A Hands-on Approach on Botnets for Behavior Exploration

João Pedro Dias[1,2], José Pedro Pinto[1,2] and José Magalhães Cruz[2]

[1]*Articial Intelligence and Computer Science Laboratory, Faculdade de Engenharia da Universidade do Porto, Paranhos, Portugal*

[2]*Faculdade de Engenharia da Universidade do Porto, Porto, Portugal*

Keywords: Computer Security, Botnets, Distributed Systems.

Abstract: A botnet consists of a network of computers that run a special software that allows a third-party to remotely control them. This characteristic presents a major issue regarding security in the Internet. Although common malicious software infect the network with almost immediate visible consequences, there are cases where that software acts stealthy without direct visible effects on the host machine. This is the normal case of botnets. However, not always the bot software is created and used for illicit purposes. There is a need for further exploring the concepts behind botnets and network security. For this purpose, this paper presents and discusses an educational tool that consists of an open-source botnet software kit with built-in functionalities. The tool enables anyone with some computer technical knowledge, to experiment and find out how botnets work and can be changed and adapted to a variety of useful applications, such as introducing and exemplifying security and distributed systems' concepts.

1 INTRODUCTION

A botnet, or zombie army, is the name given to any collection of compromised hosts 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, and they can vary from small ones with hundreds or low thousands of infected machines to larger ones with millions of compromised hosts (Cooke et al., 2005).

There are specific cases where bots perform beneficial activities, for example, the participants of SETI@home initiative (Search for Extraterrestrial Intelligence) which are, voluntarily, a part of a large botnet used to analyze radio telescope data in order to track evidence of intelligent extraterrestrial life (Anderson et al., 2002).

Attackers usually install bots by exploiting vulnerabilities in software or by using social engineering tactics to trick users into installing malware. So, users are unaware that their computers are being used for malicious purposes (Abraham and Chengalur-Smith, 2010).

Botnets can be used for various activities but the most traditional and common use is for DDoS (Distributed Denial of Service) attacks. Other uses for botnets are spamming, sniffing traffic, keylogging and even manipulating online polls and games (Bächer et al., 2005).

As of today, there should be a straightforward way to learn about botnets, what they are, how they work and what can be done about them. Even though there are tools, mechanisms and even open-source projects that make it possible to build our own botnets, the complexity and obscurity of these alternatives is preventing, or even making it impossible, to learn their inner functionalities and details (Barford and Yegneswaran, 2007). For that reason, we conceived and implemented a simple and open-source botnet kit with a set of built-in functionalities for anyone interested. It can be set up in a controlled environment and experimented with, finding out how can it be changed and adapted for many purposes.

The paper is organized as follows: Section 2 gives a background on botnets, Section 3 describes our approach on a base implementation of a botnet laboratory and some closing remarks and further work are presented in Section 4.

Dias, J., Pinto, J. and Cruz, J.
A Hands-on Approach on Botnets for Behavior Exploration.
DOI: 10.5220/0006392404630469
In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 463-469
ISBN: 978-989-758-245-5

2 BACKGROUND

The term botnet was widely used to describe a situation where several IRC bots were linked and setting channel modes on other bots and users while keeping IRC (Internet Relay Chat) channels free from unwanted users. This is where the term is originally from, since the first illegal botnets were similar to legal ones, operating over IRC Channels (Cooke et al., 2005). As of today, botnets are widespread in the terms of complexity, functionalities and are adapted to a variety of devices, from mobile phones up to Linux embedded systems (Internet of Things) (Bertino and Islam, 2017).

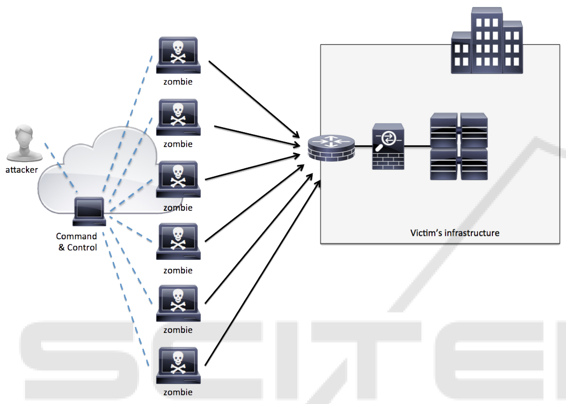

Figure 1: Overview of a DDoS attack using a botnet.

Botnets take advantage of computers whose security defenses have been breached and control conceded to a third party. Each compromised device, known as a "bot", is created when a computer is penetrated by software from a malicious party. The botnet *herder* (or master) is able to control the activities of these compromised computers through communication channels (Room, 2003). Once the attacker is in control, he can send any task to the bots, as, for example, DDoS-ing individuals or organizations, or sending spam (Cooke et al., 2005). A pictorial representation of one botnet attack is given in Figure 1

A bot typically runs as a background process and uses a covert channel to communicate with its C&C (Command & Control) server. Generally, the perpetrator has compromised multiple systems using various mechanisms which themselves can be auto-replicators, in the sense that a machine that was infected and has become a bot, could infect other machines, and so on. In a more basic approach, bots simply make use or steal computing resources of other machines. This process as a result of a system being joined to a botnet is sometimes referred to as "scrumping".

One important note is that some botnets have self-defense mechanisms against things that try to disrupt their normal operation. 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 relying on peer-to-peer relationships (Ollmann, 2009). 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 (DSLReports.com, 2009).

The existent literature about botnets is very disperse in the way that different authors characterize, detail and categorize botnets in a variety of different ways. Still, based on the existent literature, the following subsections try to make a summary of a typical botnet anatomy, malicious uses and detection techniques.

2.1 Botnet Anatomy

Botnets have a typical lifecycle that consists mainly of five phases, from creation to maintenance (Feily et al., 2009):

- Initial infection phase: the attacker scans a target subnet for known vulnerabilities, infecting machines through different exploitation methods.
- Injection phase: fetches the actual bot binary from a specific location, via File Transfer Protocol(FTP), HyperText Transfer Protocol(HTTP) or Peer-to-Peer Protocol (P2P), and installs itself on the victim machine. From this point the victim computer becomes a "Zombie".
- Connection phase: the bot program establishes a C&C channel and connects the zombie to the command and control server, becoming part of the botnet.
- Malicious command and control phase: the botmaster uses the C&C channel to disseminate commands to the nodes of the botnet, and the bot programs execute this commands, in order to conduct various illicit activities.
- Update and maintenance phase: the botmaster takes care of maintaining the bots live and updated. These updates are made, in the majority of situations, to evade new detection techniques or add new functionalities.

Botnets can be classified into three different categories based on the C&C models (Tanwar and Goar, 2014; Li et al., 2009a):

- Centralized Model: The master takes advantage of Internet Relay Chat (IRC) or the HTTP protocol as the C&C channel to communicate and control the bots. This approach is similar to the first appearances of botnets. The main disadvantage of this model is that this server is the single point of failure of the botnet, making it easier to be detected and destroyed (Li et al., 2009b; Seenivasan and Shanthi, 2014). Examples of this model are the AgoBot, SDBot, and Zotob (Li et al., 2009a).

- Peer-To-Peer Model (Decentralized): The botnet consists of a decentralized peer-to-peer network of nodes. Here, the connection does not depend on any particular server, making this type of botnet highly robust because the network is resilient (Dittrich and Dietrich, 2008; Seenivasan and Shanthi, 2014). The master usually shares specific commands to each and every bot in the network and the bots frequently communicate between them and send keep-alive messages (Dittrich and Dietrich, 2008). Grizzard et al. make a case study over the Trojan.Peacomm P2P botnet (Grizzard et al., 2007).
- Unstructured Model (Random Model): The bot program will not actively contact other bots or its master. Instead, it will just listen to incoming connections from the master. The master, when in need, scans the network and passes along an encrypted message. When received by a bot, the message will be decrypted and its commands executed (Li et al., 2009a). The main advantage of this approach is the easy implementation, however, the model is only theoretical and no real use cases have been discovered (Rataj, 2014).

2.2 Malicious Uses of Botnets

A botnet is nothing more than a tool and there are many different motives for using it. The most common uses are criminally motivated or for destructive purposes. Some of the possible uses of botnets can be categorized as listed below.

- Distributed Denial-of-Service Attacks: Often botnets are used for DDoS attacks. Such, is an attack on a computer system or network that causes a loss of service to users, by consuming the bandwidth of the victim network or overloading the computational resources of the victim's system (Mirkovic and Reiher, 2004; Sharma et al., 2016).
- Spam (bulk email): Some bots offer the possibility to open a SOCKS proxy on a compromised machine in order to be used for spamming. With the help of a botnet and thousands of bots, an attacker is able to send massive amounts of bulk emails. Some bots also implement a special function to simultaneously harvest email-addresses increasing the amount and impact of this functionality (Leech et al., 1928; Bächer et al., 2005).
- Sniffing Traffic: Bots can also use a packet sniffer to watch for interesting clear-text data passing by a compromised machine, like usernames and passwords. If a machine is compromised more than once and is now a member of more than one botnet, the packet sniffing allows to it to gather the key information of the others botnets. Thus it is possible to sniff other botnets using bots (Bächer et al., 2005).
- Keylogging: Bots can implement keylogger functionalities in order retrieve sensitive information from the zombie machines, that can lead to massive identity theft (Bächer et al., 2005). One example of a botnet that implements this functionality is SpyBot (Barford and Yegneswaran, 2007).
- Spreading new malware: Botnets typical implement auto-replication mechanisms, used to spread new bots, but can also have functionalities to spread other malware. 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 (Bächer et al., 2005).
- Google AdSense[1] abuse: An attacker can leverage his botnet to click on these advertisements in an automated fashion and thus artificially increment the click counter (Bächer et al., 2005; Cole et al., 2007).
- Manipulating online polls and games: Online polls and games are getting more and more attention and its rather easy to manipulate them with botnets. Since every bot has a distinct IP (Internet Protocol) address, every vote will have the same credibility as a vote cast by a real person (Bächer et al., 2005; Cole et al., 2007).
- Bitcoin mining: Due to the appearing of the blockchain-based digital currencies like Bitcoin, botnets combined computing power is being used for bitcoin mining activities. One example is ZeroAccess botnet that already implements this feature (Wyke, 2012).

2.3 Botnet Detection

Botnet detection deals with the identification of bots in a machine or network so that some sort of patch can

[1]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.

be applied. Despite the long existence of malicious botnets, just a few formal studies have analyzed this problem. One of the pioneering studies of the botnet thematic was the Honeynet project (Bächer et al., 2005).

Due to the increasing impact of the botnets (and related attacks) this has become a major research topic in recent years. As suggested by Feily (Feily et al., 2009) there are mainly two ways of botnet detection and tracking. Setting up honeynets, that is specially important for the understanding of the characteristics of the botnet, and passive traffic monitoring and analysis, which is the most used approach for detecting bot infections, both approaches are usually integrated with Intrusion Detection System (IDS) (Gu et al., 2008b).

As for passive traffic monitoring and analysis there are some relevant techniques being used (Feily et al., 2009):

- Signature-based Detection: Uses the signatures[2] of current botnets for their detection. The main problem with this method is that it's only able to detect well known botnets. Thus, unknown botnets can't be detected by this method (Feily et al., 2009; Bächer et al., 2005).
- Anomaly-based Detection: Botnet detection is performed by considering several different network traffic anomalies, including high network latency, high traffic volume, traffic on unusual ports, and other unusual system behavior that could indicate the presence of malicious bots in the network (Feily et al., 2009; Bächer et al., 2005; Gu et al., 2008b).
- DNS-based Detection: This technique bases itself on particular DNS (Domain Name System) information generated by a botnet, being similar to anomaly detection techniques (Feily et al., 2009; Choi et al., 2007).
- Mining-based Detection: Because identification of botnet traffic, is a difficult task, due to the similarities with normal traffic, this approach uses several data mining and machine learning techniques to detect botnet traffic. One example is the BotMiner which uses clustering techniques in order to detect botnet traffic (Gu et al., 2008a).

3 IMPLEMENTATION

With the goal of producing a real hands-on tool for testing and developing purposes we created a base core for a botnet lab which comprehends, a communication protocol, a command and control center, encryption capabilities and a basic bot implementation with some built-in features.

The botnet built using this laboratory matches the general architecture for any botnet based on a C&C architecture. Our actor is the (Bot) Herder or (Bot) Master, that operates using a special IRC client (which is part of this laboratory) and connects to a IRC-Server (IRCD-Hybrid) where all the bots also connect to.

Whenever the Master sends a message to the IRC Server, it will broadcast it to all the connected bots. By connected bots, we mean bots that are already hosted in a machine (which for the purpose of this tool can be a local, virtual or foreign machine) and have started and connected themselves to the server has then acknowledged them. Having received a message, the bot will execute the requested action.

3.1 Architecture

The main network components that interact with the botnet are showed in fig.2. This figure also illustrates a practical use of the lab, having a relation of one master to many bots, which can be deployed in several machines.

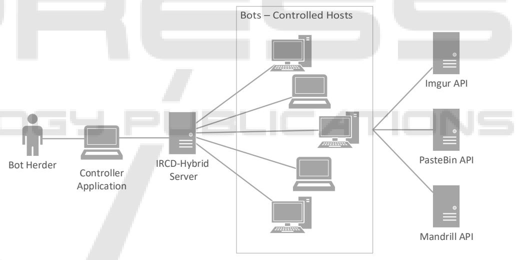

Figure 2: Network diagram.

This project neither exploits software vulnerabilities nor mention ways of using them to place bots in remote machines escaping detection from their owners. With this in mind, when using this lab a bot should be placed on a machine whose owner willingly and knowingly accepts it. After it has been successfully deployed, its typical communication process with the master, through an IRC server, goes like the following:

```
<- :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,
```

[2]Signature is an algorithm or hash that uniquely identifies a specific virus.

```
    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
<- :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 replies with 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. 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 the bot establishes a connection with the server, it will remain there listening to commands from the master and replying to them accordingly. This initial connection and command broadcasting, by the master, can be visualized in the sequence diagram in fig. 3.

3.2 Technologies

Concerning the IRC server we used a *IRCd-hybrid* server - "*A lightweight, high-performance internet relay chat daemon.*" (Team, 2015).

As our development tool we used Python 2.7 (Foundation, 2015) with the help of some external libraries. With it, we built the bots, both master and slaves, along with the whole lab infrastructure.

Having a IRC server set up, we had the need for an IRC Client to manually communicate with the bots as well as to maintain a visual representation of the tool. Such client was built using Python 2.7 (Foundation, 2015) as well.

As mentioned before, it was not feasible to build all the tools from scratch for most of our needs during the development of the lab. We had available to us many external, public available, tools and services, namely API's, with proved stability and reliability. Of those, we used the following:

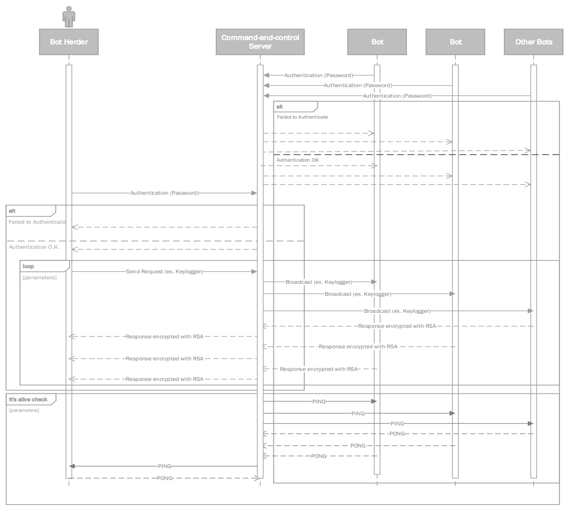
Bot Herder
Command-and-control Server
Bot
Bot
Other Bots
Authentication (Password)
Authentication (Password)
Authentication (Password)
alt
Failed to Authenticate
Authentication OK
Authentication (Password)
alt
Failed to Authenticate
Authentication O.K.
loop
[parameters]
Send Request (ex. Keylogger)
Broadcast (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 3: Basic use sequential diagram.

- **Freegeoip**[3]: provides a public REST 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.
- **Google Static Map API**: creates a map based on URL parameters sent through a standard HTTPS request.
- **Mandrill**[4]: is a reliable, scalable, and secure API for delivering and manage transactional emails.
- **Pastebin**[5]: a website where we can store text for a certain period of time.
- **Imgur API**[6]: an image storage website.

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

3.3 Details

Amongst all the base functionalities provided for this lab, there are some which we'll highlight given their more complex and unusual behavior. Specifically the features of spam and screenshot/webcam request.

[3]https://freegeoip.net/
[4]http://mandrill.com/
[5]http://pastebin.com/
[6]https://api.imgur.com/

First, the Spam feature. To avoid the trouble of setting up a SMTP (Simple Mail Transfer Protocol) server on every bot we used the *Mandrill* API to send e-mails. This might seem unusual, because we are centralizing all the traffic on one e-mail sender API with low free quotas and so risking the account being blocked. However, by sending an API Key in the request sent to a bot, whenever we request the spam feature, if an API Key is blocked a different API Key is sent in future request. Additionally, the *PasteBin* service is used, taking advantage of its anonymous and hidden file ability to host relevant data, like the e-mail sending list, the API Key and message that the master wants to send as email spam. With this service, all that is left to send is the URL that compiles this information to the bot. Figure 4 show how this particular feature would function in a practical use.

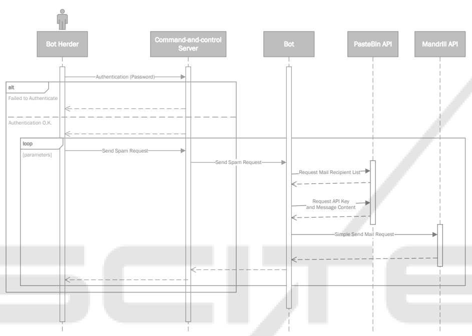

Figure 4: Spam case sequential diagram.

In the functionality of Screenshot/Webcam, the bots use the *Imgur* API in order to store images, either an image file of a screenshot or a photo from the webcam. Following the request sent by the botmaster to a given bot, the bot will then store an image file online using *Imgur* and report back to the master just the URL of the hosted picture as we can see on fig. 5.

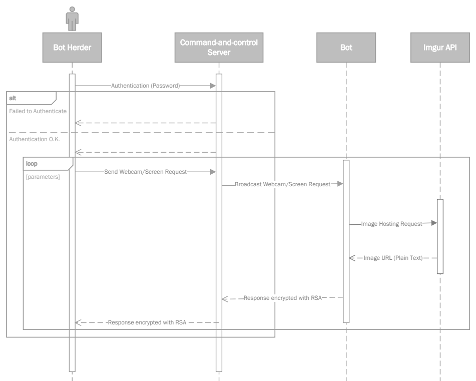
Bot Herder
Command-and-control Server
Bot
Imgur API
Authentication (Password)
alt
Failed to Authenticate
Authentication O.K.
loop
[parameters]
Send Webcam/Screen Request
Broadcast Webcam/Screen Request
Image Hosting Request
Image URL (Plain Text)
Response encrypted with RSA
Response encrypted with RSA

Figure 5: Webcam/Screenshot case sequential diagram.

Additionally, in order to better check the bots that are working and where they are located at a certain point, we use the *Freegeoip*'s public REST (Representational State Transfer) API to search geolocations of IP addresses and Google's *Static Map* API for retrieving and displaying the relative world position of the controlled hosts (fig.6). The result of both these two tools put together is a map of the world with pinpoints on all the bots active in that instance of the botnet lab.

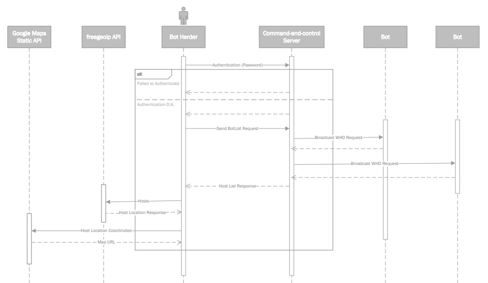

Figure 6: Location request case sequential diagram.

Also, we use RSA encryption (R.L. Rivest and Adleman, 1978) so that the master is the only one capable of decrypting the messages sent by the bots given it is the private key owner. The bots encrypt the messages using the public key defined by the master.

4 CONCLUSION AND FURTHER WORK

As of today, botnets are a big issue in computer security. Botnets are responsible for many malicious activities, including spam and DDoS, targeting every type of device, from normal computers up to embedded Linux devices.

After proceeding to an evaluation of the research being done in the area of botnets, we found the lack of a simple way to build and modify a botnet. In order to overcome this lack of research, we implemented a simple core tool for anyone who wants to setup a botnet laboratory, following the base build principles of common botnets. The resulting work is *open-source* on GitHub (https://github.com/jpdias/botnet-lab).

There is, however, some future work that can be done in order for the tool to reach its full potential.

First off, the addition of some out-of-the-box functionalities (such as keylogging and DDoS) for the base tool besides the existing ones. Following that, we could implement some resilience and auto-replication techniques, resorting to, for example, the exploit of any known software vulnerability.

Also, the security functionalities of the botnet could be improved, like, for example, the implementation of self-defense mechanisms against third-parties.

Given the nature of this tool it would be interesting to improve the whole setup of the environment to ease the user (developer or researcher) experience. This can be accomplished with, for example, the favor of addition of new functionalities in a framework-like way and the possibility of personalize every setting in the core tool.

REFERENCES

Abraham, S. and Chengalur-Smith, I. (2010). An overview of social engineering malware: Trends, tactics, and implications. *Technology in Society*, 32(3):183 – 196.

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. (2002). Seti@home: An experiment in public-resource computing. *Commun. ACM*, 45(11):56–61.

Barford, P. and Yegneswaran, V. (2007). *An Inside Look at Botnets*, pages 171–191. Springer US, Boston, MA.

Bächer, P., Holz, T., Kötter, M., and Wicherski, G. (2005). Know your enemy: Tracking botnets.

Bertino, E. and Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2):76–79.

Choi, H., Lee, H., Lee, H., and Kim, H. (2007). Botnet detection by monitoring group activities in dns traffic. In *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*, pages 715–720.

Cole, A., Mellor, M., and Noyes, D. (2007). Botnets: The rise of the machines. In *Proceedings on the 6th Annual Security Conference*, pages 1–14.

Cooke, E., Jahanian, F., and McPherson, D. (2005). The zombie roundup: Understanding, detecting, and disrupting botnets. *SRUTI*, 5:6–6.

Dittrich, D. and Dietrich, S. (2008). P2p as botnet command and control: A deeper insight. In *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, pages 41–48.

DSLReports.com (2009). What is a botnet trojan?

Feily, M., Shahrestani, A., and Ramadass, S. (2009). A survey of botnet and botnet detection. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pages 268–273.

Foundation, P. S. (2015). Python programming language.

Grizzard, J. B., Sharma, V., Nunnery, C., Kang, B. B., and Dagon, D. (2007). Peer-to-peer botnets: Overview and case study. *HotBots*, 7:1–1.

Gu, G., Perdisci, R., Zhang, J., Lee, W., et al. (2008a). Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security Symposium*, volume 5, pages 139–154.

Gu, G., Zhang, J., and Lee, W. (2008b). Botsniffer: Detecting botnet command and control channels in network traffic.

Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L. (1928). Socks protocol version 5.

Li, C., Jiang, W., and Zou, X. (2009a). Botnet: Survey and case study. In *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pages 1184–1187.

Li, C., Jiang, W., and Zou, X. (2009b). Botnet: Survey and case study. In *innovative computing, information and control (icicic), 2009 fourth international conference on*, pages 1184–1187. IEEE.

Mirkovic, J. and Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53.

Ollmann, G. (2009). Botnet communication topologies - understanding the intricacies of botnet command-and-control.

Rataj, M. (2014). *Simulation of Botnet C&C Channels*. PhD thesis, Ph. D Dissertation, Faculty of Electrical Engineering-Department of Computer Science and Engineering, Czech Technical University in Prague.

R.L. Rivest, A. S. and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems.

Room, S. I. R. (2003). Bots & botnet: An overview.

Seenivasan, D. and Shanthi, K. (2014). Categories of botnet: a survey. *Int. J. Comput. Control Quantum Inf. Eng*, 8(9):1589–1592.

Sharma, S., Garg, S., Karodiya, A., and Gupta, H. (2016). Distributed denial of service attack. 4.

Tanwar, G. S. and Goar, V. (2014). Tools, techniques & analysis of botnet. In *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*, ICTCS '14, pages 92:1–92:5, New York, NY, USA. ACM.

Team, I.-H. D. (2015). Ircd-hybrid.

Wyke, J. (2012). The zeroaccess botnet: Mining and fraud for massive financial gain. *Sophos Technical Paper*.