A hands-on approach on botnets for a learning proposal

Computer Systems Security (EIC0072)

Eduardo José Valadar Martins (ei11104@fe.up.pt)
João Pedro Matos Teixeira Dias (ei11137@fe.up.pt)
José Pedro Vieira de Carvalho Pinto (ei12164@fe.up.pt)
João Carlos Teixeira de Sá (ei11142@fe.up.pt)

Group 6 - Theme 12

December 7, 2015

Contents

1 Introduction

A botnet is, by definition, the name given to any collection of compromised hosts (PCs) controlled by an attacker remotely. Botnets generally are created by a specific attacker or small group of attackers using one piece of malware to infect a large number of machines. The individual machines that are part of the botnet are, generally, called *bots*, *nodes* or *zombies*. There are botnets of various sizes (there is no minimum number of infected machines to the group be called a botnet), and they can vary from small ones with hundreds or low thousands of infected machines and larger ones with millions of compromised hosts [29].

There are specific cases where bots perform beneficial and even vital activities [31], for example, the use of web crawlers by search engines to index webpages can be considered as a type of bot, also, the participants of SETI@home initiative (Search for Extraterrestrial Intelligence), are, voluntary, part of a large botnet used to analyze radio telescope data in order to track evidence of intelligent extraterrestrial life [32].

Attackers usually install bots by exploiting vulnerabilities in software or by using social engineering tactics to trick users into installing malware. Users are often unaware that their computers are being used for malicious purposes [31].

Botnets can be used for various activities but the most traditional and common use is for DDoS (Distributed Denial of Service) attacks. These attacks rely on the computing power and bandwidth of hundreds or thousands of PCs to send huge amounts of traffic at a specific Web site in an effort to knock the site offline. Other uses for botnets are, for example, spamming, sniffing traffic, keylogging and even manipulating online polls and games [17].

As of today, we think that it's essential that there is a simple way to learn about botnets, what they are, how they work and what we can do about it. For that we built a two modules project. The first one consists of a *wiki* with information regarding botnets, its anatomy, architecture and impact on the technological world. The second part is a more technical one, consisting on a laboratory with a simple and open-source botnet kit with a set of built-in functionalities for anyone who is interested and want to setup a laboratory at home and play with it, finding out how can it be changed and adapted for almost anything.

2 Research and background

2.1 Botnet anatomy

The term botnet is widely used when several IRC bots have been linked and may possibly set channel modes on other bots and users while keeping IRC channels free from unwanted users. This is where the term is originally from, since the first illegal botnets were similar to legal botnets.

Botnets sometimes compromise computers whose security defenses have been breached and control conceded to a third party. Each such compromised device, known as a “bot”, is created when a computer is penetrated by software from a malware (malicious software) distribution. However, it could also be someone (or a spider) that hacks into a computer. The controller of a botnet is able to direct the activities of these compromised computers through communication channels formed by standards-based network protocols such as IRC or Hypertext Transfer Protocol (HTTP) [37].

After successful exploitation, a bot uses Trivial File Transfer Protocol (TFTP), File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), or CSend (an IRC extension to send files to other users, comparable to DCC) to transfer itself to the compromised host. The binary is started, and tries to connect to the hard-coded master IRC server. Often a dynamic DNS name is provided rather than a hard coded IP address, so the bot can be easily relocated. Some bots even remove themselves if the given master server is localhost or in a private subnet, since this indicates an unusual situations.

Once an attacker is in control of the bots, they can do whatever they want with the bots: Searching for sensitive information on all compromised machines and DCC-sending these files to another machine, DDoS-ing individuals or organizations, or enabling a keylogger and looking for PayPal or eBay account information. These are just a few possible commands, other options have been presented in the previous section. The IRC server that is used to connect all bots is in most cases a compromised box. This is probably because an attacker would not receive operator-rights on a normal chat network and thus has to set-up their own IRC server which offers more flexibility.

2.1.1 Organization

A botnet’s originator (known as a “bot herder” or “bot master”) can control the group remotely, usually through IRC, and often for criminal purposes. Though rare, more experienced botnet operators program command protocols from scratch. These protocols include a server program, a client program for operation, and the program that embeds the client on the victim’s machine. This server is known as the command-and-control (C&C) server. The bots communicate over a network, using a unique encryption scheme for stealth and protection against detection or intrusion into the botnet.

A bot typically runs hidden and uses a covert channel to communicate with its C&C server. Generally, the perpetrator has compromised multiple systems using various tools (exploits, buffer overflows, as well as others; see also RPC). Newer bots can automatically scan their environment and propagate themselves using vulnerabilities and weak passwords. Generally, the more vulnerabilities a bot can scan and propagate through, the more valuable it becomes to a botnet controller community. The process of stealing computing resources as a result of a system being joined to a “botnet” is sometimes referred to as “scrumping.”

A few important aspects about to note how botnets have countered measures against them are:

- Botnet servers are typically redundant, linked for greater redundancy so as to reduce the threat of a takedown. Actual botnet communities usually consist of one or several controllers that rarely have highly developed command hierarchies; they rely on individual peer-to-peer relationships[33].

- Botnet architecture evolved over time, and not all botnets exhibit the same topology for command and control. Advanced topology is more resilient to shutdown, enumeration or discovery. However, some topologies limit the marketability of the botnet to third parties [18]. Typical botnet topologies are star, multi-server, hierarchical and random.

- Some botnets are scaling back in size to minimize detection. As of 2006, the average size of a network was estimated at 20,000 computers [19].

2.1.2 Types of botnets

- **Agobot (parent of Phatbot/Forbot/XtremBot)**

  This is probably the best known bot. There are more than 500 known different versions of Agobot and this number is increasing. The bot itself is written in C++ with cross-platform capabilities and the source code is put under the GPL. The latest available versions of Agobot are written in tidy C++ and show a really high abstract design. The bot is structured in a very modular way, and it is very easy to add commands or scanners for other vulnerabilities. Agobot uses libpcap (a packet sniffing library) and Perl Compatible Regular Expressions (PCRE) to sniff and sort traffic. Agobot can use NTFS Alternate Data Stream (ADS) and offers Rootkit capabilities like file and process hiding to hide its own presence on a compromised host. Furthermore, reverse engineering this malware is harder since it includes functions to detect debuggers (e.g. SoftICE and OllyDbg) and virtual machines (e.g. VMWare and Virtual PC). In addition, Agobot is the only bot that utilized a control protocol other than IRC. Furthermore, the Linux version is able to detect the Linux distribution used on the compromised host and sets up a correct init script [17].

- **SDBot (parent of RBot/UrBot/UrXBot/etc.)**

  This family of malware is at the moment the most active one. SDBot is written in very poor C and also published under the GPL. It is the father of RBot, RxBot, UrBot, UrXBot, JrBot,.. and probably many more. The source code of this bot is not very well designed or written. Nevertheless, attackers like it, and it is very often used in the wild. It offers similar features to Agobot, although the command set is not as large, nor the implementation as sophisticated [17].

- **mIRC-based Bots - GT-Bots**

  We subsume all mIRC-based bots as GT-bots, since there are so many different versions of them that it is hard to get an overview of all forks.

mIRC itself is a popular IRC client for Windows. GT is an abbreviation for Global Threat and this is the common name used for all mIRC-scripted bots. These bots launch an instance of the mIRC chat-client with a set of scripts and other binaries. One binary you will never miss is a HideWindow executable used to make the mIRC instance unseen by the user. The other binaries are mainly Dynamic Link Libraries (DLLs) linked to mIRC that add some new features the mIRC scripts can use. The mIRC-scripts, often having the extension ".mrc", are used to control the bot. They can access the scanners in the DLLs and take care of further spreading. GT-Bots spread by exploiting weaknesses on remote computers and uploading themselves to compromised hosts [17].

These three last examples of bots are probably the most common ones, in terms of usage and also que more capable ones as for their features and implementation. There are however other types of bots which given their very interesting features are worthy of mentioning in this topic.

- **DSNX Bots**

  The Dataspy Network X (DSNX) bot is written in C++ and has a convenient plugin interface. An attacker can easily write scanners and spreaders as plugins and extend the bot's features. Again, the code is published under the GPL. This bot has one major disadvantage: the default version does not come with any spreaders. But plugins are available to overcome this gap. Furthermore, plugins that offer services like DDoS-attacks, portscan-interface or hidden HTTP-server are available [17].

- **Q8 Bots**

  Q8bot is a very small bot, consisting of only 926 lines of C-code. And it has one additional noteworthiness: It's written for Unix/Linux systems. It implements all common features of a bot: Dynamic updating via HTTP-downloads, various DDoS-attacks (e.g. SYN-flood and UDP-flood), execution of arbitrary commands, and many more. In the version we have captured, spreaders are missing. But presumably versions of this bot exist which also include spreaders [17].

- **kaiten**

  This bot lacks a spreader too, and is also written for Unix/Linux systems. The weak user authentication makes it very easy to hijack a botnet running with kaiten. The bot itself consists of just one file. Thus it is very easy to fetch the source code using wget, and compile it on a vulnerable box using a script. Kaiten offers an easy remote shell, so checking for further vulnerabilities to gain privileged access can be done via IRC [17].

- **Perl-based bots**

  There are many different version of very simple based on the programming language Perl. These bots are very small and contain in most cases only a few hundred lines of code. They offer only a rudimentary set of commands (most often DDoS-attacks) and are used on Unix-based systems [17].

2.1.3 Malicious uses of botnets

A botnet is nothing more than a tool, there are as many different motives for using. The most common uses are criminally motivated or for destructive purposes. The possibilities to use botnets can be categorized as listed below. And since a botnet is nothing more than a tool, there are most likely other potential uses that we have not listed.

- **Distributed Denial-of-Service Attacks**

  Often botnets are used for Distributed Denial-of-Service (DDoS[1]) attacks. A DDoS attack is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system. In addition, the resources on the path are exhausted if the DDoS-attack causes many packets per second (pps). Most commonly implemented and also very often used are TCP SYN and UDP flood attacks [6].

  Note that DDoS attacks are not limited to web servers, virtually any service available on the Internet can be the target of such an attack. Higher-level protocols can be used to increase the load even more effectively by using very specific attacks, such as running exhausting search queries on bulletin boards or recursive HTTP-floods on the victim's website. Recursive HTTP-flood means that the bots start from a given HTTP link and then follows all links on the provided website in a recursive way. This is also called spidering [6].

- **Spamming**

  Some bots offer the possibility to open a SOCKS v4/v5 proxy - a generic proxy protocol for TCP/IP-based networking applications (RFC 1928[2]) - on a compromised machine. After having enabled the SOCKS proxy, this machine can then be used for nefarious tasks such as spamming. With the help of a botnet and thousands of bots, an attacker is able to send massive amounts of bulk email (spam). Some bots also implement a special function to harvest email-addresses. Often that spam you are receiving was sent from, or proxied through, old Windows computers sitting at home. In addition, this can, of course, also be used to send phishing-mails since phishing is a special case of spam [1] [17].

- **Sniffing Traffic**

  Bots can also use a packet sniffer to watch for interesting clear-text data passing by a compromised machine. The sniffers are mostly used to retrieve sensitive information like usernames and passwords. But the sniffed data can also contain other interesting information. If a machine is compromised more than once and also a member of more than one botnet, the packet sniffing allows to gather the key information of the other botnet. Thus it is possible to “steal” another botnet [17].

- **Keylogging**

  If the compromised machine uses encrypted communication channels (e.g. HTTPS or POP3S), then just sniffing the network packets on the victim’s computer is useless since the appropriate key to decrypt the packets is missing. But most bots also offer features to help in this situation. With the help of a keylogger it is very easy for an attacker to retrieve sensitive information. An implemented filtering mechanism (e.g. “I am only interested in key sequences near the keyword ‘paypal.com’”) further helps in stealing secret data. And if you imagine that this keylogger runs on thousands of compromised machines in parallel you can imagine how quickly PayPal accounts are harvested [17].

- **Spreading new malware**

  In most cases, botnets are used to spread new bots. This is very easy since all bots implement mechanisms to download and execute a file via HTTP or FTP. But spreading an email virus using a botnet is a very nice idea too. A botnet with 10.000 hosts which acts as the start base for the mail virus allows very fast spreading and thus causes more harm. The Witty worm, which attacked the ICQ protocol parsing implementation in Internet Security Systems (ISS) products is suspected to have been initially launched by a botnet due to the fact that the attacking hosts were not running any ISS services [17].

- **Installing Advertisement Addons and Browser Helper Objects (BHOs [20])**

  Botnets can also be used to gain financial advantages. This works by setting up a fake website with some advertisements: The operator of this website negotiates a deal with some hosting companies that pay for clicks on ads. With the help of a botnet, these clicks can be “automated” so that instantly a few thousand bots click on the pop-ups. This process can be further enhanced if the bot hijacks the start-page of a compromised machine so that the “clicks” are executed each time the victim uses the browser [17] [20].

- **Google AdSense abuse**

  A similar abuse is also possible with Google’s AdSense [24] program: AdSense offers companies the possibility to display Google advertisements on their own website and earn money this way. The company earns money due to clicks on these ads, for example per 10.000 clicks in one month. An attacker can abuse this program by leveraging his botnet to click on these advertisements in an automated fashion and thus artificially increments the click counter. This kind of usage for botnets is relatively uncommon, but not a bad idea from an attacker's perspective [17] .

- **Attacking IRC Chat Networks**

  Botnets are also used for attacks against Internet Relay Chat (IRC) networks. Popular among attackers is especially the so called "clone attack": In this kind of attack, the controller orders each bot to connect a large number of clones to the victim IRC network. The victim is flooded by service request from thousands of bots or thousands of channel-joins by these cloned bots. In this way, the victim IRC network is brought down - similar to a DDoS attack [17].

- **Manipulating online polls/games**

  Online polls/games are getting more and more attention and it is rather easy to manipulate them with botnets. Since every bot has a distinct IP address, every vote will have the same credibility as a vote cast by a real person. Online games can be manipulated in a similar way [17].

- **Mass identity theft**

  Often the combination of different functionality described above can be used for large scale identity theft, one of the fastest growing crimes on the Internet. Bogus emails ("phishing mails") that pretend to be legitimate (such as fake PayPal or banking emails) ask their intended victims to go online and submit their private information. These fake emails are generated and sent by bots via their spamming mechanism. These same bots can also host multiple fake websites pretending to be Ebay, PayPal, or a bank, and harvest personal information. Just as quickly as one of these fake sites is shut down, another one can pop up. In addition, keylogging and sniffing of traffic can also be used for identity theft [17].

This list demonstrates that attackers can cause a great deal of harm or criminal activity with the help of botnets. Many of these attacks – especially DDoS attacks - pose severe threats to other systems and are hard to prevent. In addition, we are sure there are many other uses we have not covered in this list.

2.1.4 Vulnerabilities exploitation

In computer security, a vulnerability is defined as: a weakness that allows an attacker to reduce a system's information assurance. Vulnerability is a intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw [37].

Due to their immense size - botnets can consist of several ten thousand compromised machines - botnets pose serious threats. Distributed denial-of-service (DDoS) attacks are one such threat. Even a relatively small botnet with only 1000 bots can cause a great deal of damage. These 1000 bots have a combined bandwidth (1000 home PCs with an average upstream of 128KBit/s can offer more than 100MBit/s) that is probably higher than the Internet connection of most corporate systems. In addition, the IP distribution of the bots makes ingress filter construction, maintenance, and deployment difficult. In addition, incident response is hampered by the large number of separate organizations involved.

Another use for botnets is stealing sensitive information or identity theft: Searching some thousands home PCs for password.txt, or sniffing their traffic, can be effective. The spreading mechanisms used by bots is a leading cause for "background noise" on the Internet, especially on TCP ports 445 and 135. In this context, the term spreading describes the propagation methods used by the bots. These malware scan large network ranges for new vulnerable computers and infect them, thus acting similar to a worm or virus.

2.2 History

The "boom" of the internet and the massive increase of the use of the e-mail made the conditions ideal for the appearance of botnets. In fact, as you will see below, most of the botnets detected until today are somehow related to sending enourmous amounts of e-mails, especially spam. Be aware that the estimated number of bots presented below are just that, estimatives, because in some countries it is common that users change their IP address several times a day.

- 2004
  - **Bagle** [3]
  - *Est. no of bots:* 230 000
  - Bagle (also known as Beagle) is a mass-mailing computer worm affecting all versions of Microsoft Windows. Bagle uses its own SMTP engine to mass-mail itself as an attachment to recipients. It copies itself to the Windows system directory and opens a backdoor.
- 2006
  - **Rustock** [12]
  - *Est. no of bots:* 150 000
  - Rustock botnet consisted of computers running Microsoft Windows and was capable of sending up to 25,000 spam messages per hour from an infected PC. To increase the size of the botnet it would use self-propagation by sending malicious e-mails with a trojan which would incorporate the targeted machine into the botnet.
- 2007
  - **Akbot** [2]
  - *Est. no of bots:* 1 300 000

  - Akbot is an IRC controlled backdoor program that allows an outside user to take control of the infected computer. It operates by joining IRC servers and then waiting for further instructions. Once installed, Akbot can be used to gather data, kill processes, or perform DDOS attacks.
  - **Cutwail** [5]
  - *Est. no of bots:* 1 500 000
  - Cutwail botnet is a botnet mostly involved in sending spam e-mails. Typically, it uses a Trojan component called Pushdo to infect a machine. It affects computers running Microsoft Windows.
  - **Srizbi** [14]
  - *Est. no of bots:* 450 000
  - Srizbi botnet is a botnet mainly involved in sending spam e-mails. It infects computers with the Srizbi trojan, which allows to send spam on command.

- 2008

  - **Mariposa** [9]
  - *Est. no of bots:* 12 000 000
  - The Mariposa botnet is a botnet mainly involved in cyberscamming and denial of service attacks. It is one of the largest known botnets with up to 12 million unique IP addresses.
  - **Sality** [13]
  - *Est. no of bots:* 1 000 000
  - Sality is a family of malicious software which infects files on Microsoft Windows systems. Machines infected with Sality are able to communicate over a peer-to-peer (P2P) network for the purpose of relaying spam, proxying of communications, exfiltrating sensitive data, compromising web servers and/or coordinating distributed computing tasks for the purpose of processing intensive tasks (e.g. password cracking).
  - **Conficker** [4]
  - *Est. no of bots:* 10 500 000+
  - Conficker, also known as Downup, Downadup and Kido, is a computer worm targeting the Microsoft Windows operating system. The botnet uses flaws in Windows OS software and dictionary attacks on administrator passwords to propagate itself.
  - **Grum** [7]
  - *Est. no of bots:* 560 000

  - The Grum botnet, also known by its alias Tedroo and Reddyb, was a botnet mostly involved in sending pharmaceutical spam e-mails. It relies on two types of control servers for its operation. One type is used to push configuration updates to the infected computers, and the other is used to tell the botnet what spam emails to send.

- 2009
  - **BredoLab**
  - *Est. no of bots:* 30 000 000
  - The BredoLab Botnet, also known by its alias Oficla, was a botnet mostly involved in viral e-mail spam.

- 2010
  - **Kelihos** [8]
  - *Est. no of bots:* 300 000+
  - The Kelihos botnet, also known as Hlux, is a botnet mainly involved in the theft of bitcoins and spamming.
  - **TDL-4** [15]
  - *Est. no of bots:* 4 500 000+
  - TDL-4 is a botnet and the name of the rootkit that runs the botnet (also known as Alureon). It infects the master boot record of the target machine, making it harder to detect and remove.

- 2011
  - **Ramnit** [11]
  - *Est. no of bots:* 3 000 000
  - Ramnit is a computer worm affecting Windows operating system. The Ramnit botnet was dismantled by Europol and Symantec securities in 2015.
  - **ZeroAccess** [16]
  - *Est. no of bots:* 2 000 000
  - ZeroAccess, also known as Max++ and/or Sirefef, is a botnet mostly involved in bitcoin mining and click fraud. It uses a Trojan horse computer malware that affects Microsoft Windows operating systems and downloads other malware on an infected machine while remaining hidden by using rootkit techniques.

- 2012
  - **Nitol** [10]
  - *Est. no of bots:* unknown

  - The Nitol botnet is a botnet mostly involved in spreading malware and distributed denial-of-service attacks. The botnet is mostly prevalent in China where an estimate 85% of the infections are detected. It was found to be present on systems that came brand-new from the factory, indicating the trojan was installed somewhere during the assembly and manufacturing process.

- 2014
  - **Semalt (aka Soundfrost)** [39]
  - *Est. no of bots:* 300 000+
  - Semalt is a botnet mainly involved mainly involved in sending spam e-mails. It visits random websites to generate referral and spies on users browsing habits.

This list demonstrates that there has been several botnets that were able to reach a big dimension infecting thousands (or even millions) of machines. Most of the botnets presented were spam sending related but it is also noticeable that there has been an evolution and they tend to do more than just that. Also botnets have become more complex making it harder to dismantle them.

2.3 Countermeasures

2.3.1 Botnet Detection

Botnet detection deals with the identification of bots in the machine or network so that some sort of remedy can be done.

In recent years botnet detection has been a hot topic in the research community due to increase in the malicious activity. According to the majority of the common characteristic of a bot malwareare related to network activities the bots require some sort of interaction with the command and control servers. Some of the common activities one could monitor to detect botnets are:

- *opening of specific ports*
- *establishing a number of unwanted network connections*
- *downloading and executing files and programs*
- *creating new processes with well-known names*
- *disabling antivirus software*

Detection techniques

Intrusion Detection System (IDS) is an approach for botnet detection that can be either a signature or anomaly-based technique [28].

- **Signature-Based**: A signature-based Botnet detection technique uses the signatures of current Botnets for its detection. This method has several advantages, such as very low false alarm rate, immediate detection, easier to implement and there is better information about the type of detected attack. Signature based detection method can only able to detect well known botnets.

  Thus, unknown Botnets can't be detected by this method. Anomaly-based detection techniques are introduced to overcome this drawback.

- **Anomaly-Based**: The idea behind anomaly-based detection approach is to perform botnet detection by considering several different network traffic anomalies, including high network latency, high traffic volume, traffic on unusual ports, and unusual system behavior that could indicate the presence of malicious bots in the network.

  Anomaly-based detection techniques are further divided into host-based detection and network-based detection.

  A host-based technique is a detection strategy which monitors and analyzes the internals of a computer system instead of network traffics on its external interfaces. In this approach the individual machine is monitored to find any suspicious behavior, including its processing overhead, and access to suspicious files. If suspicious activity is detected the Host Intrusion Detection Systems will alert the user or administrator. It takes a snapshot of existing system files and matches it to the previous snapshot. If the critical system files were modified or deleted, an alert is sent to the administrator to investigate.

2.4 Botnet countermeasures

Defense against botnets is carried out by application of certain strategies. All the Internet users are responsible for defense, starting from home or business computer users, system administrators, developers, up to web administrators and ISPs. The defense must be considered as a permanent and comprehensive process in which all the activities must be proactive. This is the only way to achieve good results and to protect computers, i.e. Web services/applications against the activities with bad intentions.

Countermeasures against *botnet treats* can be broadly classified into two approaches, namely, technical approaches or social and regulatory approaches [21].

- **Technical Approach**: Most of botnet countermeasures focus on the commandand-control infrastructure of botnets by filtering botnet related traffic, sinkholing domains with the assistance of DNS registrars or obtaining the shutdown of malicious servers in data centers. The technical botnet defense approach includes Blacklisting, Packet Filtering, Reverse Engineering and Port Blocking [21].

- **Blacklisting**: A blacklist may provide single IP addresses of malicious hosts or whole subnets showing suspicious activities. A blacklist can be used to block all traffic from included addresses and also to filter websites with suspicious or proven malicious contents [21].

  For example, the Spamhaus Project provides various real-time lists that assist in identifying and blocking attempts by malicious activities. Spamhaus Block List (SBL) and the Domain Block List (DBL) contain a collection of IP addresses and domain names respectively from which incoming e-mail should not be accepted [21].

- **Packet Filtering**: The packet filtering can be applied at a host, network and ISP level. A typical component that performs packet filtering at host level is a desktop firewall. Its purpose is to monitor the network activities of all active processes. As the amount of traffic at host level is usually manageable, deep-packet inspection is applicable. Often, user or administrator interaction is required to allow or deny network access for certain applications, if no suitable rules have been specified for them yet [21].

- **Reverse Engineering**: Recovering the functionality of a program without the source code is known as reverse engineering. The malware reverse engineering technique helps in extracting the details of the installation and spreading of malware. The process involves static analysis and dynamic analysis. In case of static analysis, the binary is not executed. This phase deals with the reconstruction of certain aspects of the functionality. The dynamic analysis deals with the execution of the sample. The behavior of the malware can be determined by monitoring the host [21].

- **Port Blocking**: Port blocking is a preventive measure that can be applied by ISPs to reducing the amount of spam mails traversing their network. The use of unauthenticated services via port 25, like direct mail exchange or open relay mail servers is almost exclusively for spam distribution purposes. Hence, blocking port 25 at ISP level has been recommended as best practice [21].

3 Implementation

3.1 Botnet Wiki

In order to build the *botnet wiki*, to reach the largest number of people, we decided to build a web platform, namely a wiki, based on W3C standards - HTML, CSS and Javascript [40]. To facilitate the addition of content and edition, we used an automatic site building tool designated Jekyll [35] and its support to markdown [26].

This part was built with the objective of transmitting all the results from the research on the topic "botnets". The content presented in the wiki is divided in:

- **Anatomy of a botnets**: Describing how bots and botnets work, types of bots and attacks and vulnerability exploitation.
- **History of botnets**: A little of botnet history including the largest botnets and the impact on the technological world.
- **Botnet countermeasures**: Detection and defense techniques against botnets.
- **Botnet Lab**: Instruction on how to use and setup the *botnet laboratory*.

3.2 Botnet Laboratory

With the intuit of having a real hands-on tool for testing and developing proposes we created a botnet lab framework, a botnet kit based on the IRC communication protocol, with built-in functionalities and an easy way of expanding functionalities, in a framework way.

The botnet built using this laboratory match the general architecture for any botnet based on a Command-and-Control (C&C) architecture. Our actor is the Bot Herder or Bot Master, it operates using the a special IRC client (that is part of this laboratory), connects to a IRC-Server (in this case a IRCD-Hybrid based one) where all the bots are connected.

Whenever the Bot Herder sends a message to the IRC Server it broadcast it to all the connected bots that executes the requested job.

Special cases are the Spam request and the Screenshot/Webcam request. In the first case, the Spam, to avoid the trouble of setting up a SMTP server on all the bots we use the Mandrill API for sending the e-mail. While this can appear strange, because of centralizing all the traffic on one e-mail sender API with low free quotas and with the risk of the account being blocked, we send the API Key in the request sended to bots in a way that if a Key it's blocked we can simple send a different API Key on the request sended to the bots. Additionally it's used the PasteBin Service and it's "anonymous and hidden file" hability for hosting relevant data like the the e-mail sending list, API Key and Message and the Bot Herder just needs to send to the bots the files URL (fig.3).

In the second case, the Screenshot/Webcam, the bots uses the Imgur API for storage the images and just send the URL of that images back to the *Bot Herder* (fig.4).

Adding to all of this, it is used the freegeoip and Google Static Map API for getting and showing the relative world position of the controlled hosts (fig.5).

Also, it's used RSA encryption [36] so the Bot Herder it's the only one capable of decrypt the messages sended by the bots because it's the Private Key owner. The bots encrypts the messages using the Public Key defined by the Bot Herder.

3.2.1 Technologies

For setting up the IRC server we used *IRCd-hybrid* - "*A lightweight, high-performance internet relay chat daemon.*"[38].

In order to build the bots we used Python 2.7 [22] with the help of some libraries.

Additionally it were the need of build a *special* IRC client, and we used Python too.

Besides this, we used several external API's, namely:

- **freegeoip**: provides a public HTTP API for software developers to search the geolocation of IP addresses. It uses a database of IP addresses that are associated to cities along with other relevant information like time zone, latitude and longitude [23].
- **Google Static Map API**: create any map based on URL parameters sent through a standard HTTPS request [25].
- **Mandrill**: is a reliable, scalable, and secure delivery API for transactional emails from websites and applications [30].
- **Pastebin**: a website where we can store text for a certain period of time [34].
- **imgur API**: an image storage website [27].

All the developed objects, namely the client and the bot are cross-platform capable of running on Windows and Linux machines. The exception goes to the IRCd-Hybrid server that needs a Linux machine to work.

3.2.2 Architecture

The principal network components that interact with the botnet are listed showed in fig.1.

A typical communication that can be observed after a successful bot deploy looks like:

```
<- :irc1.XXXXXX.XXX NOTICE AUTH :*** Looking up your hostname
<- :irc1.XXXXXX.XXX NOTICE AUTH :*** Found your hostname
-> PASS secretserverpass
-> NICK [urX]-700159
-> USER mltfvt 0 0 :mltfvt
<- :irc1.XXXXXX.XXX NOTICE [urX]-700159 :*** If you are having
problems connecting due to ping timeouts, please type /quote
pong ED322722 or /raw pong ED322722 now.
<- PING :ED322722
-> PONG :ED322722
<- :irc1.XXXXXX.XXX 001 [urX]-700159 :Welcome to the irc1.XXXXXX
 IRC Network
 [urX]-700159!mltfvt@nicetry
<- :irc1.XXXXXX.XXX 002 [urX]-700159 :Your host is irc1.XXXXXX,
 running version IRCd-Hybrid
```

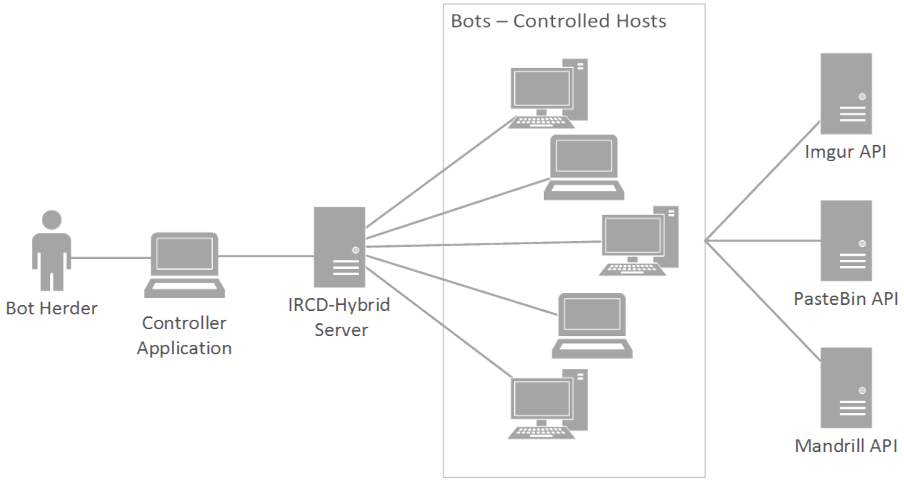

Figure 1: Network Diagram

```
<– :irc1.XXXXXX.XXX 003 [urX]−700159 :This server was created
 Set  8 18:58:31 2015
<– :irc1.XXXXXX.XXX 004 [urX]−700159 irc1.XXXXXX.XXX IRCd−Hybrid
 iowghraAsORTVSxNCWqBzvdHtGp lvhopsmntikrRcaqOALQbSeKVfMGCuzN
```

Afterwards, the server accepts the bot as a client and sends him RPL_ISUPPORT, RPL_MOTDSTART, RPL_MOTD, RPL_ENDOFMOTD or ERR_NOMOTD. Replies starting with RPL_ contain information for the client, for example RPL_ISUPPORT tells the client which features the server understands and RPL_MOTD indicates the Message Of The Day (MOTD). In contrast to this, ERR_NOMOTD is an error message if no MOTD is available.

On RPL_ENDOFMOTD or ERR_NOMOTD, the bot will try to join his master's channel with the provided password:

```
–> JOIN #botnet channelpassword
–> MODE [urX]−700159 +x
```

After this the bots keep listening for commands sent by the *Bot Herder*. This communication is showed in fig.2.

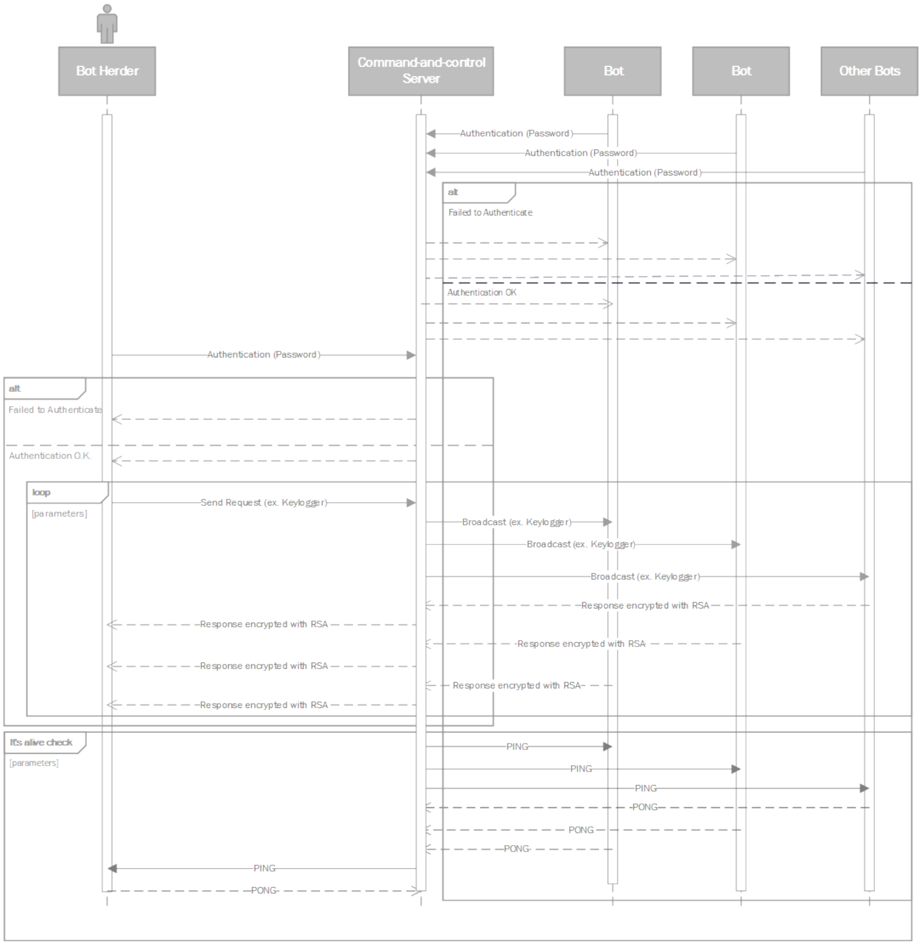
Bot Herder
Command-and-control Server
Bot
Bot
Other Bots
Authentication (Password)
Authentication (Password)
Authentication (Password)
alt
Failed to Authenticate
Authentication OK
Authentication (Password)
alt
Failed to Authenticate
Authentication O.K.
loop
[parameters]
Send Request (ex. Keylogger)
Broadcast (ex. Keylogger)
Broadcast (ex. Keylogger)
Broadcast (ex. Keylogger)
Response encrypted with RSA
Response encrypted with RSA
Response encrypted with RSA
Response encrypted with RSA
Response encrypted with RSA
Response encrypted with RSA
It's alive check
[parameters]
PING
PING
PING
PONG
PONG
PONG
PING
PONG

Figure 2: Basic use sequential diagram.

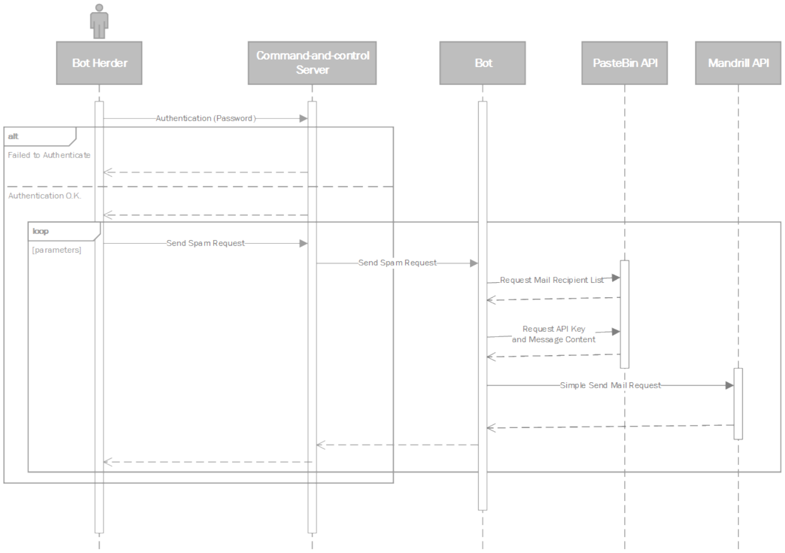

Figure 3: Spam case sequential diagram

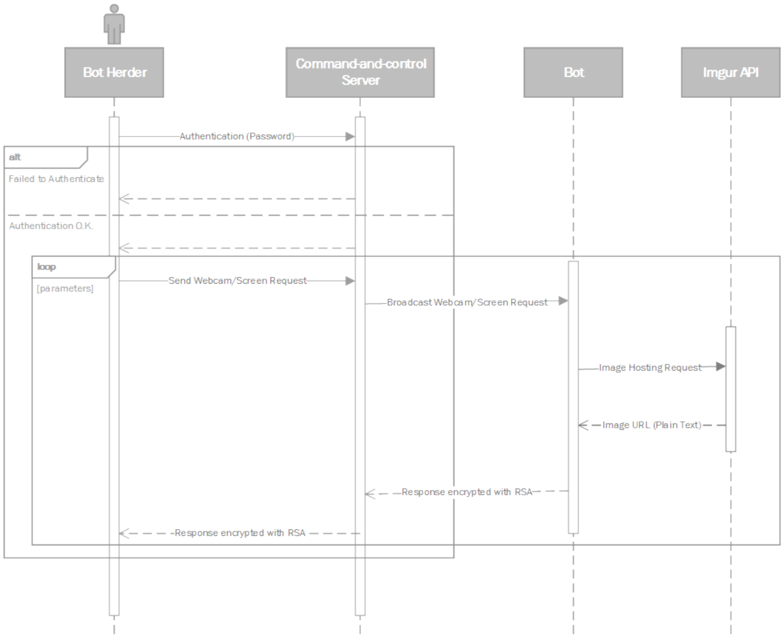
Bot Herder
Command-and-control Server
Bot
Imgur API
Authentication (Password)
alt
Failed to Authenticate
Authentication O.K.
loop
[parameters]
Send Webcam/Screen Request
Broadcast Webcam/Screen Request
Image Hosting Request
Image URL (Plain Text)
Response encrypted with RSA
Response encrypted with RSA

Figure 4: Webcam/Screenshot case sequential diagram.

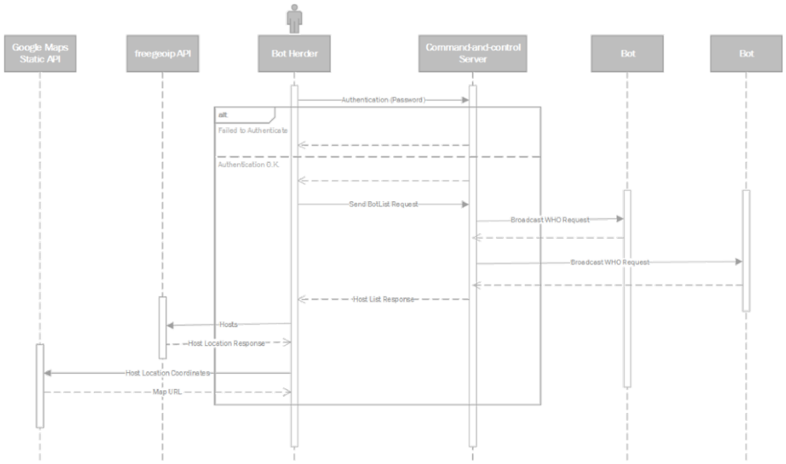

Figure 5: Location request case sequential diagram.

4 Conclusion and Further Work

Botnets are a big issue in computer security being responsible for sending huge amounts of spam and being involved in many cybercrimes including DDoS.

As a result of our research work we present a complete wiki that encompasses the anatomy of a botnet, a list of botnets created along the history and some countermeasures against them making it a useful guide for any person who wants to know more about botnets. As a future work we would like to refer the maintenance of the information presented in it.

We also constructed a very useful tool for anyone who wants to try some features related to botnets. This tool includes all the functionalities that we referred in our initial proposal.

It's also important to refer that all the functionalities, both of wiki and the lab, are working and in the case of the lab there is even support for different operating systems (Linux and Windows).

Also, an important note, all the project and its development was and will be in the future completely open-source because it represents the idea behind our initial proposal for a learning tool for everyone. The wiki will also be openly available for viewing proposes as well as to ease addition of contributions and improve maintenance.

So, in conclusion, we present a complete work that allies the theory and the practice where the wiki complements the lab work.

4.0.3 Further Work

There is, however, some future work that can be done and was not possible due to time limitations.

First off, the wiki could of course be further developed in terms of design quality and content quantity. Also maintaining the information present would also enhance the wiki itself given that botnet's information changes at a relative great speed. Next, and most obvious, for the lab, the addition of some out-of-the-box functionalities beside the existing ones. Following that, we had the intention of exploiting any given vulnerability but since that was not covered it would be a good opportunity for future work to do so. Security of the system, besides using the RSA encryption for communication, is still a priority, so later on, enhancement of the communication channel as well as every other aspect that protects the module against external entities is something to certainly work on.

Given the nature of this tool we've developed, it is only appropriate that we improve the whole setup of the environment to easy the experience. In addition, facilitate the addition of new functionalities and the personalization of every possible setting. And final but also important, removing deprecated dependencies and upgrading the whole system from Python 2.7 to 3, which is crucial to a healthy and maintainable piece of software.

References

[1] "Socks protocol version 5," 1928. [Online]. Available: https://www.ietf.org/rfc/rfc1928.txt

[2] "Akbot," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Akbot

[3] "Bagle (computer worm)," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Bagle_computer_worm

[4] "Conficker botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Conficker

[5] "Cutwail botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Cutwail_botnet

[6] "Denial-of-service attack," December 2015. [Online]. Available: http://en.wikipedia.org/wiki/Denial-of-service_attack/

[7] "Grum botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Grum_botnet

[8] "Kelihos botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Kelihos_botnet

[9] "Mariposa botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Mariposa_botnet

[10] "Nitol botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Nitol_botnet

[11] "Ramnit," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Ramnit

[12] "Rustock botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Rustock_botnet

[13] "Sality botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Sality

[14] "Srizbi botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/Srizbi_botnet

[15] "Tdl4 botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/TDL4_botnet

[16] "Zeroaccess botnet," 2015. [Online]. Available: https://en.wikipedia.org/wiki/ZeroAccess_botnet

[17] P. Bächer, T. Holz, M. Kötter, and G. Wicherski, "Know your enemy: Tracking botnets," 2005. [Online]. Available: https://www.honeynet.org/papers/bots

[18] L. DaileyPaulson, “Hackers strengthen malicious botnets by shrinking them,” 2006. [Online]. Available: http://www.computer.org/csdl/mags/co/2006/04/r4017.pdf

[19] DSLReports.com, “What is a botnet trojan?” April 2009. [Online]. Available: https://www.dslreports.com/faq/14158

[20] D. Esposito, “Browser helper objects: The browser the way you want it,” January 1999. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb250436%28v=vs.85%29.aspx

[21] T. W. Felix Leder and P. Martini, “Proactive botnet countermeasures – an offensive approache,” 2009. [Online]. Available: https://net.cs.uni-bonn.de/fileadmin/user_upload/leder/proactivebotnetcountermeasures.pdf

[22] P. S. Foundation, “Python programming language,” 2015. [Online]. Available: https://www.python.org/

[23] freegeoip.net, “freegeoip,” 2015. [Online]. Available: https://freegeoip.net/

[24] Google, “Adsense product page.” [Online]. Available: https://www.google.com/adsense/

[25] ——, “Google static maps api,” 2015. [Online]. Available: https://developers.google.com/maps/documentation/static-maps/

[26] J. Gruber, “Jekyll,” December 2004. [Online]. Available: https://daringfireball.net/projects/markdown/

[27] imgur, “imgur,” 2015. [Online]. Available: http://api.imgur.com/

[28] A. Inc., “Intrusion detection systems,” February 2014. [Online]. Available: https://www.dmoz.org/Computers/Security/Intrusion_Detection_Systems

[29] K. Lab, “What is a botnet?” April 2013. [Online]. Available: http://blog.kaspersky.com/botnet/1742/

[30] mailchimp, “Mandrill,” 2015. [Online]. Available: https://www.mandrill.com/

[31] Microsoft, “What is a botnet?” 2015. [Online]. Available: https://www.microsoft.com/security/sir/story/default.aspx#!botnetsection

[32] U. of California, “What is seti@home?” 2015. [Online]. Available: http://setiathome.berkeley.edu/

[33] G. Ollmann, “Botnet communication topologies - understanding the intricacies of botnet command-and-control,” 2009. [Online]. Available: https://www.damballa.com/downloads/r_pubs/WP%20Botnet%20 Communications%20Primer%20%282009-06-04%29.pdf

[34] Pastebin, “Pastebin,” 2015. [Online]. Available: http://pastebin.com/

[35] T. Preston-Werner, “Jekyll,” 2015. [Online]. Available: http://jekyllrb.com/

[36] A. S. R.L. Rivest and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” 1978. [Online]. Available: https://people.csail.mit.edu/rivest/Rsapaper.pdf

[37] S. I. R. Room, “Bots & botnet: An overview,” August 2003. [Online]. Available: https://www.sans.org/reading-room/whitepapers/malicious/bots-botnet-overview-1299

[38] I.-H. D. Team, “Ircd-hybrid,” 2015. [Online]. Available: http://www.ircd-hybrid.org/

[39] J. van den Boezem, “Semalt & soundfrost caught spying,” July 2014. [Online]. Available: http://blog.nabble.nl/post/93407897762/semalt-soundfrost-caught-spying

[40] W3C, “The web standards model - html css and javascript,” March 2014. [Online]. Available: https://www.w3.org/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript