A Parallel Cyber Universe: Botnet Implementations over TOR-Like Networks

Hüseyin Yağcı, Çağatay Yücel, Ahmet Koltuksuz
Department of Computer Engineering, Yaşar University, Izmir, Turkey
huseyin.yagci@stu.yasar.edu.tr
cagatay.yucel@yasar.edu.tr
ahmet.koltuksuz@yasar.edu.tr

Abstract: The first bot implemented in the history of computers was the Eggdrop (Fisher J, 1998). The first instance of this kind was benign; it was an automated management tool for Internet Relay Chat (IRC) rooms. It wasn't much later when Internet users experienced the first botnet attack. The GTbot family was the first known malicious automated attack network on IRCs (Bächer et al. 2009), and new era for bots had begun.

Botnets can be practically defined as a network of infected smart devices. As a result of the infiltration attacks made on a victim's computer with different malwares and zero-day attacks, the control of the computer is confiscated without the victim being aware of it. Confiscated machines are connected to Command and Control (C&C) centers. In the case of a single infection, this attack is nothing more than a data theft or privilege escalation. However, when the number of the infected devices scales up to thousands, the attack becomes a mass destruction weapon on global companies' networks.

Amazon, Spotify, Twitter, and many more companies were affected by DDoS attacks by the Mirai botnet in October 2016 (Allison Nixon, John Costello, 2016). The Mirai botnet was conducted by a malicious network utilizing the IoT devices. Moreover, an even worse fact was the announcement of more, similar botnet attacks after that October (Paganini 2016, Anon 2016).

Today, honeypot-based, signature-based, and host-based defenses, as well as active and passive monitoring techniques, are being developed against botnets (Silva et al. 2013). Botnets are fighting back for their existence by using binary obfuscation, fast-flux networks, domain generation algorithm (DGA) techniques, and polymorphism, while ciphering, IP spoofing, multi-hopping, and email spoofing (Rodríguez-Gómez et al. 2013, Wang et al. 2016). Another important technique for botnets is to utilize The Onion Routing (TOR) networks where the communication scheme of the bot network is anonymized in the layers of the TOR scheme. The name of this network comes from a reference to the multi-layered structure of an onion.

This research presents a novel implementation of a hidden botnet mechanism over like networks to The Onion Routing (TOR) ones. The focus is on creating parallel cyber universes with TOR-like structures and hiding the existence of the botnets in the blind range of the Internet. The design of such a network and the attack vector is explained in detail for the first time in the literature.

Keywords: Cyber warfare, Cyber security, Botnet, TOR, Network, Anonymity

1. Introduction

A bot is software that is designed for running automated processes. The name "bots" originated from the Slavic language, wherein the word *"robota"* is the name for a forced laborer. Today, bots are used in different fields such as editing articles, checking human behaviors on the internet forums, stealing information, conducting DDoS attacks and mining bitcoins on victim systems.

Networks created by joining bots around the globe are known as botnets. Botnets work under the *union is strength* principle. Almost every machine that has a Network Interface Controller (NIC) device can be included in a botnet. In most recent cases, it has been seen that Internet of Things (IoT) devices are included in botnets. Adversary botnet developers strengthen their hands with that of the IoT devices and the effectiveness of botnets are highly increased. The adversaries can now offer advertising services from botnets such as cryptocurrency mining services, as well as provide counterfeit DDoS services on the Internet's black market.

The TOR project, which was developed in the Naval Laboratories of the United States of America, almost guaranteed the anonymity of its clients. It became publicly available at some time in Internet history, unfortunately the details about the TOR project being free has never been available (Dingledine et al. 2004). The TOR project consists of relay nodes that can provide an environment for creating an encrypted circuit in the anonymous structure of its network. This means the destination and source do not know any descriptive information about each other.

Botnets built on the anonymity of TOR networks have become quite popular. Even in the cases when the bots of a network are compromised, TOR botnets are still resilient to such attacks, as the anonymity provided by the network aids it to stay under the radar if the encryption relay circuits hold.

This research presents a novel methodology to create several networks similar to TOR ones on the TCP/IP stack by configuring the communication between such networks. The methodology provided in this paper reveals information about how to create parallel cyber-anonymous TOR networks and how to control their botnets over every one of them separately, thus allowing several botnets to reside on the same structure. Since every one of these networks utilizes a relay circuit to survive, the idea has become analogous to the theory of parallel universes. Therefore, the authors named it "Parallel Cyber Universes".

This paper was organized in the following manner: Section 2 presents a brief explanation about the offensive and defense mechanisms of botnets, Section 3 describes the TOR network structure. Full details of the completed work are elucidated in Section 4. The related work of the implementations of botnets over TOR networks is discussed in Section 5. The pearls and pitfalls are mentioned in Section 6.

2. Botnets

After it became clear to the adversaries that overtaking a home user computer is easier than overtaking a fully equipped server, adversaries have given their attention to hacking many home computers. When the number of computers included in a botnet reaches into the thousands, the attack capacity is beyond the capacity of a single supercomputer. The analyzed attack mechanisms show that most botnets have an implementable life-cycle model (Feily et al. 2009, Rodríguez-Gómez et al. 2013). In the model, the stages of the life-cycle are motivation, design, recruitment, interaction and marketing.

The design of a botnet is directly affected from the motivation of the developers. The design includes the design of C&C server communications and the network structure of botnets, which can be centralized, decentralized or hybrid. The recruitment of bots is affected from design similarly, and also can use malicious software such as worms, zero-day exploiters, brute force attacks and social engineering techniques. This is the phase where the number of bots are increased exponentially.

The interaction stage is the stage on which the bots and their masters are decided to communicate. Communication protocols plus network interactions between the bots and the bot-master are defined at this phase. Marketing is the final stage of the botnet life-cycle model. It is the time where the bot network reaches a substantial surface and harvests the goods from the network. In this stage, the developed botnet is marketed as a service or do-it-yourself (DIY) type of source code. Renting a service from a botnet works like renting a domain name from a provider, but only the rented service can do DDoS attacks or provide anonymity (Symantec 2010). However, interestingly, DIY kits are available as an open source code. The aim of these is to hide the actual developer by revealing the source code and making it public. This is controversially opposed to botnets that provide anonymity by hiding the information, since the source code provides anonymity by becoming public.

The research on the detection of botnets gained momentum in the recent years. Researches can be grouped in multiple categories: Honeynets, signature-based and anomaly-based detection systems.

Honeynet is a part of the honeypot family which originated from intelligent human techniques. This technique refers to a strategy where an attractive female or male agent is depraving vulnerable human victims and exploiting their sexual relationship to force them to cooperate with them. This technique is used in cyber space with the same terminology. A decoy-based information gathering system is prepared as a vulnerable member of a network. These vulnerabilities lure the attackers into traps. While the attackers exploit a victim, the honeypot system is analyzing and logging all the behaviors of the adversaries. Honeynets are used as information gathering mechanisms on botnets all over the cyber domain. At the same time, these kind of mechanisms have some problems which were inherited from their core theory (Anon 1999). The discoverability of the botnet span and zero-day vulnerabilities are two limitations of the detection systems with Honeynets.

Intrusion detection systems (IDS) are another important functional type of tool against botnets. IDS mechanisms are based on two main defense techniques: signature-based detection systems and anomaly-based detection systems (Y. Kugisaki, Y. Kasahara, Y. Hori 2007). Signature-based detection systems are basically comparison tools for the defined peculiarities of known bots. These properties are determined by the monitored botnet traffic, the communication pattern, utilized ports, protocols and services. Although signature-based detection systems can rapidly detect known, analyzed bots, they can never be successful for even slightly different bots and vulnerabilities. Theoretically the technique cannot detect a bot's activity before an attack occurs.

Anomaly-based detection systems are a favorite topic of study in botnet detection (Binkley & Singh 2006). The technique is based on capturing suspicious activities within the network and the host. Any usage of pre-defined ports, services and unusual network traffic latency can be counted as a suspicious activity. The main challenge about anomaly detection is defining what is normal and an anomaly. The method can detect new kind of botnets with a good configuration, and at that point, anomaly-based detection systems are separated in two main categories, host-based and network-based approaches. The anomaly-based detection technique involves a learning process where the anomaly is statistically defined. This process is also the weakness of the technique, as the open sourced botnets are versioned on the dark market almost immediately, then the learning process becomes a nightmare. For these matters, security specialists use hybrid approaches of all mechanisms.

The comprehensive detection and defense mechanisms are highly developed, mainly because there are even more developed deception and hiding techniques on the market. These hiding mechanisms can be categorized again as host-based and network-based. Common host-based techniques are listed below:

- Binary obfuscation is the effective anti-reverse engineering technique applied on the source codes of bots.
- Versioning the bots' source codes within modules and switching between them when IDS or anti-virus programs are triggered is known as polymorphism.
- IP and Email spoofing mechanisms are also used for mystification against detection systems.

The scope of this paper is focused mainly on the network-based mechanisms. Contemporarily known techniques, including multi-hopping, ciphering, domain name generation (DGA) and fast-flux networks, can be counted as network-based hiding mechanisms.

- Multi-hopping is a networking mechanism for making it difficult to track the source of an attack or C&C server. When based on the C&C server or the botnet's operator, multiple proxies are used before reaching the bot or any interface of botnet hierarchy.
- Ciphering relies on the power of cryptography. The communication channel between bots and C&C are encrypted, and even if the botnet has been compromised, the encrypted packages need to be cracked.
- Domain name generation (DGA) is yet another technique that improves the covertness of the botnet. In the mechanism, randomly generated domain names are used as meeting points. The number of generated domains can reach up to tens of thousands of domains. This includes potential rendezvous points which can disable blacklisting mechanisms.
- Fast-flux networks attach hundreds to thousands of IP addresses on a fully qualified domain name and switch them in a pre-defined frequency. Such a website that switches on a set of frequencies from the fast-flux network is much more than a domain. The domain becomes a nice hiding corner for the C&C and the frequency changes protect the actual identity of the user.

Taking the advantage of anonymity provided by TOR networks is a new trend among botnet developers. The first known implementation of such a botnet was in 2010. A modified version of the Zeus (Falliere & Chien, 2009) botnet with extra capabilities for anonymity had been published through Reddit (Casenove & Miraglia, 2014). The anonymity of TOR networks and the dangers of a botnet has been combined since then.

3. **TOR Networks**

TOR networks originally showed up as a third-generation onion routing project in U.S. Navy, as a purpose of protecting governmental communications. Then, the TOR networks were publicized to worldwide usage. An increasing number of volunteers and participants increased its anonymity and efficiency. Anonymity based on cryptographic protocols was implemented in a multi-layered structure. Utilizing this type of structure, TOR layouts create a confidential communication medium between the senders and receivers.

TOR networks run as computer processes. The user-level processes can be configured as an onion proxy (OP) or onion router (OR) on every installed node. OP nodes which are defined as clients can be entry or exit nodes and handle the connections of the user. OR nodes are used to manage and establish TOR networks in the form of a relay, a hidden service or a directory server. OR/OP machines communicate from specified ports with a specified data structure. TOR networks originally used SOCKS ports and TLS connections while establishing communication mediums (IETF® 1996, IETF® 2008). The communication between the nodes of each network was made with size-fixed cells and organized circuits. TOR networks' traffic stands on the specified cells defined by the configuration. The cells are examined in two subcategories: control cells and relay cells. The header of the circuit identifier specifies whether it is a control or a relay cell. Control cells manage the communications between OP/OR and OR/OR, as well as relay cells which control end-to-end stream data with specified commands.

Creating a circuit with source and destination cells is the beginning of the process. The source then starts to construct a circuit with a symmetric key and at every step (hop) of the way, it adds another symmetric key and encryption. Once the circuit has reached its destination, a message containing half of the Diffie-Hellman handshake, which was carried by the circuit, is delivered. Then a connection is established with responses from the destination.

TOR network structures use Rendezvous Points (RP) to connect the clients and services at a certain level of anonymity. Web services on a clear network turn into a hidden service (HS) (as well as an OR) in the TOR network. The structure of communication between a HS and OP is created by specifying introduction points and making advertisements on the nodes to advertise its public key. The HS then builds a circuit on each of the advertisement nodes and waits for requests. When the service recipient wants to connect to the HS, it chooses a RP to connect with and builds a circuit to that RP. When the RP is ready, a stream is opened for that advertisement point, and another RP is created by the user. Once the two RPs are linked and the stream information is transmitted, the HS builds a circuit to the new RP via the client and creates a stream circuit. Then finally, nodes start to communicate anonymously.

Another important point is that there is a need for management to increase the reliability of the system and sustainability of the relays. Therefore, on the second-generation TOR networks, Directory Servers (DS) are generated. A group of reliable, well-known ORs are used for tracking changes in the network topology, including keys and exit policies. Directory Servers work as HTTP servers. Clients can easily get updated about the list of the services. Other ORs can upload their states into that HTTP server as well, ORs in the network send updates of their signed information frequently to their directory servers. The directory servers process this information and store it as a list of the network nodes and their detailed states.

A client, becoming an internal node as an OR, pulls the list from the chosen directory server and gets informed about the services. Before the creation of second generation TOR networks, this list of details is flooded into the network by the ORs. Putting in the directory servers not only increased the efficiency, it has also increased the secrecy of the TOR networks.

TOR networks' provided anonymity is not perfect. There are lots of exploitable parts in the structure. Some of these parts are caused by misconfigured onion routers and onion proxies. The other weak parts are due to the nature of the security. An adversary who owns the directory server can track down its private information and communication paths. From that point of view, if the adversary owns enough of the directory server, it is possible to take over the whole secrecy of the network activities and deplete anonymity. Other attack types are listed and discussed in detail in the source paper about the core TOR design (Dingledine et al. 2004).

4. **The Work Done**

TOR networks not only provide anonymity to internet users, they also are an open source tool. There are several implementations and customizations of TOR networks for several aims. One of them is JonDonym which uses the same cryptographic processing but differs in relaying data (JonDos 2011). A second one is Whonix which was designed for a TOR network. It works as a shell operating system for additional anonymization, and it avoids DNS leaks (Patrick Schleizer, Fortasse, HulaHoop, Troubadoour, Jason Ayala, IronSoldier 2012).

The research presented in this paper is a methodology to configure the TOR network structure to create parallel cyber networks with the same structure. Also, an abstract schema of the accomplished work is shown in Figure 1.

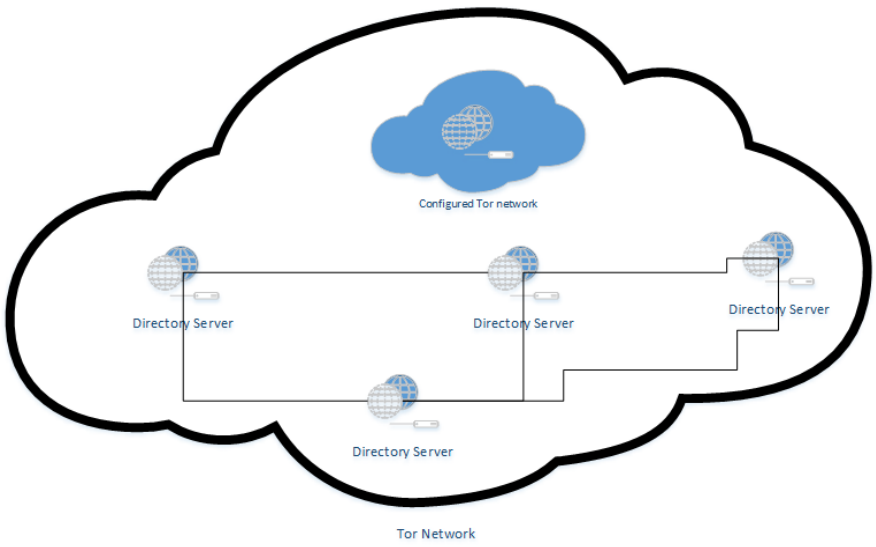

Figure 1: Abstract design of research

The main motivator behind this research is separating directory servers from the original TOR networks. To hide these special DSs from others, the default configuration file of the TOR structure must be changed. In a regular TOR network, there are different types of nodes which are mentioned in the below sections. A node can be a relay, an exit node, or if found trustworthy (depending on the requirements), it can be a director server. By the properties of a TOR structure, completely separate and individualized networks can be implemented. This process requires remaking all customizations in the configured TOR network. An example configuration file is shown below:

```
AssumeReachable 1
PathsNeededToBuildCircuits 0.25
TestingDirAuthVoteExit *
TestingDirAuthVoteHSDir *
V3AuthNIntervalsValid 2
RunAsDaemon 1
ConnLimit 60
ShutdownWaitLength 0
ProtocolWarnings 1
SafeLogging 0
DisableDebuggerAttachment 0
```

Default configuration commands

Specified directory server commands

```
DirAuthority test000a orport=5000 no-v2 hs v3ident=
3028F321F8F02FC07EE85667F4726E8FCCF37ACC 192.168.1.100:7000
F23ADC68E07AD5B54FB6908A067E3ACDE967E041
```

```
ControlPort 8024
SocksPort 9024
CookieAuthentication 1
```

Node specific commands

2.1

```
ControlPort 8000
SocksPort 0
OrPort 5000
Address 192.168.1.100
DirPort 7000

AuthoritativeDirectory 1
V3AuthoritativeDirectory 1
```

Authorty Server specific commands

2.4

```
ControlPort 8027
SocksPort 0
Address 192.168.1.100

HiddenServiceDir /..
HiddenServicePort 6667 127.0.0.1:6667
```

Hidden service specific commands

2.3

```
ControlPort 8004
OrPort 5004
Address 192.168.1.100
DirPort 7004
```

Relay specific commands

2.2

Figure 2.1, 2.2, 2.3, 2.4: A TOR network's configuration files including a client (entry node), relay, hidden service, and an authority server

The following stages are necessary for the creation of cyber-TOR domains:

- Stage I - A botnet structure is coded and implemented based on a botnet life-cycle.
- Stage II - A customized TOR network structure is implemented on the test nodes.
- Stage III involves the implementation of a botnet over a customized TOR network.
- In the last stage, Stage 4, an experimental implementation is established on an internal network, and implemented botnet activities and communications above that kind of network structure is verified.

After an implementation of such customized TOR networks, to control the botnet an Internet Relay Chat (IRC) protocol is appended to the network. The implementation schema is shown in Figure 2.

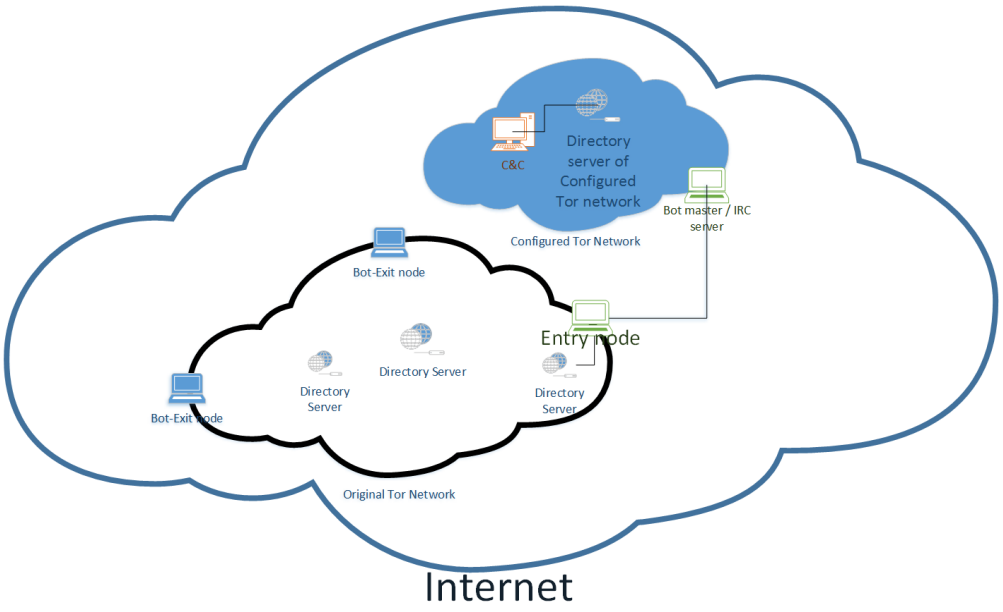

Figure 2: The implemented work's conceptual schema

The customizations mentioned in Figures 2.1, 2.2, 2.3, and 2.4 were applied successfully in an isolated lab. Fifteen machines consisting of real and virtual devices have been configured to implement a plan of three directory servers and ten relay nodes (two real ones and two virtual ones), for test clients in a Linux environment. The anonymity provided by the TOR network structure has been applied as well. Tests of the anonymity of the configured TOR network was done with network monitoring programs and network forensic techniques (Meghanathan et al. 2010).

5. Related works

The related work on botnets using a TOR approach will be presented in this section. Implemented live botnets have been using the power of anonymity provided by the TOR network and it seems to be that this approach will continue to exist in the future.

In Sanatinia & Noubir (2015), analyses, mitigations, and detections of botnets' activities over TOR networks are discussed. In this research, when a hidden botnet was detected over a TOR network, counter bots were created to attack the hidden structure to track down malicious activities. However, in our approach, overlapping TOR networks had completely separate configurations. In that circumstance, it becomes impossible to infiltrate the underlying structure. One needs to collect all anonymized traffic information in whole, global TOR networks, and it is practically impossible to detect hidden botnets over TOR networks to begin with. Another study focused on HS addresses which were compromised due to lack of design. The vulnerabilities mentioned in this research were caused by the first steps of the botnet's life cycle. When the malware exploits a specified computer, it must be communicating with the C&C in order to download initial configurations (Casenove & Miraglia 2014). In a Ghafir et al. publication (2014), a campus network designed as a TOR network was studied. This research aimed to capture the anonymous network traffic using one of the fundamental weaknesses of a TOR structure by taking over all the DSs to identify all anonymous traffic. In using this specified technique, it's possible to avoid anonymized network activities, however, there is also the certain need to attack all captured circuits in the onion traffic with an offline dictionary-style decryption procedure to acquire necessary information.

6. Conclusion

With the performed structure, the weaknesses of a TOR network which were mentioned in Section III (the last paragraph) can be annihilated. An implemented communication mechanism is well suited for botnet control and management. The implementation provided in this paper focused more on botnet activities and attack mechanisms. Another important result of this structure is to create a cyber intelligence network where the information flow policies are on parallel cyber domains.

The main ramification of this research is that all of the implemented structures still have the same issues as the TOR network structure mentioned in the seventh section of "TOR: The second-generation onion router" (Dingledine et al. 2004). Another ramification is that the maintenance of the implemented structure needs a lot of effort.

Maintenance issues will be solved by using dynamic configuration scripts with LinuxContainers (LXC) (Linuxcontainers.org 2013), however this is left as a future work in this research. Other shortcomings are inherited from the TOR design. Solving that issue will bring an anonymity and efficiency tradeoff in every trial.

As a conclusion, this research provided a novel point of view to anonymize communications in a parallel fashion to the created TOR structure. The study also experimented with adding another layer to increase secrecy and anonymity. To the best of our knowledge, there are no such researches about these kind of possible implementations. This article is trying to create awareness about the depth of this topic.

The main statement of this study is a communication medium which is designed for concealing itself by utilizing a TOR network. The design provides an environment which is capable of hiding itself from TOR networks by using an altered version of one. The number of such possible networks within a bigger TOR one is infinite. Therefore, even if a complete TOR architecture is compromised one day, these parallel anonymized spaces are able to stay under the radar. On the other hand, given the configuration technique explained in this research, it is impossible to know if such networks have ever existed or not.

References

Allison Nixon , John Costello, Z.W., 2016. Flashpoint. Available at: https://www.flashpoint-intel.com/action-analysis-mirai-botnet-attacks-dyn/.

Anon, 2016. Malwaremustdie. Available at: http://blog.malwaremustdie.org/2016/11/linux-malware.html.

Anon, 1999. The Honeynet Project.

Bächer, P. et al., 2009. Know your Enemy : Tracking Botnets. , pp.1–49.

Binkley, J.R. & Singh, S., 2006. An Algorithm for Anomaly-based Botnet Detection. *Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet SRUTI'06*, p.7.

Casenove, M. & Miraglia, A., 2014. Botnet over tor: The illusion of hiding. *International Conference on Cyber Conflict, CYCON*, pp.273–282.

Dingledine, R., Mathewson, N. & Syverson, P., 2004. Tor: The second-generation onion router. *SSYM'04 Proceedings of the 13th conference on USENIX Security Symposium*, 13, p.21. Available at: http://portal.acm.org/citation.cfm?id=1251375.1251396.

Falliere, N. & Chien, E., 2009. *Zeus : King of the Bots*, Available at: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf.

Feily, M., Shahrestani, A. & Ramadass, S., 2009. A survey of botnet and botnet detection. *Proceedings - 2009 3rd International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2009*, (July), pp.268–273.

Fisher J, 1998. Eggdrop.

Ghafir, I., Svoboda, J. & Prenosil, V., 2014. Tor-Based Malware and Tor Connection Detection. In *International Conference on Frontiers of Communications, Networks and Applications (ICFCNA)*. pp. 1–6.

IETF®, 1996. SOCKS Protocol Version 5.

IETF®, 2008. The Transport Layer Security (TLS) Protocol.

JonDos, 2011. JonDonym. Available at: https://anonymous-proxy-servers.net/.

Linuxcontainers.org, 2013. Linux Containers. Available at: https://linuxcontainers.org/.

Meghanathan, N., Allam, S.R. & Moore, L. a., 2010. Tools and techniques for Network Forensics. *International Journal of Network Security & Its Applications (IJNSA)*, 1(1), pp.14–25.

Paganini, P., 2016. security affairs. Available at: http://securityaffairs.co/wordpress/52845/malware/linuxirctelnet-malware.html.

Patrick Schleizer, Fortasse, HulaHoop, Troubadoour, Jason Ayala, IronSoldier, E., 2012. Whonix.

Rodríguez-Gómez, R. a., Maciá-Fernández, G. & García-Teodoro, P., 2013. Survey and taxonomy of botnet research through life-cycle. *ACM Computing Surveys*, 45(4), pp.1–33. Available at: http://dl.acm.org/citation.cfm?id=2501654.2501659.

Sanatinia, A. & Noubir, G., 2015. OnionBots: Subverting Privacy Infrastructure for Cyber Attacks. In *Proceedings of the International Conference on Dependable Systems and Networks*. pp. 69–80.

Silva, S.S.C. et al., 2013. Botnets: A survey. *Computer Networks*, 57(2), pp.378–403.

Symantec, 2010. SYMANTEC ENTERPRISE SECURIT Y Symantec Internet Security Threat Report Trends for January – June 07. *Methodology*, XII(April), p.134.

Wang, T.-S., Lin, C.-S. & Lin, H.-T., 2016. DGA Botnet Detection Utilizing Social Network Analysis. *2016 International Symposium on Computer, Consumer and Control (IS3C)*, pp.333–336. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7545203.

Y. Kugisaki, Y. Kasahara, Y. Hori, K.S., 2007. Bot Detection Based on Traffic Analysis. , pp.303–306.