connections and managing the network. P2P can be understand as an overlay network, sitting above the IP network much in the way a VPN or WAN connecting many branches of a company does.

There exist different technologies to achieve the goals of an overlay network. WASTE [WAS] and Kademlia based networks [MM02] are amongst the most widely used. Basically they use all a list of peers which is constantly managed and updated. Using hashes and routing tables, overlay networks are routing requests on their own without the help of DNS.

This is the most complex communication method. Distributed networks can be unstable which must be addressed when wanting to design a stable overlay network. Then there are other problems like security and trust. Nodes must know each other and route commands and information to their peers only. Instead of sending commands directly to the bot clients, as is done in the other communication modes, a P2P network shares commands among the bot clients. Because of this, commands are not sent to every single bot client, but are distributed in the whole botnet. Commands are thus not immediately executed but executed asynchronously.

P2P has the added benefit that the bot herder can inject the commands anywhere in the botnet, without the need for a central C&C server and without revealing his position. This consequently means that bot clients must only accept commands which are originating from the legitimate bot herder. Otherwise everybody could inject commands.

4.10.6. Voice over IP (VoIP). Using VoIP to control and command a botnet is more of a theory than used in practice. The theory describes the botnet using the centralised control model. Of course it is also possible to use the other C&C models.

Implementing such a communication protocol would be quite complex and there would be a few problems to be solved for a working system. One of the problems is to implement the bot communication (and infection) so that it complies with the protocol so that communication works in an already deployed system. But this is an interesting thought nevertheless. Especially since many companies are depending on VoIP traffic, in the same way they depend on HTTP, and can't block that traffic for the same reasons.

4.10.7. Others. It does not matter which communication protocol is used. DNS can be used or something absolutely new. There must only be some way for the bots to communicate with either each other or a C&C so that they can receive new commands and orders.

4.11. General effects of architectural choices

Some aspects of the chosen architecture have a direct influence on the strategies to detect, capture, analyse and annihilate a botnet. Table 4.2 shows which design most influences which step in the lifecycle of a botnet.

That table will be used as a reference in the following chapters about the detection, analysis and annihilation of botnets and the capturing of bots. An "x" means that there is a significant influence, an "o" means that there is some influence.

Architecture	Detection	Capturing	Analysis	Annihilation
Infection mechanisms	x	x		x
Usage	x		o	
C&C mods	x		x	x
Rally mechanisms	x	x	x	o
Update mechanisms	x	x	o	x
Bot defence mechanisms	x	x	x	x
Communication protocols	x		x	o

TABLE 4.2. Influences of architectural choices on the lifecycle of a botnet

CHAPTER 5

Botnet detection

5.1. Overview

When connecting a computer to the Internet, it will be scanned and attacked immediately. Most of these attacks are automated scans from infected computers which are commanded to search new victims.

The same happens with public email addresses. Once an email address is used and thus known to others, the chance is high that it gets harvested by some malware and used to send spam to. Some of these spam mails contain viruses or attempt to lure the recipient onto an infected website with the goal to infect the recipients computer with malware.

The magic of botnet detection is to find a structure in these scans and spam mails. It is about finding new and formerly unknown botnets and new bot clients.

5.2. Effects of architectural choices

Detection of botnets can be easy or very difficult based on the architecture of a botnet and its bots. The main strategy behind detecting a botnet is to find anomalies in the network and on a host itself. Depending on which architectural choice was taken detection will work different.

5.2.1. Infection mechanisms. The infection mechanism is a critical factor because the infection is the first step in the lifecycle of a botnet. If the infection process is not fast enough bots will be cleaned faster than new computers get infected. If the infection rate is too high the bot will raise suspicion because of the unusually high traffic it produces and will be detected too early. It is the ultimate goal for the botnet designer to find the right infection rate for the right use.

5.2.2. Usage, C&C modes, rally mechanisms and communication protocols. Depending on the usage of a bot, the bot will generate noise on the host itself and on the network. When the bot wants to go unnoticed the bot herder needs to find the right amount of usage and silence.

If the botnet is using widely known C&C communication protocols, ports and servers, intrusion detection software will usually find the bots very quickly.

5.2.3. Update mechanisms. If the bot update mechanisms are not fast enough, AV software will be updated quickly enough to detect and identify the bot client. Updating the bot too quickly increases the risk of being detected because of the unusual network traffic.

5.2.4. Bot defence mechanisms. Much can be gained with using the right defence mechanisms because AV is not the full solution. If the bot can deactivate the AV software without the user noticing, the user will never become suspicious because he thinks the AV software is running (and detecting malware) as expected, instead the AV software is deactivated by the bot client.

Another problem with AV software is that it cannot detect if it has been installed on an infected system. For example, some bot client is installed on a computer before AV software is installed. A week later the user is installing an AV client. Because the computer already is infected, the bot can control the installation of the AV software. The bot can place files with known signatures of different malware on the host and have the AV software recognise that bait while hiding itself. The user would then scan a system, find some malware, have the AV software remove the bait and feel safe again. Without noticing the bot client which still runs in the background.

5.3. General detection mechanisms

5.3.1. Human Intelligence. Although not directly anomalies based, human intelligence can be useful in detecting a botnet. Especially when the botnet owner or bot herder starts to brag about how big his botnet is and on which systems he owns bots [BS006].

5.4. Local detection mechanisms

5.4.1. Behaviour detected by user. Users working on a system could notice that the computer is behaving strangely. The problem with users detecting malware is that once they detect that something is fishy the malware already is running and has its defences prepared. There is also the problem of users suspecting something which is usually not there, which results from a lack of know-how and experience.

Generally said, users suspect an infection because of these factors:

- There is unknown software installed or running on the system. Unfortunately malware is often named like system utilities and users typically don't know which software is legitimate and which is not.
- A system is becoming slow. Unfortunately speed is a very subjective feeling and computers which are not regularly maintained have the tendency to become slow over time because users install utilities and services run in the background (completely unrelated to bots).
- Suspicious registry keys (Windows) or configuration files.
- Strange system warnings or errors.
- Commercial popups or changed website content or a new start page in the browser.

Unfortunately people tend to act irrationally when facing unknown threats. Our perceived risks rarely match the actual risks, as described in [Sch03]. Malware exists which displays commercial popups and ads in webpages for virus removal tools when running on an infected host. Criminals authoring and distributing malware like that have a good chance of turning in quite some profit because users tend to pay for the first software which promises to clean up the system and bringing it back to the state it had before infection.

This approach is even more successful when the malware injects links to websites for its own "removal tool" when the user is searching for help on Google or another search engine. This is a very simple form of social engineering where the malware author makes use of the stress situation of the user.

5.4.2. AV Software. According to tests published by the German newspaper c't the effectiveness of AV software has fallen off, and more and more malware can now slip past these barriers [hei07]. The article states that AV software protection is now worse than a year ago. Mostly thanks to malware becoming more complex and the massively growing number of new malware being released daily.

AV software has two different ways to search for malware, the reactive and the proactive approach. Most AV software nowadays contains a mixture of both approaches for best performance.

Reactive. AV software historically used signature based analysis. Which means every malware was defined with some unique signature (binary, keys, location) and this signature was then used by the AV software to inspect a system for an infection. This method is only as good as the signature files. New malware typically can't be detected automatically and the AV vendor needs to (manually) create a new signature for every new malware file.

Things get more complicated because new malware often contains some kind of packer or compiler which makes sure that the signature of the malware is changing regularly. Requiring the AV vendors to manually analyse the code to make sure every new signature variation can be detected as well.

There is also another problem with this method. The amount of malware variants is increasing. One of the reasons for this is the availability of virus creation toolkits and source code for malware on the Internet. It is quite simple to find and download source code for malware. AV vendor companies are hence struggling to keep up with their signature databases. According to [Hru08] F-Secure's database had grown to include over 500'000 examples of malware by the end of 2007.

Proactive. Since early days of AV software heuristical scanning was implemented to mitigate the downsides of signature based malware detection. Some algorithms tried to find similarities and known patterns in files to try to identify potentially malicious (and still unknown) software. The problem with this approach is that the false-positives rate can be quite high.

In the last few years behaviour blocking technology was actively developed. Instead of trying to identify patterns in a file, such a behaviour blocker system (or HIPS) is implementing sensors directly into the host system. The HIPS can then analyse what commands an application is executing and prevent the execution of a possible malicious activity.

But the main problem remains with both proactive strategies. Proactive detection strategies don't know the scanned software and try to classify an application based on assumptions. Which means that there is always the risk for false-positives or malware which is not detected because it does not behave maliciously (false-negative).

5.4.3. Host-Based Intrusion Detection System (HIDS). Most of today's HIDS regularly check the file system for modifications. These checks are based on cryptographical signatures (hash) of files which are compared with known good values. If the current and last known good signatures of a file do not match, the administrator will be warned. This file change detection mechanism does work quite well, no matter what software is used, but it needs a knowledgeable system administrator.

If a bot is installing itself on a computer and changing the system so that the bot gets started with the system the next time the system is booted, then a HIDS is able to detect the changes to the system boot up procedure and warns the administrator of the changes.

The main problem with such a file comparison is that the database of known-good values needs to be trustable. The whole protection mechanism of file change detection is based on the validity of the known-good hashes. If a HIDS gets installed after an infection occurred, and the malware is able to manipulate the known-good hashes, the infection will never be noticed.

Besides checking the signatures of critical files, HIDS also check for running but unknown software and for network ports which are unknown but which are in use. So a HIDS is basically software trying to detect many different anomalies within a system. Something which is also done by AV software. It is no surprise that both types of software do also share the problem of false positives. There are scenarios where an intrusion detection system is warning of a possible intrusion even nothing really happened. This can become annoying if it happens in the middle of the night. Too many false positives can lead to the software being de-installed or in the real attack happening unnoticed.

HIDS have another big problem. They are running on the same host they try to protect. This means if an attacker successfully captures a computer, he is usually also in the position to fool the HIDS. The attacker can use the same procedure to trick the HIDS as he can apply to fool the administrator.

5.4.4. Virtual Machine Detection. Virtual machines like VMware, Parallels or VirtualPC are often used to analyse malware. The main reason for doing so is the easy handling of virtual computers. Once a virtual machine and a virtual host are configured, the virtual host can simply be backed up. After an infection has occurred the system can be halted, analysed and overwritten with the clean backup copy. Making the process of analysing malware quite simple.

Malware authors noticed that virtual machines became a trend in the AV business and have begun to develop countermeasures. Some malware is now able to detect if it is running in a virtual environment or on a real host. These defence mechanisms all work on the concept that every virtual machine can be detected in one way or another [**?**]. Section 7.4.2 goes into more details.

Once malware detects that it is running in a virtual environment, it starts its defence. To protect itself, the malware either shuts itself down or behaves differently than when running on a clean system. There has evolved an arms race between developers of virtual machines and malware authors over detecting virtual machines.

This whole topic shows that malware researchers need to make sure that what they are analysing is what is also happening in the wild. And this also shows that the malware researchers will have to look into other methods on how to clone computers so they can benefit of the features of a virtual machine without the drawbacks of having the host being flagged as suspicious by the analysed malware.

5.5. Network detection mechanisms

5.5.1. Classical network analysis. Classical network analysis is about detecting connections to unknown hosts or unknown ports. And to detect connections to hosts and ports which are flagged as suspicious or malicious.

This classical view has the problem that stealthy communication will go unnoticed. Because botnet hunters are scanning networks for IRC based communication, bot herders start to use other communication channels like HTTP. HTTP is the basis for web based traffic and quite common nowadays. Botnets using HTTP as C&C protocol will thereby go unnoticed as long as they keep their traffic down. And botnet hunters can't just block every HTTP based traffic because the ports and protocol used for HTTP traffic are quite common and in wide use.

5.5.2. Communication analysis. Communication analysis is quite common and mostly used in collaboration with the classical network analysis. If a suspicious connection gets detected, a botnet hunter starts to analyse the communication.

Because IRC and HTTP traffic are very common, there exist many tools which can sniff and help to analyse such network traffic. Based on the analysis of the traffic, a botnet hunter can tell if a connection is suspicious or not. In [GH07] a method is described which identifies IRC based botnet traffic based on evaluation of the nicknames used within the IRC communication.

The downside of this method is that bots using a slightly different version of the protocol will require some additional work by the botnet hunters. If the bot herder manages to make his changes to the protocol subtle enough, the traffic will probably go unnoticed.

5.5.3. Communication signatures analysis. Because the classical network analysis has downsides, some researchers started to search for other methods to find and track C&C traffic. During interbotnet communication and during infection, bots are exhibiting unique communication signatures. This fact can be used for a mechanism to (early) detect infections within a network.

Some projects like [GH07] detect IRC based botnet communication through normal network analysis. And there was research done by [KRH07] which developed an anomaly-based passive analysis algorithm that was reported to have been able to "detect IRC botnet controllers achieving less than 2% false-positive rate".

Such detection mechanisms are not IRC specific. Software can analyse any communication behaviour as described in [PSY07]. It is interesting to see that the behaviour of a computer when being infected represents some kind of unique fingerprint.

Similar experiments were done by [SWLL06]. But instead of having to use sensors in every single network their results suggest that one can find evidence of botnet activity by monitoring network traffic only at various core locations in the Internet.

5.6. Global detection mechanisms

Some botnets show anomalies on a global scale. Fast-flux networks are such an example [All07]. Such networks are having an extremely short Time-To-Live (TTL) value in their DNS Resource Records. Looking globally at regular changes of DNS records or TTL values could point to a fast-flux domain which most probably is used for malicious and illegal purposes.

Such a global behaviour can be detected without the need to be present on one specific system. This is quite different to the host and network detection mechanisms discussed before.

5.7. The threat of detection

I think it is a good question to ask if the detection of a botnet is a problem to the detected botnet. But the question of what happens to a botnet if it gets detected cannot be answered generally. Much of this depends on the botnets architecture, the timing and what the botnet is used for and what the botnet master tries to hide (IRC server, network architecture, communication protocol and so on).

While the bot developer can try to hide architectural facts, there is no perfect protection. There will probably never be. To mitigate the risk of having the innards of a botnet being inspected by a botnet hunter, the bot developer needs to apply some architectural designs which minimise that risk.

Another question to ask is the influence of the time of detection. There are some situations in the lifecycle of a botnet where the whole network is more vulnerable than in other situations. One

such precarious moment is during the initial release and growth. A botnet which has not reached the critical mass is at risk of being disrupted prematurely. Removing important nodes or elements could stop the growth and kill the network.

It is always important who is detecting a botnet. Depending on the motivation and skills of the person detecting the botnet, the consequences will be different. While some researchers just try to hide in a botnet and to learn about the bot herder and the workings of a botnet, there are others which try to take over a botnet, to use it themselves, or forward the information to law enforcement.

There is the question of what is a botnet used for. If the botnet is used primarily for spamming purposes then detection will probably not be much of a problem. Cleaned hosts can be compensated with infecting new hosts. If the botnet is used for a sneak attack, like a blackmailing attempt or similar, then it can be vital to keep a low profile until everything is over. Risking the exposure of the botnet can endanger the attack.

I think detection is a problem which every botnet faces. Every botnet will be detected eventually. But many of the threats to the botnet described above can be mitigated with a good architecture:

- Trust mechanisms help in keeping unauthorised commands out of the botnet.
- Encryption can help in keeping out snooping eyes.
- Stealth technologies and environment aware bots can make the analysis of a bot client much more difficult.
- Code obfuscation can make decompiling even more complex.
- Regular updates to the network protocol can make earlier findings useless.

I believe that a botnet master can be one step ahead of his "enemies" if he watches his steps and stays agile in that he regularly changes parts of his botnets architecture. A similar behaviour can be detected with the human body and viruses. Once the body is infected, it creates defensive forces which will protect it from a further infection of the same kind. But viruses keep on mutating. And once a mutation is different enough to the last virus, the body will become infected again. The defensive forces will be useless and everything restarts from the beginning.

I strongly believe that as long as a botnet master is not making any wrong moves, stays anonymous (using a few hosts to hide his real identity and source address) and does not act suspiciously (bragging around at the local bar) he will be safe. Unfortunately.

CHAPTER 6

Bot capturing

6.1. The capturing process

6.1.1. Access method. To analyse the innards of a botnet it can be informative to study the source code or binaries of a bot. There are several methods to get access to the bot binary or to its source code. A passive and an active method.

Passive. The passive method is easier since it is all about waiting. Some recording mechanism is set up to wait for an infection and once the infection occurs and the malware gets delivered the recording stops and the malware gets analysed. A honeypot is a typical implementation of such a passive capturing method.

Active. The active method is far more complex. It is actively searching for malware and analysing everything which is detected during the search. While the active method is more complex than the passive method, the active method finds malware which is more aggressive and stealthier. Such malware would probably not have been found without an active capturing method.

A possible implementation of the active method could scan all incoming mails of an email account receiving much spam. Once a suspicious email is received, this mechanism could analyse the mails content and then try to download the malware without the need for user interaction.

6.1.2. Dropper. There is a problem with capturing bot clients. The infection of a computer usually starts with receiving a dropper as explained in figure 2.1 on page 13. The dropper starts the infection but does not contain the bot client itself. Only after successful infection, the dropper downloads the actual bot client and removes itself.

This results in a problem which needs to be solved when wanting to analyse bot clients. A researcher looking into bot clients always has to make sure that he not only collected the dropper, but also the real bot software. This is one of the main reasons why some researchers are building automated analysis infrastructures where droppers are regularly run on real hardware to make sure that they download the actual bot client.

6.2. Effects of architectural choices

Infection and update mechanism. The infection mechanism of a bot has some effect on how the bot clients can be captured. While bot clients which are actively trying to infect other hosts will be detected by passive capturing processes. Bot clients trying to lure users to download and install malware will only be detected by active capturing processes.

The update mechanism has similar effects. When botnets spread their updates widely, it will become easier to capture (and analyse) these updates.

Rally mechanism. The rally mechanism has not much of an effect on the capturing process. A noisy rallying mechanism might help in detecting a botnet. But the rallying is about bringing already infected computers together and not to spread software. This is happening in earlier (infection) and later (updates) phases.

Defence mechanism. Depending on the defence mechanisms, botnet capturing processes can run into problems. If a botnet actively attacks nosey capturing mechanisms, this can lead to network outages and other problems for the infrastructure running the capturing mechanism. Some botnets attack an IP address automatically if that IP address tries to download the malware too many times [McA08]. A passive detection infrastructure is probably less threatened by the defence mechanism of a botnet than by an active detection infrastructure.

Said that, there are possible technical solutions which can be used when there is the potential of being on the wrong end of a DDoS attack. Threatened organisations like CastleCops and SpamHaus are renting network services from companies like Prolexic [War07] which make sure

that DDoS attacks are stopped before they are reaching the target. But at the end this is still only an arms race between the botnet hunters and the botnet masters.

6.3. Receive

6.3.1. E-mail with attachment. Some bot clients are sent to the potential victims via e-mail. The goal with this distribution strategy is for the bot client to exploit a vulnerability in the e-mail client without the need for the user to do anything or to trick the user into clicking on the dropper and start the infection.

Not that many bot clients are sent as attachments now.

6.3.2. E-mail without attachment. Other botnets use emails without attachments for spreading. They do not send the malware within the mail body but they lure the receiver of the email into visiting an infected website as described in subsection 4.4.2.

Botnet researchers implemented mechanisms which can interpret such emails and are able to automatically download and analyse the malware. This method can be quite successful since new bot clients can be detected quickly and without manual intervention.

6.4. Capture

6.4.1. Honeypot. A honeypot is a computer which acts like a normal victim but its sole use is to catch, and sometimes analyse the behaviour of, malware [HP08]. There are different kinds of honeypots, each kind having its own advantages and weaknesses. Honeypots have three special characteristics. These characteristics can be combined to create a honeypot for a special purpose:

- Interaction (high vs. low)
- Character (virtual vs. physical)
- Target (service vs. client)

Interaction. Honeypots either are high interactive or low interactive. Low interactive honeypots emulate system services. Because of this they are easy to deploy and the risks running them is fairly low.

High interaction honeypots are running the real services. Therefore they look more convincing to an attacker, but because of this they are also riskier to run. It is more difficult to deploy many high interactive honeypots because every service needs to be set up as if it would be used for a live system.

Character. Honeypots are either virtual or physical. Virtual honeypots are running within some kind of virtual machine. Many honeypots can thereby share one single physical host. Virtual honeypots are scalable and can be easily maintained.

Physical honeypots on the other hand are run without emulation, because of this they are more complicated to be managed but appear more authentic.

Target. Classical honeypots were emulating services like web, ftp, file sharing and others.

Newer honeypots start to emulate client behaviour. They act like computer users, browse on the Internet and click on links as a normal user would do. The idea behind client honeypots is to collect malware which specifically targets users and not services.

6.4.2. Dilemma. There is a dilemma for security experts working with honeypots which is explained in [ZC06]. It is the question about the liability of security experts running honeypots which are then compromised and eventually used as a relay or starting point for attacks against a third party.

To mitigate such risks, security experts started to implement filters and bandwidth shaping for their honeypots. The problem with this is that other bots can be used as sensors [ZC06] which can determine if a bot is honeypot or not. Bots can send themselves some faked traffic which the honeypot admin cannot classify as either malicious or unmalicious so he has to disable such communication.

Another interesting thought is to implement a trust mechanism like the PGP web of trust: Bots learn about other bots how many hosts they infected. Depending on the number of successful infections, the trust level raises. This strategy could turn out to mitigate the risks for Sybil attacks and other scenarios.

6.5. Obtain

6.5.1. Find and download source code. The easiest way to analyse the workings of a bot is to read its source code. Source code to some of the wider known bot clients is freely available under open source licences. For example, Agobot and some of its family can be readily found on the Internet.

While source code can make the life of a botnet hunter much easier, it also worsens the situation because the availability of malware source code typically provokes a heap of free-riders and script kiddies which have a head start thanks to the freely available source code. They are able to create new viruses and worms without the need to learn something about the technology they use. Which they would need to do if they would not have access to the source code.

Having source code freely available has another, less pleasant effect. Such source code is often taken and enhanced in some or the other way. This means somebody studies that code and creates something new. Probably even something more powerful and even "more evil" than the original version.

6.5.2. Ask for it. AV companies ask for samples of malware which they receive from researchers and from online scanning applications. They then analyse that malware and add the corresponding signatures to their signature database. They also receive prototypes for new infection strategies, for new stealth mechanisms or other advancements. The AV companies analyse these prototypes to learn more about upcoming threats and trends.

There is also another group which asks for malware, persons wanting to run a botnet but not being skilled or able to write their own bot client. They are obviously willing to pay for their bot software, support and updates.

6.6. Discussion

Many botnet researchers are currently making use of passive honeypots. These honeypots are waiting until somebody tries to infect them with malware. This unfortunately leads to the fact that only some fractions of all the bot clients will ever get caught. Botnets which require an active capturing method will be detected less frequently. There are no numbers available and it is not clear how many botnets are undetected because they are not visible to the current detection mechanisms. This definitely would be an interesting topic for further research.

There are efforts being done to develop active capturing infrastructures. These systems are complicated and the development is costly. The problem with them is that they need to contain some kind of intelligence which interprets and understands input. Analysing emails and downloading the advertised malware is a simple example of an active capture. Unfortunately such a capturing mechanism can be made unusable when slightly changing the bot spreading mechanisms. For example, using images instead of text could irritate the code and make the email parser useless because it can't interpret the email's content anymore.

We need more active capturing methods so that we can capture all kinds of bots. It must be the goal to catch as many different bot clients from as many different botnets as possible. Only achieving that goal will help us in being able to observe most of the botnets and learn about the motives and plans of the botnet herders.

Intelligence is a key in fighting the criminal activities emerging from botnets. Banks need to learn about where phishing is happening and who is involved in the activities. ISPs need to learn about their customers computers becoming infected and turning into zombies. Credit card companies need to know about their credit card numbers being sold on the black market.

I feel confident that organisations like the Shadowserver Foundation are an important factor when trying to prevail over the current situation. They are collecting data and more importantly, generating knowledge and awareness about what is going on and what we will see tomorrow. Technical means to collect and observe botnets and botnet activity is important, but processing that information and sharing it with the stakeholders are just as important. Capturing methods will change, they will need to adapt to the development of botnets, but cooperation amongst the botnet hunters will stay unchanged.

CHAPTER 7

Botnet analysis

7.1. What is botnet analysis?

Botnet analysis is about learning how a bot client and the botnet are designed, about their features, the commanding structure and the behaviour. This analysis can give important facts about how big and dangerous a botnet is. Depending on the findings, a countering strategy can be developed and executed.

When analysing software, the researcher sets up a taxonomy to describe the findings. In doing so, he first identifies the application boundaries. Once the boundaries are defined, the researcher finds entry and exit points to and from the application. This is to learn about where input is coming from and where an application is writing or sending data to. These entry and exit points are then analysed to define the attack surface.

Defining the attack surface is an important tool to formulate a strategy on how an application could be attacked. Based on all these findings and the analysis of the behaviour of the application under different circumstances, the analyser can then define a strategy to annihilate single bots or the whole botnet.

7.2. Effects of architectural choices

Usage and Update mechanisms. These two architectural choices are interesting to learn about what the botnet is used for, but they do not play a significant role in bringing down the network.

The update mechanism could be used by the botnet herder to constantly change the binaries of his bots, which would make the life of the analyser much harder. But it is important to note that this means no real protection to the botnet.

C&C modes and Communication protocols. C&C modes and the communication protocol are very important for the analysis process. When understanding these parts of a botnet, an analyser can write his own tools to take over the whole botnet. This can be compared with a community. Once an external person understands what a closed group is talking about and learns the lingo of that group, the external person will be accepted within the group and will have influence on the others.

IRC and similar protocols can be automatically analysed since there already are many tools and utilities for doing so. Other protocols might be more complicated. This means that when botnets are following standards they are developed much quicker, but the analyser also has a less difficult job because he can use standard tools for the analysis process.

Rally mechanism. The rally mechanism of a botnet is a very critical aspect because it is a weak spot. If the rallying mechanism is not secure, an attacker can destroy the rallying mechanisms infrastructure making it impossible for new bots to join the network. Consequently the botnet will shrink and eventually disappear.

Or an attacker can inject clones into the network. Which, when not being detected by the botnet master, could lead to a successful Sybil attack [Dou02]. Sybil attacks are explained in section 8.8.2.

Defence mechanism. Understanding and analysing the defence mechanisms is important because once the defence is understood it becomes possible to work around the defence mechanisms.

It is important to stress the fact that every defence can be subverted; it is only a matter of time and available resources.

7.3. Malware analysis methodologies

As is described in [BB07], an analyser can never be sure if the analysed malware is acting maliciously or if the malware is detecting the analysis and therefore running its self defence. This

means that malware always needs to be looked at from many different angles to make sure that the software behaves as it is expected to.

Generally said, the botnet owner wants to keep all the information about his botnet secret. As example, an IRC based botnet requires all the bots to know about the IRC server to connect to. As long as the bots can keep that central server secret, the C&C structure will not be taken down. Which means that in this example it is in the interest of the botnet developer to have his bots detect when they are being analysed so they do not reveal the location of the IRC server.

The bot can analyse its "neighbourhood" and find out if it is running on the right OS. The bot client can do a timing attack to find out if the host runs as fast as the hardware normally would. It could detect network throttling or missing network at all. Some of these tricks are described in [?] in detail.

Malware analysis is all about learning to understand the behaviour and functionality of a bot and its communication within the botnet. Generally said, there are two ways to analyse the bot. Analyse it when it is running on some system or analyse its source or binary code. [HM04] describes both analysis methods and a mixture of both as black, white and gray box testing.

A typical analysis follows a simple flow. The bot client will first be analysed. It will be let run, probably some decompilation occurs. After the binary was looked at, it is time for the network. The bot will be analysed how and where it is connecting to, as well as what it is sending over and receiving from the network.

7.4. Black Box Testing

7.4.1. Introduction. Usually the first step in analysing a bot client is the black box testing method. In black box testing the analysed software is being run and the analyser is analysing what the software is doing. The analyser is injecting some input into the application to analyse the behaviour on different "stimulations".

If there are several versions of the same application, it can also be interesting to compare the binaries of the different versions [HM04]. Such an analysis can reveal which parts of an application were changed over time.

Looking at the binary of an application can reveal which middleware and libraries were used, sometimes it is possible to recognise parts which were compiled from known source code. Sometimes it is possible to learn about the development tools which were used to compile the binary. And some developers forget that they added hardcoded directory paths and other information which can reveal details about their identity.

7.4.2. Behaviour based analysis with virtual machines and sandboxes. Virtual machines are convenient to analyse malware because it is easy to set up a new system after the old one was infected. Using tools like VMWare, this normally consists of the simple step to reload the original, uninfected disc image to revoke all changes made by the bot client.

Because of the nature of the complexity of virtual machines and sandboxes, every such mechanism can be detected [?]. Some virtual machines do not implement the whole hardware or software stack they emulate. Malware can then try to detect these deficiencies. The malware can look at the memory handling, namely the Local Descriptor Table and the Global Descriptor Table[1] [QS06]. If these memory structures are different than expected, the malware is probably running in a virtual environment.

Besides exploiting the architecture or programming errors, there sometimes exist even simpler methods. Some versions of VMWare can be detected by looking into the Windows Registry and searching for this key:

```
"SOFTWARE\VMware, Inc.\VMware Tools".
```

When this key is present, there is a good chance that the system is running in a virtual environment.

Now that many botnet hunters are making use of one or another virtual machine technology, an arms race started between malware authors and the botnet hunters. Bot clients started to become aware of virtual hosts. Currently some bots either try to deactivate the analysing software or behave differently. Even make-your-own-botnet tools like Shark3 allow for the inclusion of some

[1]The Global Descriptor Table (GDT) and Local Descriptor Table (LDT) are memory management structures on the x86 processor platform. Both descriptor tables contain Segment Descriptors which are used to translate a logical memory location to a linear location. While the GDT contains global memory segments, the LDT contains memory segments which are private to a specific application.

form of VM detection code [Dan08], which makes that technology available also for the technically challenged botnet masters. This means that there is a need for better virtualisation software and mechanisms to analyse malware which is aware of virtual environments.

It is possible for the malware to directly attack the virtual machine it is running in [D.G08]. Every virtual machine has defects which the malware can exploit to either shut the whole virtual machine down or to break out of the virtual environment and to infect the host. Once a host is infected, the whole analysis becomes useless because the results can't be trusted.

This trend will become interesting when client computers regularly use virtual machines. Currently our computer systems are moving towards virtualisation and more and more systems will depend regularly on such technology. Even client computers and not only servers. Computers running at home could use virtual environments for any reason. This trend would mean that the botnet developers would need to accommodate to this scenario and they would need to update their virtual machine detection mechanisms so that they distinguish when the bot client is analysed or when it is run on a regular (unprotected and non-hostile) client computer. This will be an interesting trend to observe.

Another interesting scenario is when bot clients make widespread use of virtual machine mechanisms to run as hosts themselves. Such a bot would run as host and move the OS into a virtual guest system. Probably during the boot up of the computer. This scenario would mean that although a system is running AV software and intrusion detection tools, the bot client acting as the host can subvert any OS activity and completely hide from detection. Such malware is currently in the wild [Kim08] and it will be interesting to see how this evolves.

7.5. White Box Testing

7.5.1. Introduction. During white box testing, the analyser is in possession of the bot client's source code and is analysing that source. Instead of obtaining the source, it is also possible to decompile a binary and then to analyse that result to understand what an application is doing.

7.5.2. Source code analysis. Source code analysis is the best possibility to analyse the inner workings of a bot. The source code is what makes the bot run, so reading and analysing the source helps in fully understanding a bot.

Bot developers and authors of malware in general are regularly adding backdoors to their code so that they can access bot clients after having them sold to customers [Wüe08]. It is therefore good practice to study malware for unknown backdoors.

There is no better defence to source code analysis than to not publish the source at all. Unfortunately there is a problem with this. Since a bot client needs distribution, so that as many hosts can be infected as possible, this also means that it becomes less difficult to get the hands on the binary. And when the binary can be obtained, it can be analysed. Which means that there is no protection for an application once it is released. Everybody owning a copy of said application can try to decompile the binary or read the assembler output to find out what the application is up to.

Of course some application's codes are obfuscated before they are published. But code obfuscation is only a distraction for the analyser of software and no real protection.

Source code analysis can be automated and there are several commercial and open source tools available. But all these automation mechanisms will always require the analyser to be a specialist so that he can interpret and understand such an analysis.

7.6. Gray Box Testing

7.6.1. Introduction. According to [HM04], gray box testing is the combination of "white box techniques with black box input testing". A simple scenario of gray box testing is to run a bot within a debugger, analysing the behaviour when the bot runs and receives input from a black box test.

Some bots like the Storm worm contain anti debugging features which can stop or deactivate a debugger, and thereby making analysis of the bot much more complicated. There is proof [IH05] that some bots do a simple check to see if SoftICE is running on a system. SoftICE is a debugger with which it is possible to analyse running software. Other strategies are used as well. Appendix B shows an example of how such a kill mechanism works.

Other bots try to encrypt their code. They are making use of a packer or some algorithmic method which obfuscates or encrypts the binary. When the bot is run, a decryption mechanism or unpacker translates such obfuscated commands to the computer.

7.7. Network Analysis

7.7.1. Introduction. Network analysis is actually some kind of black and gray box testing. To analyse a botnet means not only to analyse the bot software but also to monitor and analyse the network behaviour of a single bot and all the bot clients connecting to the same network. The importance of network analysis becomes clear when thinking about the connected nature of botnets. A botnet is only as good as the sum of all the bot clients. Which means that the botnets power lies in the stability and working of the interconnection of the single bots.

Because the network is an important factor for the analysis, botnet masters started to implement botnet level defence mechanisms as described in subsection 6.2. Storm worm nodes include automated and manual mechanisms to attack researchers which are too nosy. The defence mechanism consists of some nodes starting a DDoS attack for a few minutes or a few hours, depending on some settings and decisions made by the botnet master.

7.7.2. Analysis Methods.

Botnet Infiltration. The best method to analyse a botnet is to infiltrate the C&C. This is usually achieved by joining the C&C with moles. These moles look as authentic as possible and log all the communication within the botnet. This method unfortunately has some drawbacks:

- Moles must be named like the real bot clients. If they look different, they will get detected by the botnet herder.
- Moles must behave exactly like the real bot clients or fear their detection. They must answer on inquiries and they must react on commands as expected.
- Although a mole sits in a C&C, this does not necessarily mean that it can see the complete traffic. The botnet can be segmented or using private communication channels. This is an important fact to consider when analysing the botnet communication.

When the botnet shares all the communication amongst all bots, this analysis method not only reveals all the bot clients which are online, but it also reveals who the botnet herder is and what kind of commands are sent from where to how many bots.

DNS Redirection. Another method which can be used for counting bots in a botnet, but not to learn about the commands sent to the bots, is DNS redirection. The method is as simple as it is effective. The DNS entry associated with the C&C is redirected to a computer under the control of a researcher [RZMT07]. That computer is then waiting for bots contacting it, counting every connection attempt.

Unfortunately there are a few problems with this method as well. It can only count connection attempts, but it gives no details about how many bots would be connected at a specific time and it cannot detect when the same bot connects multiple times from different network locations.

7.7.3. Aspects to be considered.

Botnet Size. When analysing a botnet, it is interesting to learn about the size of the network. Unfortunately this is quite difficult. A botnet is constantly growing and shrinking. New bots are connecting, old bots are disconnecting. This behaviour has a simple reason. New bots are getting infected, already infected machines get cleaned or shut down over night. Which means that looking at the number of currently connected bots will not reveal the real size of a botnet. Some bots could be temporarily shut down over night.

From this it follows that to analyse the size of a botnet, the analysis should be done over some time, logging all the different bots connecting to the C&C.

Unfortunately this method has a problem as well. Some computers will change the IP address when reconnecting to the Internet. Most common reason for this is that ISPs give out different IP addresses to clients each time they connect to the ISP's network. This means that a previously unseen IP address could actually be an already detected bot reconnecting after a shutdown.

In [RZMT07] the authors discuss this problem of real botnet footprint versus the life population. While there is no real solution to this problem yet, it shows that size actually matters and that estimates about botnet sizes should always be taken with a grain of salt.

Geographical distribution. The geographical distribution of a botnet's clients has many different influences on a botnets nature. The simplest impact is that when all bots reside in the same time zone, the population of the network will grow and shrink dramatically because computers are shut down for the night at the same time and other reasons.

But the geographical spreading can have different impact on the botnet as well. When being planned accordingly, the botnet can target its attacks according to some diurnal model. It is less of a waste of resources and there is better probability to find an active and infectable computer when the attacks are timed to the time of day where people are sitting behind their computers. Such an attack can make use of the cumulative online population during the evening hours to attack the private computers of a specific time zone. Such attack behaviour would have an effect on the analysis of the bot client. It could be that some bots do only attack during special hours or some similar scenario.

Such distribution strategies are described in [BW007] which talks about zealous propagation. Identifying and understanding such a geographical or diurnal attack model can help in an early detection of an attack. Once such an attack scenario is identified, it becomes possible to start a better defence.

IP distribution. An interesting aspect to be considered when looking into botnet analysis is to learn about the distribution of bots. It can be insightful to analyse if a bot is only attacking servers, only clients or if it just attacks IP addresses with no preferences at all.

It can also be interesting to analyse the distribution algorithm to find out if some IP blocks are favoured over others. This information can be used to build up the defence strategy.

7.8. Discussion

Bot client and botnet analysis is difficult and time consuming. Considering the number of different (and known) botnets out there, it should be obvious that it is impossible to analyse all of them manually. An automated analysis is required.

The second problem is about how bots are analysed. An analyser can never be sure that he has detected everything. A bot could sleep for 90% of the time and only wake up and do "its work" on a specific time in the month (when the researcher is looking away). There is an old saying which goes like this: "An absence of evidence is not the evidence of absence". You can never be sure that you know about all the features within a bot client.

Another problem is that bot clients can monitor their environment. A bot can try to detect if it is being run in a VM, if a user is sitting at the computer (mouse and keyboard activity) and if the network is filtered. If the bot thinks that it is being observed, it can behave inconspicuously or crash to put the analyser off the scent.

Botnet analysis sometimes sounds like a game between the botnet developer and the botnet hunter. While the botnet hunter tries to find out about the workings of the bot client, the bot developer tries to hide as much information as possible. But no defence can be completely perfect and thus everything ends in a never ending build-up of arms between the bot developer and the botnet hunter.

CHAPTER 8

Botnet annihilation

8.1. Introduction

The end of the botnet lifecycle is when either the C&C infrastructure does not function or there are no bots left to control.

Attacking a botnet can be difficult because the different bots are usually located in different countries (with different jurisdictions). And the C&C is probably located in a different country than the person trying to close down the botnet.

There is an old Soviet doctrine which goes something like this. First kill one third, disrupt the second third and the last third will fall down as a result. Unfortunately that doctrine does not directly apply to the botnet scenario. Although removing bots from their botnet will reduce the size of the botnet, this does not mean the botnet will be annihilated when there will only be one third of the bots left. While some P2P based botnets can be susceptible to such attacks, some centrally commanded networks will be immune against it. This shows that everything comes back to the architecture of a botnet.

All of the annihilation strategies follow one simple rule, find the weakest link in a botnet infrastructure and attack that single point with the hope to bring down the whole infrastructure. The energy required to bring down the botnet should not surpass the worth (or threat) of the botnet. It probably does not make sense to invest millions to annihilate a small botnet when the botnet is only of small annoyance. Which means that the annihilation strategy is more efficient when the budget and resources for the botnet hunter are smaller than the budget and resources for the defence of the botnet.

It is also important to stress the fact, that there are circumstances where it is undesired to annihilate a botnet. If for example the forensics team of a bank gains access to a botnet which is actively used to defraud the bank, then it can be desired to keep the botnet up. The motivation for doing so is the chance to learn about the motives and the moves made by the criminals. Observing the C&C of a botnet also sometimes reveals important information which can be used by law enforcement.

8.2. Causes for annihilation

8.2.1. Natural causes. Botnets experience a constant natural annihilation. Computers get patched, some computers get shut down during the night and there is still the possibility of power failures or network outages [DGZ+05]. There is also the possibility that the botnet master looses interest in his network and stops all management tasks. The botnet will then continuously shrink and eventually cease to exist.

8.2.2. Manual take down. There are several possibilities to bring down a botnet. This chapter looks into the different strategies on manually taking down a botnet.

8.3. Motivation

Botnets are a threat to the security and privacy of the Internet. They threaten businesses and people all around the globe. Botnets use resources from third parties without paying for them, and they use these resources to follow illegal activities. From a legal perspective this is motivation enough to target botnets.

During the InBot'08 conference in Germany, Freed0 from the Shadowserver Foundation made an interesting statement. He said that most victims which are targeted by botnets will never contact law enforcement and will never demand a criminal investigation. Not contacting law enforcement is clearly an act of endorsing the criminal activity. It is the same as when paying the ransom when

being blackmailed. Giving in on such a demand legitimates the blackmailing and will finally result in being targeted again.

There may be different reasons not to report an offence. An infection can stay undetected, users are overstrained and don't know where to report to, companies can be ashamed of having to publicly admit that they were infected. And most often evidence is erased before somebody had the chance to analyse the infected system.

But there are many more reasons why somebody could want to bring down a botnet. Many of these reasons depend on the annihilator's motivation.

- A rival botnet herder could be interested to take over the botnet to integrate the bots into his own botnet.
- A researcher could be interested to learn more about the architecture of a botnet and how the architecture plays a role in the botnet's defence.
- The Shadowserver Foundation has the mission to improve the security of the Internet. Other teams have similar goals.
- Law enforcement will want to annihilate the network because of legal aspects.

These actors all have different ways in which they confront a botnet. While rivalling botnet masters will prefer to directly attack the botnet, researchers and groups like the Shadowserver Foundation will only report on their findings (to the authorities). Law enforcement will try to stop the computers running the C&C and will investigate on the botnet masters.

For the time being the coexistence of groups like the Shadowserver Foundation and law enforcement will persist. Law enforcement is dependent on the details they receive from the researchers. Mostly because they lack the resources to investigate the details on their own. There is also another aspect. Botnets are operating globally, law enforcement mostly locally. Which means that either all law enforcement agencies around the globe are constantly reinventing the wheel, or they use the advantage of having access to global information. Globally sharing resources is clearly the better solution. And because the Shadowserver Foundation (as example) does only gather intelligence without acting themselves, there is also no conflict with local laws.

8.4. Strategies against active defence mechanisms

Attacking and trying to bring down a botnet can be a risky task. Depending on the motivation of the botnet master and the monetary aspects of the botnet the botnet master could become very angry. Which means that there is some potential harm to the botnet hunter's systems as well as potential physical harm involved.

From an ethical and legal standpoint it is questionable if bringing down botnets is legal in all cases and if this is ethically and morally okay to do so. Analysing a botnet and then informing the legal authorities is certainly the better strategy than trying to do that on your own.

8.5. Effects of architectural choices

Infection mechanisms. The infection mechanism per se is not that important to the analysis of a botnet and the bot binary. But the infection mechanism can be interesting to analyse and can help in selecting an annihilation strategy.

C&C modes and communication protocols. The design of the C&C modes and the communication protocols has a strong influence on how easy or complex it is to analyse a botnet. While communication has little effect on the analysis of the bot binary, it has a huge impact on the whole network. If the communication is decentralised and encrypted, the analysis becomes very complex and most probably will require the development of analysis tools.

If on the other hand the C&C is based on the implementation of an RFC compliant IRC protocol then the analyser can use standard tools and dissect the workings relatively easily.

Rally mechanisms. The rally mechanism can have an impact on the analysis so far as that when trying to infiltrate moles into a botnet the architecture of the rally mechanism can require the analyser to write his own tools instead of being able to use standard tools and utilities.

Update and defence mechanisms. The architecture of the update and defence mechanisms obviously have a significant impact on the analysis. If a bot gets updated regularly, changes its behaviour and old binaries are closed out from joining the network, then a researcher analysing the botnet will be forced to regularly redo the analysis to reflect the new conditions.

A defence mechanism like the automated DDoS attacks from the Storm network can prevent analysers from looking into the workings of a botnet simply because they cannot afford or do not wish to be attacked.

8.6. General annihilation strategies and thoughts

Attacking a botnet can mean simply bringing down single bots and trying to decimate the botnet. This strategy will work a small percentage of the time. Successful attack strategies can be found in military tactics and the analysis of the workings of terrorist cells as described in [NA05].

When trying to bring down a botnet it is important to look into the topology of the network. It is interesting to see the similarities between a distributed P2P botnet and distributed small terrorist cells. Both systems can't be annihilated by simply attacking the central communication infrastructure. It is also difficult to bring them down with infiltrating moles because one single mole will always only see some part of the whole network.

So there are three important points to be looked into when analysing a botnet:

- What strategy is used to control the botnet.
- What kind of trust mechanisms are built into the botnet.
- What strategy is used to replace lost bots.

Control mechanism. As was discussed before, the control mechanism is important. If there is any way to subvert the control mechanism it becomes possible to inject false commands into the control structure which can disrupt parts or the whole of a botnet.

Trust mechanism. The trust mechanism decides on how bots interact with new or already known bots. In simple botnets where there is only one server commanding the clients there is no need for a trust mechanism. Every bot implicitly trusts the central C&C. But such a scenario does not ask for an enhanced attack strategy because it suffices to attack the central C&C to bring down the whole network.

In a much more complicated P2P network, as example, there must exist a trust mechanism. There must be some way in which the network accepts new bots and in which new commands are fed into and distributed through the botnet so that an infiltrated mole cannot take control over parts or the whole of the botnet.

Replenishment strategy. The replenishment strategy is very important. If new bots are all added to one side of a network it can be a possible scenario to isolate all new bots from the already existing botnet to disrupt the communication channels and successfully attack the replenishment process.

On the other hand if there is some clever defence mechanism like the cliques described in [NA05] then annihilation becomes quite difficult. Cliques are similar to the cell structure often used in revolutionary warfare. Such cells are operating independently from other cells and are only contacting other cells for coordinating their forces. It is interesting to see that the same strategy successfully used by many insurgent groups can be applied to the botnet defence as well.

8.7. Strategies against the technology

8.7.1. Attack communication infrastructure. An attack against the C&C infrastructure tries to disrupt the communication between the botnet master and the bots. To disable the communication in a botnet there are many strategies possible.

Centralised Communication. If the botnet has a centralised communication infrastructure and the bots are locating the central server with the help of DNS, then it may be possible to take over the DNS name used by the botnet. Once the DNS name is taken over, the domain name entry can be changed to 127.0.0.1 which basically redirects all bots to themselves. The botnet will then collapse because it becomes headless.

Another possibility is to take down the central C&C server. This can be achieved with the help of the ISP where the server is running. Shutting down the C&C server has the same effect as changing the DNS name.

Decentralised Communication. Attacking the communication infrastructure with decentralised botnets is more complicated. There is no central entity which can be attacked. There is no silver bullet for this problem. Current strategies try to inject false commands or to pollute the communication amongst the bots. Such a strategy is only effective when the architecture of the communication protocol allows for command injection.

Becoming headless. Most of the strategies targeting the communication infrastructure have in common that while they solve the immediate problem, they will not solve the long term problem. Even when a C&C is not available anymore, the bots are still infected and it is only a question of time until somebody else will capture them and they will then join another botnet instead.

Botnets can defend themselves against attacks targeting their communication infrastructure. Bot developers can change their botnet's architecture so that there is no central C&C anymore. They can harden the communication protocols. This means that somebody wanting to annihilate the network will have to address single bots or invest resources and try to find an exploit within the command structure.

8.7.2. Update injection. It could also be interesting to use the automatic patching system of the botnet where the existing communication infrastructure is used to distribute patches to the client. It could be tried to inject an "insurgent" update into the botnet. The bots would then automatically patch themselves and the botnet would cease to exist.

There are many ethical and legal aspects in this strategy for obvious reasons. Such an update can fail and leave the computer unoperateable. Since the update would be run without the users consent it could lead to legal actions against the person injecting the update into the botnet. Even when it was done with reputable motives.

8.8. Strategies against the organisation

8.8.1. Follow the money. Shutting down the C&C might either be not a possibility or it is the goal to find the botnet herder to bring him to justice. On both counts it can be a good strategy to follow the money. Following the money means that law enforcement tries to follow the routes the money takes from somebody renting a botnet until the money reaches the bot herder. There are different reasons why somebody needs to pay a bot herder. Somebody can pay for clicks on the ads on his own website (click-fraud) or he can pay for the sending of spam or a victim of a DDoS attack can decide to pay a ransom for stopping the attack.

Depending on the steps and the countries involved and the intelligence of the botnet herder following the money can become quite difficult. Although banks are required by law to report suspicious money transfers (at least here in Switzerland), successfully laundering money is not that difficult when using the right methods. When laundering money it is important to alternately use different means to transmit the money. Which is why money transfer using Western Union[1] is popular. The money is taken from one bank account, transmitted to the next, withdrawn from the second account, deposited to the third and then transmitted via Western Union to a foreign country. As more steps and countries are involved, as more difficult it becomes to follow the money.

8.8.2. Destroy reputation. While some botnets exist simply because the botnet herder feels like doing so, there are also botnets which are there for commercial purposes. Owners of a commercial botnet are dependent on the reputation of their services and the quality of their network. While this automatically provokes mob wars between such commercial botnets, at the end all of those networks want to be the biggest so they can earn the most, this also opens up an interesting attack against the reputation of a botnet and its owner.

Let's think about a possible scenario where a botnet like, for example, Storm is known for their spam runs. The botnet herder would be known for the performance in which his bots can send out millions of spam mails in a very short time. The response rate would be very high because of the good quality of email addresses. The owner of that network can then ask for a good price for his services.

To make that botnet less attractive, there could now be two possible scenarios. One is to flood the database with bad email addresses. Addresses which will not work or which are spam traps so that the next spam run will result in a much lower response rate. How to achieve this would mostly depend on the method the botnet herder gathers the addresses.

The other scenario could be to start a Sybil attack [Dou02]. Sybil attacks are about creating multiple identities and infiltrate them into a system. In a scenario this could mean that a botnet hunter is creating many faked bots and have them join the botnet. Once the Sybils joined the network, the botnet size would look good to the botnet owner. But once the botnet master sends

[1]Western Union sometimes is called the "high speed train of money laundering".

a command to the botnet, only a part of the bots would actually do what they are told to. The Sybil bots would only fake their engagement but actually do nothing.

The second scenario can be evaded by the botnet master when building a robust trust mechanism into his botnet where it becomes difficult for a botnet hunter to infiltrate Sybils into the botnet.

There are many other variations of Sybil attacks possible [FPPS07]. They all tend to address the reputation of the botnet master and try to destroy the market the botnet master is active in. The inversion of a Sybil attack can be watched on sites like eBay and others where some sellers are creating multiple entities (users) which are then used to raise the value of a seller using the Sybils to create faked credentials.

8.9. Discussion

Looking at the different annihilation strategies it should become clear that there is no right strategy for every case. While the technical take down is probably the easiest solution in many cases, it often does not solve the problem but only fights the symptoms.

A take down is driven by architecture. This should also make clear why a good analysis is needed to decide which strategy could be effective. This does not mean that every single botnet and bot should be analysed into every detail. But this means that the botnet problem must be constantly addressed and new trends should be analysed in detail. Reports about new findings must be shared amongst the researchers so that protection mechanisms can be developed and implemented.

Besides the technical side there must also be a communicative side where information about the botnet problem is shared with the public. At the end the botnet problem can only live as long as there are computers which can easily be captured and turned into zombies.

Part 3 of this thesis contains some more thoughts about botnet defence and annihilation strategies.

Part 3

Corollary

CHAPTER 9

Trends and future development

9.1. General thoughts

9.1.1. Trends. Nobody can tell the future, but there are strong indicators which point in the general direction where we expect to be in the near future.

There are broadly two groups of botnet trends. The evolutionary ones which are just the incremental improvement of existing systems. These trends just enhance what's already there. They represent the "normal" evolution and won't bring about any step change in behaviour. There are the sophisticated trends which will be developed by the malware authors which think out of the box. These trends will introduce new aspects to the botnet landscape. These trends will really challenge the botnet hunters.

The incremental trends will avoid old programming faults and design errors. The tendency will be towards a general robustness of botnet software and will contain these topics:

- General professionalism [FdP07].
- Hiding tracks.
- Use of computer resources for computational tasks.
- Growing recklessness.

The sophisticated trends will contain advanced attacks against the OS and the infrastructure itself. Botnet masters and bot developers will use findings of research papers and do their own research to enhance their bot software.

9.1.2. Ever-growing attraction. While there is a general disagreement regarding the raise of the numbers of infected hosts [Bar07] it should become clear that the attraction of botnets will grow in the future.

Malware turned into a lucrative business with an attractive black market some while ago and this will not change that fast [FPPS07], [FA07]. There is too much money at stake. Being lucrative means that there will be the need for more specialists developing and running botnets. There will be a growing infrastructure need and the fields of operation will become more professional to what we see now.

But no matter how things evolve. It should be clear that we will need to work on mitigation strategies to face the upcoming threats from the coming years. And these strategies will need to become pro-active instead of being reactive as they are today.

9.2. Evolutionary trends

9.2.1. Botnet segmentation. Instead of having one huge botnet, botnets will be segmented [VA06]. Segmented botnets can be leased to spammers and others. Reducing the size of a single botnet reduces impact of losing that botnet [NA05]. There are reports that this is already happening with Storm [FSJ08] and other botnets.

Another benefit of botnet segmentation is described in [Ayc07]. Segmented botnets need less intra botnet communication which makes the botnets existence less obvious.

9.2.2. Enhanced P2P. Another natural trend will be the technical advancement of the P2P based botnets. While some botnets are built with Kademlia based techniques [MM02] the trend will probably be towards overlay service networks like Tapestry [ZHS+04]. These P2P networks will then be used like an overlay network where traditional network services will be replaced by their own equivalents within the overlay network itself.

Such coordinated networks, more resembling a system in the traditional sense than a network, can then be used to attack the traditional network services like DNS and others. If done well,

an overlay network will only require basic functionality from the network stack below and can therefore attack computers outside the network without fear of being affected as well.

This attack will gain in popularity when kits become available which allow to easily migrate from IRC to P2P based communication. Currently P2P is just too complicated for some bot developers.

9.2.3. Extortion. Bots encrypt files on the local host with a cryptographic algorithm and the botnet master extorts the owner of the computer to pay a sum for a decryption program which can be used to revert the encryption mechanism. This can be countered with clean and recent backups.

A similar scenario is when the botnet master is not asking for money for a decryption program but to not make public the data found by a bot. This can obviously not be countered by making regular backups but only with using strong crypto to store important data in the first hand.

9.2.4. Improved spam.

Personalisation. The general trend will be towards more convincing and better looking messages. If a spam message looks better and is personalised to the recipient and the current circumstances (world news), a social engineering attack will become more effective. Spam will improve and not only look legitimate but botnet masters will start to mine emails they find on infected computers and automatically use that data to forge improved mails which look more convincing to the recipients [AF06].

Another special kind of personalisation is not directly spam related but can be adjusted accordingly. [AFA07] describes the personalisation of attacks, where a botnet master could decide to target one localisation, like Berne, Switzerland. He would only need to identify all computers from one specific geographic location and could then start a DDoS attack. Or he could decide to just infect computers residing in England (or time the attack with the diurnal rhythm). Or send spam only to specific regions.

Events. Reacting to events is a powerful tactic to make a spam run much more effective. If there is a disaster somewhere in the world, people are tending to overlook obvious warnings. This is a true and tested tactic which was employed by the Storm worm (hence it's name) and many others before (and afterwards). The same can be done at seasonal events like Christmas and Easter.

The storm botnet started its Christmas season 2007 spam run the day the Russian domain name registrar nic.ru closed its doors for Christmas and New Years Eve. Because of that move, the botnet masters were able to use the recently registered domain names in their spam mails without the fear of having them shut down by nic.ru during their spam run.

9.2.5. Consistent use of stealth. The use of stealth communication can be an absolute requirement. As long as something is not seen or detected, as long it can operate without interferences. So if a botnet herder tries to build a botnet silently so he can attack some huge system out of the blue, he will probably want to stay hidden as long as possible.

There are many more reasons why a botnet herder may want his botnet to be as silent as possible. Many of those were discussed before.

For C&C and rallying. Some botnets already make use of some form of stealth communication. Some make use of HTTP based traffic which is often overseen because it does not look suspicious in log files. Another problem with HTTP is that this protocol is probably the most used protocol on the Internet any many companies rely on using that protocol. This means that HTTP cannot be blocked on company borders. Which makes it an interesting protocol when wanting to have bots in company environments.

The sky is the limit in misusing protocols for C&C communication. NNTP, the protocol used for Usenet messages, can be used for communication. While the Usenet was in wide use some years ago, this lately changed a bit.

But then there is also the possibility to use ICMP for communication purposes. ICMP is the protocol which is used for network analysis where systems can be pinged to look how long it takes for a packet to reach them on the network.

An interesting idea is to use DNS as a communication channel. DNS is the protocol which is used for domain name translation where a domain name like www.somap.org is translated into an IP address which is then used to contact the server in question. Commands in that protocol can be very difficult to be detected. The problem with using DNS as a protocol to hide information in

will probably be that somebody will become suspicious for the increased amount of DNS traffic. This can be countered by small segmented botnets with low interbotnet communication to keep the "noise" down.

Another interesting strategy is to mimic a legitimate protocol and use the exact same ports as the original, but to do something completely different [WSZ07]. Honeyd, a honeypot client, does this by trying to lure malware into connecting to the honeypot. The same strategy can be used for the bot developers to mislead botnet hunters about the real bots and their purposes.

On Host. There are some tricks which can be used on the client to stay undetected as well. This starts with rootkit technologies hiding the presence of the bot from the file browser and the process monitor. Another simple tactic is to only eat resources when it will go unnoticed.

But there are other tactics which currently are either not widely used or which will be probably introduced real soon now. Malware already experimented with installing a virtual machine and moving the users system into a virtual guest environment [Kim08]. This means that the users OS runs as usual. But the bot client actually runs as the host, hosting the users OS as a guest system. Since the bot hosts the users OS, there is only a slight chance that the user will detect this behaviour because the bot can simulate whatever it wants to. This attack is currently successful because the Windows OS allows writes to the MBR. This is obviously a security risk.

Another very interesting idea is to include some patched version of the free AV software ClamAV or another patched AV software on the bot. The bot client is then installing the patched AV software into the system as ordinary virus scanner. Windows as example would then tell the user that he runs a virus scanner and the patched AV would (wrongly) state that everything is okay. While the bot runs undetected in the background.

9.2.6. Honeypot detection. As was discussed before. The technology to detect if software is run within a virtual environment will be constantly enhanced. Not only is it possible to find out if a bot is running on the real software or some emulation. But it is also possible to analyse if the computer is run by a human being or an automated process [ZC06]. The idea behind this is to analyse the mouse and keyboard events to find out if somebody is sitting behind the computer or if the system just tries to be as real as possible although it is a computer trying to lure malware into running their exploits and to then automatically analyse that malware.

The same mechanism will then probably be built into the trust mechanism of some botnets. It is important for the trust mechanism of a botnet to make sure that no Sybils or moles can enter the network. Filtering out computers which look suspiciously will help in reducing the risk of the bot being analysed by a botnet hunter.

9.2.7. Avoid signature detection. Morphing binary structure is something which is already being done by many bots. This means that some mechanism makes sure that the binary of the bot is changing constantly so that the signature detection by the AV software will not detect the bot because the time between the changes is faster than the time the AV vendors need to collect the malware and to update and release new signatures.

The problem with this approach is that some AV companies are using heuristic mechanisms and behaviour detection to find such morphing structures. Although many of these mechanisms are working badly [hei07], it is a start and definitely the right way to go.

Another problem the bot developers face is that the code which is used to do the morphing is sometimes not changing itself and the AV vendors can look for occurrences of such morpher programs to detect the malware.

Unfortunately this is another arms race and will not change until one side gains some new insight into the rearmament and develops a completely new defence strategy.

9.2.8. Better botnet protection. Some botnets are already "protected" today. They stay as low profile as possible or they start a DDoS attack against the ones trying to analyse them. Automatic update mechanisms are already in wide use. Botnets update their binaries regularly. But there are other possible solutions which can be used as protection mechanisms.

Botnets could start to behave less traditional and instead of attacking a system with a DDoS attack a botnet could start to cancel all connections to a computer. To do so the bots of a botnet would need to contain the same code which is used in Intrusion Prevention Systems (IPS). Such IPS send a "connection close" network event to the source and target of a connection if they detect suspicious or malicious activity. The same could be done by bot clients who are near their target.

The bots could start to inject packets into the network. These packets would look like they come from the victim computer, but instead these packets are generated by the attacking bots, resetting all connections the victim has open. This is an attack behaviour which could be very difficult to mitigate. Another strategy could be to use DNS poisoning or other attacks against the DNS infrastructure.

9.3. Sophisticated trends

9.3.1. NAT traversal and UPnP. UPnP, or universal plug and play, is a protocol with which a device can configure the network it is running on. A user located behind a firewall can configure the firewall so that he can connect to the Internet. The goal of this protocol is to make network configuration as easy for home users as possible.

This creates interesting new attacks which bots could make use of. Instead of shutting down the firewall, a bot client would just reconfigure the firewall so that the bot can connect the Internet without being stopped by the firewall. Besides opening the inside, the bot can also open the outside interface so that all clients on the same internal network can be attacked from the Internet.

The problem with this UPnP scenario is that it will go unnoticed while the network still works. The user will only get suspicious if something breaks and he has to look at the configuration.

There recently was an attack against routers [McM08] which was similar to the above scenario. Client computers behind a firewall were tricked into visiting a website. That website contained script code which was consecutively executed on the client and which automatically and silently re-configured the router in the network the client resided in. The result was that every client in the same network was exposed to attacks from the Internet.

9.3.2. Subvert cryptography.

Install Root Cert. The botnet master can install a new root certificate on the bot [HAJ07]. With such a certificate in place, phishing becomes very easy as all the protective measures which are taught by banks and others are void and useless. Because the root certificate is installed in the browser, the user will not see anything suspecting. The symbol of the SSL lock in the browser window will be closed because there is no reason for the browser to alarm the user. The server's certificate is signed by a root certificate which the user is trusting. So everything looks all right.

A user starting his e-banking client on a computer with such a bot installed can be redirected to a phishing website under the control of the botnet master. This website would look exactly like the original banking website and the user would fall for a phishing attack.

Crack cryptography. A very interesting thought is to use the power of a botnet to crack cryptographic keys. Much in the way seti@home users are cooperating in finding traces of extraterrestrial life. This idea was described in [HAJ07] and that paper analyses how long it would take to brute force the code of the private key of a root CA. Once the private key of the CA is found, the key can be used to sign SSL certificates of phishing sites. This scenario would not require to install a new root certificate within the user's browser, making a phishing attack even simpler.

Such a scenario would threaten everybody because it would also fool users without malware installed. Everybody connecting such a site would see an SSL certificate which was signed by a trusted root CA. As long as the root CA is not revoking their root certificate (and every user updated his browsers) as long an attack with such a certificate would be successful.

Currently the threat of such an attack is not that big. Mostly because computers still are too slow to successfully attack an RSA key in a reasonable time, even in a huge group like a botnet. But the computation power of processors is constantly increasing and it is absolutely possible that such a scenario becomes reality in the foreseeable future.

9.3.3. Use of trust mechanisms. To protect the botnet from moles and prevent the infiltration of Sybils a botnet will need to have some trust mechanisms. This is especially true in decentralised networks where new members to the network are joining an already existing group of hosts and where there is no central authority controlling everything.

There are mechanisms for such scenarios like the PGP web of trust or EigenTrust. Such mechanisms should make sure that it is not possible to illegally inject commands into the network.

Another such mechanism comprises of mob tactics which are known from movies like "The Godfather". Every bot joining a decentralised botnet would need to prove that it is really a legitimate bot. To do so the botnet could support many different mechanisms and update a bots

reputation or legitimation accordingly. Successfully attacking and infecting other computers could lead in a raised reputation. Much in the way that a new member of a mafia clan needs to shoot somebody to show that he is no cop. This could work in the botnet world because many botnet hunters configure their systems so that a honeypot cannot be used to attack other computers.

Another strategy could be to analyse the environment the bot is running in, to try to find out if it runs within a virtual machine and other tests. Every such test could then raise the reputation.

9.3.4. Automation. Automation on the side of the malware is a significant risk to botnet hunters. If bots start to become capable of doing simple tasks themselves, things start to turn very ugly. One such idea is to teach bots to automatically develop their own exploits with tools like the Metasploit framework [FdP07]. Such bots could constantly try to find new exploits and then use them to infect even more computers.

This would be an extremely disturbing trend because it would make the botnet very difficult to defeat. The botnet would constantly change its attack behaviour and probably make use of previously unknown exploits which first would be needed to be analysed by specialists before some patch could be released. Once a patch is available the botnet could already use another attack to exploit another vulnerability.

This could turn out to become even worse when some bots automatically develop new exploits while others try to silently build up a list of vulnerable hosts without attacking them [WPSC03]. The goal would be to compile a huge list of possibly vulnerable hosts. Once such a list is big enough, another group of bots could start to attack all hosts in a coordinated effort, resulting in a form of flash attack with the potential for an extremely rapid infection.

9.3.5. P2P Botnets and Overlay networks. If a P2P botnet is reaching critical size and distribution, there will be a few attacks which could become realisable. Some of these trends were mentioned before.

Drop traffic. Instead of attacking a single host or system, the bots which are strategically located best could just drop the traffic coming from and going to the victim host [BW007]. Such bots would basically behave like IPS systems but instead of closing potentially malicious connections, such bots could stop legitimate traffic. This is better than a DDoS attack because there is not much which can be done against such an attack apart from cleaning the Internet from all infected hosts.

Packet Injection. Instead of defacing a web server, bots located in between a victim and a web server inject HTML code into web page requests [BW007]. This is no classical MITM attack but some injection of packets without interrupting the direct connection between the victim and the web server. The result for the victim would basically be the same. But finding the problem will be much more complex.

An even more disturbing scenario could be when a big botnet starts to inject spoofed DNS packages into the network. When bots are placed in strategic locations, this could lead to serious network problems. Botnet masters could redirect any legitimate traffic to their servers. Due to the fact that such an attack leaves no marks, researchers and security experts would have a hard time to detect and stop it.

Rogue code / Poisonous communication. This is something which is tried on distributed hash table based P2P botnets. Some researchers inject poisoned hashes into the botnet communication to disrupt the C&C. Currently this only slows down the botnet for a short time and is therefore no more than an interesting experiment.

Assuming the bot developer is clever enough, it should be no problem to develop protective measures against such attacks. But when thinking about the future of P2P based botnets, injecting poisonous communication could be one of the most effective strategies.

CHAPTER 10

Conclusion

When looking into botnets it quickly became obvious to me that many different aspects need to be addressed and, more important, to be understood. Botnets are a complicated mixture of many different subject matters. I quickly learned that I needed to widen my focus and to investigate topics like networking protocols, host security, social engineering, psychology, warfare, legal aspects, P2P, trust mechanisms, virtualisation and hardware technologies as well as many other subjects.

Botnets are becoming more robust, communication is encrypted and trust and stealth mechanisms are built in. While there always were and probably always will be some amateurs participating in the game, there will also be the pros which know what they are doing and which will always be one step ahead of the crowd.

Many problems which current botnet architectures are struggling with are solved in other areas. Researchers are constantly looking into network protocols, stealth technologies, encryption algorithms and overlay network architectures. Their findings are published and can be read by everybody interested in the topic. I don't say this is a problem and should not be done, I actually believe in freely sharing knowledge. But we need to be aware of the fact that advancements can be a double edged sword.

Let's take the problem of trust. A P2P based botnet needs some way to trust new and existing nodes so that it is not possible to inject unauthorised commands into the botnet. There is some literature about introducing trust mechanisms into distributed networks. It will only be a matter of time until the first P2P based botnet will introduce a trust mechanism which makes command injection attacks and communication poisoning attempts on botnets futile.

While the current situation is interesting to look into, it is also very scary. Criminals follow the money and take every opportunity. They have the capability to run decentralised private networks and the current situation is that only a small percentage of botnet masters ever get caught and only a few of those get convicted. Since we keep on adding even more hosts to the Internet, e.g. fridges, cars, mobile phones, this threat will even increase. A growing connectivity and accompanying complexity will result in more possibilities for the criminals. I am tempted to say that we only stand at the beginning and things will become more problematic than they already are.

When looking at possible trends I realised, that the sky is the limit. I was especially fascinated by the thought of using the botnet resources to attack cryptography, using the bots as a number cruncher to work on calculation intensive tasks.

Because to understand the threats coming from botnets also needs some degree of technical knowledge, it becomes difficult to raise awareness of the immediate threat. Meanwhile criminals are getting better, they learn fast and try out new technologies and capabilities.

Criminals will also work together. A market already exists where they do not only share credit card details and other data, but they also started to specialise and to sell dedicated services. This trend hints to a similar tendency as can be identified in the "normal" software industry. Companies started to specialise in specific subjects and sell software and services tailored to dedicated needs.

Botnets don't know any borders and different jurisdictions and because of this it becomes difficult to do something against a specific attack. Therefore it is my strong believe that we need to organise our defence on an international level. Groups like the Shadowserver Foundation are a step into the right direction. They bring together specialists from many different areas of expertise and analyse data they collect from sensors located all over the world. Sharing that data with the stakeholders is very important. Only when understanding the immediate threat and understanding how the criminals think and operate will we learn how we can protect our communication infrastructure.

There are three interest groups which should work together:

- Private companies are having a need but lack the intelligence.

- Botnet hunters do have the intelligence but they can't do anything with it.
- Law enforcement would need the intelligence (and information about concrete threats) to investigate on botnets.

All three interest groups together can share knowledge and do something against the threat from botnets. Intelligence is the key, but only worth the cost if we share it and work together.

This brings me back to what I wrote in the introduction: It is important that the above mentioned stakeholders discuss different scenarios. A discussion is important. It does not only raise awareness but it helps in developing and refining ideas about new trends. Scenarios are important because they help in understanding a complex system. With scenarios new threats and trends can be analysed.

I learned from many discussions during the InBot'08 conference and from private communication with many botnet hunters and other stakeholders that there is really an interest to do something. The defence is forming which is a good thing. But still, it remains to be a very long way to go. We need to find a common language and we need to learn to know each other so that we know who is doing what.

The problem I faced with my objectives is that I could have spent days, weeks and years on looking into botnets. I could have written sentences, pages and books about them. I wanted to show a picture as complete as possible. This unfortunately comes with the cost of depth. The challenge therefore was it to keep on topic without losing myself into details of one aspect only. But it is my opinion that I achieved my goals. I was able to work and report on all of my objectives. And having pointers to many of the technologies and techniques used, it should be not much of a problem to dig deeper. I believe that once the basics are understood, diving into details is always about learning new facts and relating them with information already learned before.

Talking about current detection mechanisms and incident handling strategies is delicate. Botnets are still an elitist's playground only and are only now entering the radar of companies and individuals. Which means that there are not many incident handling strategies and we are still mostly reacting to incidents but not proactively mitigating the threat. There is definitely backlog demand and this objective would be a perfect topic to do further research on.

Our networks are very vulnerable and we rely heavily on our communication infrastructure. I learned about many ugly possibilities and the ones in this thesis are definitely frightening. This surely would also be an interesting topic to dive into with another project.

Part 4

References & Appendices

Bibliography

[AF06] J. Aycock and N. Friess. Spam zombies from outer space. Technical Report 2006-808-01, University of Calgary, Computer Science, February 2006.

[AFA07] R. Acton, N. Friess, and J. Aycock. Inverse geolocation: Worms with a sense of direction. *pcc*, 0:487–493, 2007.

[Ale07] Alexey. Don't update with that update.exe. Website, December 2007. `http://www.f-secure.com/weblog/archives/00001308.html`.

[All07] The Honeynet Project & Research Alliance. *Know Your Enemy: Fast-Flux Service Networks*, July 2007.

[AM07] J. Aycock and A. Maurushat. Future threats. In *17th Virus Bulletin International Conference*, pages 275–281, 2007.

[Ayc07] J. Aycock. Covert zombie ops. In *17th Virus Bulletin International Conference*, 2007.

[Bar07] D. Barroso. Botnets – the silent threat. Technical report, Institution, November 2007. ENISA Position Paper No. 3.

[BB07] P. Barford and M. Blodgett. Toward botnet mesocosms. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[BBC08] Kidnapped irishman held to ransom. Website, January 2008. `http://news.bbc.co.uk/1/hi/northern_ireland/7073108.stm`.

[Bon99] V. Bontchev. Re: Features versus security. Mailinglist, March 1999. `http://www.security-express.com/archives/ntbugtraq/1998-1999/msg00443.html`.

[Bra04] P. Brauch. Geld oder netz! *c't*, 2004(14), 2004. `http://www.heise.de/ct/04/14/048/`.

[BS006] For-profit botnet. Website, February 2006. `http://www.schneier.com/blog/archives/2006/02/forprofit_botne.html`.

[BS07] E.F.G. Bergande and J.F. Smedsrud. *Using Honeypots to Analyze Bots and Botnets*. Norwegian University of Science and Technology, June 2007.

[BW007] Curious yellow: The first coordinated worm design. Website, September 2007. `http://blanu.net/curious_yellow.html`.

[CDC07] Medialist. Website, December 2007. `http://www.cultdeadcow.com/news/medialist.htm`.

[CJM05] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, pages 39–44, June 2005.

[Dan07] D. Danchev. The russian business network. Website, October 2007. `http://ddanchev.blogspot.com/2007/10/russian-business-network.html`.

[Dan08] D. Danchev. The shark3 malware is in the wild. Website, January 2008. `http://ddanchev.blogspot.com/2008/01/shark3-malware-is-in-wild.html`.

[D.G08] D.Goodin. Vmware vuln exposes the perils of virtualization. Website, February 2008. `http://www.theregister.co.uk/2008/02/25/vmware_critical_vuln/`.

[DGZ+05] D. Dagon, G. Gu, C. Zou, J. Grizzard, S. Dwivedi, W. Lee, and R. Lipton. A taxonomy of botnets. c., 2005.

[Dou02] J. R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[DSo07] N. Daswani, M. Stoppelman, and others. The anatomy of clickbot.a. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Berkeley, CA, USA, 2007. USENIX Association.

[EO007] robot. Website, December 2007. http://www.etymonline.com/index.php?term=robot.

[FA07] N. Fries and J. Aycock. Black market botnets. Technical Report 2007-873-25, Institution, July 2007.

[FdP07] A. Fucs, A. P. de Barros, and V. Pereira. *New botnet trends and threats*, March 2007.

[FPPS07] J. Franklin, V. Paxson, A. Perrig, and S Savage. *An Inquiry into the Nature and Cause of the Wealth of Internet Miscreants*, November 2007.

[FSJ08] It security threat summary for h2 2007: Bulk amounts of malware, storm, apple, and databases. Website, January 2008. http://www.f-secure.com/2007/2/index.html.

[GH07] J. Goebel and T. Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Berkeley, CA, USA, 2007. USENIX Association.

[Gos07] A. Gostev. Malware evolution: January - march 2007. Website, May 2007. http://www.viruslist.com/en/analysis?pubid=204791938.

[Got07] G. Goth. Fast-moving zombies: Botnets stay a step ahead of the fixes. *IEEE Internet Computing*, 11(2):7–9, 2007.

[HAJ07] R. Hemmingsen, J. Aycock, and M. Jacobsen. *Jr. Spam, Phishing, and the Looming Challenge of Big Botnets*, May 2007.

[hei07] heise Security. Antivirus protection worse than a year ago. Website, December 2007. http://www.heise-security.co.uk/news/100900.

[HEP08] Laut bka nehmen schäden durch phishing rasant zu. Website, March 2008. http://www.heise.de/newsticker/Laut-BKA-nehmen-Schaeden-durch-Phishing-rasant-zu--/meldung/105104/.

[HM04] G. Hoglund and G. McGraw. *Exploiting Software, How to break code.* Addison Wesley, 2004.

[Hol08] T. Holz. Measuring the success rate of storm worm. Website, January 2008. http://honeyblog.org/archives/156-Measuring-the-Success-Rate-of-Storm-Worm.html.

[HP08] T. Holz and N. Provos. *Virtual Honeypots.* Addison Wesley, 2008.

[Hru08] J. Hruska. A/v companies struggling to keep up with malware variations. Website, January 2008. http://arstechnica.com/news.ars/post/20080105-av-companies-struggling-to-keep-up-with-malware-variations.html.

[IH05] N. Ianelli and A. Hackworth. *Botnets as a Vehicle for Online Crime*, December 2005.

[Int06] Trend Micro International. *Taxonomy of Botnet Threats.* A Trend Micro White Paper, November 2006.

[ITN07] ITNews.com.au. Storm worm botnet more powerful than top supercomputers. Website, September 2007. http://www.itnews.com.au/News/60752,storm-worm-botnet-more-powerful-than-top-supercomputers.aspx.

[Jos07] Jose. Trick or treat with stormy helloween. Website, December 2007. http://www.f-secure.com/weblog/archives/00001304.html.

[Kim08] Kimmo. Mbr rootkit, a new breed of malware. Website, March 2008. http://www.f-secure.com/weblog/archives/00001393.html.

[KL03] J. Kannan and K. Lakshminarayanan. *Implications of Peer-to-Peer Networks on Worm Attacks and Defenses.*, 2003.

[KRH07] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.

[Mar07] V. Martínez. *PandaLabs Report: MPack Uncovered*, 2007.

[McA08] R. McArdle. Storm gets new toys for christmas. Website, January 2008. http://blog.trendmicro.com/storm-gets-new-toys-for-christmas/.

[McM08] R. McMillan. Flash attack could take over your router. Website, January 2008. http://www.pcworld.com/article/id,141399-pg,1/article.html.

[Mes07] MessageLabs. Latest stormworm developments through worldwide botnet of 1.8 million computers. Website, September 2007. http://www.messagelabs.com/resources/press/4501.

[MM02] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *In Proceedings of IPTPS02*, March 2002.

[Mon07] P. Montoya. Cybercrime... for sale (i). Website, April 2007. `http://pandalabs.pandasecurity.com/archive/Cybercrime_2E002E002E00_-for-sale-_2800_I_2900_.aspx`.

[Mor05] C. Morris. 40,000 euros offered for identities of online blackmailers. Website, December 2005. `http://www.heise.de/english/newsticker/news/63238`.

[MSB02] D. Moore, C. Shannon, and J. Brown. Code-red: A case study on the spread and victims of an internet worm. In *In Proceedings of the ACM Internet Measurement Workshop*, November 2002.

[NA05] S. Nagarja and R. Anderson. The topology of covert conflict. Technical Report UCAM-CL-TR-637, University of Cambridge, July 2005.

[NBL06] V. Nivargi, M. Bhaowal, and T. Lee. *Machine Learning Based Botnet Detection*, 2006.

[New07] News.com. Storm worm rivals world's best supercomputers. Website, September 2007. `http://news.com.com/8301-10784_3-9774071-7.html`.

[Oll04] G. Ollmann. The phishing guide: Understanding & preventing phishing attacks. Website, September 2004. `http://www.ngssoftware.com/papers/NISR-WP-Phishing.pdf`.

[Oll05] G. Ollmann. The pharming guide: Understanding & preventing dns-related attacks by phishers. Website, July 2005. `http://www.ngssoftware.com/research/papers/ThePharmingGuide.pdf`.

[PSY07] P. Porras, H. Saidi, and V. Yegneswaran. *A Multi-perspective Analysis of the Storm (Peacomm) Worm*, October 2007.

[QS06] D. Quist and V. Smith. Detecting the presence of virtual machines using the local data table. Technical report, Offensive Computing, 2006.

[Reg07] The Register. Trial in 419-related murder under way. Website, April 2007. `http://www.channelregister.co.uk/2007/04/10/nigerian_murder_trial/`.

[RZMT06] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM on Internet Measurement (IMC)*, pages 41–52, 2006.

[RZMT07] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 5–5, Berkeley, CA, USA, 2007. USENIX Association.

[Sav05] S. Savage. Large-scale worm detection. In *In ARO-DHS Special Workshop on Malware Detection*, August 2005.

[Sch03] B. Schneier. *Beyond Fear*. Copernicus Books, 2003.

[Sha05] Shadowserver foundation. Website, March 2005. `http://www.shadoserver.org/`.

[SPW02] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the internet in your spare time. In *Proceedings of the 11th USENIX Security*, 2002.

[SWLL06] W.T. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting botnets with tight command and control. *lcn*, 0:195–202, 2006.

[Tea05] Team cymru. Website, March 2005. `http://www.team-cymru.org/`.

[Tun07] L. Tung. Infamous russian isp behind bank of india hack. Website, September 2007. `http://news.zdnet.co.uk/security/0,1000000189,39289057,00.htm`.

[VA06] R. Vogt and J. Aycock. Attack of the 50 foot botnet. Technical Report 2006-840-33, Department of Computer Science, University of Calgary, August 2006.

[War07] G. Warner. Cut the head off the snake. Website, June 2007. `http://www.birmingham-infragard.org/meetings/talks/presentations/DDOS.Cut.the.Head.Off.The.Snake.pdf`.

[WAS] Waste. Website. `http://waste.sourceforge.net/index.php?id=news`.

[Wika] Wikipedia. Click fraud. Website. `http://en.wikipedia.org/wiki/Click_Fraud`.

[Wikb] Wikipedia. Malware. Website. `http://en.wikipedia.org/wiki/Malware`.

[WPSC03] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 11–18, New York, NY, USA, 2003. ACM.

[WSZ07] P. Wang, S. Sparks, and C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, Berkeley, CA, USA, 2007. USENIX Association.

[Wüe08] C. Wüest. Phishing for easter eggs. Website, March 2008. `http://www.symantec.com/enterprise/security_response/weblog/2008/03/phishing_for_easter_eggs.html`.

[XFO] Xforce. Website. `http://xforce.iss.net/xforce/alerts/id/advise8`.

[ZC06] C. C. Zou and R. Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 199–208, June 2006.

[ZHH+07] J. Zhuge, T. Holz, X. Han, J. Guo, and W. Zou. *Characterizing the IRC-based Botnet Phenomenon*, December 2007.

[ZHS+04] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

APPENDIX A

Evidence of distributed attacks

The following lines show an extract of a distributed brute force attack on the computer "host" on October 22, 2007 (the original hostname was replaced with "host"). One single IP address only tried between one and three passwords before giving up. The distributed scan ran for a few hours. These log entries look like originating from a botnet since the connections are TCP oriented and faking the sending IP address would be of no help.

```
[...]

10:05:51 host sshd: Authentication failure for user mysql from 81.2.220.43

10:07:29 host sshd: Authentication failure for user mysql from 74.232.154.114

10:10:06 host sshd: Authentication failure for user mysql from 83.13.20.252

10:12:05 host sshd: Authentication failure for user mysql from 216.221.95.27

10:14:04 host sshd: Authentication failure for user mysql from 81.75.126.101

10:16:38 host sshd: Authentication failure for user mysql from 82.234.183.60

10:18:51 host sshd: Authentication failure for user mysql from 200.81.233.18

10:20:31 host sshd: Authentication failure for user mysql from 80.218.30.113

10:23:01 host sshd: Authentication failure for user mysql from 85.214.54.182

10:25:04 host sshd: Authentication failure for user mysql from 87.54.26.146

10:27:09 host sshd: Authentication failure for user mysql from 62.212.121.156

10:30:09 host sshd: Authentication failure for user mysql from 202.106.60.24

10:31:49 host sshd: Authentication failure for user mysql from 200.79.37.194

10:34:43 host sshd: Authentication failure for user mysql from 200.172.166.2

10:36:29 host sshd: Authentication failure for user mysql from 64.180.238.88

10:38:24 host sshd: Authentication failure for user mysql from 83.17.126.94

10:41:15 host sshd: Authentication failure for user mysql from 200.207.9.57

10:43:05 host sshd: Authentication failure for user mysql from 200.204.141.237

10:44:58 host sshd: Authentication failure for user mysql from 212.190.88.173

10:47:54 host sshd: Authentication failure for user mysql from 206.83.201.107

10:49:36 host sshd: Authentication failure for user mysql from 83.3.138.50

[...]
```

APPENDIX B

Agobot3

This is an extract of the Agobot3 source code. It shows the method which is used to kill running AV software processes. Please note the comment in the header of the source code where the virus author describes where he found the list of common names for processes he would like to kill.

This is very simple code which tries to kill a process with every name it knows without looking up if this process actually runs or not.

```
/*

This kills all active Antivirus processes that match

Thanks to FSecure's Bugbear.B analysis @

http://www.f-secure.com/v-descs/bugbear_b.shtml

*/

void KillAV()

{

    #ifdef WIN32

        const char szFilenamesToKill[455] = { "ACKWIN32.EXE", "ADVXDWIN.EXE",

                                              "AGENTSVR.EXE", "ALERTSVC.EXE",

                                              [...]

                                              "zapro.EXE", "zonealarm.EXE", NULL };

        for(int i=0; szFilenamesToKill[i]!=NULL; i++)

        KillProcess(szFilenamesToKill[i]);

    #else

        KillProcess("tcpdump"); KillProcess("ethereal");

    #endif

}
```

APPENDIX C

MPack v0.94

The following examples are from the index.php file from the MPack software. MPack is a web server based software which tries to infect its visitors with many different exploits.

The first example shows that the application tries to find out which browser variant the visitor is using and which OS the browser is running on. Depending on the client specifications, MPack sends different exploits to the visitor (source code slightly edited for formatting purposes):

```
//exploits combination

if ($browser[name]=="MSIE")

{

   if ($browser[os]!="Windows NT 5.0")

   {

      AddIP("0day");

      include 'crypt.php';

      include 'megapack1.php';

   }

   if ($browser[os]=="Windows NT 5.0")

   {

      AddIP("jar");

      include 'ms06-044_w2k.php';

      include 'megapack1.php';

   }

   //'ms06-044_w2k.php'; ex crypt

}

if ($browser[name]=="Firefox")

{

   AddIP("firefox");

   include 'ff.php';

}

if ($browser[name]=="Opera")
```

```
{

   if (substr($browser[version], 0, 1)<"8")

   {

      AddIP("opera7");

      include 'o7.php';

   }

}
```

It is also interesting to see that there are still some developer and debug comments within the source code (slightly edited):

```
// Windows NT 5.0 = Win2000

// Windows NT 5.1 = WinXP sp0,1

// Windows NT 5.1 SP2 = WinXP sp2 (Windows NT 5.1; SV1) under IE

// Windows NT 5.2 = Win2003

[...]

//if ($browser[name]!="Opera") && ($browser[name]!="Firefox") && ($browser[name]!="MSIE")

//{

//    include 'megapack1.php';

//}

//echo getenv("HTTP_USER_AGENT")."<br>";

//echo "Browser: ".$browser[name]."<br> Browser Ver: ".$browser[version]."<br>OS: ".$browser[os];
```