Analysis of Malware and Domain Name System Traffic

Hamad Mohammed Binsalleeh

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

July 2014

CONCORDIA UNIVERSITY

Division of Graduate Studies

This is to certify that the thesis prepared

By: **Hamad Mohammed Binsalleeh**

Entitled: **Analysis of Malware and Domain Name System Traffic**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Christian Moreau

_____ External Examiner
Dr. Nadia Tawbi

_____ Examiner to Program
Dr. Lingyu Wang

_____ Examiner
Dr. Peter Grogono

_____ Examiner
Dr. Olga Ormandjieva

_____ Thesis Co-Supervisor
Dr. Mourad Debbabi

_____ Thesis Co-Supervisor
Dr. Amr Youssef

Approved by _____
Chair of the CSE Department

_____ 2014 _____
Dean of Engineering

ABSTRACT

Analysis of Malware and Domain Name System Traffic

Hamad Mohammed Binsalleeh

Concordia University, 2014

Malicious domains host Command and Control servers that are used to instruct infected machines to perpetuate malicious activities such as sending spam, stealing credentials, and launching denial of service attacks. Both static and dynamic analysis of malware as well as monitoring Domain Name System (DNS) traffic provide valuable insight into such malicious activities and help security experts detect and protect against many cyber attacks.

Advanced crimeware toolkits were responsible for many recent cyber attacks. In order to understand the inner workings of such toolkits, we present a detailed reverse engineering analysis of the Zeus crimeware toolkit to unveil its underlying architecture and enable its mitigation. Our analysis allows us to provide a breakdown for the structure of the Zeus botnet network messages.

In the second part of this work, we develop a framework for analyzing dynamic analysis reports of malware samples. This framework can be used to extract valuable cyber intelligence from the analyzed malware. The obtained intelligence helps reveal more insight into different cyber attacks and uncovers abused domains as well as malicious infrastructure networks. Based on this framework, we develop a severity ranking system for domain names. The system leverages the interaction between domain names and malware samples to extract indicators for malicious behaviors or abuse actions. The system utilizes these behavioral features on a daily basis to produce severity or abuse scores for domain names.

Since our system assigns maliciousness scores that describe the level of abuse for each analyzed domain name, it can be considered as a complementary component to existing (binary) reputation systems, which produce long lists with no priorities.

We also developed a severity system for name servers based on passive DNS traffic. The system leverages the domain names that reside under the authority of name servers to extract indicators for malicious behaviors or abuse actions. It also utilizes these behavioral features on a daily basis to dynamically produce severity or abuse scores for name servers.

Finally, we present a system to characterize and detect the payload distribution channels within passive DNS traffic. Our system observes the DNS zone activities of access counts of each resource record type and determines payload distribution channels. Our experiments on near real-time passive DNS traffic demonstrate that our system can detect several resilient malicious payload distribution channels.

DEDICATION

To my deceased mother who taught me invaluable lessons in life,

To my father who emphasized the importance of education,

To my wife for all of her incredible support, patient, inspiration, and love,

To my kids who have been giving me the strength through their sweet smiles,

To my brothers, and sisters who have been very supportive.

ACKNOWLEDGEMENTS

First and foremost, all praises to Allah for blessing, protecting and guiding me throughout this period. I could never have accomplished this without the faith I have in Allah.

I would like to thank my supervisors Prof. Mourad Debbabi and Prof. Amr Youssef for their indispensable and incredible guidance. Our research objectives would not have been achieved without the professional and experienced guidance and support of my supervisors. My gratefulness extends to members of the examining committee including Dr. Olga Ormandjieva, Dr. Peter Grogono, Dr. Lingyu Wang, and Dr. Nadia Tawbi for critically evaluating my thesis and giving me valuable feedback.

My special thanks go to my fellow labmates Amine Boukhtouta, Mert Kara, Son Dinh, Taher Azab, Elias Bou-Harb, Claude Fachkha, and Nour-Eddine Lakhdari for the stimulating discussions, for the good moments we spent during my studies.

I would like to acknowledge the financial support from the Government of Saudi Arabia under the scholarship of Imam Mohammed Bin Saud University, which enabled me to undertake my PhD studies.

Last but not the least, I take this opportunity to express my profound gratitude to my beloved parents, my wife and my two lovely daughters for their moral support and patience during my studies. Also not forgetting my brothers and sisters for their endless love, prayers and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

A	Address record
AAAA	Internet Protocol v6 Address Record
API	Application Programming Interface
AS	Autonomous System
AV	Anti-Virus
C&C	Command and Control
CNAME	Canonical NAME
DCG	Discounted Cumulative Gain
DDoS	Distributed Denial of Service
DGA	Domain Generation Algorithm
DKIM	Domain Keys Identified Mail
DLL	Dynamic Link Library
DNS	Domain Name System
DNSBL	DNS Black List
EDNS	Extension mechanisms for DNS
EP	Entry Point
FFSN	Fast-Flux Service Network
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HTTP	Hyper Text Transfer Protocol
IDS	Intrusion Detection System
IKE	Internet Key Exchange
IP	Internet Protocol

IRC	Internet Relay Chat
ISP	Internet Service Provider
IV	Initialization Vector
KNC-plot	K-Neighborhood Connectivity plot
LMS	Longest Meaningful String
MAC	Media Access Control
MAC times	Modification, Access, and Creation times
MX	Mail eXchange record
NAT	Network Address Translation
nDCG	normalized Discounted Cumulative Gain
NS	Name Server
NXDOMAIN	Non-existent Internet Domain Name
P2P	Peer-to-Peer
PDF	Probability Density Functions
PE	Portable Executable
PTR	Pointer record
RCE	Reverse Code Engineering
RR	Resource Record
RDP	Remote Desktop Protocol
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
TTL	Time-To-Live
TXT	Text record
URL	Uniform Resource Locator

WOT	Web Of Trust
XML	Extensible Markup Language
XOR	Exclusive-OR

Chapter 1

Introduction

Malicious networks are increasingly abusing Internet infrastructure to perform illicit activities. Recent studies [2] indicate that botnets are the primary platform through which cyber criminals create global cooperative networks that are instrumental in most cyber criminal attacks. A bot is a software robot or a malware instance that runs automatically on a compromised machine without being noticed by the victim user. The bot code is often written by skilled programmers and usually supports several kinds of malicious functionalities [3] that are instrumental in a variety of attacks and malicious activities. The term botnet, derived from the word bot, is a network of bots that are controlled by an attacker called a botmaster or botherder. A botnet is generally considered as a generic platform for online criminal attacks which affect the Internet economy [4].

The alarming increase in the power of botnets and their infectious effects have turned botnets into one of the biggest threats to Internet security [5]. Currently, botnets are considered as the main cause of most Internet attacks and malicious activities. Although the existence of botnets has been noticed for a long time, it is the recent growth of cyber crimes, which are mediated by botnets, that has attracted the attention of IT security researchers. Botnets are normally used to distribute malware and other harmful software. According

to a recent report [6], one botnet illegally installed adware on hundreds of thousands of computers in the U.S., including some belonging to the military.

Most of the botnets are designed to steal sensitive information (e.g., identities, credit card numbers, passwords, or product keys) from a victim's local machine. This can be achieved by employing keyloggers and screen capturing utilities. In 2013, the FBI reported that 10 members of an international cyber crime ring were arrested for using botnets to steal more than $850 million after obtaining personal financial information from compromised computers [7].

Advanced crimeware toolkits were responsible for many recent cyber attacks. These crimeware toolkits were behind more than 60% of the malicious domains as reported by Symantec [8]. In order to understand the inner workings of such toolkits, we present a detailed reverse engineering analysis of the Zeus crimeware toolkit to unveil its underlying architecture and enable its mitigation. Our analysis allows us to provide a breakdown for the structure of the Zeus botnet network messages.

In addition to analyzing malware samples, monitoring the Domain Name System (DNS) traffic also provides valuable insight into such malicious activities and helps security experts detect and protect against many cyber attacks. Since the DNS is a core component of Internet activities, it has been increasingly abused by malicious networks to operate different activities. For instance, malicious domains host Command and Control (C&C) servers that are used to instruct infected machines to perpetrate malicious activities, such as sending spam, stealing credentials, and launching Distributed Denial of Service (DDoS) attacks. Moreover, malicious domains may serve as repositories of stolen credentials [9], hacked software [10], and attack payloads [11].

In this thesis, we study the problem of DNS abuse by utilizing malware analysis and monitoring the DNS traffic. The proposed solutions can be employed to investigate different

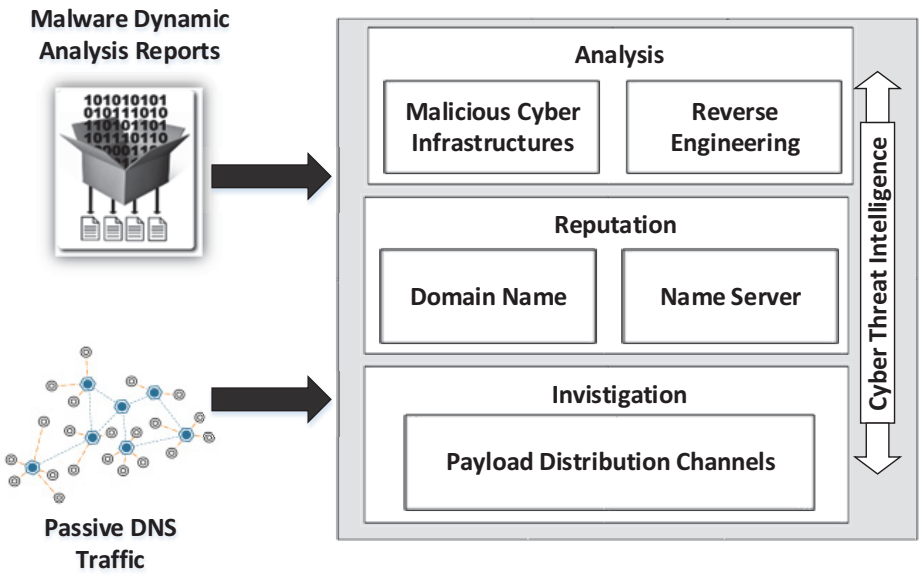


Figure 1.1: Overview of the thesis components.

cyber crime attacks and provide insightful intelligence, recommendations, and maliciousness indicators. As depicted in Figure 1.1, the analysis of DNS abuse starts by building a framework that extracts insights into malicious networks from the dynamic malware analysis reports. This framework can help investigators to collect preliminary information about different cyber attack incidents and then guide them in shaping the investigation process. A typical investigation starts by collecting simple statistics such as active malware families and most abused Internet Service Providers (ISPs). In addition, an investigator may be interested in discovering the structure of the suspicious networks, which can be achieved by our framework by applying network analysis techniques. Furthermore, our framework provides geolocation information about compromised machines in order to gain deeper insight into targeted attacks.

DNS abuse incidents are increasing dramatically as reported by Architelos (see Figure 1.2) [1]. Security professionals adopt domain name blacklisting, which is regarded as one of the basic defense lines against DNS abuses. However, domain blacklists are growing

progressively limited and ineffective in fighting the ever increasing number of malicious domain names appearing every day. To overcome this problem, blacklisting approaches must provide more information about each blacklisted domain in order to facilitate and prioritize the investigation process. Blacklisting can be extended to include detailed maliciousness indicators to focus on specific types of attacks while dealing with cyber attacks. To achieve this in our thesis, we apply a statistical reputation system to generate severity scores for domain names.

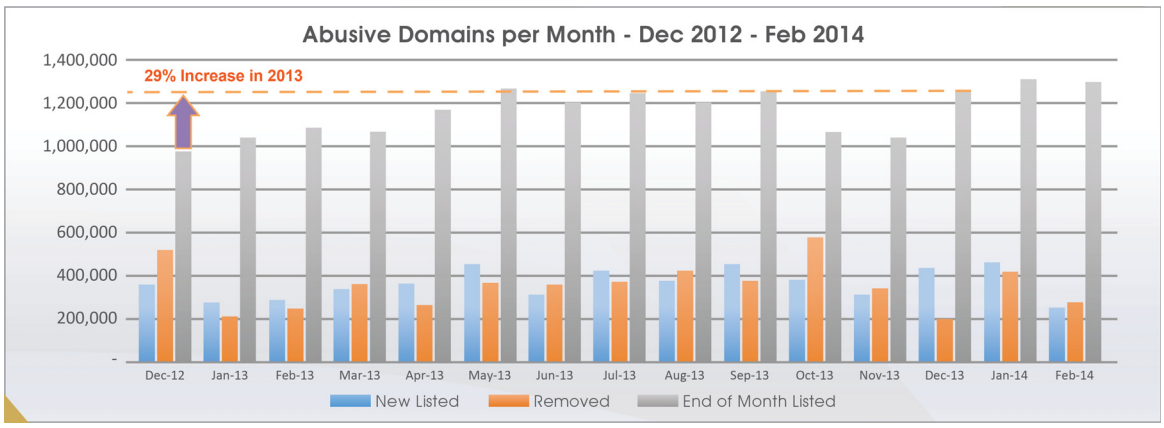


Figure 1.2: Domain abuse monitoring based on major blacklists and security services from November 2013 to February 2014 [1].

Name servers play an important role in the DNS infrastructure to provide the necessary information about domain names. In recent times, authoritative name servers have been abused to amplify DDoS attacks toward different victims [12]. Moreover, cyber criminals host their malicious domains on bulletproof name servers equipped with various techniques, which impede takedown operations [13]. In fact, the abuse of name servers has increased by advanced techniques requiring the control over name servers. To tackle this problem, reputation systems can be built around name servers to fight the root cause of malicious domains and minimize their hosting power. On the other hand, name servers have been abused to distribute malicious payload to compromised machines. Malware families such

as Morto [14], Katusha [15], and Feederbot [16] have been identified as using the DNS protocol to hide their communications. As a countermeasure, we propose a solution to identify the malicious name servers that serve payload distribution domains.

The rest of the chapter is organized as follows: Section 1.1 presents the motivation and problem statement. Section 1.2 lists the contributions of the thesis. The structure of the thesis is given in Section 1.3.

1.1 Motivation and Problem Description

In this section, we briefly discuss the motivations of the current study and identify the challenges faced by the security community in analyzing malicious networks.

1.1.1 Intelligence Extraction

One of the basic defense lines against cyber attacks is analyzing malware binary samples statically and/or dynamically to understand their behavior and thence develop detection mechanisms. However, the tremendous number of malware variants for the same malware family is severely affecting the utilization of the extracted information from the analysis. For example, Mcafee reported more than 100,000 new malware samples every day in the year of 2012 [17]. Since the number of malware samples is rapidly growing, the dynamic analysis reports produce huge amounts of valuable information that need to be utilized effectively for further analysis.

1.1.2 DNS Reputation

Domain name blacklisting is one of the basic defense lines against DNS abuses. Throughout past years, several improvements were introduced to blacklisting in order to overcome some

of the weaknesses in responsiveness and completeness [18]. Blacklists suffer from high false negative rates and false positive rates [19]. Furthermore, cyber criminals are utilizing the power of their zombie machines to operate short-lived domains, which dramatically increase the size of blacklists. For instance, the Confiker.C malware family used about 50,000 domains per day [20]. This behavior overwhelms takedown operations with a large number of malicious domain names.

In addition, DNS has been abused by cyber criminals with techniques that require more control over name servers to be implemented effectively. Cyber criminals have begun to use agile behaviors that result in a tremendous number of short-lived domains, which are used for malicious purposes for a certain period and then disposed. For example, some botnets (e.g., Torbig [21], Karken [22]) employed Domain Generation Algorithms (DGA) to periodically create and register domain names that point to the botnet infrastructure. On the other hand, malicious networks usually operate on compromised machines, which suffer from availability problems. Consequently, botmasters adopted the load balancing technique, known as Fast-Flux [23], in utilizing these compromised machines for malicious activities. Such behaviors usually rely upon frequent updates on the authoritative name servers, which are responsible for managing the malicious domains. Despite the fact that DNS abuse has been known for a long time, it remains as one of the greatest challenges in fighting cyber crime.

1.1.3 Payload Distribution Channels

A common approach to bypass network defense borders involves tunneling the communication through existing protocols. Such tunneling can effectively defeat traditional firewalls and Intrusion Detection Systems (IDS). Malicious network operators (botmasters) often

prefer tunneling to keep their communications under the radar. In the early stages of botnets, botmasters mostly used Internet Relay Chat (IRC) channels (e.g., Agobot) to operate and control their activities. The advancement of newer protocols such as Session Initiation Protocol (SIP), Peer-to-Peer (P2P), and Hyper Text Transfer Protocol (HTTP) has rendered the use of IRC channels obsolete [15]; see e.g., Zeus [24] (HTTP-based), Storm [25] (P2P-based). As a natural extension to exploiting common protocols for tunneling, DNS comes into play due to its wide availability. DNS is a query-response protocol, which responds to each query with the corresponding pre-defined resource record. The simple but robust architecture of DNS at times attracts botnets to abuse the system for different malicious activities [26, 27, 28, 16]. Botmasters take advantage of DNS tunneling to conduct malicious activities such as C&C or payload distribution. For example, in payload distribution channels, botmasters use DNS query and response packets to carry out malicious instructions and payload updates to individual bots. Malware families such as Morto [14], Katusha [15], and Feederbot [16] have recently been identified as using the DNS protocol to hide their communications.

Compared to other protocols, the inherent nature of DNS renders the protocol quite inefficient as a payload distribution channel [29]. On the other hand, DNS is widely available, which explains the recent exploitation of DNS as an attack channel despite the narrow transmission bandwidth. However, in comparison to P2P botnets [25], DNS abuse by malware has not been comprehensively studied thus far, and previous work on DNS abuse mainly focused on specific malware families (e.g., [16]).

1.1.4 Objectives

Our main objective is to generate timely, relevant and actionable cyber threat intelligence. This can be achieved by:

- Understanding most prominent threats to extend the knowledge about malware inner workings, behaviors and enabling techniques and technologies.
- Deriving cyber threat intelligence from malware analysis to reveal more insights about different cyber attacks.
- Analyzing the severity of domain names and name servers to overcome problems associated with traditional blacklisting approaches, one must provide more information about each blacklisted domain in order to facilitate and prioritize the investigation process.
- Investigating the malicious use of DNS to transport payload.

1.2 Contributions

We have developed a set of methods to address the objectives mentioned above. Our contributions can be summarized as follows:

Reverse engineering of one of the most prominent crimeware toolkits: We analyze one of the famous crimeware toolkits by reverse engineering techniques. More precisely, we present a detailed reverse engineering analysis of the Zeus crimeware toolkit to unveil its secrets and enable its mitigation.

Design and implementation of a framework to investigate abused domains and malicious infrastructure networks: We propose a framework that extracts intelligence from the dynamic malware analysis reports. We design and implement our framework based on a live daily feed of dynamic malware analysis reports, which contains an average of 6000 reports per day. Our framework has been used to investigate the abused domains as well as the malicious infrastructure network [30].

Design and implementation of a dynamic severity system that ranks domain names: We propose a dynamic severity system that ranks domain names by observing malware behavioral features in a controlled environment, which reveals the level of abuse of the observed domains by malware samples. The obtained behavioral features help to uncover malicious domains and assign dynamic severity scores to them. We design and implement a proof-of-concept for our system and evaluate it using a 10-month malware dataset, where we analyzed more than 14 million malware reports. This extensive real-world evaluation confirms that our system can assign a high severity score to malicious domain names.

Design and implementation of a dynamic severity system that ranks name servers: In general, malicious networks tend to reuse their resources, such as source code [31] and network infrastructure [32], to launch different attacks. When an important element of a malicious network, such as name server, has been taken down, the activities of that specific network is negatively affected. We propose a dynamic reputation system that ranks the severity of name servers. Some statistical features help to uncover malicious domains and then assign dynamic reputation scores for the observed name servers. We extend the domain name reputation system to observe domain features from DNS traffic that reveal various types of abuse.

Investigation of payload distribution channels in DNS traffic: In order to investigate the abuse of DNS for payload distribution, we present a comprehensive analysis of malicious payload distribution channels using a 1-year malware dataset covering Jan.-Dec. 2012. Our study reveals the effectiveness of distributing payload over DNS by malware instances. In addition, we also introduce a detection solution for payload distribution channels using passive DNS traffic by utilizing the access counts of resource records. The evaluation of the proposed method was conducted on near real-time passive DNS traffic over a 30-day period provided by Farsight Security Inc. [33].

The work in this thesis was published in [34, 35, 36]. In addition, other work, which was executed during this PhD, but not included in the thesis, appeared in [37, 38, 39, 40, 30].

1.3 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 gives an overview of the necessary knowledge required throughout our work. In addition, it provides a discussion on the current literature about the subjects that are related to the problems addressed in this thesis. Chapter 3 describes a detailed reverse engineering analysis of the Zeus crimeware toolkit. Chapter 4 proposes the intelligence extraction framework built based on the dynamic malware reports. Chapter 5 introduces a new severity system for domain names that produce domain ranking scores. Chapter 6 defines the payload distribution detection mechanism. Chapter 7 concludes the thesis and identifies directions for future research.

Chapter 2

Background and Related Work

2.1 Background

In this section, we review some of the required concepts that are used throughout our thesis. This section is organized as follows. Section 2.1.1 reviews the basic information about botnets, while Section 2.1.2 provides the required knowledge about DNS. Section 2.1.3 presents details about reputation systems.

2.1.1 Botnet Overview

A bot typically uses a combination of existing advanced malware victimizing. For example, a bot can use keylogger and rootkit techniques. Analogous to worms, a bot has the potential capability of increasing its size and propagating over the Internet. It can also spread by employing existing social engineering techniques and systems, such as instant messaging and email communication systems. Bots have recently adopted phishing techniques to trick victims into downloading specified malware [41]. As a result of these multiple propagation vectors, the attacker obtains control over many victim machines within a short time span. The regular size of most botnets currently ranges from tens to hundreds of thousands of bots,

including exceptionally large botnets comprised of several millions of bots [42]. Compared to other intrusion systems, botnets are distinct in two ways: first, bots are goal-directed, with the purpose of most attacks (such as spamming and DDoS attacks) focusing on the gain of financial profits [43]; second, the botmaster (the owner of botnet) can interact with his/her bots via C&C servers.

Botnet Life Cycle

As depicted in Figure 2.1, the life cycle of a typical botnet begins with an ordinary personal computer that is initially infected by certain propagation vectors. The infected system first launches malicious activities locally, followed by attempts to communicate with the botnet infrastructure. From the botnet infrastructure, the infected machine will have an opportunity to update itself with the latest malware binaries. The acquired binaries instruct the infected system to communicate with the rest of the botnet infrastructure in addition to other modules. In general, the infected system is directed to download the updates through a variety of protocols used for file transfer such as File Transfer Protocol (FTP) and HTTP. At a specific point in time, the botmaster tries to instruct its botnet infrastructure (C&C) with all the necessary information to launch an attack. After a successful attack, the botnet may lose some of its systems due to the presence of detection and mitigation systems inside the networks. To fill the gap, the botnet attempts to recruit more systems to be used in future attacks.

Bot Propagation and Infection Techniques

To avoid being detected by various deployed detection systems, bots are designed in a specific way that enables them to change their propagation mechanisms over time. For instance, an infected bot may apply scanning to all possible ranges of IP space for computers

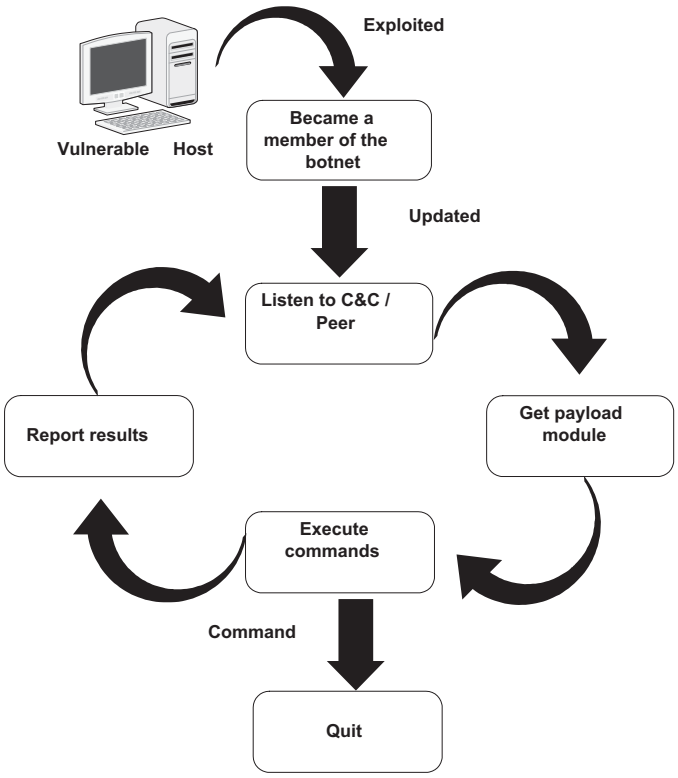


Figure 2.1: Botnet life cycle.

running exploitable services. These exploitable services allow the previous infected hosts to inject a small amount of shellcode into the victim machine. Unlike worms, botnets are difficult to be classified on the mere basis of their scanning techniques. Botnets are adopting such behaviors, and the existence of a human controller (botmaster) allows them to cause more targeted attacks while maintaining a high level of stealth.

With the development of more sophisticated protection and detection techniques, the method of botnet attacks is now more similar to that of a Trojan horse. Trojans trick victims into installing malware under the false belief that the malware is useful and beneficial software. Similarly, bot infection spreads by transmitting binaries as email attachments. Once received, the naive user may open malware executables and thus become infected. These infection scenarios are comparatively less alarming, as they require some action on the part of a user for the completion of the infection cycle. Another effective infection technique,

known as web drive-by downloads, involves the user visiting a web site that can exploit targeted web browsers and thus infect the user's system. In addition, P2P file sharing technologies are successfully exploited by botmasters to distribute malware binaries.

Botnet C&C Architecture

Every new generation of bots introduces itself by exploring different C&C techniques through which bots can be updated and directed by the botmaster, who can use different kinds of C&C mechanisms in terms of communication protocols and network structures. A good choice for attackers would be to employ the IRC protocol, which is now widely used. HTTP would be another good choice, as it hinders the detection process since HTTP traffic is for the most part allowed in network policies. More advanced botnets do not rely on centralized C&C mechanisms and are instead using distributed control techniques to avoid the single point of failure problem, such as the usage of peer-to-peer networks to organize and control botnets more consistently [44, 25, 45, 46].

Botnet Reverse Engineering

Reverse Code Engineering (RCE) is the process of analyzing the disassembly of an executable with the purpose of recreating the actual source code, or a pseudo-code representation of the executable's behavior. The RCE can disclose every decision taken as well as every algorithm used in a program, but the process can be very time consuming. Compilation of an executable will strip most of the meaningful information from comments, variable names, and so forth. Furthermore, the use of optimization by compilers will also diffuse the structure of the disassembly compared to the source code. For this reason, RCE is often used as an addition to the behavioral analysis by investigating only the interesting sections of the disassembly. These sections can be found, for example, by looking for the

use of strings appearing in the bot's network communication or by finding sections where Application Programming Interface (API) calls are made behind some interesting behavior.

Debugging software is the process of running the code with a debugger attached, permitting the use of breakpoints and variable inspection at any point in the code. Debugging is a very helpful addition to inspect the disassembly. It provides good clues to implementations of functions, if the actual input and output to the functions are traced.

2.1.2 The Domain Name System

The DNS protocol is designed to play as a translation service for the Internet infrastructure. The protocol receives a query for a specific Resource Record (RR), which describes the characteristics of a domain, and then responds with the corresponding answer based on a zone file. A zone file defines the services running under a particular domain. A domain name usually has multiple resource records dedicated for different purposes. These records consist of five main components: name, class, type, time-to-live (TTL), and data. The resource record name is a Fully Qualified Domain Name (FQDN) that consists of several labels. The right-most label is the top-level domain, and each label represents a level within the DNS hierarchy. The label that comes after the top-level domain represents the second-level domain, unless it is used as a sub-zone of a top-level domain, e.g., co.uk. Any label that comes after the second-level domain is considered as a sub-domain of the second-level domain. These sub-domains are defined and mapped to the corresponding resource record in the zone file. In some cases, a wildcard can be used to return the same resource record for any sub-domain [47]. A sub-domain can be set up to have its own zone file with a dedicated name server. In this case, the name server of the second-level domain delegates queries to the name server of the sub-domain. This technique is called zone delegation and is often used for easing the management of different sub-zones under a domain [48].

The length of a resource record name cannot exceed 256 bytes and each label length is limited to 63 bytes [49]. The resource record class defines name spaces that are used for different purposes within the DNS protocol. The resource record type indicates the type of information carried by the DNS message. In Table 2.1, we list some of the resource record types used in our work. The TTL value is a time period used by DNS servers to determine how long to cache the response before discarding it. The resource record data is the response information assigned to a resource record name.

The main actors within the DNS system architecture are the Name Servers (NSs). These servers are authorized to provide information about a set of domain names. Each server is aware of other name servers according to the hierarchy of DNS. The right-most label of any query corresponds to the top-level domain within the hierarchy. Therefore, when a query is made, the first query goes to one of the root name servers, which returns the address of the name server of the top-level domain. The query traverses the hierarchy until it reaches the name server of the second-level domain to receive the requested resource record.

Resource Record (RR) type	Description
Address record (A)/ IPv6 address record (AAAA)	IPv4/IPv6 address
Name Server (NS) record	Authoritative name server
Mail Exchange (MX) record	Mail server
Text (TXT) record	Text information associated with a name
Canonical Name (CNAME) record	Canonical name or an alias name

Table 2.1: Examples of DNS resource records investigated throughout the thesis.

Passive DNS

Passive DNS is a technique that replicates DNS activities in order to investigate the DNS traffic in near real-time. The inconsistency between Address record (A) and Pointer record

(PTR) [50], which is caused by dynamic IP addresses, requires a technique to track changes to resource records. Therefore, passive DNS is introduced to establish a real-time replication mechanism [50]. It is designed to be deployed on a local DNS resolver to observe the DNS traffic. For instance, this historical database is used to retrieve reversed queries about domain names to gain intelligence information.

Payload Distribution via DNS Hierarchy

In this section, we outline key differences between DNS tunneling and payload distribution channels. We then discuss the use of these channels both for legitimate and malicious purposes.

Recently, DNS became a target to distribute malicious payloads for two main reasons. First, DNS traffic is often allowed to pass through corporate networks without inspection, as it is considered to be a core element of Internet activities. Second, DNS protocol contains certain fields that are defined to be more flexible, which opens the doors for other unintended uses. Malicious payload can be stored in different resource records (e.g., NULL, TXT, or CNAME). Distributed data can be managed by TTL record for caching purposes. In addition, the labels within the resource record name can be used to store Base32 encoded data. RFC 1464 paved the way for payload distribution by opening the possibility of storing arbitrary information within DNS messages [51]. However, it recommends storing key-value pairs to pass only some operational data between servers. The feasibility of using DNS resource records to distribute payload has been proven by the DNS tunneling technique, which shows that DNS can be used for transmitting any type of information after simple encoding operations. However, attackers face some limitations due to the low data transmission rate through resource records. In general, payload distribution channels operate similarly to the DNS tunneling technique [29].

Payload distribution through DNS is a relatively new concept and has a very limited number of legitimate uses. Some organizations have been inspired by the evolution of DNS tunneling, and have begun to use DNS as a means to channel part of their operational data to enhance their systems.

Legitimate Use Cases: In 2007, Trend-Micro Inc. proposed a method to distribute malicious code signature updates through the DNS protocol [52]. The intention of this technique is to feed Anti-Virus (AV) software clients with signature updates through DNS as an alternative update mechanism. The signature updates are divided into several chunks, which can be identified by an identifier number. These pieces are encoded with Base64 and assigned to the zone file as TXT resource records of a specific domain name. When the client requires an update of a malicious code signature, it sends a query with an identifier number of the signature as the FQDN label. The server then responds with the corresponding AV signature in TXT records. In general, each signature update can span over many TXT records, which makes the client generate many queries to retrieve all the pieces necessary to complete the whole update. Finally, the client combines all TXT records and then forms the actual update of the malicious code signature.

In 2009, Devicescape Software Inc. introduced a methodology of a public hotspot authentication system for mobile devices [53]. In their model, there are sets of public WiFi hotspots placed across various locations, such as coffee shops and restaurants. The authentication system for these hotspots is managed through a centralized scheme. The DNS protocol is used as a channel to transfer authentication parameters between mobile devices and a credential server. The client prepares a DNS query, which consists of six subdomain labels to carry different parameters, e.g., the Media Access Control (MAC) address of the client's machine. When the name server receives the query, it forwards the parameters embedded in these labels to the back-end credential server. Based on the parameters, the

credential server prepares the corresponding credentials to be transported back to the client. Finally, the client verifies the response and then submits it to the authentication server in the local hotspot network.

Malicious Use Cases: The crucial component of any malicious network is the control and communication method within the network. DNS has been used by malicious networks for updating clients with recent payload data (i.e., module updates, command instruction). In 2011, Dietrich et al. [16] reverse engineered the Feederbot botnet that uses DNS as a C&C channel. Another example of abuse of the DNS protocol is the Morto worm [14] that uses DNS TXT records to transmit a single piece of information that can fit in one packet to the client bot. The embedded information is a Uniform Resource Locator (URL) that points to the real attack payload as explained by Symantec [14].

2.1.3 Reputation Systems

In information retrieval, reputation is a type of collective measure for satisfaction based on member ratings in a given community. Reputation is considered a soft security mechanism that has been integrated successfully in many applications [54]. There are many reputation models proposed in the literature to address issues such as trust, quality of service, and security [55]. The main components of any reputation system are participants, rating centers, and reputation engine. Since ratings are the basic input for reputation systems, the collection of ratings from participants depends on the nature of the application. For instance, centralized applications require dedicated rating centers to collect ratings to be used by the reputation engine. On the other hand, distributed applications require the ratings to be kept by participants and each individual responsible for deriving any reputation score.

Reputation engines are the main component of integrating the rating values to reflect the participants' opinions. Based on the format of rating values, reputation engines can

vary from one to another. For example, a user's preference can be represented in either binary or multinomial format. The binary format reflects the amount of negative and positive opinions, while the multinomial format provides more descriptive information about the opinions. There are many models proposed in the literature to address different requirements as discussed in [55]. The model used throughout our work is based on the Bayesian system, which computes the reputation scores by statistical update of Dirichlet multinomial probability density functions (PDF) [56]. The Dirichlet system adopts flexible multinomial rating possibilities, which enable the participant to reflect detailed subjective opinions.

Reputation System Evaluation

In order to measure the effectiveness of any reputation system, we must evaluate the resulting scores against known judgments from public or third party knowledge. Effectiveness measures differ based on the results of the evaluated reputation system, which can be ordered or unordered sets. In our case, there is a ranked set of objects which are ordered based on the opinions of the participants. Another parameter to choose the right measure is the nature of the public judgments used in the evaluation. Public knowledge can be represented as binary (e.g., Good or Bad) or graded notion. The graded notion is normally used to quantify the level of importance among the other rated objects. In general, one proper evaluation measure that can be used to measure the effectiveness of graded notion is the Discounted Cumulative Gain (DCG) [57]. The DCG accumulates the gain starting at the top of a ranked list of length n and then discounts the gain at lower ranks. More precisely, DCG is defined as:

$$DCG = rank_1 + \sum_{j=2}^{n} \frac{rank_j}{\log_2 j} \tag{2.1}$$

where $rank_j$ is the graded ranking level of an object at rank j, and $log_2 j$ is a reduction

factor for the gain. The DCG is normalized (nDCG) by the *ideal* DCG value, which reflects the *perfect* ranking [57]. The normalized form of DCG is defined as:

$$nDCG = \frac{DCG_s}{DCG_r} \tag{2.2}$$

where DCG_s and DCG_r denote the DCG values for the evaluated system and the reference knowledge, respectively. When there is a large number of objects to evaluate, it is more practical to evaluate the quality of the top ranked list, which holds the important objects among the others. The nDCG truncated at k^{th} position is denoted as $nDCG@k$.

2.2 Related Work

In this section, we present a review of state-of-the-art techniques developed in the areas of detection and mitigation of different types of DNS abuse.

This section is organized as follows. In Section 2.2.1, we discuss some previous works in the area of reverse engineering. Section 2.2.2 reviews the latest works that explore malicious infrastructures. Section 2.2.3 presents the efforts to fight different DNS abuses. In Section 2.2.4, we give an overview of the related works in analyzing the payload distribution through DNS.

2.2.1 Botnet Reverse Engineering

Bot reverse engineering can be considered as one of the primary factors that feed the learning curve about the underground community. Valuable information can be obtained by analyzing malware binaries, network traces, and infected system behavior. In the botnet literature, researchers provided case studies of famous botnet variants. Their studies aimed to gain knowledge about botnet behaviors and develop some methods to evade them. There

are three main categories of botnets based on their C&C architecture: IRC, HTTP, and P2P. For IRC botnets, [3] evaluated four different instances of IRC botnets. This comprehensive study reveals a great deal of knowledge about botnet capabilities in controlling the infected system, C&C, propagation, attacks, updates delivery, obfuscation, and deception mechanisms. Throughout their work, they determine possible infection interactions of IRC botnets and their command strings.

With the rise of HTTP botnets, researchers studied a few instances to understand the inner details of these new botnet behaviors. Nazario [58] introduced one of the first HTTP botnet analysis studies about *BlackEnergy*, which is known as a web-based bot tool. This analysis provided the research community with complete information about the botnet architecture, commands, and communication patterns. Like any web-based bot tool, there was a PHP based C&C which collects statistics about the botnet, and the bot binaries were built by a customizable malware builder program. The main threat behind this botnet was DDoS, but no significant attacks have been noticed. Another analysis was presented by Chiang and Lloyd [59] for Rustock rootkit, which contains a spam bot module. They noticed from the network traces that the communication was encrypted by the *RC*4 algorithm, which makes it difficult to be detected by detection systems. Moreover, the rootkits and the multiple levels of obfuscation further complicated the detection process by traditional AV systems. Generally, this rootkit was used mainly for spamming purposes, which can be updated through C&C servers. In their analysis, they were able to extract the encryption key of the C&C communication patterns. HTTP-based botnets were recently involved in many click frauds. One of the famous botnets responsible for click fraud attacks is the Clickbot.A. Daswani and Stoppelman [60] presented a detailed case study about clickbot.A which reveals new techniques for click fraud activities. Their analysis uncovered the main components of this botnet and their management techniques, commands, and configuration.

Botnet herders clearly realized the challenges to hide their existence with centralized control approaches. As a consequence, they shifted to decentralized techniques, P2P communication protocol, which provided them with huge scalability and led them to become more resilient against traditional detection and tracking techniques. Recent contributions aim to understand and analyze the existing P2P based botnet instances. For example, Porras et al. [61] reverse engineered the *Storm* botnet variant to uncover its capabilities to control bots and hide binary distribution as well as its obfuscation techniques. From their analysis, the Storm botnet manages its C&C communication by the Overnet protocol with various customized communication patterns. Generally, the Storm botnet is used for sending spam and has capabilities for DDoS attacks. From another perspective, Holz et al. [25] and Grizzard et al. [44] reported on the Storm botnet by exploring the encryption key generation algorithm that is used by each Storm variant to establish secure communication with other peers in the botnet.

Another example of P2P-based botnets is Nugache [62], which controls its army by a customized P2P protocol architecture. Dittrich and Dietrich [63] reported information about their analysis for the Nugache instance. They analyzed the communication patterns, which involved the key exchange process (Rijndael algorithm) to encrypt the C&C communications. Using an encrypted P2P network, the botherder instructs the botnet to listen to a specific IRC channel for DDoS commands. Nugache was later updated to use P2P networks for all its communications. Again, Dittrich and Dietrich [64] addressed extra aspects of analysis and investigated the size estimation of the Nugache botnet using a customized bot client crawler.

2.2.2 Malicious Infrastructure Analysis

Within an interactive community, there are many relationships that can be leveraged to gain insightful knowledge about the nature of the community. Malicious network communities have been explored from different perspectives. For instance, criminal network infrastructures can be discovered by monitoring the DNS traffic, spam emails, or URLs. Recently, Konte et al. [65] investigated the use of dynamic changes for Fast-Flux networks and their role to characterize the hosting infrastructure for different online scam campaigns. Nagaraja et al. [66] extracted P2P botnet communities from network traces by modeling the interaction information between peers as a communication graph. Invernizzi et al. [67] proposed an approach to identify malicious download attempts in large-scale networks. The suspicious downloads that exhibit similar behavior aggregated to construct different malicious neighborhood graphs, which can be used to recognize various malware distribution infrastructures. Nadji et al. [68] constructed criminal networks by correlating the passive DNS and several indicators for malicious activities. Using a graph-based approach, they identify criminal communities and study their structural properties. They identified the critical nodes to take down the network by utilizing the eigenvector centrality.

In addition to exploring the relation between domain names and IP addresses to uncover malicious infrastructures, our work differs from previous works by investigating the interaction between malware samples and domain names to understand the level of malicious abuse.

2.2.3 DNS Reputation

Domain name blacklisting is one of the oldest and most effective techniques to fight cyber criminal activities and track the abuse of DNS. Most of the current blacklisting services are based on two approaches to collect malicious IP addresses or domain names. The first

approach is based on collecting IP addresses and domain names from complaints or abuse reports by victims [69]. The other approach utilizes the behavioral analysis of DNS traffic based on finding evidence of malicious activities [70, 71, 72].

Dietrich and Rossow [73] studied the effectiveness of various IP blacklists and reported the need for dynamic blacklisting for optimization purposes. Zhang et al. [74] introduced dynamic domain blacklisting by cross-correlating DShield [75] attack log files from different providers. They build relevant IP blacklists for each data provider as well as attack severity scores based on frequencies of accessing pre-defined ports. The relevance concept has been improved by Soldo et al. [76] using recommendation systems to predict attack sources. As a proactive solution, Felegyhazi et al. [77] use domain name registrations to infer sets of malicious, not-yet-blacklisted domains based on initial seed domain names. Similarly, Sato et al. [78] extended blacklists by studying the co-occurrence of DNS queries with known blacklisted domains. Recently, Antonakakis et al. [79] presented Notos, a dynamic reputation system for domain names using behavioral features from passive DNS traffic. Similarly, Bilge et al. [80] introduced EXPOSURE, a system to detect malicious domain names involved in malicious activities. Both Notos and EXPOSURE extract their evidence from passive DNS datasets and then apply data mining algorithms to calculate domain reputation. In order to keep up with the fast growing number of malicious domains, KOPIS [81] has been proposed to extend and detect new related malicious domains by observing DNS patterns from the upper DNS hierarchy. Recently, GZA [82] used game theory during the dynamic analysis of malware instances to reveal its alternative domains. These studies focused on the construction or the extension of blacklists. However, there is no work on dynamically evaluating the level of maliciousness of the abused domains included in these blacklists. Certain works use the knowledge of blacklists to construct group reputations. For instance, Stone-Gross et al. [83] address the identification

of rogue networks by checking the presence of a large number of long-lived, misbehaving hosts within three different sources of malicious activities. Similarly, Kalafut et al. [84] show the possibility to derive the reputation of ISP networks by combining different existing public blacklists. Collins et al. [85] use the fact that malicious networks tend to abuse weakly protected networks. Based on this observation, they defined a network based quality uncleanliness indicator to quantify the existence of malicious hosts. Roveta et al. [86] presented a visualization tool to explore the behavioral pattern inside rogue AS in order to facilitate the identification of malicious events.

The Fast-Flux technique has captured the attention of many researchers to study and analyze botnets that are using this mechanism. For example, Holz et al. [87] introduced a complete study and analysis of Fast-Flux Service Networks (FFSNs) that are used by spamming botnets. They presented different metrics, which can be used to identify malicious fast-flux activities. Furthermore, they studied the characteristics of FFSNs and developed detection algorithms that first extract URL links inside spam emails and then, using a linear classifier, identify FFSNs based on the number of unique IP addresses in DNS queries, the number of unique Autonomous System (AS) [88] numbers of those IP addresses, and the number of *NS* records in a single lookup. In another work, Nazario and Holz [89] continued the analysis with live network traces, which reveal botnet memberships and domain names. They extended the heuristic features of FFSNs that are used by botnets. Passerini et al. [90] introduced a set of features to characterize FFSNs. Their approach, called $FluXOR$, collects suspected domain names from different sources, monitors their DNS response messages over specific periods of time, and then uses a trained naïve Bayesian classifier to classify these domain names as either benign or malicious FFSNs. Similarly, Caglayan et al. [91] presented a distributed architecture of web services to detect FFSNs activities using active and passive monitoring of DNS properties.

DGA was recently introduced by botnets to add another layer of DNS abuse. Infected machines use DGA to periodically generate domain name lists and then reach the botnet infrastructure. Stone-Gross et al. [21] reported the experience of such botnets by over-controlling the *Torbig* botnet by reverse engineering the DGA algorithm. Bilge et al. [80] captured DGA domains by observing some behavioral features from the passive DNS traffic. Domains that are generated by DGA may not always be registered by the botmasters for different reasons. Pleiades [26] has been proposed to utilize this behavior by observing Non-existent Internet Domain Name (NXDOMAIN) DNS response messages. Domains with NXDOMAIN responses were inspected against certain statistical features to detect C&C servers and cluster botnets.

Unlike previous works on reputation systems, we do not only detect malicious domain names; rather, we are proposing a dynamic scoring mechanism for evaluating the severity and the maliciousness of each domain and name server.

2.2.4 Payload Distribution via DNS

The use of DNS as a communication medium for payload distribution is relatively new and research activities on this topic are limited. Although these studies are scattered, they can be roughly grouped under four categories: the malicious channels in DNS protocol, the feasibility of using DNS in malicious activities, the detection of DNS tunnels, and using DNS traffic for detecting other malicious activities.

Malicious Channels in DNS Protocol: Dietrich et al. [16] are the first to discuss the existence of botnets that tunnel the C&C channels through DNS. They discovered a malware family Feederbot that exfiltrates data within DNS query sub-domain labels and infiltrates the attack payloads in DNS response packets. Their detection method introduces the extraction of several features from the response data. While their work showed promising

results, it is limited to the detection of aggressive DNS tunnels for C&C channels. Malware families are using more resilient methods for receiving the attack payloads through DNS rather than DNS tunneling [14]. Furthermore, their work focused on the assumption that there will be a certain degree of traffic, while our analysis showed that some families use the DNS to receive a very limited amount of payload, such as the Morto family. Moreover, we found that malware might not receive Base32 or Base64 encoded payload, but rather clear text in TXT records.

Feasibility of using DNS for Malicious Activities: Xu et al. [92] introduced a resilient mechanism for bots to create covert channels through DNS for C&C communications. They designed a stealthy C&C that supports two different modes. The *Codeword mode* creates a uni-directional communication channel that pulls the attack payload. The *tunneled mode* creates a bi-directional communication channel between bots and the C&C server. They also mentioned techniques to increase the stealth of these channels to make these queries virtually undetectable from the host perspective. In fact, during our analysis in passive DNS and malware datasets, we found that their proposed methods are already used by some malware families, such as Feederbot [16] and Morto [14]. While their technique can easily defeat host-based detection mechanisms, we are able to detect these malicious channels in passive DNS traffic. We also discovered that malware families using bi-directional channels are often easily detected due to extensive traffic. Similarly, Raman et al. [93] proposed a network penetration technique that uses DNS tunneling to infiltrate the attack payload. Their technique is based on establishing a tunnel via an exploit code. Our system can detect the payload distribution channel in passive DNS regardless of the format of the payload, as we do not inspect the content of DNS messages. There are several studies on building a covert channel using DNS query and response packets [94, 95, 96, 97, 29]. They discuss the possibility of sending and receiving data through DNS query response packets as well

as the performance analysis of existing DNS tunneling tools.

Detection of DNS Tunnels: There are some proposed methods for detecting DNS tunneling within a network by using the n-gram analysis [98, 99]. Promising results were presented in terms of detecting the tunnels; however, malicious payload distribution channels often do not have extensive upstream data, and thus do not show this characteristic feature of DNS tunneling tools. Therefore, any string based analysis on the queries might not reveal significant differences between regular and malicious queries to detect these channels. Our system also detects the payload distribution channels regardless of the syntax by using DNS zone activities.

Detecting other Malicious Activities in DNS: Choi et al. [100] proposed an algorithm to detect the botnet activities based on DNS queries. They targeted the similarity of queries of bots from the same botnet. Although they are focusing on query similarities, we focus on query and response patterns as well as the DNS zone activities. Finally, there is also some work on detection of malicious activities in passive DNS [79, 26, 80, 101]. These proposed methods focused on analyzing other abuses of the DNS protocol, such as the domain reputation problem and DGA-generated domain names.

Chapter 3

On the Analysis of the Zeus Botnet Crimeware Toolkit

In this chapter, we present a case study on the reverse engineering steps necessary to understand the inner working of the Zeus crimeware toolkit and its components.

The Zeus crimeware toolkit has become one of the favorite tools for hackers because of its user-friendly interface and its competitive price in the underground communities. This crimeware allows attackers to configure and create malicious binaries, which are mainly used to steal users' Internet banking accounts, credit cards, and other sensitive information that can be sold on the black market [102]. It also has the ability to administrate the collected stolen information through the use of a control panel, which is used to monitor, control, and manage the infected systems. In fact, this prediction was confirmed in July 2009 when a security publication from Damballa positioned Zeus as the number one botnet threat with 3.6 million infections in the US alone (roughly 19% of the installed base of PCs in the US [103]). It was also estimated that Zeus is guilty in 44% of banking malware infections [104]. Symantec Corporation referred to this crimeware toolkit as the "King of the Underground Crimeware Toolkits" [105]. To the best of our knowledge, there has been

no reverse engineering attempt to de-obfuscate and analyze Zeus before the publication of our work [34].

The remainder of this chapter is organized as follows. Section 3.1 is dedicated to the description of the Zeus crimeware toolkit components and how they are integrated. Section 3.2 details the network behavior analysis that is inferred from observing the network traffic between a bot instance and the associated C&C server. In Section 3.3, we detail the four obfuscation levels and explain how they have been uncovered. This step led to the actual un-obfuscated code of the bot and to later revealing the infection/installation process, as well as the encryption key that makes it possible to decrypt the C&C communications between the infected machine and the botnet infrastructure. We also present a sample decrypted communication session between an infected machine and a C&C server. Our conclusion is given in Section 3.4.

3.1 Description of the Zeus Crimeware Toolkit

The Zeus crimeware toolkit is a set of programs that have been designed to set up a botnet over a high-scaled networked infrastructure. Generally, the Zeus botnet aims to make machines behave as spying agents with the intent of obtaining financial benefits. The Zeus malware has the ability to log inputs that are entered by the user as well as to capture and alter data that are displayed into web-pages [102]. Stolen data can contain email addresses, passwords, online banking accounts, credit card numbers, and transaction authentication numbers. In our analysis, we examine the Zeus crimeware toolkit v.1.2.4.2, which is the latest stable publicly available version in the underground community. As depicted in Figure 3.1, the overall structure of the Zeus crimeware toolkit consists of five components:

1. A control panel containing a set of PHP scripts that are used to monitor the botnet

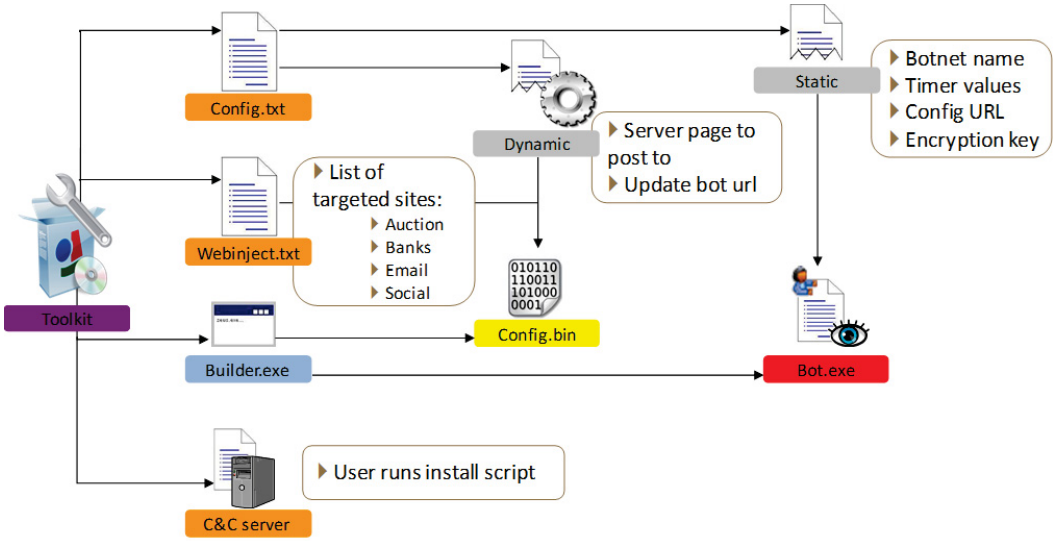


Figure 3.1: The Zeus crimeware toolkit components.

and collect the stolen information into a MySQL database and then display it to the botmaster. It also allows the botmaster to monitor, control, and manage bots that are registered within the botnet.

2. Configuration files that are used to customize the botnet parameters. It involves two files: the configuration file `config.txt` that lists the basic information, and the web injects file `webinjects.txt` that identifies the targeted websites and defines the content injection rules.

3. A generated encrypted configuration file `config.bin`. This holds an encrypted version of the configuration parameters of the botnet.

4. A generated malware binary file `bot.exe`. This is the bot binary file that infects the victims' machines.

5. A builder program that generates two files: the encrypted configuration file `config.bin` and the malware (actual bot) binary file `bot.exe`.

On the C&C side, the crimeware toolkit can easily set up the C&C server through an installation script that configures the database and the control panel. The database is used to store information related to the botnet and any updated reports from the bots. These updates contain stolen information gathered by the bots from the infected machines. The control panel provides a user-friendly interface to display the content of the database as well as to communicate with the rest of the botnet using PHP scripts. The botnet configuration information is composed of two parts: a static part and a dynamic part. In addition, each Zeus instance retains a set of targeted URLs that are fed by the web injects file `webinject.txt`. Instantly, Zeus targets these URLs to steal information and to modify the content of specific web pages before they get displayed on the user's screen. The attacker can define rules that are used to harvest data from web forms. When a victim visits a targeted site, the bot steals the credentials that are entered by the victim. Afterward, it posts the encrypted information to a drop location used to store the bot update reports. This server decrypts the stolen information and stores it into a database.

3.2 Zeus Botnet Network Analysis

In this section, we explain the network communication that occurs between the C&C server (the server containing the control panel) and an infected machine. Such an analysis can be used to write IDS rules and AV detection routines. In order to perform the network analysis, we built a sandbox environment to collect and analyze the network traces generated from the communication between the C&C server and one of the bot instances. We configured a web server, which acts as the C&C server and the drop location. This server hosts all resources that are required to operate the botnet (`config.bin` file, PHP scripts and the MySQL database). To customize the malware, we used the builder program to generate the malware binary file, which is configured to communicate with the C&C server. Within

our environment, fake websites are generated to reflect real scenarios of botnet attacks. All necessary entries of the configuration file as well as the web injects scripts are modified to target the fake website. After infecting and running a machine manually with the bot binary file, we collected network traces for one day. During this session, the user of the infected machine visited the targeted website and used login credentials, personal information, and credit card information for testing purposes.

By analyzing the bot network communications, we can learn the overall behavior of the Zeus botnet. The network behavior of the Zeus botnet constitutes a starting point where we can dig into the crimeware toolkit functionalities. Since the Zeus botnet is based on the HTTP protocol, it uses a pull-method to synchronize the botnet communications. From the collected network traces between a bot and a C&C server, we observe that the bot periodically checks a specific server for an up-to-date configuration and bot binary files. The HTTP communication messages between the two entities are encrypted. By observing the network trace, we managed to determine the following communication pattern between the C&C server and the infected machine:

1. The infected client starts the communication by sending a request message `GET /config.bin` to the C&C server. This message is a request to fetch the configuration file for the botnet.

2. The C&C server replies with the encrypted configuration file `config.bin`.

3. The client receives the encrypted configuration file and decrypts its contents using an encryption key, which is embedded inside the bot binary file.

4. In a situation where the botmaster wants to involve the infected machine to manage the botnet, the infected machine has to provide its external IP address and report any use of Network Address Translation (NAT). In order to know the external IP address

that is seen by the botnet servers, the infected machine makes a request to a specific server. Subsequently, this server informs the infected machine about their externally facing IP address. The server's URL is provided in the static configuration file.

5. The bot posts the stolen information and its updated status reports to the C&C server `POST/gate.php`.

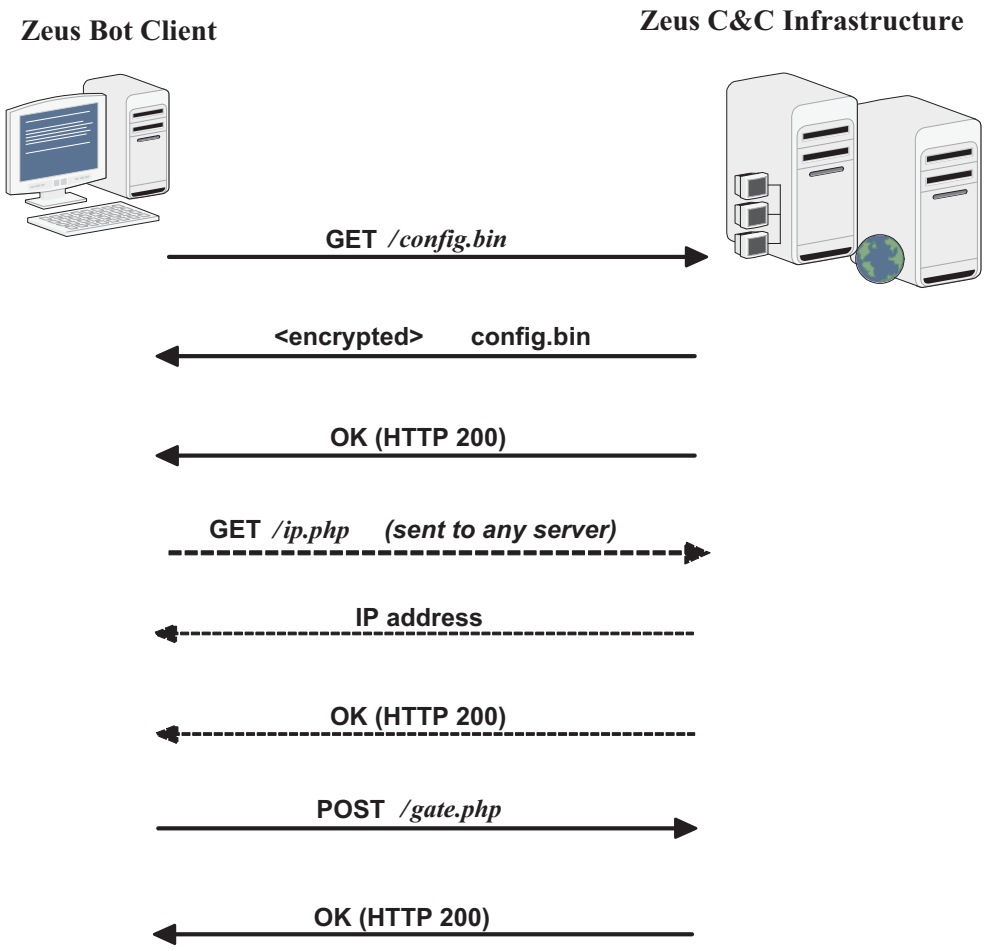


Figure 3.2: Communication pattern of Zeus.

Figure 3.2 illustrates the communication pattern between the C&C server and the infected machine. The communication pattern is repeated frequently depending on a timing variable, which is defined in the botnet configuration file.

3.3 Reverse Engineering Analysis

The increasing use of malicious software has pushed security experts to try to find the secrets related to the development of malware design. A common technique to detect the existence of a given malware is to track system modifications. The changes include what an operating system runs at startup, default web pages, generated traffic, infection of processes, packing/unpacking of binaries, and changes to the registry keys. One way to look for these changes is to reverse engineer the malware and try to reveal what is hidden behind the assembled code. In our case, this kind of analysis provides invaluable insight into the inner working of the crimeware toolkit in general and the malware binary in particular. In this line of thought, we investigate the builder program and malware binary file. To this end, we mainly employ "IDA Pro" [106] to disassemble the binaries and debug them to understand their business logic. The analysis is two-fold: first, an analysis that is related to the builder program; second, an analysis that is linked to the malware binary file.

3.3.1 The Zeus Builder Program Analysis

The builder is one of the components of the Zeus crimeware toolkit. It uses the configuration files as an input to generate the bot binary file and the encrypted configuration file. The builder component resides in the hands of the botmasters to customize and create new malware samples to be delivered to the victims.

We first analyze the builder program, as it uses a known obfuscation technique that can be easily removed. In addition, the GUI allows us to categorize different subroutines, which make up the builder program functionalities. Using the "PaiMei" reverse engineering framework [107] (a reverse engineering framework that provides many reverse engineering tasks, such as fuzzer assistance, code coverage tracking, and data flow tracking), we were able to see exactly which functions of the builder program are invoked by a specific action.

This significantly aids in simplifying the reverse engineering efforts as it allows us to focus on a few key subroutines at a time. In what follows, we summarize the reverse engineering analysis of the functionalities of the builder program.

Building the Configuration File Functionality: This function is responsible for encoding the clear text of the configuration files of the botnet into a specific structure. It subsequently encrypts the whole structure with the RC4 encryption algorithm using the configured encryption key.

Building the Malware Binary File Functionality: The main function of the builder program resides within this functionality, which is responsible for building the customized malware binary files. In general, it builds the malware executable file into a portable executable (PE) standard format. It also sets some parameters according to the current configuration file and then produces the malware binary file.

Malware Infection Removal Functionality: The builder has a functionality that ascertains the presence of the Zeus bot and removes it. When this functionality runs, it performs a detection routine by checking the existence of special registry keys that are inserted during the bot infection process. It also detects the presence of some files in the system. If these files are detected, the builder program cleans some registry keys and instructs the bot to shut itself down and then deletes the stored Zeus binary file from the system. Upon reception of the shutdown command, the expected behavior of the bot is to disinfect itself from the currently running processes. The analysis reveals the names of files whose presence in the system is checked by the builder. Table 3.1 represents these file names with their descriptions.

File	Description
C:/WINDOWS/system32/sdra64.exe	A copy of a bot, which has infected "system32" folder.
C:/WINDOWS/system32/lowsec/local.ds	A data storage file, used to store the configuration file that is used locally by a given bot in the system.
C:/WINDOWS/system32/lowsec/user.ds	A data storage file, used to log the users' activities that have been recorded by the bot.

Table 3.1: Description of the files that are created during the bot infection.

3.3.2 Zeus Bot Binary Analysis

As depicted in Figure 3.3, the bot binary file contains four segments: a "text/code" segment, an "imports" segment, a "resources" segment, and a "data" segment. We begin our analysis at the malware Entry Point (EP) that resides in the "text/code" segment. The initial analysis of the disassembly reveals that only a small part of the "text/code" block is a set of valid computer instructions. The remainder of the binary is highly obfuscated, preventing the computer from using these segments directly unless they are de-obfuscated at some stages.

De-obfuscation Process

Using the "IDA Pro" debugger, we were able to debug the malware and walk through the instructions to analyze and understand the logic of the de-obfuscation routines. Each routine reveals specific information used by the other routines until all obfuscation layers are removed. The first de-obfuscation routine contains a 4-byte long decryption key and a one-byte long seed value. These two values are used to decrypt a block of data from the "text/code" segment and then write the decrypted data in the virtual memory. The result of the first de-obfuscation routine revealed some new code segments. These segments contain three de-obfuscation routines as shown in Figure 3.4. During our analysis, the initial offset

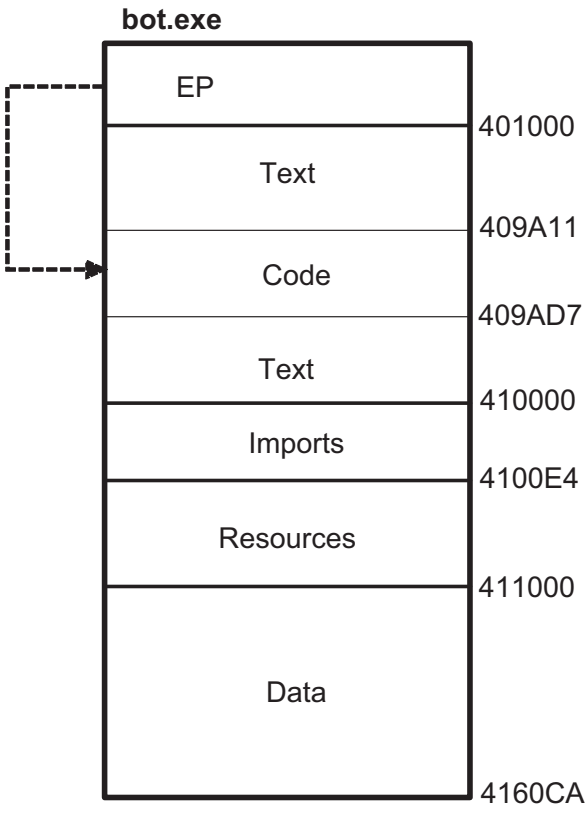


Figure 3.3: Segments of the `bot.exe` binary file.

address of the memory for the code segments was `0x390000`. After the address space of the second de-obfuscation routine, there was an 8-byte key that the "IDA Pro" incorrectly identified as code instructions. Figure 3.5 illustrates the location of the 8-byte key. In what follows, we explain the main logic of the second de-obfuscation routine.

1. First, the routine copies two binary blocks from the "text/code" segment, concatenates them, and writes them into the virtual memory. The first text block contains data with many zero value bytes that will be filled by the next text block as shown in Figure 3.6.

2. Second, the routine scans every byte in the first text block and when it encounters a "hole" (zero byte), it overwrites the zero byte with the next available byte in the

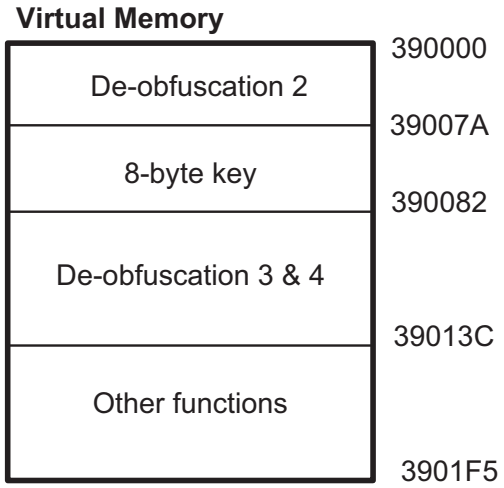


Figure 3.4: De-obfuscated code in the virtual memory.

"filler" text block. This is repeated until all "holes" are filled (See Figure 3.7).

The filled text segment turns out to be the main outcome of the second de-obfuscation routine. However, this text segment is still not readable and is not considered as computer instructions. By utilizing the 8-byte key, the third de-obfuscation routine starts by decrypting the output of the second de-obfuscation. Similar to the first de-obfuscation routine, this routine utilizes the 8-byte key and performs an Exclusive-OR (XOR) operation instead of an addition operation. Finally, the fourth de-obfuscation layer contains heavy computations to initialize and prepare parameters for the rest of the malware operations. It uses the decrypted bytes revealed by the previous routines to modify the rest of the "text/code" segment. After this routine is completed, we can observe the real starting point of the Zeus malware. Although the "text/code" segment is now valid, the Zeus bot binary employs two additional layers of obfuscation. These two layers are de-obfuscated during the installation procedure. They consist of logical loops that transform arbitrarily long strings into a readable text. The first layer is performed on a set of strings that the malware uses to

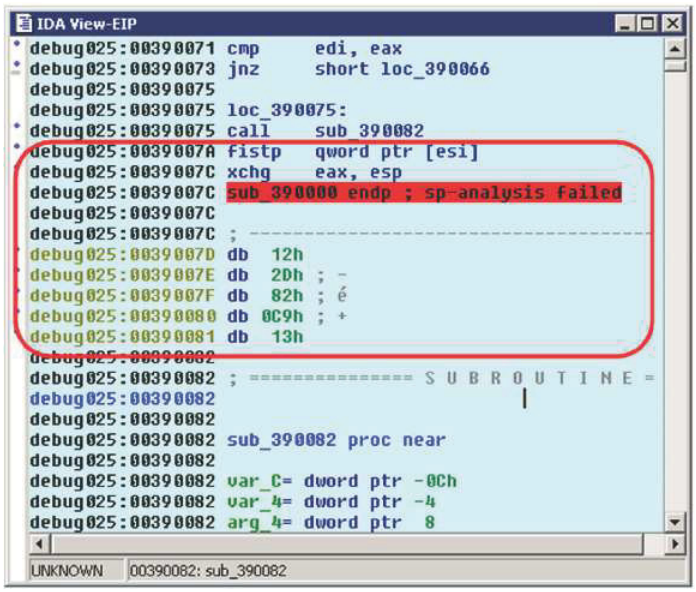


Figure 3.5: The 8-byte key.

load the Dynamic Link Library (DLL), retrieve function names, and for other purposes during the installation process as described in Algorithm 3.3.1. Similarly, the second layer is used to decrypt URLs in the static configuration of the configuration file as summarized in Algorithm 3.3.2.

Algorithm 3.3.1: DECRYPT_STRING(*enc_string*)

seed = 0xBA;

String new_string = new String(enc_string.length());

for $i = 0$ **to** enc_string.length()

do {
- new_string[i] = (enc_string[i] + seed) %256;
- seed = (seed + 2);

return (*new_string*)

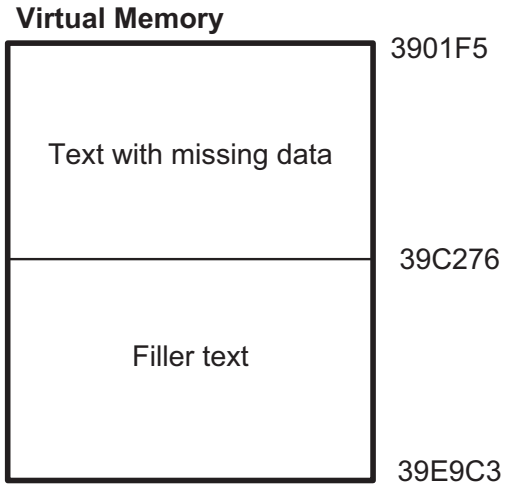


Figure 3.6: The virtual memory used by the second de-obfuscation routine.

Algorithm 3.3.2: DECRYPT_URL(*enc_url*)

```
String new_url = new String(enc_url.length());
for i = 0 to enc_url.length()
do { if (i%2 == 0)
       then
     new_url[i] = (enc_url[i] + 0xF6 - i * 2) %256;
     else
     new_url[i] = (enc_url[i] + 0x7 + i * 2)%256;
return (new_url)
```

Bot Installation Process

Following execution of the first four de-obfuscation routines, the malware begins the installation process, which aims to prepare and then launch the malicious activities of the

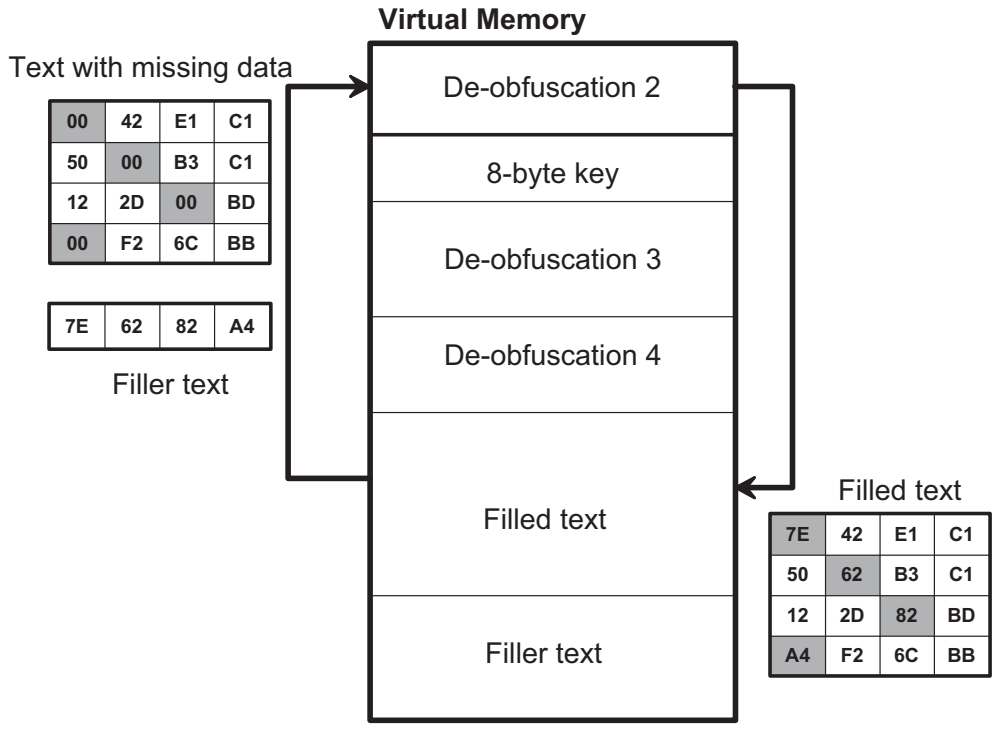


Figure 3.7: The result from the second de-obfuscation routine.

malware. In what follows, we explain the main procedure of the installation process.

1. The Zeus malware dynamically loads the `LoadLibrary` and the `GetProcAddress` methods from `Kernel32.dll` library.

2. The Zeus malware decrypts the set of strings, which become DLL methods names, into the virtual memory according to Algorithm 3.3.1.

3. The `LoadLibrary` and the `GetProcAddress` methods are used to load further methods, as decrypted in step 2, from the Windows DLLs.

4. The Zeus malware enumerates the current process table searching for targeted processes, such as the main process name for the Outpost personal firewall application from Agnitum Security [108] `outpost.exe` and the main process name for the personal firewall of the ZoneLabs Internet security [109] `zlclient.exe`. If any of these

processes are found, the Zeus malware aborts the installation process.

5. The Zeus malware appends the path `C:/Windows/System32/sdra64.exe` to the `HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/WindowsNT/CurrentVersion/Winlogon/Userinit` registry key. This entry enables the Zeus malware to initiate its installation process again during Windows startup.

6. Finally, the Zeus malware injects its entire binary file from the memory address `0x400000` to `0x417000` into the virtual memory of the `winlogon.exe` process. Following this, Zeus passes control to this process by creating a new user thread, which is immediately executed.

Similarly, the bot uses these steps when the infected machine is restarted. However, certain steps are performed only during the initial Zeus installation process. These steps involve the creation of a local copy of the malware and storing it on the infected system for further activities. In what follows, we list the main processes of creating a local copy of the malware.

(a) The Zeus malware searches for any existing copies of previous Zeus infection files `sdra64.exe` and erases it from the infected machine. This behavior occurs when the Zeus binary file is being updated with a newer version of the malware.

(b) The bot makes an exact copy of itself and saves it to `C:/Windows/System32/sdra64.exe`. To evade signature-based detection systems, it appends some randomly generated bytes to the end of the file.

(c) In order to hide itself, the bot duplicates the Modification, Access, and Creation times (MAC times) information from the `Ntdll.dll` library, and applies them to the `sdra64.exe`.

The purpose is to make `sdra64.exe` appear as a system file that has been present since Windows was first installed.

(d) In another level of hiding the created file, the bot sets the `sdra64.exe` file attributes to system and hidden, so that the user cannot see the file using the standard file explorer.

At this stage, the malware is already injected within the `winlogon.exe` running process. The currently running bot exits and leaves the control to the injected process. However, the installation procedure is continued by the user thread that was started in the `winlogon.exe` process as described in step 6. From the injection process, we infer that the entire Zeus binary file is copied into the `winlogon.exe` process. Therefore, the injected Zeus instance starts by removing the remaining two layers of the obfuscation by applying Algorithm 3.3.1 and Algorithm 3.3.2 as described in Section 3.3.2. When the injected malware decrypts all the strings, the Zeus instance employs the piggyback thread technique (to control the infected system through a legitimate process) within the `winlogon.exe` process. However, Zeus instances only perform a few tasks before they create another thread and exit themselves. This is yet another attempt made by the designers of the Zeus malware to evade detection. Subsequently, the Zeus instance starts injecting itself into another process, namely the `svchost.exe` process. This injected process initiates a communication channel with the C&C server to download the latest updates for the configuration file and the malware itself. The targeted processes later get injected with the latest malware payload and then activate the process of stealing information through API hooking techniques. During the malware update process, the following changes were observed in the file system:

1. A new folder is created at the path `C:/Windows/System32/lowsec`. Hiding techniques similar to those applied to the `sdra64.exe` are also applied to the newly created folder.

Command	Purpose	Return Value
1	Retrieve Zeus version number	4 bytes in a buffer
2	Retrieve name of the botnet	Ascii string in buffer
3	Uninstall Bot	n/a
4	Open the `local.ds` file or create it if it does not exist	n/a
5	Close the `local.ds` file	n/a
6	Open the `user.ds` or create it if it does not exist	n/a
7	Close the `user.ds`	n/a
8	Close the `sdra64.exe`	n/a
9	Open the `sdra64.exe`	n/a
10	Retrieve loader file path	Wide character string
11	Retrieve configuration file path	Wide character string
12	Retrieve log file path	Wide character string
13	Crash the `winlogon` process intentionally	n/a

Table 3.2: List of the Zeus malware commands.

2. Two new files, `local.ds` and `user.ds`, are created and placed in the new folder. The `user.ds` stores the dynamic configuration file, and the `local.ds` logs the stolen information until the Zeus malware is ready to send it to the drop location.

The malware that resides in the `winlogon.exe` process acts as the brain for the Zeus malware activities. It communicates and coordinates all the infected processes using the named pipe `_AVIRA_2109`. Table 3.2 shows the list of numerical commands that are supported by the Zeus malware.

Key Extraction

As mentioned in Section 3.1, the Zeus botnet uses a configuration file that contains static information. This part of the configuration is stored inside the malware binary file in a specific structure. During the de-obfuscation processes, this structure is recovered and placed

in the virtual memory (in our analysis, starting at `0x416000`). All information in the structure is completely de-obfuscated, except for two URLs: `url_compip` and `url_config`. These URLs can be de-obfuscated using Algorithm 3.3.2. The `url_compip` is the web location to determine the IP address of the infected host, and the `url_config` is the web location to download the configuration file for the botnet. The static configuration structure also contains an RC4 substitution table that is generated by the encryption key specified in the configuration file. Throughout our analysis, we noticed that the substitution table was generated by the RC4's key-scheduling algorithm, and we verified that the encryption employed by Zeus is done by the RC4 algorithm. The recovered static configuration can be used in different ways to gain partial control over the botnet. The most valuable piece of information is the substitution table, which can be used to decrypt all communication of the Zeus botnet. Moreover, it can be used to decrypt the configuration file as well as the stolen information. In order to recover the static configuration structure described above, we must go through all of the de-obfuscation phases discussed in Section 3.3.2. This requires executing the malware until it completes all of the de-obfuscation layers. Emulation techniques are considered as safe and fast procedures to achieve our goals. Using Python scripting language along with the "IDAPython" plugin [110], we were able to emulate all of the de-obfuscation routines and extract the substitution table from the static configuration structure. These extracted keys allow for decryption of the botnet communication traffic and all of the encrypted files. Similarly, they allow us to extract any information from the static configuration structure, such as the URLs for any future updates, which point to the C&C servers. Our experimental results demonstrate that any subversion of Zeus (v.1.2.x.x) can be fully analyzed using our methodology as it holds the same logical blocks.

3.3.3 Packet Decryption

After extracting the RC4 encryption key as described in Section 3.3.2, we used it to decrypt the botnet communications. By decrypting the transmitted HTTP payload, we were able to uncover the structure of the messages between the bot and the C&C server. We analyzed the structure of the HTTP POST messages (`POST /gate.php`), which carry all the updates and reports from the bots to the C&C server. Each bot posts a variable number of encrypted bytes based on the data sent to the C&C server in a specific structure. The payload is encrypted using only an RC4 encryption algorithm. As depicted in Figure 3.8, we restore the structure of the messages as follows:

1. Each message starts with a header that consists of 28-bytes. This header contains an MD5 hash value for the rest of the message.

2. As shown in Figure 3.8, the rest of the message follows in the form of repeated data blocks, where each block consists of the following:

   (a) An entry header of 16-bytes that contains information about the current data entry. The first 4-bytes serve as the type of the reported information, which can be recognized by the bot and the control panel. The third 4-bytes determine the length of the carried information.

   (b) A variable number of bytes that is specified in the entry header. These bytes represent one piece of the information that is transmitted within this packet.

It should be noted that the encrypted communication of the Zeus botnet is vulnerable to the RC4 keystream reuse attack because there is no Initialization Vector (IV) setup in every session, i.e., the same RC4 keystream is reused to encrypt all messages.

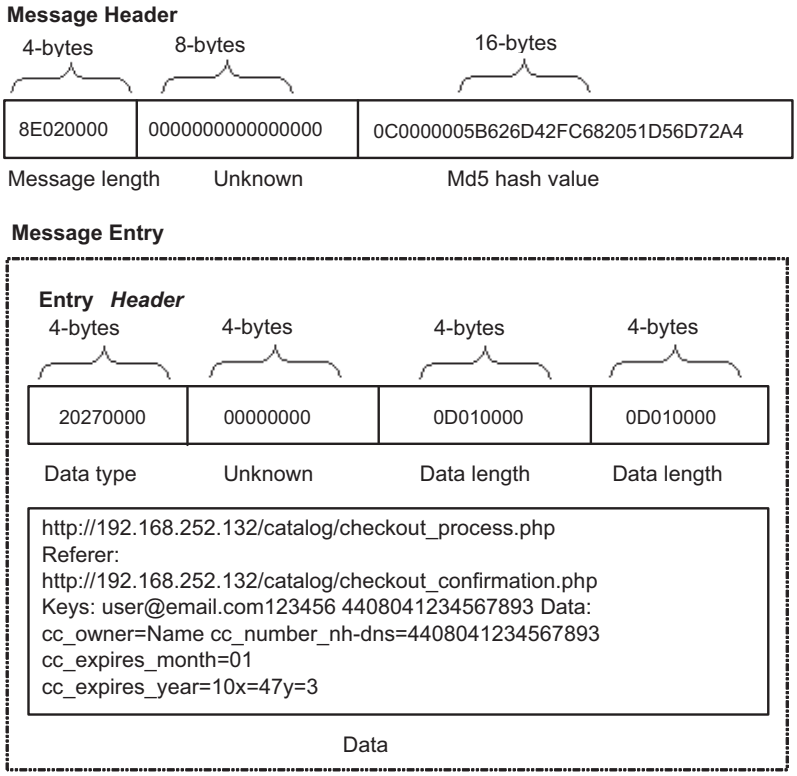


Figure 3.8: A decrypted sample message.

3.4 Conclusion

The Zeus crimeware toolkit is an advanced tool used to generate very effective malware that facilitates criminal activities. The integrated toolkit technology impedes the detection of the malware at the host level. The use of encrypted HTTP messages for C&C makes it difficult to detect any clear behavior at the network level. Moreover, the multiple levels of malware obfuscation represent a burden to the analysts to find information about the C&C servers or to generate binary signatures. In this work, we presented a detailed reverse engineering analysis of the Zeus crimeware toolkit to unveil its underlying architecture and enable its mitigation. Furthermore, we provided a breakdown for the structure of the Zeus botnet network messages.

Our analysis of the C&C communications indicates that the RC4 algorithm is used in

a poor way to encrypt these communications (keystream reuse). In addition to the knowledge of the network messages structure, we can launch active countermeasures by interacting with the botnet servers using the extracted encryption key. For example, we can inject falsified information into the botnet communication for various purposes, such as defaming the botnet business model by reducing the effectiveness of their services [111, 112]. A useful extension to our work is to use the extracted encryption key mechanism to analyze and track down the Zeus C&C servers or to defame the toolkit, e.g., by returning fake (invalid) credit card numbers.

Chapter 4

Cyber Security Intelligence Extraction from Malware Analysis

Dynamic analysis of malware samples can be used to generate useful cyber security intelligence. The reports produced during the dynamic analysis process represent an immediate source for collecting preliminary information about a given malicious threat. Since malicious networks tend to reuse their resources such as source codes, domain names and IP addresses for different purposes, the network analysis enables us to capture relations between the shared resources. For instance, cyber criminals abuse IP addresses and domain names and adopt techniques such as Fast-Flux and DGA; these leave traces, which can be backtracked by simply correlating their activities. Moreover, the structure of malicious networks manifests information about the abused resources behavior and the importance of each individual resource. In this chapter, we present a framework for extracting cyber threat intelligence from the reports that are generated during the dynamic analysis of malware samples. Our framework utilizes the extracted information from the malware dynamic analysis reports to explore the infrastructure properties of malicious networks.

The rest of this chapter is organized as follows. Our framework is described in Section 4.1. Section 4.2 explains our dataset and presents the insights produced from our framework via an experiment on real-world malware dynamic analysis reports. Finally, Section 4.3 provides concluding remarks.

4.1 Framework Description

The process of dynamic analysis monitors the analyzed malware samples and records a detailed behavioral report about various observed activities. The monitored activities include, but are not limited to, system changes, file records, registry entries, and network activities. Our framework utilizes dynamic analysis reports that are produced by analyzing malware samples in a controlled environment. The framework accepts as input these reports for each analyzed malware and produces insights that can be used in various cyber attack investigations. The main components of our framework are shown in Figure 4.1. Once the reports are generated by the dynamic analysis, the pre-processing module is responsible for extracting information from the reports and storing it in a graph database. The graph database enables us to keep track of relations between the analyzed malware samples, domain names, and IP addresses in order to conduct various network structure analyses. The database is also supported by external information that helps in localizing resources and tag activities to possible malware families. Given any cyber attack investigation, the statistical analysis provides valuable insights. The network structure analysis investigates the relations between abused resources and evaluates the importance of each individual resource to understand the key components in the malicious infrastructure. In what follows, we provide a detailed description of each component of our framework.

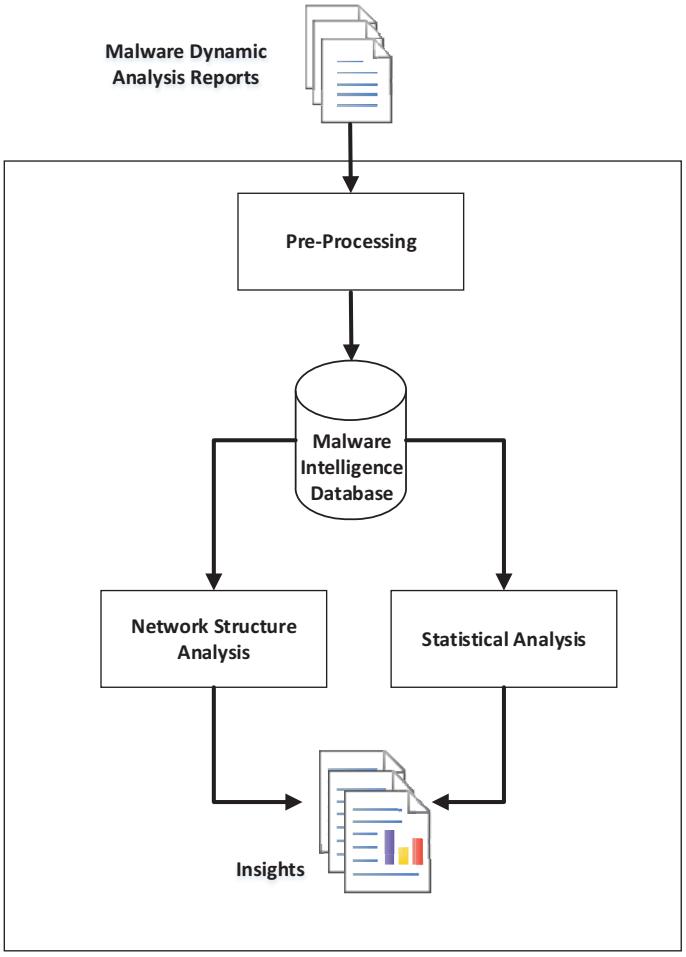


Figure 4.1: Intelligence framework overview.

4.1.1 Pre-Processing

Our framework processes malware dynamic analysis reports, which are generated in a controlled environment. Each report contains detailed information about the activities conducted by the analyzed malware sample in Extensible Markup Language (XML) format. The reported activities cover the local system activities and the network interactions. System information records any observable changes to the file system, registry keys, memory operations, and loaded libraries. The network information captures the interaction between the observed malware sample and any remote or local network resource. The sandbox system used in the dynamic analysis process decodes various network protocols, such as DNS,

HTTP, FTP, Simple Mail Transfer Protocol (SMTP), and IRC. It also stores the raw network packets for further inspection. In our framework, we focus only on the network activities; we do not evaluate local system events.

The parsing engine begins by creating a unique profile for each report using the hash value of the malware binary. The parser then extracts any possible communication activities to external resources and records all observed IP addresses and domain names. These IP addresses and domain names are used to conduct various activities by different network protocols as summarized in Table 4.1.

Category	**Extracted Information**
FTP	User name, password, FTP command, ports
HTTP	Method, URL, ports
IRC	User name, password, host name, server name, real name, nick, channel name, channel password, private message, notice message
DNS	Name server, query name, query type, query class, answer name, answer type, answer class, answer TTL, answer length, NS name, CNAME, TXT, PTR, ports
Plain connection [1]	Data sent, data received
SMTP	User name, password, mail from, recipients, ports
Downloaded files	Download URL, local file name, local file hash

Table 4.1: Example of extracted information from malware dynamic analysis reports.

Our framework utilizes additional external resources to complement the information extracted from the dynamic analysis reports. Since we are dealing with malware samples, knowledge of the malware family of each analyzed malware sample is required in order to enrich the cyber investigation process. Security experts and AV companies have long faced the challenging task of consistently naming malware families. With the number of new malware samples growing by the day, the problem of classifying malware by a common

[1] In plain connection, the underlying protocol cannot be decoded by the sandbox system.

name is becoming more and more difficult. Many attempts have been made by researchers and AV vendors [113, 114] to solve this issue; however, very little success has been shown. To overcome this problem, we propose a solution that takes into consideration all of the reported family names from AV companies, and extracts the commonly used family name accordingly. For each observed malware sample, we retrieve the family names from a public web service called VirusTotal [115], which collects different AV companies (43 vendors) scanning reports for a specific malware sample. We then extract the most frequent string from all of the reports. In the following, we summarize the process of naming a given malware sample:

1. We retrieve the naming reports from different AV companies for a given malware hash value.

2. Malware names contain strings that indicate more information about a given malware sample, such as the targeted platform (e.g., Win32), the malware version (e.g., Zeus.B), or some classification (e.g., Trojan or Adware). These strings are considered as extra information, which can be ignored while extracting the malware family name.

3. Finally, we calculate the frequencies of the remaining strings inside all of the retrieved reports from AV companies. The most common string is selected as the possible malware family name.

4.1.2 Statistical Analysis

Statistical analysis of the extracted information from malware dynamic analysis reports provides important insights, especially in the early stages of cyber investigations. Information about targeted countries, targeted organizations, suspected malware families, and possible

cyber criminals support and guide the process of taking the right steps towards a successful investigation. For instance, monitoring the daily behavior of malware samples while communicating Internet resources (e.g., domain names and IP addresses) gives an overview of the level of abuse of such resources.

Occasionally, botnets exploit or take advantage of specific network protocols. The monitoring of malware communication with remote resources helps uncover possible malicious activities. Network protocols contain detailed information that can be digested to measure the level of abuse for any protocol or service. As an example, some malicious activities can be distinguished by using special protocol configurations, which help recognize them, such as payload distribution through DNS [14] and port0 activities [30].

Cyber criminals have many targeted objectives and they operate different campaigns to increase their profit. The functionality of malicious networks can therefore differ from one to another. For example, the Waledac botnet mainly operates spam activities [116], while the Sality botnet is responsible for many scanning incidents [117]. By observing the daily activities of malware samples that belong to different malware families, we can infer the nature of the malware activities and their evolution with time. Additionally, malware families can be correlated with the abused resources, such as domains and IP addresses, to guide cyber crime investigation.

4.1.3 Malicious Networks Analysis

The malware dynamic analysis reports include many relations between different Internet resources that can be mapped to a graph. For instance, malware samples are contacting domain names that resolve to IP addresses and are used to conduct specific activities. The relation between malware samples, domain names, and IP addresses can be abstracted as a directed graph that consists of a tuple $G_{multi} = \langle M \cup D \cup IP, E \rangle$, where:

- $M = \{m_1, m_2, \cdots, m_n\}$ is a finite set of malware sample nodes
- $D = \{d_1, d_2, \cdots, d_l\}$ is a finite set of domain name nodes
- $IP = \{ip_1, ip_2, \cdots, ip_k\}$ is a finite set of IP address nodes
- $E \subseteq (M \times D) \cup (D \times IP)$ is a finite set of pairs of distinct nodes, called edges.

G_{multi} is a 3-mode network composed of three different types of actors. Multi-mode networks tend to form communities by utilizing shared properties. The overlapping properties are also used to simplify the network and focus on one actor in the network. In general, multi-mode networks can be transformed into many one-mode networks in order to apply most of the network analysis notions and compare networks. For example, a network of domain names can be formed by considering the shared IP addresses or by considering the malware instances, which access the domains. The transformation of a multi-mode network to many one-mode networks is called *projection* [118], which produces weighted networks by defining the weights as the number of common neighbors in G_{multi}. The produced weighted networks contain all of the structural information from the original network [119].

From G_{multi}, we can derive different one-mode networks that represent many aspects of the network. To limit the scope of our analysis, we focus on the following two graphs that have domain names as the main actor.

Abused domains: $G_{DM} = \langle D, E_{DM} \rangle$, where $E_{DM} \subseteq (D \times D)$. The graph G_{DM} represents the graph of domain names connected by a number of shared malware samples. A domain d_i is connected to another domain d_j when there is at least one malware sample that accessed both d_i and d_j. The graph G_{DM} can be used to quantify the abuse of domain names by malicious networks and also to group the domains that are abused by the same malicious networks.

Malicious infrastructure: $G_{DIP} = \langle D, E_{DIP} \rangle$, where $E_{DIP} \subseteq (D \times D)$. The graph G_{DIP} models the relation between domain names that are resolved to a number of shared IP addresses. A domain d_i is connected to another domain d_j when there is at least one malware sample that mapped both d_i and d_j to the same IP address. The graph G_{DIP} reports the structure of malicious infrastructure and their robustness.

Given the above two networks, we can introduce a set of metrics that help us assess the overall network structure and measure the importance of individual nodes.

Network Structure and Centrality Measures

There are several basic metrics that can be used to reveal the configuration of any given network and to quantify the relative importance of individual nodes. In what follows, we study the k-neighborhood connectivity plot, degree centrality and betweenness centrality metrics. The graph k-neighborhood connectivity plot characterizes graphs to identify the strength of relationships between nodes. The degree and betweenness centrality are used to identify the critical vertices in the graph. These metrics guide the cyber crime investigation to understand the properties and the structure of malicious networks.

K-Neighborhood Connectivity plot (KNC-plot): Given a weighted graph G, where the weights k represent the number of sharing neighbors, KNC-plot is an algorithm that measures the connectivity of G_k as a function of k [120]. More precisely, the KNC-plot is defined as a function of k that shows the decreasing size of the largest component and the increasing number of components. The connectivity of a graph provides a global understanding of the captured network and a means to study the robustness of the network. Given a weighted graph G, two nodes are k-neighbors if they share an edge with the weight of at least k. As an example, the graph G_1 contains all the nodes that share edges with weights greater than or equal to one. This graph is considered

to be a very highly connected graph. Similarly, G_2 is defined on the same nodes, but it might be less connected. When increasing the degree of connectivity k, the graph will become increasingly sparse and less connected until it becomes completely disconnected. The analysis of different G_k graphs enables us to understand the structure of sharing resources within malicious networks.

Degree Centrality: Node degree is defined as the number of edges that are shared with neighboring nodes [121]. In a weighted graph, where edges have weights, nodes have an important metric that measure their strength; this is computed for each node by accumulating all the weights for all the direct neighboring nodes. The degree centrality represents the connectivity of a node i in the network, which can be defined by utilizing the node degree and strength as follows [122]:

$$C_D(i) = \left(\sum_{j=1}^{n} a_{ij} \right)^{1-\alpha} \times \left(\sum_{j=1}^{n} w_{ij} \right)^{\alpha} \tag{4.1}$$

where $a_{ij} = 1$ when node i and node j are direct neighbors and $a_{ij} = 0$ otherwise. The term w_{ij} denotes the weight between the two adjacent nodes i and j. The term α is a positive parameter that can be used to balance the importance between the node degree and the strength, while calculating the degree centrality.

Betweenness Centrality: Betweenness centrality measures the importance of a node i being in the shortest path between two other nodes. When a node is included in many shortest paths between other nodes, it has more control over the network by serving as a bridge between nodes. This node can be considered as an intermediate node between different communities. The betweenness is defined as the ratio of all shortest

paths passing through node i as follows:

$$C_B(i) = \sum_{k \neq i \neq j \in N} \frac{\sigma_{kj}(i)}{\sigma_{kj}} \tag{4.2}$$

where σ_{kj} is the sum of all shortest paths between node k and node j, and $\sigma_{kj}(i)$ is the number of shortest paths that pass through node i to connect node k and node j.

The calculation of the shortest paths between nodes in weighted graphs is achieved via Dijkstra's algorithm [123] by considering the inverse of the weights between nodes as the cost of including each edge [122].

The centrality measures discussed above, defined at the node level, can be extended to reflect the graph centrality as follows:

$$C_G = \sum_{i=1}^{n} (C(n^*) - C(n_i)) \tag{4.3}$$

where n is the number of nodes within the graph, $C(n_i)$ is the centrality value of node i, and $C(n^*)$ is the largest centrality value in the graph. The group centrality enables us to compare different communities. To compare the group centrality of different graphs, the C_G value of each graph must be normalized with the maximum possible sum of differences of node centralities calculated by Equation 4.3. The normalization values for degree and betweenness are given by $(n-1)(n-2)$ and $(n-1)$, respectively [121].

4.2 Experimental Results

In this section, we present some of the intelligence that we extracted from the malware dynamic analysis reports. We discuss some of the insights that are utilized from the statistical analysis and the malicious network analysis. During our analysis, we use a dataset of

dynamic analysis reports provided by ThreatTrack Security Inc [124] that spans from the 1^{st} of January 2014 to the 4^{th} of March 2014. In Table 4.2, we show some facts about the dataset. Our framework utilizes the Neo4j graph database [125] to store the extracted information. We choose a graph database because it supports most of the functionalities and algorithms of graph theories while maintaining flexible and scalable storage capabilities.

Number of reports	1,573,214
Number of domains	49,375
Number of IP addresses	44,746
IRC connection	288,346
DNS connection	96,265
Plain connection	1,541,949
SMTP connection	458
HTTP connection	1,165,904
Downloaded files	25,585

Table 4.2: Statistics of the dataset used to evaluate the framework.

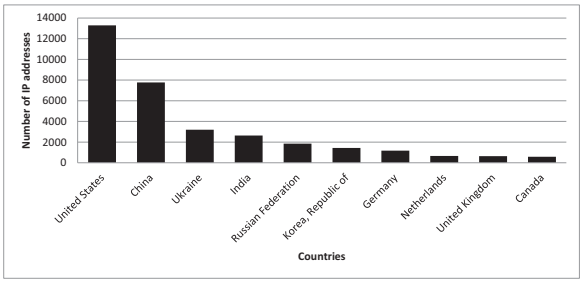
4.2.1 Statistical Insights

The first step in investigating any cyber crime attack is to collect statistics about the incident. Our framework analyzes the malware dynamic analysis reports on a daily basis and populates the Neo4j graph database with the extracted information. While populating the database, we also include some external information to support the knowledge about malicious activities. For instance, each observed IP address is correlated with the Maxmind databases [126] to retrieve the geographic location and the ISP responsible for each IP address. In addition, the framework also provides information about domain name registration, such as owner information, by correlating the observed domain names with the public WHOIS records [127]. Figure 4.2 shows an example of the geolocation of some malicious networks resources, while Figure 4.3 illustrates an example of statistics that are produced by our framework.

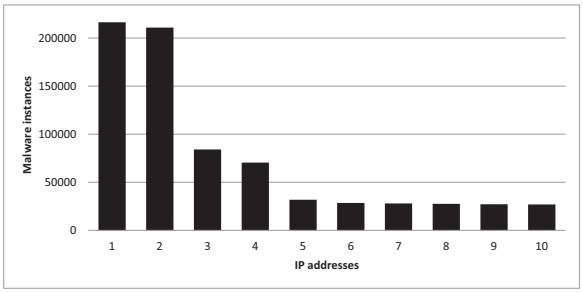
Figure 4.2: Geo-locating malicious networks.

Our framework provides complete information about connections conducted by IP addresses. For example, FTP connections may point to drop or deposit locations that may contain stolen credentials. On the other hand, IRC connections reveal the plain C&C communication between bots and C&C servers. In addition, the protocol information helps us investigate other abuses such as those involving the DNS protocol, which will be addressed in Chapter 6.

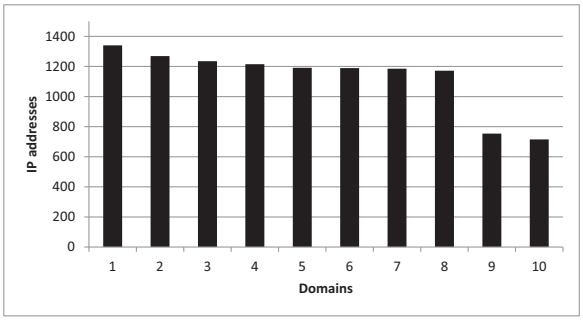
Using the observed malware samples, we detect their corresponding families from VirusTotal [115] by leveraging our proposed approach for naming the malware families as discussed in Section 4.1.1. We recognize more than 3500 malware families, which corresponds to 40% of the observed malware samples. The remaining malware samples were either not recognized by VirusTotal, or no common name exists. In Figure 4.4, we show the top 20 active malware families that are recognized during analysis. Such insight can help us estimate the spread of different malware families. Certain malware families are specialized in conducting specific activities such as spamming (e.g., Virut), scanning (e.g., Sality), or fake software downloaders (e.g., LoadMoney). Through analyzing the distribution of malware families, we can learn the current trend of malware activities.



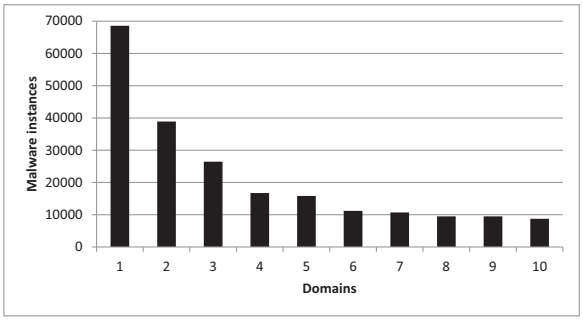
(a) Top 10 targeted countries by malware samples.



(b) Top 10 IP addresses that received connections from malware samples.



(c) Top 10 domain names that mapped to different IP addresses.



(d) Top 10 domains contacted by different malware samples.

Figure 4.3: Example of cyber intelligence extracted using the developed framework.

4.2.2 Malicious Networks Analysis

Studying the structure and properties of malicious network resources provides important insights for a cyber crime investigation. In our analysis, we consider two different graphs: abused domains G_{DM}, and malicious infrastructure G_{DIP}. Given these two graphs, we analyze their structural properties, investigate the importance of individual network resources, and compare the networks centrality. In Table 4.3, we illustrate some of the general properties of the two graphs.

Abused Domains Networks

In this section, we explore domains abused by observed malware samples. During our experiments, we extracted 1,402 different networks composed from 19,572 domain names.

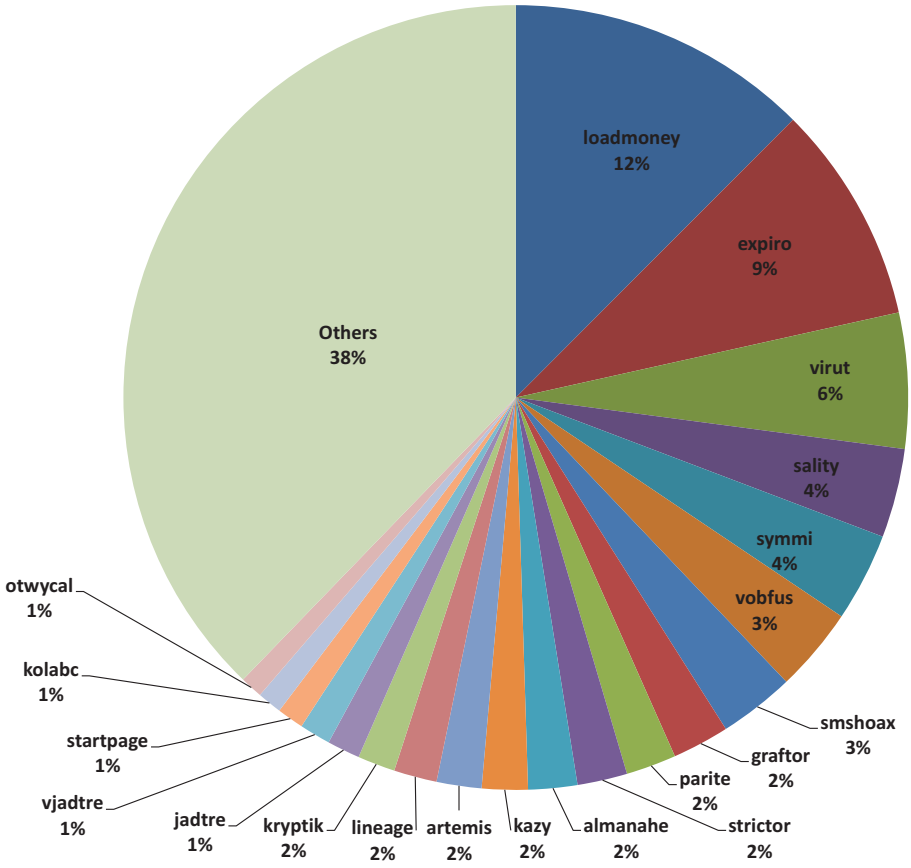


Figure 4.4: Top 20 observed malware families.

Within these networks, two domain names are connected with each other when at least one malware sample has visited both domain names, and the strength of their relation is weighted by the number of such malware samples. Studying the connectivity of the abused domains by the KNC-plot enables us to analyze the strength of networks and understand their level of involvement in malicious activities. In Table 4.4, we show some network examples with the value of k for which the size of the largest subgraph in the network is 50%, 25%, 10% of the original network. Some networks may contain many domains (e.g., ID = DM_1), but are weakly connected. This kind of network consists of domains that are not largely abused by the malware community, or are part of domains randomly probed by malware samples. In contrast, small networks can be very heavily abused in malicious

	Abused domains	Malicious infrastructure
Number of domains	19,572	13,986
Number of networks	1,402	2,500
Average network size	13.9	5.6
Largest network size	14,671	3,148
Average degree	23.3	44.3
Average strength	242.1	105.7

Table 4.3: Abused domains and malicious infrastructure network statistics.

activities (e.g., network ID = DM_2, DM_3, DM_4). Networks might contain a small number of domains and be equally abused by groups of malware samples (e.g., ID = DM_5, DM_6). These networks hold domains that are essential for operating the malicious network. For instance, malware samples are normally configured to visit certain domain names to reach the malicious infrastructure. An increase in the number of malware samples visiting a specific network indicates that the associated domains are correlated (i.e., targeted) by malware community. In the KNC-plot, the domains that remain in the last connected component of the network as k increases are considered as the most abused domains in their network.

ID	Network size	50%	25%	10%	Largest k value
DM_1	14,671	2	5	15	21810
DM_2	23	1,755	2,200	2,208	2246
DM_3	24	6	28	31	45
DM_4	49	4	12	24	44
DM_5	28	50	51	0	52
DM_6	10	145	0	0	165

Table 4.4: Value of k at which the largest subgraph in each network of G_{DM} represent 50%, 25%, and 10%.

In Figure 4.5, we demonstrate an example of the KNC-plot of an abused network DM_4. As indicated in Table 4.4, 10% of the domain names in the network have at least 24 malware samples in common. These domains hold larger connectivity (more abused) compared with the other domains in the network. On the other hand, Figure 4.6 illustrates an example of a network where the underlying domains are equally important in preserving

its connectivity.

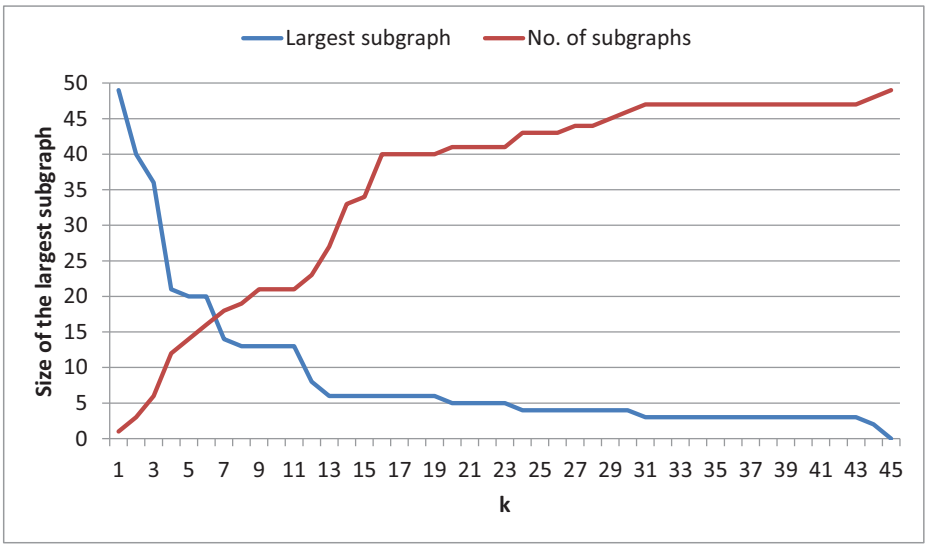


Figure 4.5: Example of KNC-plot for the abused domains network DM_4.

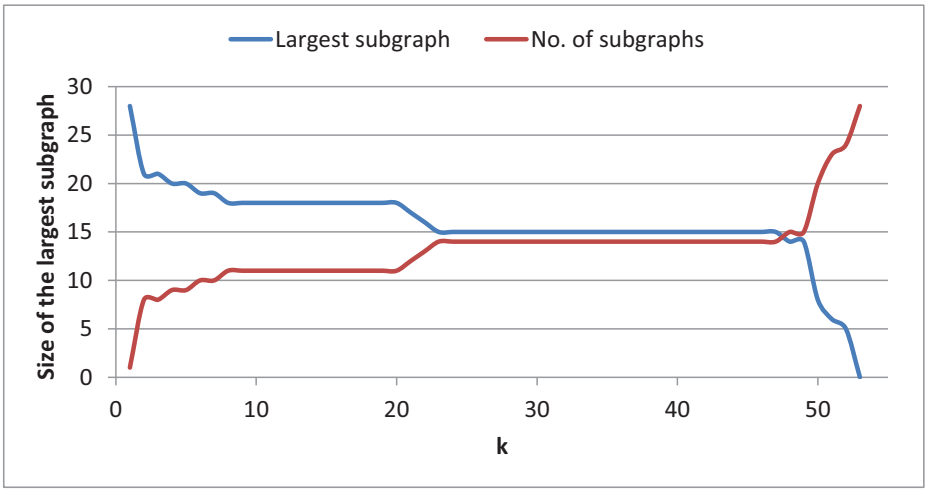


Figure 4.6: Example of KNC-plot for the abused domains network DM_5.

After studying the structure of abused domains networks, we investigate different metrics to evaluate the importance of each domain in the network. In Table 4.5, we present the group centrality values (calculated using Equation 4.3) for some of the abused networks. In general, the group centrality of a network reflects the centrality of its nodes. The degree centrality is an indicator of the existence of domains that are heavily abused by the malware

community. The two networks DM_2 and DM_6 have a high degree centrality, which supports the KNC-plot statistics previously discussed. Some networks might contain sub networks that are connected by intermediate nodes that can be captured by the betweenness centrality measure. For disturbing any network, good candidates to start with would be the domains that have high betweenness centrality. The group betweenness centrality is used to capture the existence of intermediate nodes in any network such as DM_1, DM_3, DM_4, and DM_6.

Network ID	Network size	Degree	Betweenness
DM_1	1,4671	0.0029	0.57
DM_2	23	0.52	0.000069
DM_3	24	0.083	0.4
DM_4	49	0.048	0.43
DM_5	28	0.076	0.098
DM_6	10	0.70	0.54

Table 4.5: Group centrality for different abused domains network G_{DM}.

Malicious Infrastructure Networks

In this section, we explore malicious infrastructures that consist of domain names connected by shared IP addresses. Our analysis reveals 2,500 different networks that contain 13,986 domain names. Two domain names are connected with each other when there is at least one shared IP address between both of them, and the strength of their relation is weighted by the number of such shared IP addresses.

The connectivity of malicious infrastructures helps in understanding the nature of the relationship between domains. Malicious networks tend to replicate their resources in order to increase their availability, which strengthens their resiliency against take down operations. This behavior is achieved by a technique called Fast-Flux, which simply assigns multiple IP addresses for domain names. Using the KNC-plot, we can analyze the strength of the relationship between domains and investigate the level of being a Fast-Flux network.

Table 4.6 shows some network examples with the value of k for which the size of the largest subgraph in the network is 50%, 25%, 10% of the original network. A weak network is one that requires less k values to disturb the connections between its domains, such as DIP_1 and DIP_4. On the contrary, heavy networks that share many IP addresses between their domains, such as network DIP_2 and DIP_3, are more resilient to IP address take down operations. On occasion, malicious networks secure all of the used domains with the same IP address to operate the network (e.g., DIP_5, DIP_6). Other networks maintain strong ties between all of the abused domains, such as DIP_7. In general, the core domains are the ones that remain in the last connected component of the network as k increases. The larger the k value required to dissolve the network, the more difficult it becomes to take down the network.

ID	Network size	50%	25%	10%	Largest k value
DIP_1	3,148	2	3	5	480
DIP_2	233	20	65	161	951
DIP_3	39	29	40	80	85
DIP_4	116	4	7	12	20
DIP_5	12	8	148	0	148
DIP_6	16	11	45	0	48
DIP_7	11	27	0	0	32

Table 4.6: Value of k at which the largest subgraph in each network of G_{DIP} represent 50%, 25%, and 10%.

ID	Network size	Degree	Betweenness
DIP_1	3,148	0.00075	0.49
DIP_2	233	0.0279	0.40
DIP_3	39	0.037	0.049
DIP_4	116	0.0065	0.34
DIP_5	12	0.46	0.74
DIP_6	16	0.079	0.061
DIP_7	10	0.083	0.2

Table 4.7: Group centrality for different malicious infrastructure networks G_{DIP}.

After studying the structure of malicious infrastructure networks, we investigate different measures to evaluate the importance of each domain in the network. In Table 4.7, we present the group centrality values (calculated using Equation 4.3) for some of the malicious infrastructure networks.

The degree centrality is considered as an estimator of the level of existence of Fast-Flux domains in the network. For instance, the network DIP_5 has a high degree centrality, which confirms the network structure properties indicating that this network is easy to be disturbed by taking down the domains with high degree centrality .

Periodically, different malicious networks connect with each other and with some domains to share information (e.g. exploits or infection modules). Betweenness centrality helps uncover the domains that play a role in connecting different malicious networks. In order to take down any network, good candidates to start with would be the domains that have high betweenness centrality. An example of networks that contain more intermediate domains are the networks DIP_5 and DIP_2.

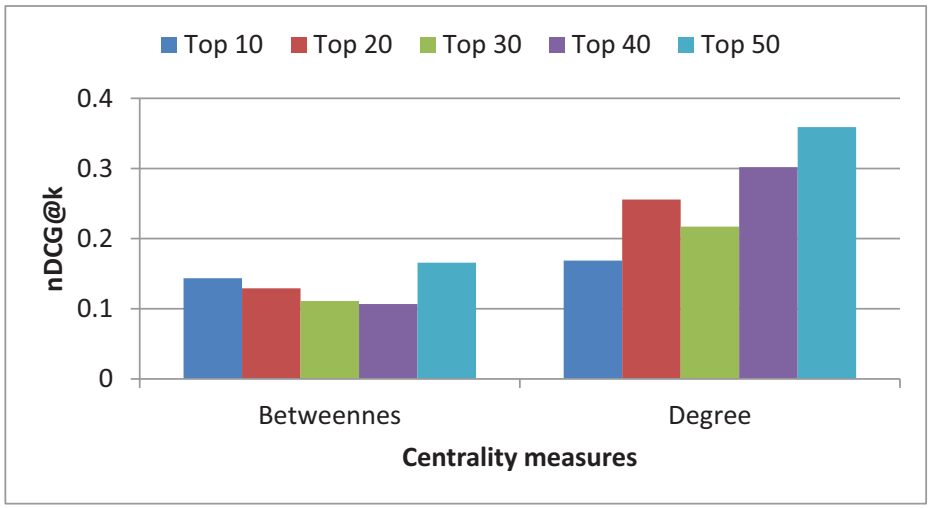


Figure 4.7: The nDCG@k evaluation measure values for degree and betweenness centrality measures for abused domains networks G_{DM}.

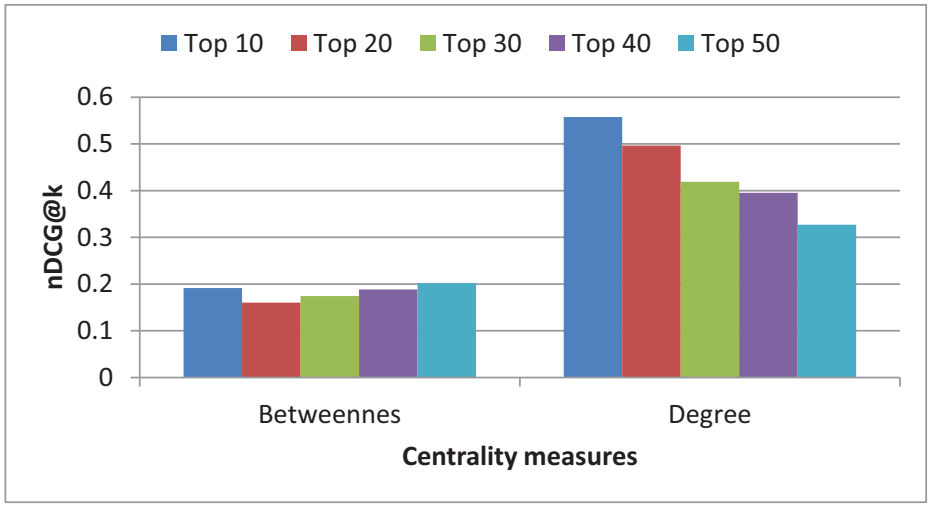


Figure 4.8: The nDCG@k evaluation measure values for degree and betweenness centrality measures for malicious infrastructure networks G_{DIP}.

Since the centrality metrics quantifies the importance of domains based on their relationships, we evaluate how these measures confirm public knowledge about the maliciousness of these domains. Since these measures order the importance of domains, we correlate them with graded maliciousness scores from the WOT reputation system [128]. The evaluation is judged by the nDCG measure explained in Section 2.1.3. In Figure 4.7, we show the evaluation of all centrality measures to rank the top domains in the abused networks. On a similar note, Figure 4.8 illustrates the evaluation of centrality for the malicious infrastructure networks. In both graphs, the degree centrality outperforms the betweenness centrality because it relies on the frequencies of being used by malware communities or the level of Fast-Flux activity. However, the betweenness centrality quantifies only the importance of the position of the domains in the network and does not determine the maliciousness of the domains. From this observation, we learn that malware interaction with domains and IP addresses shows promising indicators of domain involvement in malicious activities.

4.3 Conclusion

In this chapter, we developed a framework to extract and build intelligence from malware dynamic analysis reports. The framework produces statistical insights that can be used in the early stages of cyber crime investigations. Moreover, it analyzes the configuration of abused networks and quantifies the relative importance of domain names. Our framework is evaluated on two months' feed of malware dynamic analysis reports, which contains an average of 6000 reports per day. Our evaluation reveals that studying the malware interaction with domain names can lead to deeper insight into malicious activities.

Chapter 5

Ranking the Severity of Domain Names based on Malware Behavior

In this chapter, we study the problem of assigning severity scores to malicious domains, i.e., domains that are abused by the malware underground community. Our main goal is to automatically assign a high severity score to domains that are involved in severe malicious activities, such as C&C servers and drop locations. The severity scores enable dynamic domain name blacklists to be more efficient in prioritizing how to deal with cyber crimes. For example, with existing blacklists, it is unclear, which domain names have been involved in more malicious activities compared to others. With dynamic severity scoring, our goal is to determine the level of maliciousness for each domain name. We formulate the problem using a system inspired by the emerging field of reputation systems.

Our work is based on the observation that behaviors of malicious networks are reflected in their interactions with domain names. In short, malware behavioral features have distinct characteristics that reveal the nature of their malicious interaction with domain names. By identifying and observing these features, our system can assign appropriate severity scores to malicious domains with, which these malware interact. Our system uses

dynamic malware analysis reports generated in a controlled environment from malware samples provided by ThreatTrack Security Inc [124]. Through analysis of such reports, our system assigns severity scores to existing and new domain names, thereby enriching the blacklists with more information about malicious domain names and their behaviors.

Previous works, discussed in Section 2.2.3, mainly focus on (binary) reputations and blacklisting of domain names. We can distinguish our proposed system from other reputation studies [79, 80, 81] by the following key points: first, our work proposes a novel approach to evaluate the reputation score of a given domain name based on the malware behavior analysis by quantifying and measuring the level of abuse. Second, our severity system updates the severity scores based on observed malware behavior and requires no training period. Third, our system benefits from the flexibility of generating customized severity scores by adjusting the weights of the features based on the investigator's priorities. Finally, our system provides a behavioral pattern for domain names that can be used to recognize domain name abuse activities.

The remainder of this chapter is organized as follows. Our system is described in Section 5.1. Section 5.2 explains our dataset and Section 5.3 demonstrates the effectiveness of our proposed system via an experiment involving real-world malware dynamic analysis reports. We discuss the results and limitations of our work in Section 5.5. Finally, Section 5.6 provides our concluding remarks.

5.1 Severity Ranking System

The goal of our system is to assign severity scores to abused domain names. For any given domain name, a high severity score is assigned if the domain is involved in extensive malicious activities, such as C&C servers and drop locations.

Our system uses malware dynamic analysis reports as a main source of information.

These reports contain detailed information about domain names and their resolved IPs. For each domain name, behavioral reports provide information concerning all communication between a malware instance and domains through different protocols, such as HTTP, FTP, IRC, and SMTP. Our dynamic malware analysis report database is updated on a daily basis using a live feed of malware samples obtained from ThreatTrack Security Inc [124].

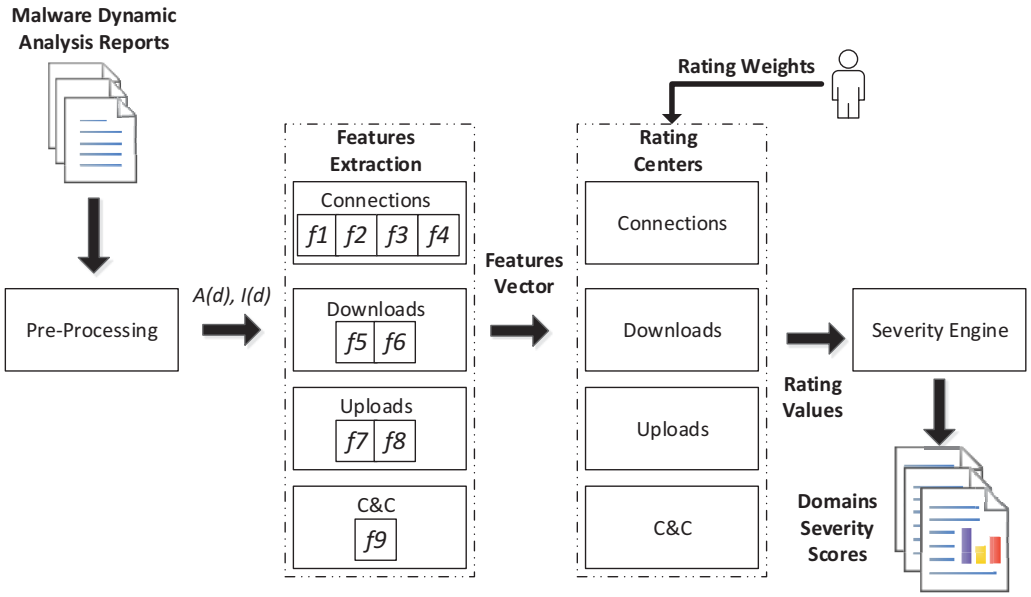


Figure 5.1: Overview of the proposed severity ranking system.

5.1.1 System Overview

In reputation systems, the main goal is to collect and combine feedback about participants' behavior. The participants convey their opinions according to a specific rating model that results in an integrated rating vector for each participant. Given a set of domain names, a rating value is associated with each domain name based on a set of malware behavioral features over a given time period. The severity system differs from traditional reputation systems in the collection of feedback. In contrast to the traditional reputation system based on subjective opinions, the rating values in our approach are implicit values inferred from

the interactions between domain names and malware samples.

Prior to discussing our proposed domain name severity system, we first introduce some basic notation. A malware sample m contacts a domain name d, represented by a set of labels separated by dots.

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of domain names, and $M = \{m_1, m_2, \ldots, m_p\}$ be the set of analyzed malware samples. Given a domain name d, we define $A(d) \subset M$ to be the set of all malware samples that contacted d. We denote $I(d)$ as the set of all IP addresses resolved from $d \in D$.

Definition 5.1. A domain name d is a *severe domain* when there is evidence that d or $I(d)$ are associated with known malicious activities. The evidence of maliciousness is recognized by the malware interaction with the domain d. The more evidence observed toward d, the higher the severity considered.

As shown in Figure 5.1, the severity of a domain name d is determined by the following steps. First, we observe the most recent set $A(d)$ of malware samples that contacted d. Then, we aggregate the set $I(d)$ for all IP addresses resolved from all possible domain name labels. Subsequently, we measure four different groups of features for each domain d. The first group contains connection-based features, which quantify the number of connections that are made by the set $A(d)$. The second group consists of download-based features, which measure the level of download activities conducted by the set $A(d)$. The third group includes upload-based features to capture any information leakage conducted by $A(d)$. The last group contains C&C-based features to quantify the level of C&C activities.

Once the above sets of features are extracted, we feed them to the rating centers, where they are transformed into a set of rating values for each domain name d. The severity engine then takes the rating values for each domain name and assigns the severity score accordingly. We now explain the details of features extraction and how the rating centers

convert them to rating values, followed by how the severity engine computes the severity score of domain names.

5.1.2 Features Extraction

Domain names are considered as pointers to various Internet services that can be abused by malicious networks to operate various activities. Our features are inspired by the life cycle of a typical botnet that begins with an ordinary infected machine. Normally, the infected system first launches malicious activities locally, followed by attempts to communicate with the botnet infrastructure. The infected machine then updates itself with the latest malware binaries. The acquired binaries instruct the infected system to communicate with the rest of the botnet infrastructure. The infected system is typically directed to download the updates through a variety of file transfer protocols FTP and HTTP. By utilizing the life cycle of malicious networks, we can extract some useful features that measure the level of abuse for the involved domains. In what follows, we explain the set of features extracted by our system.

Connection-based Features

The initial phase of botnet infection involves calling home infrastructure. The connection-based features describe the frequency of accessing a domain d by different malware samples. A large number of connections established by malware samples to a domain d implies high involvement of the domain d in malicious activities. In order to measure the number of visits to a domain d, we observe features that capture successful connections to different services on the set $I(d)$. During our feature extraction, we evaluated the number of FTP, HTTP, IRC, and other connections to non-standard ports (denoted by f_1, f_2, f_3, and f_4 respectively).

Download-based Features

A crucial stage in every malicious network involves updating the modules or the functionalities of the malware. We use the download-based features to measure the process of updating information for the infected systems. A domain name is considered as a *data provider* when it serves as a direct or an intermediate step to host data downloaded by a malware sample. This set of features thus reflects the severity of being a *data provider* in malicious networks. Given a domain name d, we extract the number of conducted downloads using HTTP GET (f_5) and FTP RETR (f_6) operations.

Upload-based Features

Certain malicious networks collect information from victims for different purposes. Upload-based features capture the existence of information leakage from infected hosts to domain names. A domain name that receives stolen information from an infected machine is called a *drop location*. By counting the number of HTTP POST (f_7) and FTP STOR (f_8) operations, we can measure the level of the domain's participation in such activities.

C&C-based Features

C&C servers are considered as the nerve system of malicious networks. C&C-based features reflect the existence of any instruction that is transmitted between infected hosts and domain names. In our system, f_9 counts the number of exchanged C&C commands using the IRC protocol between malware samples and the domain under consideration.

5.1.3 Rating Centers

In reputation systems, feedback collected from raters reflects participant behaviors. In our context, features *(f1-f9)* are fed to dedicated *rating centers* to generate rating values. For

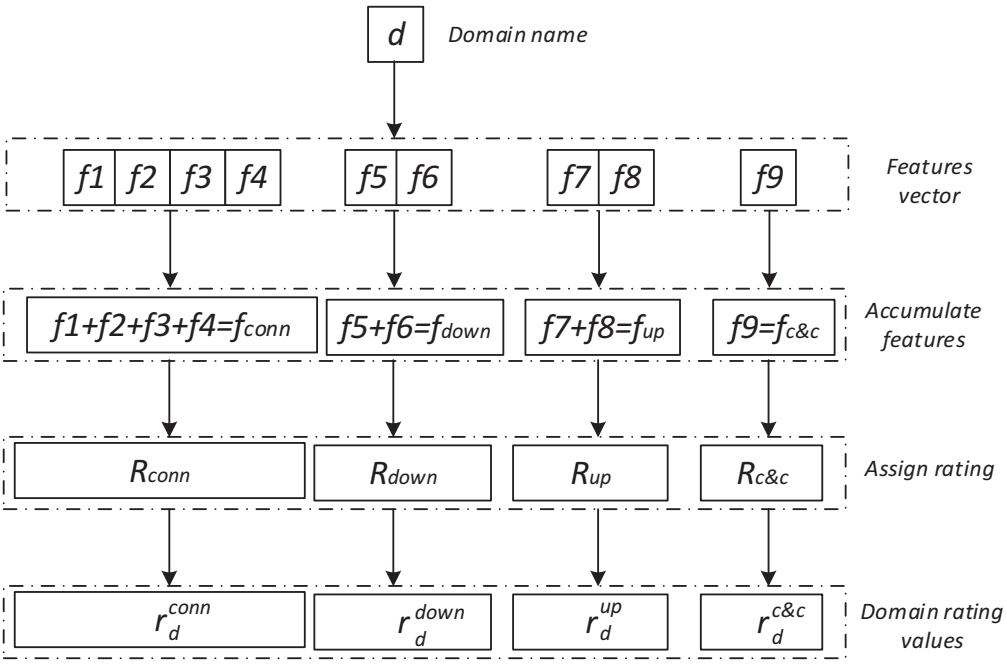


Figure 5.2: Rating center functionalities.

any given period of time, rating centers produce rating values that reflect how each domain behaves according to the collected set of features as depicted in Figure 5.2. First, the rating centers store an accumulation of the features for each domain name d. The rating centers then sort all of the observed domain names in descending order based on the calculated features. Finally, rating centers arrange domain names into different rating levels. Based on Definition 5.1, these levels reflect the position of a domain name with respect to all observed domain names based on considered features. The assigned level for a domain name is considered as the inferred rating value, which can be used in the severity score computation.

Rating Freshness. Domain names can change their involvement in malicious activities from one day to another. The severity score should therefore reflect this behavior by prioritizing attention to recent rating values. Rating values are usually aggregated by simple vector addition over a pre-specified time interval. Previous rating values can be aged by a

freshness factor $\lambda \in [0,1]$ to give recent rating values more importance over the old ones. When λ has low value, the old ratings have less influence on the severity scores, and vice versa. Let $\vec{r_{d,t}}$ be the aggregated rating values at time t; we then define the accumulated rating values with the freshness value λ at the time $t+1$:

$$\vec{r}_{d,(t+1)} = \lambda.\vec{r}_{d,t} + \vec{r}_{d,(t+1)}, \quad where \quad 0 \leq \lambda \leq 1. \tag{5.1}$$

5.1.4 Severity Engine

The severity engine is responsible for determining the maliciousness level of a given domain. To accomplish this goal, the severity engine receives rating values from designated rating centers based on the malware features. Our goal is to derive a severity function that satisfies the following design criteria: first, the severity function should be dynamic and updated over time. Second, the severity function should give more attention to the recent rating values over the old ones. Third, the severity function should be customizable to serve the purpose of a cyber crime investigation process.

Our severity engine uses Multinomial Bayesian systems (Dirichlet reputation systems) [129]. Dirichlet reputation systems provide a strong and sound mathematical model for addressing our design criteria. It also allows for multinomial rating levels that are used to represent feedback about domain names. Let $L = \{l_1, \ldots, l_q\}$ denote the set of possible q rating levels. A domain name d receives a rating value $r_d^c \in L$ from each of the four rating centers c as shown in Figure 5.2. Let $\vec{r_d^c} = (r_d(1), r_d(2), \ldots, r_d(q))$ denote the rating of center c corresponding to domain d, where $r_d(i) = 1$ if the rating level i is chosen by c and $r_d(i) = 0$ otherwise.

Since the extracted features constitute the source of the rating values, we can place

emphasis on a subset of the features for our severity score calculations. Let w_c be a weighting factor that reflects the investigator's interest in each set of features. The accumulated rating values from c rating centers is defined as:

$$\vec{R}_d = \sum_c w_c(\vec{r_d^c}) \tag{5.2}$$

Definition 5.2 (*Severity Score Vector*)**.** The severity score vector S_d of a domain d with rating values $\vec{R}_d$, as specified in Equation 5.2, and q rating levels in L is defined as [129]:

$$\vec{S}_d : \left(S_d(i) = \frac{R_d(i) + C\alpha_d(i)}{C + \sum_{j=1}^{q} R_d(j)} \quad , i = 1 \ldots q \right) \tag{5.3}$$

The updated (*posterior*) severity score is initialized using an a *priori* severity score (α_d) to reflect previous expert knowledge about each domain name. The new rating values (R_d) update the score based on the extracted features. C denotes a priori constant that weighs the importance of considering previous information about domain names while updating the new score. When a large value of C is chosen, new ratings have less impact on the current scores.

Equation 5.3 represents the severity score in q different probability values for each $l_q \in L$. In order to represent the severity score with a single value for concise representation [55], we assume a given rating level l_i has point value that reflects its importance between the other levels $\vec{V} = (v(l_1), v(l_2), \ldots, v(l_q))$, where $v(l_i) = \frac{i-1}{q-1}$. The severity score vector $\vec{S}_d$ is then multiplied by the corresponding $\vec{V}$ and accumulated as follows:

$$S_d = \sum_{i=1}^{q} v(i) S_d(i) \tag{5.4}$$

Since the severity system is based on updating a priori information, each fresh domain

name d has to be initialized with a score that is defined by the common base rate vector $\vec{\alpha}$. The base rate injects the a *priori* knowledge about domain names. When the system is bootstrapped, a default base rate $\alpha_q = \frac{1}{q}$ is used.

The base rate vector can be updated dynamically based on the severity scores observed by all domain names. This vector reflects certain information about a domain name with additional bias to either low or high severity scores. To calculate the dynamic base rate at a specific time t, we aggregate all rating values for the observed domains as follows:

$$\vec{R}_t = \sum_d \vec{r}_{d,t} \tag{5.5}$$

We can then calculate the dynamic base rate for time period $t+1$ using Equation 5.3.

To explain the proposed approach, assume that we have 10 domains for which to evaluate their severity throughout the first two days as shown in Table 5.1 and Table 5.2. We initialize the system by setting $q = 5$, $C = 5$, $\alpha = 1$ with dynamic base rate, $\lambda = 0.5$, and $w_c = 1$.

In the first day, d_6, d_5, d_3, and d_6 are the most abused by malware samples to conduct download, upload, C&C, and successful connection activities respectively. Given the features for the observed domains, rating values are inferred based on a scale of five levels, where l_5 denotes the most abused level. For example, the domain d_1 conducted low (l_1) download activities, medium-low upload and successful connection activities, and medium C&C activities. Each rating level has an individual score value that is calculated by Equation 5.3. Finally, the score estimation (S_d) is computed by Equation 5.4, which is used to produce the ranked list of domain names. In fact, the domains d_5, d_6, and d_4 occupy the top three positions in the first day. As we monitor the domains throughout the second day, d_3 has become inactive and a new domain (d_{11}) is observed. The newly observed domain (d_{11}) integrated into the system by taking the community base rate while receiving current

rating values, which qualify it to be in the 6^{th} position. On the other hand, d_3 has been penalized for being inactive, which is reflected by discounting the scores with λ. Due to the drop in activity of the d_5 domain, it has been downgraded to second place in the ranked list; since d_6 continues to receive more high rating values, it qualifies to be at the top of abused domains.

	Features				Rating					Score						
Domains	F_{down}	F_{up}	$F_{c\&c}$	F_{conn}	l_1	l_2	l_3	l_4	l_5	l_1	l_2	l_3	l_4	l_5	S_d	Rank
d_1	10	20	10	50	1	2	1	0	0	0.22	0.33	0.22	0.11	0.11	0.39	5
d_2	3	1	5	10	3	1	0	0	0	0.44	0.22	0.11	0.11	0.11	0.31	7
d_3	5	8	21	30	2	1	0	0	1	0.33	0.22	0.11	0.11	0.22	0.42	4
d_4	56	32	1	98	1	0	0	3	0	0.22	0.11	0.11	0.44	0.11	0.53	3
d_5	34	52	13	130	0	0	1	1	2	0.11	0.11	0.22	0.22	0.33	0.64	1
d_6	73	34	5	150	0	1	0	1	2	0.11	0.22	0.11	0.22	0.33	0.61	2
d_7	2	4	5	30	2	2	0	0	0	0.33	0.33	0.11	0.11	0.11	0.33	6
d_8	32	4	6	50	1	2	1	0	0	0.22	0.33	0.22	0.11	0.11	0.39	5
d_9	2	4	6	29	3	1	0	0	0	0.44	0.22	0.11	0.11	0.11	0.31	7
d_{10}	4	4	4	25	4	0	0	0	0	0.56	0.11	0.11	0.11	0.11	0.28	8

Table 5.1: Running example of ranking the severity of domain names: first day.

	Features				Rating					Score						
Domains	F_{down}	F_{up}	$F_{c\&c}$	F_{conn}	l_1	l_2	l_3	l_4	l_5	l_1	l_2	l_3	l_4	l_5	S_d	Rank
d_1	3	16	2	22	3	0	1	0	0	0.42	0.24	0.24	0.05	0.05	0.27	8
d_2	9	5	9	26	3	0	1	0	0	0.61	0.15	0.14	0.05	0.05	0.2	10
d_3	0	0	0	0	0	0	0	0	0	0.167	0.11	0.06	0.06	0.11	0.21	9
d_4	90	20	8	140	0	0	2	1	1	0.15	0.05	0.23	0.43	0.14	0.6	3
d_5	20	40	10	110	0	1	2	0	1	0.05	0.14	0.33	0.15	0.33	0.64	2
d_6	65	40	10	190	0	0	1	1	2	0.05	0.15	0.14	0.24	0.42	0.71	1
d_7	10	30	9	50	1	1	1	1	0	0.33	0.33	0.14	0.14	0.05	0.31	5
d_8	20	29	15	80	0	1	1	1	1	0.15	0.33	0.24	0.14	0.14	0.45	4
d_9	10	12	16	59	1	2	0	0	1	0.43	0.33	0.05	0.05	0.14	0.29	7
d_{10}	7	9	1	20	3	1	0	0	0	0.71	0.14	0.05	0.05	0.05	0.15	11
d_{11}	2	4	17	30	3	0	0	0	1	0.56	0.14	0.05	0.08	0.19	0.3	6

Table 5.2: Running example of ranking the severity of domain names: second day.

5.2 Data Collection and Analysis

The basic source of data for our dynamic severity system is the malware dynamic analysis reports provided by ThreatTrack Security Inc [124]. Each report summarizes all of the behavioral activities conducted by a single malware sample in a controlled environment. The reported behavioral activities cover the system level and the network interactions. We

analyzed an average of 6,000 distinct reports of different malware families on a daily basis as shown in Figure 5.3. Our database collected over 14 million unique reports during the period of January 1^{st}, 2013 to November 16^{th}, 2013.

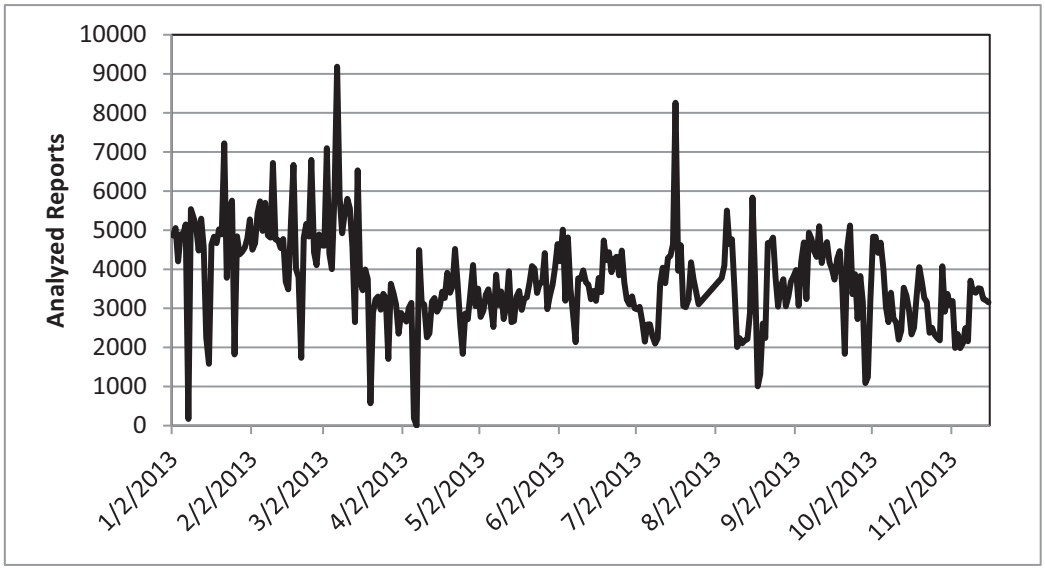


Figure 5.3: Number of analyzed malware samples.

By simple measurement of our dataset, we leverage important insights from our selected behavioral features. During our experiments, malware samples contacted between 200 to 1,000 unique domain names per day, which accumulated to roughly 132,000 domains with an average of 11% new domains per day as depicted in Figure 5.4. Since malicious networks are recycling domain names, it is necessary to monitor and observe their actions on a daily basis to limit and minimize domain recycling.

Reusing domain names in malicious activities does not prevent the use of fresh or new domains that appear for short periods of time. Figure 5.5 shows the distribution of active days for observed domains.

Figure 5.6 shows that domain names are mostly contacted by malicious networks to download updates to infected machines after successful connection. Leaking information from the victim is the second most performed action when contacting malicious domains.

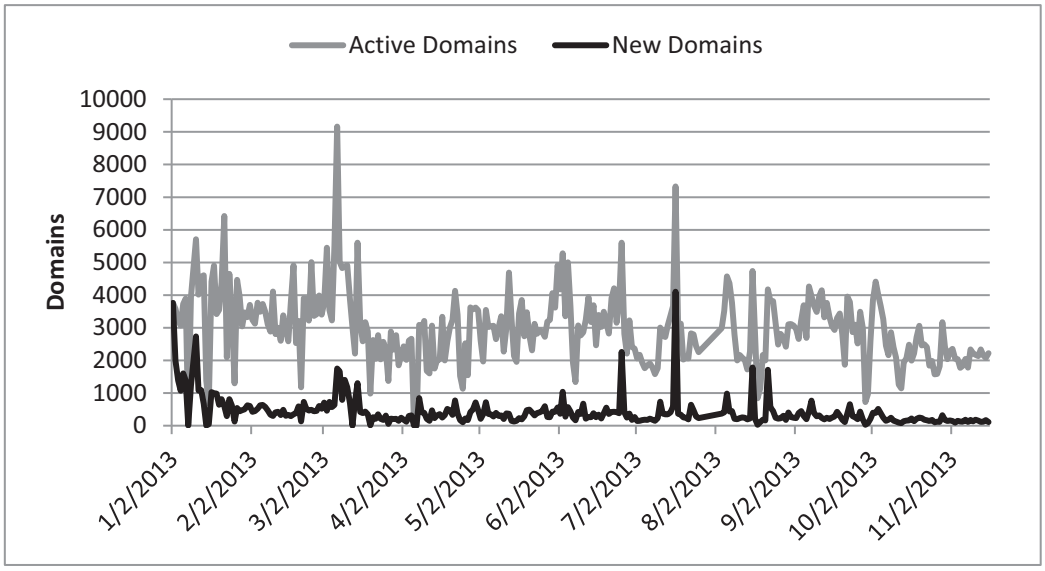


Figure 5.4: Number of observed and new domain names.

5.3 Domain Name Severity System Evaluation

In this section, we present an evaluation of the domain name severity system. We show how the system tracks the behavior of domain names and assigns dynamic severity scores based on the interactions of the domains with malware samples. Reputation systems are evaluated based on existing or known maliciousness judgments. In our case, we must build common knowledge about domain names from existing reputation solutions to represent the maliciousness judgments. Web of Trust (WOT) [128] is a community-based safe surfing tool and is considered as one of the most popular reputation systems, which assigns scores to domain names based on user complaints and sources of blacklists. Domain names are rated based on a scale between [0-100]. WOT scores represent users' preferences and their perceptions about domain names. Such scores also give a graded reputation that reflects the level of abuse of domain names, which can be used to judge the maliciousness of domains under consideration.

Our experiments fall into three sections: first, we show the effectiveness of our system

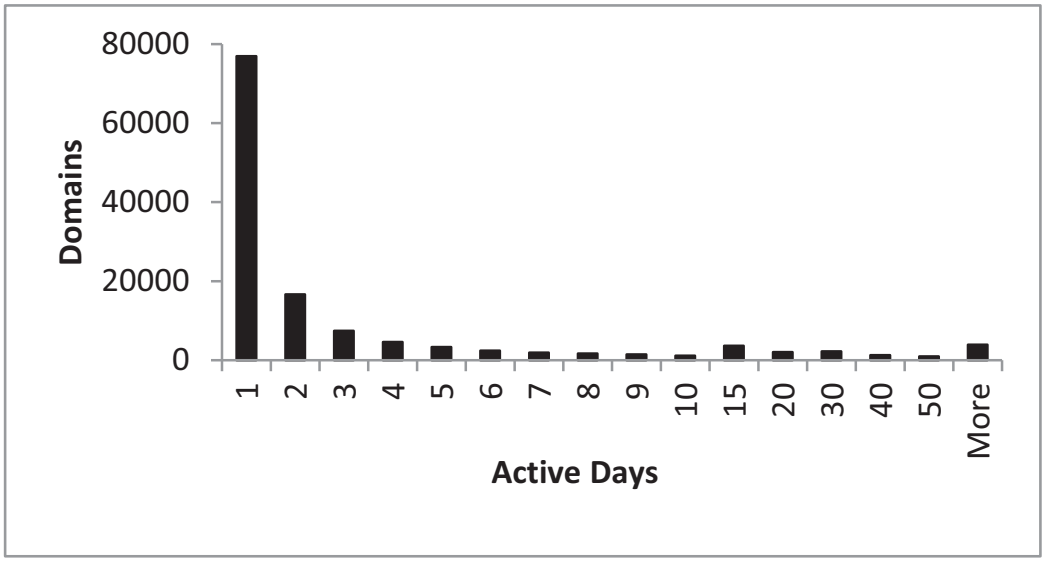


Figure 5.5: Distribution of the number of observed domains according to the number of active days.

by contrasting the obtained scores with the WOT scores. Second, we explore different factors that can affect the system, such as the domain age, features customization, and the domain's behavioral patterns. Finally, we investigate the level of maliciousness inside popular top ranked domain names.

5.3.1 Effectiveness of Domain Name Severity

The effectiveness performance of our severity system is evaluated using known maliciousness judgments from the WOT reputation system [128]. Since the WOT maliciousness indicators are represented in a non-binary notion, one evaluation measure that can be properly used to measure the effectiveness of our system is the nDCG, which is described in Section 2.1.3.

We experiment with a dataset that spans the period of January 1^{st}, 2013 to November 16^{th}, 2013. The system calculates the severity scores for each domain name as described in Section 5.1.4 on daily basis, and then produces a ranked list of severe domains. The

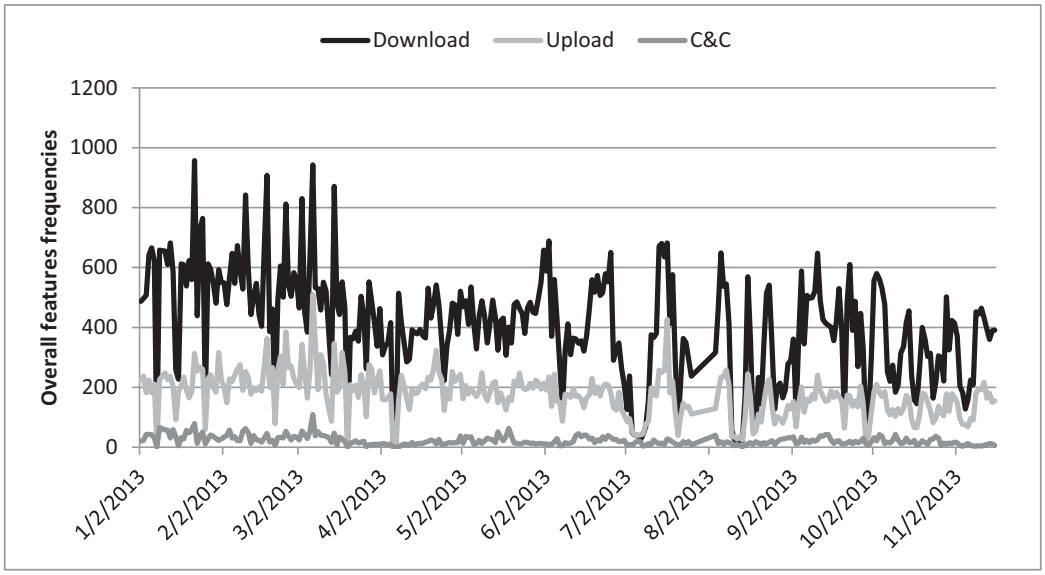


Figure 5.6: The average distribution of the extracted features.

nDCG values for the top 10 ranked domain names of each day are shown in Figure 5.7. The average and variance values of nDCG over the observed interval are 0.72 and 0.0079 respectively. In Figure 5.8, we show the average *nDCG@k* of the system for different values of *k*. Since the behavioral features capture domain name abuse from different perspectives, we evaluate the domain name severity system based on different combinations of these features as shown in Figure 5.9.

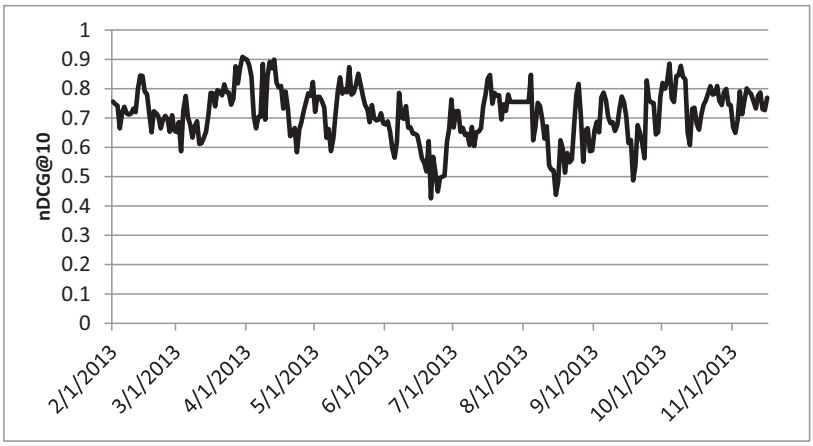


Figure 5.7: The nDCG@10 across the dataset.

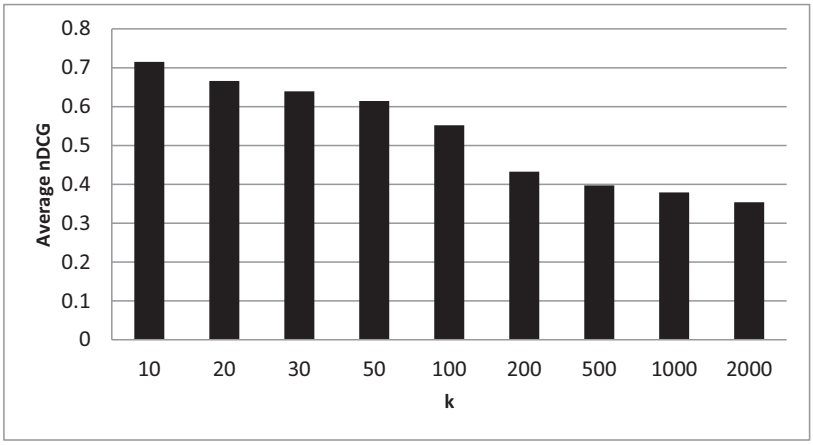


Figure 5.8: The average nDCG@k for the analyzed dataset.

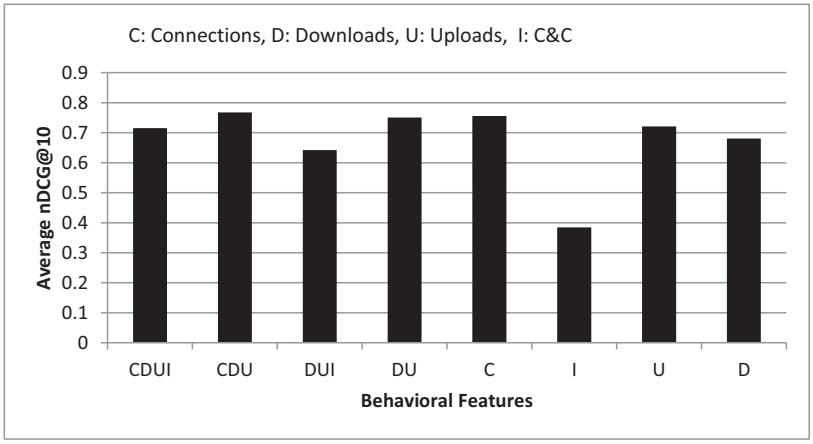


Figure 5.9: The nDCG@10 with different combinations of rating centers.

When a domain name under consideration is not found in the WOT list, we set the mal_j value in Equation 2.2 to zero. In Figure 5.10, we show the percentage of domains that were not found in the WOT list in the 30 top-ranked domains. We investigate all of the missing domains to search for clear justification for the appearances of these domains within the top ranked domains. Figure 5.11 shows the distribution of the behavioral features that are conducted by the domains with no judgments in WOT. Most of these domains fall under C&C activities. Since the WOT is based on user complaints, IRC domains are less likely to be rated by end-users for two possible reasons: one, IRC domains are usually perceived by end-user as harmless; two, there is no direct communication between these domains and

end-users. Furthermore, these domains belong to known IRC service providers.

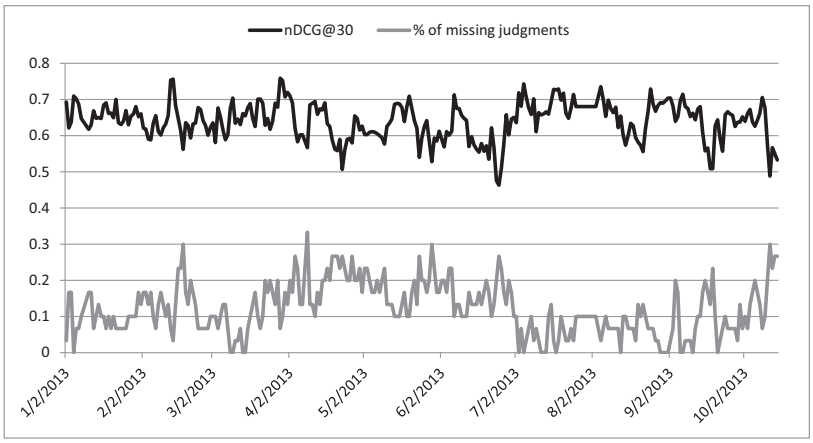


Figure 5.10: Effect of missing judgments on the *nDCG*@30.

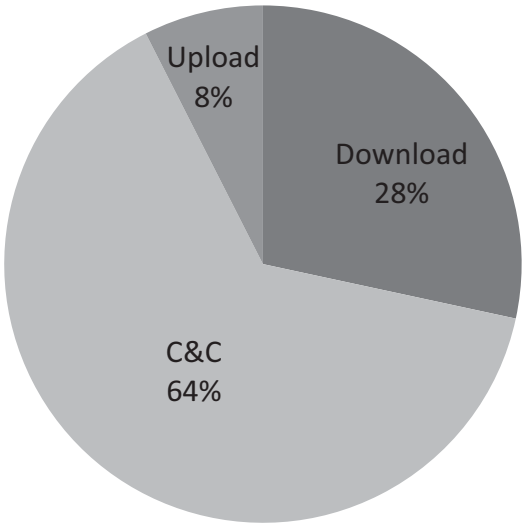


Figure 5.11: Distribution of behavioral features for domains with no maliciousness judgments in WOT.

The domain name severity system depends on the period and the freshness of an abuse. Domain severity scores are discounted based on their age of being abused by the malware community. Short-lived domains, which appear for less than 10 days, represent a large portion of the abused domains as discussed in Section 5.2. The severity system can be configured accordingly to focus on the fresh domains by adjusting the age weights (λ)

in Equation 5.2. Figure 5.12 shows the average nDCG of the system with different values of the age discount factor. When the system uses the age discount factor, it can adjust the importance or the level of contribution of past events to the current severity score. It is clear that the performance dropped in the absence of the age factor discount ($\lambda = 1$), which shows the importance of taking into consideration how long a domain name is abused. In contrast, the system performs better when the age discount factor is set to ignore past scores ($\lambda = 0$), which considers only the fresh domain names or the current day rating values. Our experiments show that the nDCG value does not significantly change for $0.1 \leq \lambda \leq 0.9$.

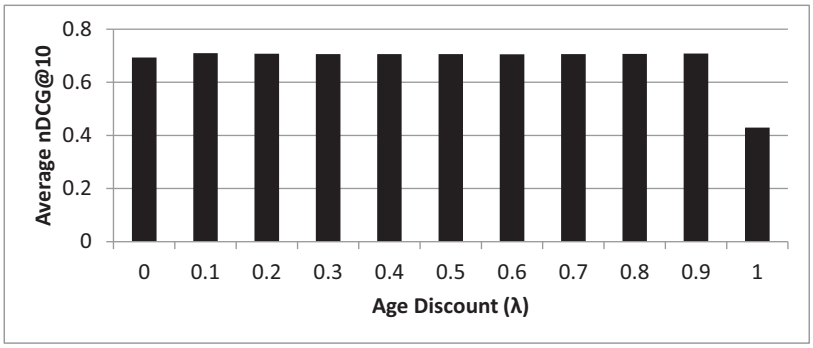


Figure 5.12: Variation of nDCG@10 with λ.

Initially, the scores of fresh domain names are initiated by adjusting the system base rate configurations. The base rate (α) can be initiated by previous expert knowledge, which can affect the evolution of severity calculation. Figure 5.13 shows variation in the average *nDCG*@10 with different configurations of the base rate α. When the system is configured to dynamically calculate the base rate from the existing community, the initial knowledge does not have any effect on the performance. However, when the system uses a fixed base rate, the initial knowledge negatively affects the performance.

To evaluate our system in assigning high severity scores to malicious domains, we collected domains listed by various sources of public blacklists [130, 131, 132]. We then compared the blacklisted domains' severity scores with the non-blacklisted domains within

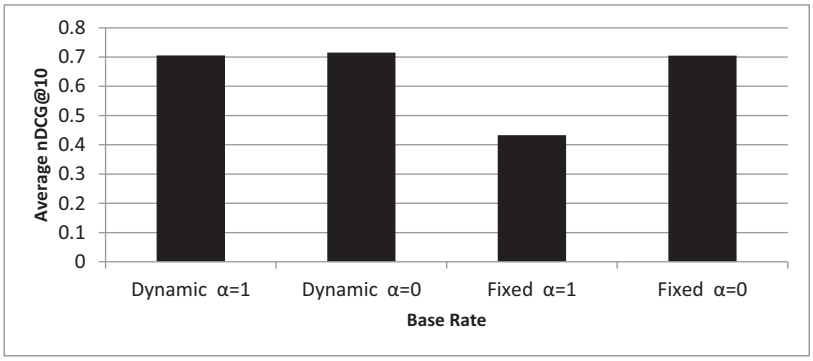


Figure 5.13: Base rate selection and nDCG@10.

the top 1,000 severe domains on the last day of our analysis. Figure 5.14 shows that our system successfully assigns high severity scores to the blacklisted domains, and also reflects the distribution of α scores that are assigned to the new observed domains.

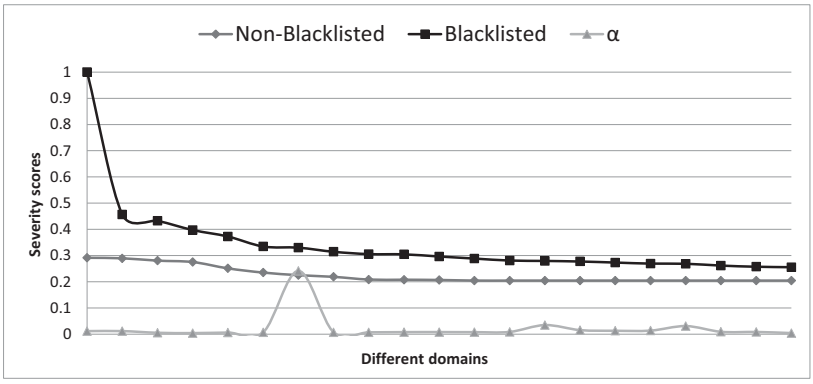


Figure 5.14: Severity score distribution for blacklisted and non-blacklisted domains.

Previous studies observed some abuse within popular domains. It is thus necessary to quantify the level of abuse within top ranked popular domain names [133]. Since our domain name severity system measures the level of maliciousness, this allows us to infer the abuse level of popular domain names. During our experiments, we examined the top 200 domains reported on a daily basis from our severity system to analyze the level of abuse within public lists of popular domains represented by Alexa [134] and Quantcast [135] lists. Figure 5.15 shows the daily percentage of the existence of popular domains within our daily severe domains lists. Among the top 200 severe domains observed from our analysis,

32% and 21% of these domains were found in Alexa and Quantcast, respectively. We also noticed that 37% of Alex and 27% of Quantcast domains have been blacklisted by public blacklists.

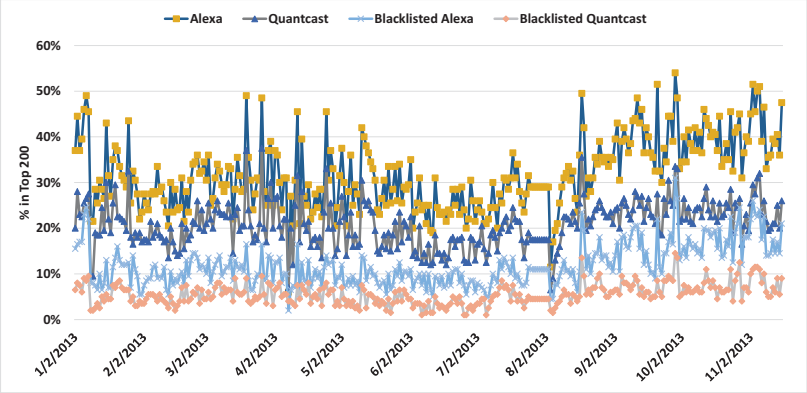


Figure 5.15: Abuse of the malware community in popular domains lists.

5.4 Ranking the Severity of Name Servers from Passive DNS

DNS has been abused by cyber criminals with techniques that require more control over name servers in order to be implemented effectively. In this section, we look at the root cause of managing malicious domain names. We extend the severity system developed throughout this chapter to assign severity scores to name servers. However, we leverage the passive DNS traffic to infer the malicious behavior of the name server. Our main goal is to assign a high severity score to name servers involved in malicious activities. The severity scores help in identifying rogue name servers, which are abused by malicious networks to conduct various activities. Given a name server *ns*, we assign a high severity score if *ns* is involved in malicious activities, such as hosting short-lived domains, Fast-Flux domains, or DGA domains. On the contrary, we want to assign a low severity score if *ns* is used mainly

for legitimate purposes. With dynamic severity scoring, our goal is to decide the level of maliciousness for each name server.

Previous works, discussed in Section 2.2.3, focused mainly on reputations and black-listing of domain names; however, we focus on the root causes by designing a reputation system for name servers instead of domains. To the best of our knowledge, our system is the first to create a dynamic reputation around name servers.

Our work is based on the observation that name server behaviors are reflected on the domain names under their authority. In short, domain name features have certain characteristics that can reveal the nature of the malicious activities operated on a specific name server. By identifying and observing the domains under an authoritative name server, our system can assign appropriate severity scores for that name server. The severity of a name server *ns* is determined by the following steps:

1. We observe the most recent set $R(ns)$ of domain names that reports *ns* as an authoritative name server.

2. We aggregate the set $S(d)$ of all observed sub-domains under d.

3. We measure two different groups of statistical features for each domain d, namely Fast-Flux-based features and DGA-based features groups.

4. The calculated features feed to the rating centers to generate rating values, which are processed by the severity engine to produce severity scores.

Once the statistical features are extracted, the rating center transforms the statistical features into a set of rating values for each name server *ns*. The severity engine then takes the rating values for each name server and assigns the severity score accordingly. In the following, we explain the details about the statistical features. Afterward, the rating centers

discussed in Section 5.1.3 are used to convert the features to rating values, and the severity engine explained in Section 5.1.4 is used to compute the name server's severity score.

5.4.1 Name Server Statistical Features

In this section, we explain the statistical features used in our name server severity system. Name servers play a crucial role in the DNS infrastructure by holding resource records for domain names. Recent abuse of DNS requires more control over name servers in order to be implemented effectively, such as Fast-Flux domains and DGA domains. Malicious networks abuse name servers to hold and operate their domains. The statistical features are inspired by the domain's abuse characteristics that are discussed in previous works [80, 89, 26, 87]. Our features focus on the abuse of DNS that requires more control over name servers. For instance, the Fast-Flux and DGA techniques are considered the most used techniques for name server abuse. Such abuse requires frequent updates to the resource records of the abused domains and more control over name servers as well. By observing the Fast-Flux and DGA abuse, we can extract a few of the statistical features that measure the level of abusiveness for the name server *ns* by malicious networks.

Fast-Flux Features

Since malicious networks are constructed mainly on top of compromised machines, botmasters need techniques to utilize these machines in order to operate malicious activities. Botmasters have recently developed several ways to make malicious networks more flexible and robust against take down actions, e.g., by using Fast-Flux techniques. Criminals introduce FFSNs, which simply host a service by many different IP addresses; i.e., they construct a proxy network on top of compromised machines that are used to build resilient hosting infrastructure using public DNS [23]. Generally, FFSNs can be classified either as

single Fast-Flux or as double Fast-Flux. The single Fast-Flux technique frequently changes the mapping of *A* records [88] of domain names to IP addresses of compromised systems in the botnet. The double Fast-Flux technique is similar to the single Fast-Flux but it has an additional layer of redundancy that continually changes the authoritative NS records [88, 23].

The Fast-Flux features report some characteristics of domain names that are abused by Single-Flux or Double-Flux techniques. Table 5.3 summarizes the features that are used in the literature to capture the behavior of Fast-Flux domain names and provide promising results in recognizing such abuse [87, 89]. Fast-Flux is used to manage many IP addresses to host a specific domain name. Therefore, the number of distinct IP addresses is one of the important signs of Fast-Flux behavior. We measure the existence of multiple IP addresses for a given domain and name server by counting the number of unique A records associated with that domain and name server (f_1, f_2). Fast-Flux behavior can also be recognized by the TTL values for resource records. Since Fast-Flux techniques need to shuffle IP addresses for domain and name servers in short periods, the TTL values of A and NS records are typically configured with low values. By setting low TTL values, DNS forces resolvers to drop the corresponding cached resource records quickly [89]. This causes new resource records to be fetched from the authoritative name server more frequently. Our features include the average TTL values for the observed A and NS records (f_3, f_4) of a specific domain.

DGA-based Features

When an infected machine must communicate with its home infrastructure, it usually contacts either some of the hard coded IP addresses or the domain names. This static contact information could be blacklisted and cause the infected machine to be isolated from reaching home. Cyber criminals overcome this challenge by introducing the DGA technique to

generate domains dynamically using predefined algorithms. By utilizing some of these domains, an infected machine has greater chances to more effectively interact with malicious networks.

The DGA-based features capture the existence of the DGA behavior reflected on DNS messages. Table 5.3 lists some of the features that were introduced in previous studies to detect DGA domains [80]. DGA domain names are usually generated from random characters or non-meaningful words. Based on these two observations, we utilize four features that can detect DGA domains. The first feature is to measure the randomness of the characters within the 2LD (f_5) using Shannon's entropy [26]. The second feature is to estimate the meaningfulness of domains (f_6) by extracting the Longest Meaningful String (LMS) using the Wordnet database [1]. The last two features evaluate the ratio of digits (f_7) and the number of unique characters (f_8) within the 2LD.

Feature Type	Features
Fast-Flux	(f_1) Number of unique A records (f_2) Number of unique A records for name servers (f_3) Average TTL value for A records (f_4) Average TTL value for NS records
DGA	(f_5) Entropy of the 2LD (f_6) Ratio of LMS to the length of the 2LD (f_7) Ratio of digits to the length of the 2LD (f_8) Number of unique characters in the 2LD

Table 5.3: Fast-Flux and DGA features.

5.5 Discussion and Limitations

Existing reputation systems produce a binary decision (Good or Bad) about domain names. We believe that our domain name severity system can serve as an extension to existing reputation systems such as Notos and EXPOSURE. Normally, reputation systems produce

[1] http://wordnet.princeton.edu

a large number of domain names to be blacklisted. Given a list of blacklisted or even regular domain names, our system can add more information about the level of abuse for these domains. Moreover, our system is based on behavioral features, which can easily be extended by adding new features to capture other behaviors of malicious actions such as spam, phishing, and other C&C communications.

Our experiments show a noticeable level of abuse for known good domains represented by Alexa and Quantcast lists. Such abuse raises an alarm against whitelists, which can be abused to deliver and participate in malicious activities. Since our severity system is based on the second level domain, the severity scores represent the collective abuse conducted under each second level domain. The malware community has recently started abusing cloud services to host malicious contents. In our analysis, we observed three major cloud service providers that were listed among the top 300 severe domains rated by our system. These domains have 665 different sites that were abused by the malware community.

Current studies focus on the abuse level of domain names, whereas our work serves as a base for future behavior analysis of name servers. Our name server severity system can complement the exiting domain name reputation systems to discover and effectively take down malicious networks. The recognition of abuse types can be adjusted by the statistical features, which can easily be extended by adding new features to capture other behaviors of malicious actions.

We are aware that our system does have certain limitations. First, the abuse level ranking is limited to the domain names observed during the dynamic analysis of the malware samples. This limitation is inherited from the nature of malware dynamic analysis, which

can be affected by different factors such as analysis time and anti-sandbox techniques. Second, the behavioral features consider all communication between domain names and malware samples as malicious. However, there are some activities that can be considered as benign, such as regular or legitimate software updates by the sandbox environment. Recently, some approaches have been introduced to detect malicious downloaded files [136, 67, 137]. Such techniques can be used to evaluate the reputation of the downloaded files in order to improve the quality of the feature extraction by our system. In the case of the name server severity system, the abuse level ranking is limited to the name servers observed by the DNS sensors. This limitation is inherited from the nature of passive DNS monitoring, which can only listen to name servers that reside in their premises. In addition, the statistical features might capture some of the legitimate services (Fast-Flux). Certain learning algorithms can be adopted to set some thresholds to recognized such legitimate services [87, 89].

5.6 Conclusion

In this chapter, we developed a severity system for domain names based on dynamic analysis reports of malware samples. The system leverages the interaction between domain names and malware samples to extract indicators for malicious behaviors or abuse actions. The system utilizes these behavioral features on a daily basis to dynamically produce severity or abuse scores for domain names. Since our system assigns maliciousness scores that describe the level of abuse for each analyzed domain name, it can be considered as a complementary component to existing (binary) reputation systems, which produce long lists with no priorities. Our system leverages the domain names that reside under the authority of name servers to extract indicators for malicious behaviors or abuse actions. The system utilizes these behavioral features to dynamically produce severity or abuse scores for name servers.

Our evaluation using real-world data, including a 10-month dynamic analysis report, shows that the list produced by our system highly correlates with those produced by the well-known Web Of Trust (WOT) reputation system. Moreover, our system allows us to identify a noticeable level of abuse for known domains represented by Alexa and Quantcast lists.

Chapter 6

Detection of DNS-based Malicious Payload Distribution Channels

In this chapter, we propose a detection mechanism for DNS payload distribution channels by leveraging features of DNS to distinguish between malicious and non-malicious domains. We use this mechanism to analyze a significant amount of DNS traffic in order to understand the extent of DNS abuse in the real world. We detected a few previously unknown long-lasting malware domains and different types of payload distribution channels. This result is significant considering that the use of DNS payload distribution channels by malicious networks is relatively rarely exploited. Our proposed technique, which is based on access counts of resource records, shows promising results regardless of the syntax formats of payload distribution channels.

This chapter is organized as follows. Our system is described in Section 6.1. Section 6.2 explains our datasets and Section 6.3 demonstrates the effectiveness of our proposal via an experiment on near real-time traffic. We discuss the limitations of our work in Section 6.4, and Section 6.5 provides concluding remarks.

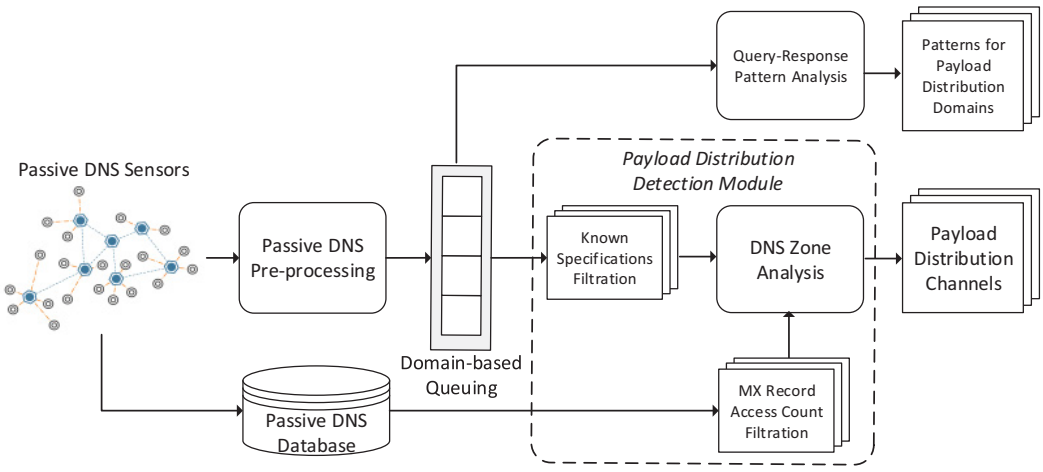


Figure 6.1: Overview of proposed approach.

6.1 Proposed Approach

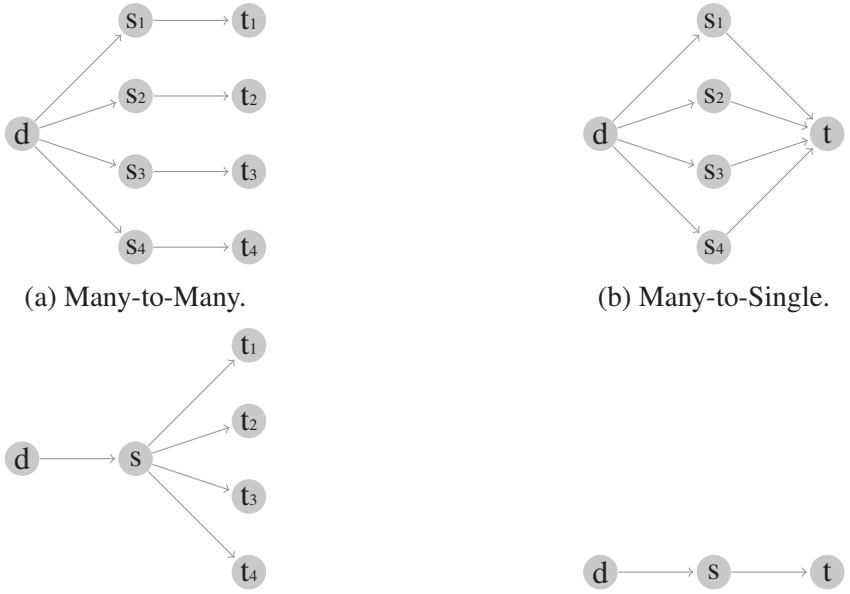
Our system monitors DNS messages in passive DNS and then characterizes and detects payload distribution channels established within these DNS messages. As shown in Figure 6.1, the system consists of two main modules, one for query and response pattern analysis and a second for the detection of payload distribution. Initially, the system pre-processes the passive DNS traffic by extracting DNS messages that have TXT resource record activities and divides the captured DNS traffic stream into specific epochs (e.g., $epoch = 1$ day). For each epoch, it aggregates the DNS query and response messages of a given domain name to be added to the domain-based queue. In parallel, the passive DNS traffic is also stored in a passive DNS database, which collects historical data about DNS messages for offline analysis. Following the pre-processing phase, the collected messages are fed to the query and response pattern analysis module, which provides characteristics of the payload distribution channels. Finally, the detection module identifies the payload distribution channels. In the following section, we provide details on how to characterize and detect payload distribution channels.

6.1.1 Query and Response Patterns

DNS protocol is based on query and response messages to manage domain name systems. A query from any client can be formed to retrieve different information from a name server, which will respond accordingly. By observing the communication between client and server, we can model the relation between query and response messages. The query and response relations can be used to distinguish between different behaviors of payload distribution. When we observe any payload distribution activity, we use three parameters to establish the channels in DNS. These parameters are the second-level domain, sub-domain, and `TXT` record. The second-level domain is used to carry out payload distribution activity. The sub-domain is used to transfer any information from the client to the server. The `TXT` record is the response information from the server to the client. During any session, the client and the server agree on a specific domain name to be used for the payload distribution activity. The second level domain parameter is thus determined before any session. We are then left with two parameters that are used to form the communication channel. Based on the nature of the payload distribution channel, these two parameters have different behaviors. The aim of the query and response pattern analysis module is to differentiate between different behaviors of payload distribution. To achieve this goal, we analyze the exchange behavior of query and response messages. This module is built based on two observations: first, payload distribution channels through DNS are forced to transfer small quantities of information with each DNS message due to the fact that DNS response packets are limited to 512 bytes of characters if Extension mechanisms for DNS (EDNS) is not used [138]. Second, transferring more information through DNS protocol results in a high rate of DNS queries and responses between the client and the name server [29].

Figure 6.2 illustrates different payload distribution scenarios that are considered the four main possible behaviors of the two payload distribution parameters as sub-domains

and TXT records. Figure 6.2a explains how the client is changing the sub-domains to send data to the name server, which will respond with the corresponding TXT records for each sub-domain (Many-to-Many relation). Figure 6.2b shows how the client is changing the sub-domains to update the name server with its status, and the server is replying with the same TXT record for all possible sub-domains (Many-to-Single relation). Figure 6.2c explains how the client is sending the same sub-domain answered by several TXT records from the server (Single-to-Many relation). This case rarely occurs within a short time frame because these responses are cached by caching resolvers for a period of time (TTL). Figure 6.2d shows how the client is sending the same sub-domain answered by only one TXT record from the server (Single-to-Single relation). In general, malware families retrieve updates from a malicious infrastructure in three different approaches: full payload updates (Many-to-Many relation), periodical updates (Many-to-Single relation), or a one-time update (Single-to-Single relation).



(a) Many-to-Many.

(b) Many-to-Single.

(c) Single-to-Many.

(d) Single-to-Single.

Figure 6.2: Examples of query and response exchange patterns.

Definition 6.1. The query-response pattern model is a tuple $G = \langle \{d\} \cup T \cup S, E \rangle$, where:

- d is the domain name node,
- S = $\{s_1, s_2, \cdots, s_m\}$ is a finite set of sub-domain nodes,
- T = $\{t_1, t_2, \cdots, t_k\}$ is a finite set of `TXT` record nodes,
- $E \subseteq (D \times S) \cup (S \times T)$ is a finite set of pairs of distinct nodes, called edges.

We model the query and response relationship for each domain using a directed graph as captured by Definition 6.1. For each vertex v in G, we define two functions: the in-degree of v, denoted by $inDegree(v)$, returns the number of *entering* edges to the node v: $inDegree(v) = |\{u \in V \mid (u,v) \in E\}|$; and the out-degree of v, denoted by $outD(v)$, returns the number of *leaving* edges from the node v: $outDegree(v) = |\{u \in V \mid (v,u) \in E\}|$.

As shown in Figure 6.2, the query and response relationship patterns share certain properties as given by Property 6.1.

Property 6.1. *$inDegree(d) = 0$, $outDegree(t_k) = 0$, $outDegree(d) = inDegree(s_m)$, and $outDegree(s_m) = inDegree(t_k)$*

In order to distinguish between the patterns shown in Figure 6.2, we define the normalized distance measure between two non-zero integer values i_1, i_2 as formulated in Equation 6.1.

$$Dist(i_1, i_2) = \frac{|i_1 - i_2|}{max(i_1, i_2)} \tag{6.1}$$

Since the query and response patterns can form complex relationships, we limited our system to recognize only one pattern for each domain. We must therefore choose a candidate node to represent the set. In fact, the system selects the *most commonly used node degree* inside the targeted set. Let $t \in T$, and $s \in S$, each query and response pattern can be recognized by the following properties:

Property 6.2. *Given that* $inDegree(t)$ *is the common value in* $inDegree(T)$*, then the Many-to-Many pattern holds when* $Dist(outDegree(d), inDegree(t)) \geq 0.5$ *and* $Dist(|S|, |T|) < 0.5$.

Property 6.3. *Given that* $inDegree(t)$ *is the common value in* $inDegree(T)$*, then the Many-to-Single pattern holds when* $Dist(outDegree(d), inDegree(t)) < 0.5$ *and* $Dist(|S|, |T|) \geq 0.5$.

Property 6.4. *Given that* $inDegree(t)$ *is the common value in* $inDegree(T)$ *and* $outDegree(s)$ *is the common value in* $outDegree(S)$*, then the Single-to-Many pattern holds when* $Dist(outDegree(s), inDegree(t)) \geq 0.5$ *and* $Dist(|S|, |T|) \geq 0.5$.

Property 6.5. *Given that* $inDegree(t)$ *is the common value in* $inDegree(T)$ *and* $outDegree(s)$ *is the common value in* $outDegree(S)$*, then the Single-to-Single pattern holds when* $Dist(outDegree(s), inDegree(t)) < 0.5$*,* $Dist(outDegree(d), inDegree(t)) < 0.5$*, and* $Dist(|S|, |T|) < 0.5$.

Algorithm 18 displays an overview of query and response pattern recognition in four steps. Step 1 (Line 1) takes a snapshot from the passive DNS traffic for a pre-defined window of time. This step produces a set of query and response messages for each domain that appears within the targeted window. Step 2 (Line 3) processes every domain name by constructing the relation graph between sub-domains and `TXT` records. Step 3 (Lines 4-6) calculates the out-degree vector for all sub-domains, the in-degree vector for all `TXT` records, and the out-degree of the domain. From these vectors, we get the commonly used degree, which is considered a strong representative for the relation between all nodes. Step 4 (Lines 7-9) counts the distinct values of sub-domains and `TXT` records. Step 5 (Lines 10-17) determines the pattern mode based on the properties of each pattern, which can be applied in arbitrary order (Property 6.2-6.5).

Algorithm 1: EXTRACTQUERYRESPONSEPATTERN

Input: A domain name d, set of sub-domains $S = \langle s_1, s_2, \ldots, s_n \rangle$, set of TXT records $T = \langle t_1, t_2, \ldots, t_m \rangle$

Output: Query and Response pattern mode, $\{Many\_Many, Many\_Single, Single\_Single\}$

```
D ← getSnapshoptFrom_pDNS(w)
foreach Domain d do
    G ← Create_Relation_Graph(d, S, T)
    SubDomain_Degree ← Common(outD(S))
    TXT_Degree ← Common(inD(T))
    Domain_Degree ← Common(outD(D))
    SubDomains_Counter ← |S|
    TXT_Counter ← |T|
    Pattern_Mode = None
    if Property 6.5 then
        Pattern_Mode = Single_Single
    if Property 6.2 then
        Pattern_Mode = Many_Many
    if Property 6.3 then
        Pattern_Mode = Many_Single
    if Property 6.4 then
        Pattern_Mode = Single_Many
return Pattern_Mode
```

6.1.2 Detection of Payload Distribution Channels

Each DNS message has a domain name *d* that consists of a set of labels. The rightmost label is called the *top-level* domain; the two rightmost labels are called *second-level* domain; the rest of the labels are referred to in the same manner. The services provided by these labels are represented in a *zone* file and stored in the corresponding authoritative name server. Name servers are capable of handling any DNS query and returning the corresponding responses, which are taken from the zone file of each domain name. As name servers are the key players in DNS, malicious networks must have access to a name server for managing the payload distribution. When a name server is appointed to be the authoritative name server for a malicious domain name, botmasters prepare the zone file of the domain to store all attack payloads to be delivered via DNS. Since DNS zone files reflect the provided services of domain names, we decided to observe DNS zone files to analyze domain name behavior.

Extraction of DNS Zones: In payload distribution channels through DNS, name servers are considered as the payload distributors. Since domain names can have multiple zones, we must recognize the responsible zones that are associated with payload distribution.

Since a domain might have multiple labels that point to different zones, the system traverses the labels from second-level to the left-most label. For each label, the NS resource record is requested from the passive DNS database to see whether or not that label is a zone. If a sub-domain label has an NS record, it is a zone under that second-level domain. In the next step, the system profiles DNS zone activities of this zone.

DNS Zone Analysis: First, we analyze the behavior of domain names by observing DNS zone activities in the passive DNS traffic. Within the zone file of each domain name, there are different types of resource records. Each resource record indicates specific services or operations associated with the domain name. An important feature of the passive DNS

database is the aggregation of the number of times each record has been requested, called *access count*. Our intuition is that domain names, which are solely used for payload distribution, show different behavior compared to regular domains. Regular domains receive queries for different resource records, whereas malicious domain names, which are mostly used for payload distribution through DNS, are only accessed to receive attack payloads. Botmasters therefore focus only on using specific resource records that are known to be used in payload distribution channels such as TXT records. Moreover, these domains do not heavily use the resource records that are normally used by regular domain names, such as A, AAAA, and MX resource records. Thus, by observing the resource records and their access counts, we can profile the DNS zone activities of a domain name.

Determining whether a zone is used for payload distribution purposes can be achieved by analyzing its resource record activities. These activities can be calculated as a function of access counts. By using the passive DNS database, we extract all accessed resource records and their access counts. The passive DNS is built in such a way that it counts the amount of times of access to each resource record for a certain period.

Let $R = R_A \cup R_{NS} \cup \cdots \cup R_{TXT}$ where $R_A = \{r_A \mid r_A \text{ is an A record}\}, \ldots, R_{NS} = \{r_{NS} \mid r_{NS} \text{ is an NS record}\}$ be the set of all resource record types that can be defined in a DNS zone file, and $P = \{p \mid p \in (R \setminus R_{NS} \cup R_{CNAME})\}$ is the set of all the resource record types that are commonly used by payload distribution channels and other uses too. Since the TXT resource record is known to be the most suitable for payload distribution, we define a set $T = \{r_{TXT} \mid r_{TXT} \text{ is a TXT record}\}$ that holds any TXT record in a given zone.

Let the sum of access counts of all resource records in P be ac_P, and let the sum of access counts of all resource records in T be ac_T. We then define μ as follows:

$$\mu = \frac{ac_T}{ac_P} \tag{6.2}$$

Passive DNS	Period	30 days
	DNS messages	$\approx$ 20 Billion
	TXT records	$\approx$ 40 Million
Malware database	Period	1 year
	No. of samples	$\approx$ 14 Million
	No. of samples with TXT activities	$\approx$ 18 Thousand

Table 6.1: Statistics of the dataset used to evaluate the proposed approach.

From Equation 6.2, μ reflects the relation between the access ratios of T and P records and thus acts as an indicator of payload distribution activities. Payload distribution channels receive significant access counts from T, which results in larger μ values. However, non-payload distribution channels receive more access counts on the resource records from P, which hence results in smaller μ values.

Filtration Steps: Certain legitimate cases can behave as payload distribution channels. In fact, there are specifications, which use TXT records to apply some security measures for mail servers such as Sender Policy Framework (SPF) [139], Domain Keys Identified Mail (DKIM) [140], Internet Key Exchange (IKE) [141], and DNS Black List (DNSBL) [142]. Since these specifications are designed for mail servers, a DNS zone file should reflect the existence of MX resource records. These legitimate services can be recognized using two different filtration steps: specifications recognition (i.e., SPF, DKIM, IKE, or DNSBL) and MX resource record activity. The first filtration process takes each domain and selects the most accessed TXT resource record using the passive DNS database. We then apply a regular expression in the TXT record based on the defined syntax of specifications [140, 139, 141, 142] to determine any possible specification string (e.g., SPF). When data from the TXT record match any defined specifications, we consider the domain name a non-payload distribution channel. In the second filtration step, we investigate the activities of MX resource records. When a domain name is associated with MX resource record activities, it is considered a non-payload distribution channel.

6.2 Datasets Description

We utilize three datasets to evaluate our system: near real-time passive DNS traffic, a passive DNS database, and a malware database.

Passive DNS Traffic: We evaluate the system using a one-month passive DNS dataset, which spans from March 19, 2013 to April 19, 2013, provided by Farsight Security Inc. [33]. We process only the packets with TXT responses. According to the system logs, the total number of packets processed by our system is around 40 million packets with an average of about 1.3 million packets daily (Table 6.1).

Passive DNS Database: Our system also builds a pDNS database that stores all the data coming from the pDNS traffic. This database recorded the pDNS traffic that we utilize for profiling the DNS zone of domains. The passive DNS database is provided by DNSDB of Farsight Security Inc. [33].

Malware Database: We observe malware samples provided by ThreatTrack Security Inc [124] over a one-year period. We receive the malware feed on a daily basis and then analyze each sample in a sandbox to generate dynamic behavioral analysis reports. In our analysis, we only consider malware samples that conduct activities using the TXT resource record. Table 6.1 shows statistics about the malware feed recorded between January 2012 and December 2012.

6.3 Experimental Results

In this section, we report the results of our experiments that we perform to test the effectiveness of our system for detecting payload distribution channels in the passive DNS dataset. We begin by demonstrating the results of the DNS zone analysis module using the passive DNS dataset. Subsequently, we elaborate on the long-running hidden domains used by the

Morto worm [14] to distribute attack payloads. Finally, we conclude the section by showing that, contrary to common knowledge (e.g., [14]), some of the attack payloads are in clear-text without any encoding or encryption. This indicates that our system can detect these channels regardless of the syntax of the distributed data.

During our experiments, we processed domain names accessed for TXT records in a time-based window, the length of which we set to one day. When the window expires, the packets are fed to the DNS zone analysis module to build the DNS zone profile of each zone for detecting payload distribution channels.

Query and Response Patterns: In the first step of our system, we determine the query and response patterns of the captured DNS traffic. To evaluate the effectiveness of each pattern to carry out payload distribution channels, Figure 6.3 compares the average distinct DNS messages with the number of malware instances per pattern as well as the number of observed domains. The Many-to-Many pattern can be considered as the best candidate for distributing large volumes of data while it was probed by a small number of malware instances during the year of 2012. This extensive payload retrieval scheme would easily alert the IDS [143]. On the other hand, the Single-to-Single pattern allows carrying small volumes of data while maintaining a low network footprint. By observing our malware samples, we discovered that most of the malware instances used this pattern to retrieve the attack payloads as punctual updates. Since each of these instances sends a single query to receive the attack payload, their queries can easily blend into the daily network traffic. Compared to other patterns, Single-to-Single is the best candidate to establish a fully resilient channel in DNS. The Single-to-Many pattern requires updating the zone file to distribute different resource data for the same query. This is technically difficult to maintain because of the caching behavior of name servers. As we see in Figure 6.3, not a single malware instance uses this pattern. Although the Many-to-Single pattern has a single response

to different queries like the Single-to-Single pattern, it creates a large number of queries that can also be noticed by IDSs.

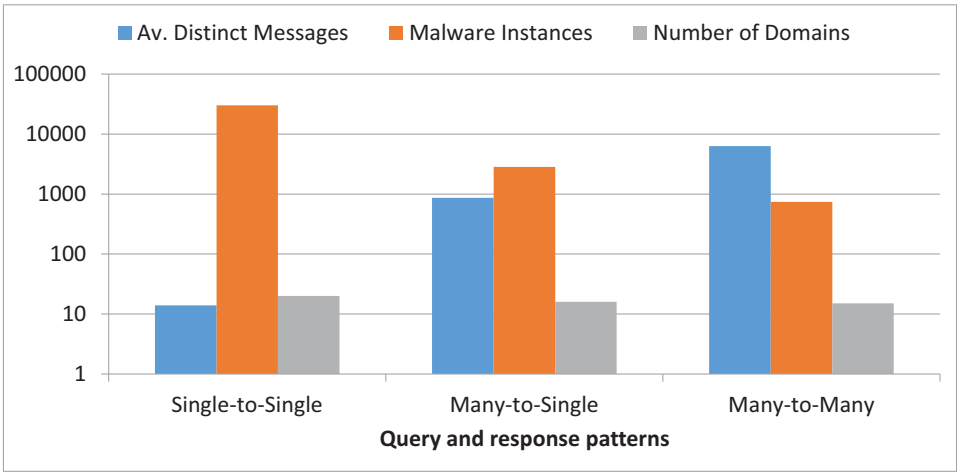


Figure 6.3: Average number of query and response messages within a one-day window.

Although the number of domains between all of the patterns is very close, there is a distinct variation in the amount of generated traffic as shown in Figure 6.3. The Many-to-Many pattern is generating the most extensive traffic compared to the other patterns. The high volume of data exchange can reveal the name server used as payload provider. In order to hide this name server, botmasters use it only for the bootstrapping phase to initiate payload distribution channels. Malware samples are heavily using the Single-to-Single pattern, which has the least amount of traffic, as it makes them difficult to be detected by traditional defense mechanisms.

It is known that malware families are likely to share functionalities by utilizing common modules [144]. We observe the same behavior between different malware families that share the same patterns in our evaluation. Figure 6.4 shows the query and response pattern distribution across different malware families. The intersection between patterns produces the number of malware families that share the same module. For example, there are nine malware families utilizing Single-to-Single and Many-to-Single patterns. These

samples use the Single-to-Single transfer pattern to contact the botmaster and receive a single packet while using the Many-to-Single pattern to check for periodic updates. There are also six malware samples that utilize a combination of all three patterns to establish more complex payload distribution channels.

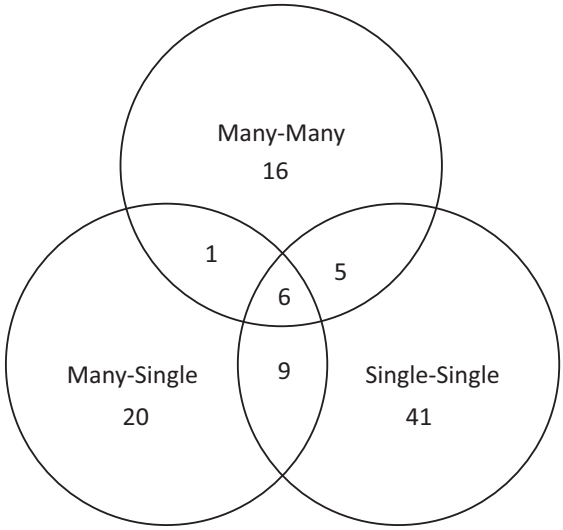


Figure 6.4: Query and response pattern distribution for observed malware families.

DNS Zone Analysis: When the query and response messages of domains are captured, they are inspected by the DNS zone analysis module. The access counts of each resource record are gathered from the passive DNS database. Equation 6.2 determines the μ values of each domain based on the access counts. During our experiments on the passive DNS traffic, we captured 2707 domains that have TXT resource record activities. Figure 6.5 shows the distribution of domain names with different μ values. According to Equation 6.2, the bigger the μ value, the more a domain name is involved in payload distribution.

Filtration: In Table 6.2, we show the number of detected domain names during the 30-day period and the effect of each filtration mechanism. 2707 domains are detected before any filtration is applied; however, some of these domains might be accessed mainly to receive specifications-related data in TXT records. Applying both filtration mechanisms reduces

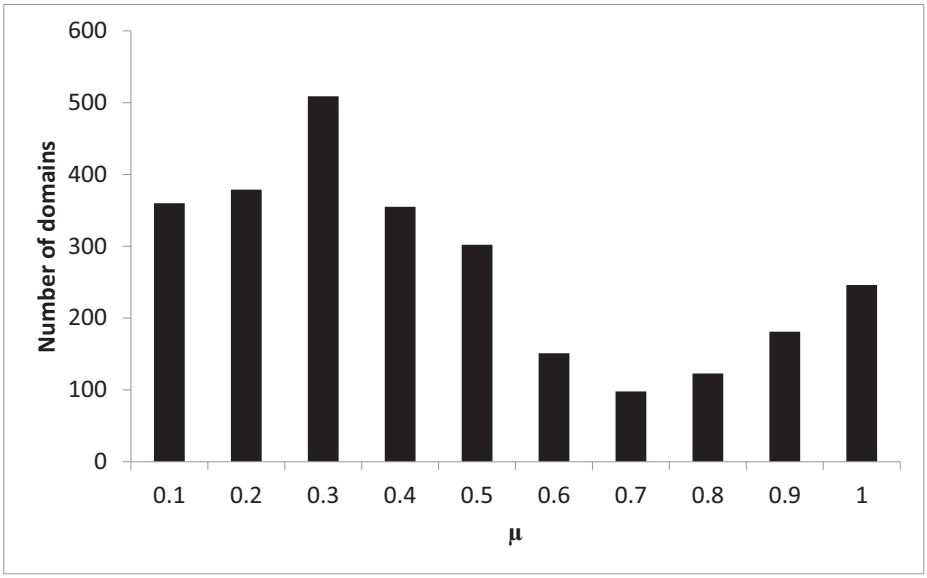


Figure 6.5: Distribution of rating values of the 2707 detected domains.

Number of observed domains	2707
Domains with MX records	2506
Domains following known specifications	2613
Domains remaining after applying both filtrations	37

Table 6.2: Statistics of detected domains within 30 days.

the number of domains to 37.

To validate the effectiveness of the filtration process, we observe our malware dataset and passive DNS database to investigate the difference between payload distribution channels and regular domains. As regular domains, we use the top 500 domains from Alexa top sites. By using our one-year malware dataset, we extract malware domains used for payload distribution. We retrieve the access counts for all resource records of each domain, from regular as well as from malware domains. These access counts represent a good measure to understand the individual resource record activities of any given domain. Figure 6.6 provides the distribution of access counts for the considered types of resource records (A, AAA, MX, NS, TXT, and CNAME). Domains from Alexa receive DNS queries for different resource records. The reason for this can be attributed to the fact that these domains

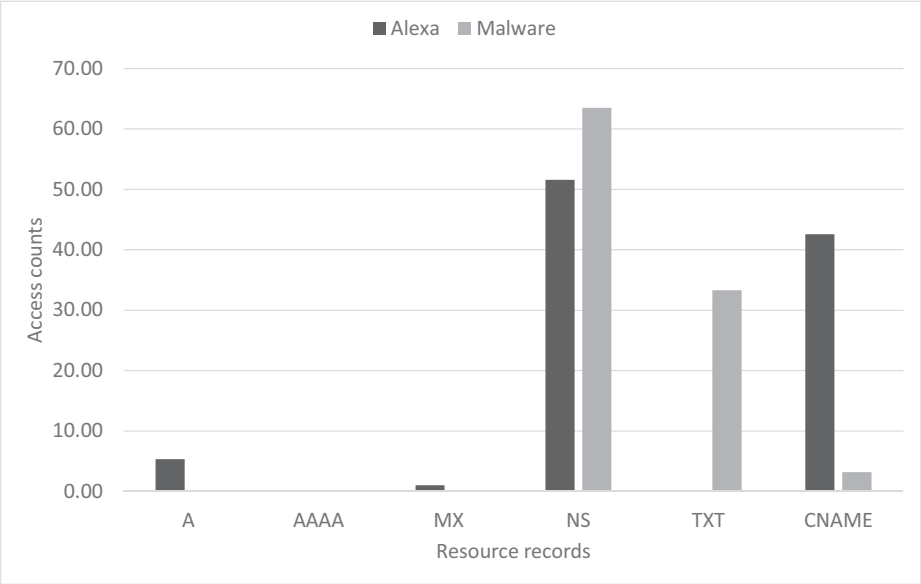


Figure 6.6: Access counts of Alexa and malware domain DNS records.

utilize DNS for enabling access to different services. On the contrary, malware domains receive an extensive number of DNS queries for TXT records. These records are used to distribute the payload as they are the most suitable resource record type within the protocol. We also investigate access to the CNAME records in malware domains. These are used to redirect queries between malicious domains as botmasters maintain a network of malicious payload distribution channels. On the other hand, malicious domains are not associated with any MX resource records, which support the second filtration process.

The Resilient Morto Domains: Morto is a malware family that targets the Remote Desktop Protocol (RDP) to gain access to host machines [14]. It is one of the malware families that uses DNS as a payload distribution channel [14]. By utilizing the passive DNS database, we detected domains used by the Morto family for more than 18 months with an average to TXT records over four million times. Past work [14] has revealed that Morto receives a Base64 encoded or encrypted URL, which points to the second payload. We noticed that Morto domains also distribute IP addresses in clear-text inside TXT records. A reverse lookup of one of these IPs in the passive DNS database reveals that it is shared with

other malicious domains. As previously mentioned, the malware authors also link together different domains through CNAME records to maintain a malicious network.

Source	No. of domains	Avg. μ value
Devicescape [53]	12	0.998
Tunneling [145]	1	0.991
Morto [14]	3	0.994

Table 6.3: Detected payload distribution domains.

Detection Regardless of Syntax: As our method discovers the Morto domains, it also detects the legitimate payload distribution channels as discussed in Section 2.1.2. It indicates that regardless of the syntax of the payload distribution channel, the μ (access count) metric is a strong feature to detect domains, which are used for these channels (Table 6.3). If botmasters start using a syntax similar to the legitimate services to blend into their traffic, they might not be detected by network monitors. However, our system may still detect them, since the system monitors the DNS zone activities of payload distribution channels rather than investigating their message syntax.

DNS Tunneling Detection: Our experiments reveal some DNS tunneling activities from a single domain (a DNS Tunneling application for Android [145]). As our system is configured to monitor TXT records, it successfully detects any DNS tunneling activities on TXT records. If the tunnel is established by using another resource record type, we expect that our system's detection ability would remain intact, as the detection is not based on the content of the resource record, but is rather based on the access counts of resource records.

In Table 6.3, we summarize our results with use cases and average μ values. After applying both filtrations, we are left with only a few domains per day, and can therefore manually investigate their traffic. The manual investigation resulted in 16 domain names that are used as payload distribution channels. The remaining domains are used for transmitting domain specific data in TXT records.

6.4 Discussion and Limitations

During our observation of the malware dataset, we find domain names that are used for payload distribution channels. These malware samples use different methods to retrieve the malicious content as discussed in Section 2.1.2. One of the interesting methods involves the use of indexed queries to receive attack payload in multiple response packets. Due to the size restriction on TXT resource records, the payload is chunked into parts, with each part placed in another TXT record. Bots start querying this series of packets in a sequential manner until the last packet is received. Some of these payloads are chunked into thousand of packets. Surprisingly, this method is very similar to the patent from Trend Micro [52]. However, our results showed that this method is not seen in our passive DNS dataset. There are two possible interpretations of the fact that this behavior is not observed in our recent dataset: either botmasters realize the significant exposure of using this mechanism, which generates a large number of messages, and therefore decide to stop using it; or, these domain names are directly resolved by their own name servers or other open resolvers, which are not captured by passive DNS sensors.

The closest work to our proposed solution is the detection of DNS traffic of a specific malware family by applying features that are based on previously seen malicious traffic syntax [16]. Our solution uses another approach by investigating the DNS zone activities of malicious domains. Even if malware changes the message syntax, the use of DNS remains the same. Our method detects the malicious traffic regardless of the syntax and malware family.

The proposed solution can be used in real-time scenarios where there is access to the DNS zone activities of domains. For example, it can be used by a domain name registrar to detect registered domains used for payload distribution. In this case, the system can be deployed on the authoritative name servers of the registrar so that it can observe the zone

activities of domains.

Since each malware family can use different domains, it is possible that our system assigns multiple pattern labels to a single malware family. As previously explained, certain malware families utilize a bootstrapping technique to initiate the payload distribution channel and then use another domain name with a different exchange pattern. In this case, our method will give two different labels to the same malware family.

Our current design includes the following limitations. First, there are domain names that use TXT resource records for legitimate services along with other activities. Malware can play the same role by operating different services on the same domain name. When a domain is used for different malicious activities (i.e. spam, phishing) as well as for payload distribution, it will be accessed for different resource records, e.g., *A* record for phishing scam websites. Since our system makes use of the fact that name servers of payload distribution channels receive requests mostly for TXT records, it might consider this domain a non-payload distribution domain. Second, the passive DNS replication [50] is a unique way to collect the global DNS traffic by sensors. Unfortunately, it suffers from a shortcoming that might affect our results. Malware might not use the caching resolver of a network, and instead send queries directly to an open resolver. In this case, the traffic would not pass through the sensors that collect DNS traffic, and would not be captured. However, this is likely to remain a limitation in all solutions based on passive DNS datasets.

6.5 Conclusion

In this chapter, we shed light on the abuse of the DNS protocol by malware for distributing attack payloads. We designed a system to characterize and detect the payload distribution channels within passive DNS traffic. Our system observes the DNS zone activities of a channel by gathering access counts of each resource record type, and determines payload

distribution channels. Our experiments on near real-time passive DNS traffic show that our system can detect several resilient malicious payload distribution channels, which were active for more than 18 months. We found that most malware instances with DNS-based payload distribution use a resilient pattern to retrieve their attack payloads, ostensibly to blend in with regular network traffic. Our system is also able to detect payload distribution channels regardless of their syntax format. The evaluation of malware dynamic analysis reports demonstrated that our method can determine payload distribution channel patterns for different malware families.

Chapter 7

Conclusion and Future Work

7.1 Summary and Conclusion

With the rise in the number of malware samples reported by Anti-Virus companies on a daily basis, security professionals should integrate methodologies to digest the extracted intelligence from analyzing such a large number of malware instances. In this context, the dynamic analysis of malware samples inside a closed environment is a promising approach to understand and observe the behavior of malicious codes and networks. This approach aims to extract information regarding an infected system and its network activities. Such extracted information can be considered a first step towards understanding malicious networks. To explore the abuse of malicious networks, we have elaborated some methodologies to extract intelligence, prioritize DNS abuse level, and detect payload distribution channels through DNS.

For the intelligence extraction, we have introduced in Chapter 3 a detailed reverse engineering analysis of the Zeus crimeware toolkit to unveil its underlying architecture and enable its mitigation. Furthermore, we provided a breakdown for the structure of the Zeus botnet network messages. Our analysis of the C&C communications indicates that

the RC4 algorithm is used in a poor way to encrypt these communications (keystream reuse). In addition to the knowledge of the network messages structure, we can launch active countermeasures by interacting with the botnet servers using the extracted encryption key. Although the reverse engineering of malware samples generates valuable insights, it is a tedious job that would be impractical to exercise on a daily basis. To keep up with the increasing demand of analyzing malware samples, we have devised, in Chapter 4, a framework to extract and build intelligence from the dynamic malware analysis reports. The framework produces statistical insights that can be used in the early stages of cyber attack investigations. Moreover, it identifies the configuration of any given network and quantifies the relative importance of individual network resources. The developed framework utilizes the power of graph-based algorithms to explore relationships between malicious resources, which are used to investigate the abused domains as well as the malicious infrastructure network.

We have also designed and implemented, in Chapter 5, a severity system for domain names based on dynamic analysis reports of malware samples. The system leverages the interaction between domain names and malware samples to extract indicators for malicious behaviors or abuse actions. The system utilizes these behavioral features to dynamically produce severity or abuse scores for domain names on a daily basis. Since our system assigns maliciousness scores that describe the level of abuse for each analyzed domain name, it can be considered as a complementary component to existing (binary) reputation systems, which produce long lists with no priorities. Our evaluation using real-world data, including a 10-month dynamic analysis report, shows that the list produced by our system highly correlates with those produced by the well-known WOT reputation system. Moreover, our system allows us to identify a noticeable level of abuse for known domains represented by Alexa and Quantcast lists. Such abuse raises an alarm against whitelists which can be

abused to deliver and participate in malicious activities. In our analysis, we observed noticeable abuse for three major cloud service providers, which raises an alarm to implement more policies around cloud services.

DNS has been recently abused by cyber criminals with techniques that require more control over name servers in order to be implemented effectively. To this extent, we extend the severity system to name servers based on DNS traffic. The system leverages the domain names that reside under the authority of name servers to extract indicators of malicious behaviors or abuse actions. The system utilizes these behavioral features to dynamically produce severity or abuse scores for name servers.

Finally, in Chapter 6, we shed light on the abuse of the DNS protocol by malware for distributing attack payloads. We developed a system to characterize and detect the payload distribution channels within passive DNS traffic. Our system observes the DNS zone activities of a channel by gathering access counts of each resource record type, and determines payload distribution channels. Our experiments on near real-time passive DNS traffic show that our system can detect several resilient malicious payload distribution channels, which were active for more than 18 months. We found that most malware instances with DNS-based payload distribution use a resilient pattern to retrieve their attack payloads, ostensibly to blend in with regular network traffic. Our system is also able to detect payload distribution channels regardless of their syntax format. The evaluation of malware dynamic analysis reports demonstrated that our method can determine payload distribution channel patterns for different malware families. The proposed solution can be used in real-time scenarios where there is access to the DNS zone activities of domains. For example, it can be used by a domain name registrar to detect registered domains used for payload distribution. In this case, the system can be deployed on the authoritative name servers of the registrar so that it can observe the zone activities of domains. The rise of DNS protocol abuse to

distribute attack payload information urges us to adjust the policies in order to regulate and manage the use of DNS resource records.

7.2 Future Work

Much progress remains to be made in addressing the DNS abuse by the malware community. In what follows, we suggest some research directions that can utilize both DNS traffic and malware analysis to address different challenges.

Take Down Optimization: Take down operations result in dismantling major criminal networks such as Rustock [146] and Citadel [147]. However, many argue over the effectiveness of such take down operations [148]. Our analysis of malicious networks reveals the importance of some structural properties and domain names. One possible improvement would be to evaluate the optimal selection of domains and IP addresses in order to efficiently take down malicious infrastructure. The evidence of maliciousness extracted from malware analysis and the volume of activities inferred from DNS traffic can help develop sound techniques for identifying effective ways to take down malicious infrastructures. This research would be extremely valuable, both to the operational community and to law enforcement, for making the best decisions against malicious networks.

Network Reputation: As demonstrated by this thesis, the severity system has been extended to evaluate the abuse level of name servers. Similarly, the severity could be applied to evaluate networks such as ISP or hosting providers. When evaluating networks, there are many challenges to take in consideration, such as the network size, malicious activities duration and volume, and the type of threat. Such techniques shall provide network operators with the threat status of their networks and help in implementing corrective measures.

Stone-Gross et al. [83] recently proposed a system to infer the reputation of networks using static blacklists, which can be extended by utilizing malware analysis and passive DNS traffic to build a comprehensive network reputation profile.

Cloud Services Abuse: Due to the reliability and scalability of cloud based platforms, cyber criminals have begun to abuse cloud services. Our study of the severity of domain names can play an important role in quantifying the abuse of public services. For instance, our evaluation demonstrates that cloud services are being abused by malicious activities. In order to protect the reputation of the abused cloud services, more research is needed to differentiate the pattern of abuse for such services. The intrusion detection system solution can be employed to study the behavior of malware samples while accessing cloud services and distinguish these behaviors from normal ones.

Bibliography

[1] The NAMESENTRY quality report: Abuse detection and mitigation service. URL http://architelos.com/wp-content/uploads/2013/11/NameSentry-Abuse-Report-Vol-2-Nov-2013.pdf. Accessed: 2014-07-30.

[2] Wenke Lee, Cliff Wang, and David Dagon. *Botnet Detection: Countering the Largest Security Threat*, volume 36 of *Advances in Information Security*. Springer-Verlag New York, 2008.

[3] Paul Barford and Vinod Yegneswaran. An inside look at botnets. In *Malware Detection*, pages 171–191. Springer, 2007.

[4] OECD. Malicious software (malware): A security threat to the internet economy, 2008. URL http://www.oecd.org/dataoecd/53/34/40724457.pdf. Accessed: 2014-07-30.

[5] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.

[6] Jaikumar Vijayan. Teen used botnets to push adware to hundreds of thousands of PCs. URL http://www.computerworld.com/s/article/9062839/Teen_used_botnets_to_push_adware_to_hundreds_of_thousands_of_PCs. Accessed: 2014-07-30.

[7] Botnets 101: What they are and how to avoid them. URL http://www.fbi.gov/news/news_blog/botnets-101/botnets-101-what-they-are-and-how-to-avoid-them. Accessed: 2014-07-30.

[8] Kelly Jackson Higgins. Crimeware toolkits driving most online malware, January 2011. URL http://www.darkreading.com/attacks-breaches/crimeware-toolkits-driving-most-online-malware/d/d-id/1135076. Accessed: 2014-07-30.

[9] Jeremy Kirk. Passwords reset after 'pony' botnet stole 2 million credentials, December 2013. URL http://www.pcworld.com/article/2069260/passwords-reset-after-pony-botnet-stole-2-million-credentials.html. Accessed: 2014-07-30.

[10] Brian Krebs. Bringing botnets out of the shadows, March 2006. URL http://www.washingtonpost.com/wp-dyn/content/article/2006/03/21/AR2006032100279.html. Accessed: 2014-07-30.

[11] Anatomy of a botnet, 2013. URL http://www.fortinet.com/sites/default/files/whitepapers/Anatomy-of-a-Botnet-WP.pdf. Accessed: 2014-07-30.

[12] Lucian Constantin. Possibly related DDoS attacks cause DNS hosting outages, June 2013. URL http://www.pcworld.com/article/2040766/possibly-related-ddos-attacks-cause-dns-hosting-outages.html. Accessed: 2014-07-30.

[13] Dancho Danchev. DIY malicious domain name registering service spotted in the wild, December 2012. URL http://www.webroot.com/blog/2012/12/03/

diy-malicious-domain-name-registering-service-spotted-in-the-wild/. Accessed: 2014-07-30.

[14] Cathal Mullaney. Morto worm sets a (DNS) record, 2011. URL http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record. Accessed: 2014-07-30.

[15] OpenDNS.com. The role of DNS in botnet command & control, 2012. URL http://info.opendns.com/rs/opendns/images/OpenDNS_SecurityWhitepaper-DNSRoleInBotnets.pdf. Accessed: 2014-07-30.

[16] Christian J. Dietrich, Christian Rossow, Felix C. Freiling, Herbert Bos, Maarten van Steen, and Norbert Pohlmann. On botnets that use DNS for command and control. In *European Conference on Computer Network Defense (EC2ND '11)*, pages 9–16, Gothenburg, Germany, September 2011. doi: http://dx.doi.org/10.1109/EC2ND.2011.16.

[17] The state of malware 2013. Mcafee, January 2013. URL http://www.mcafee.com/us/resources/misc/infographic-state-of-malware.pdf. Accessed: 2014-07-30.

[18] D. Oro, J. Luna, T. Felguera, M. Vilanova, and J. Serna. Benchmarking IP blacklists for financial botnet detection. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 62 –67, aug. 2010. doi: 10.1109/ISIAS.2010.5604040.

[19] Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shades of grey: On the effectiveness of reputation-based blacklists. In *3rd International Conference on Malicious and Unwanted Software, MALWARE 2008.*, pages 57–64. IEEE, 2008.

[20] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. An analysis of conficker$\check{S}$s logic and rendezvous points. Technical report, 2009.

[21] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

[22] Paul Royal. Analysis of the kraken botnet. Technical report, 2008.

[23] Know your enemy: Fast-Flux service networks. URL `http://www.honeynet.org/papers/ff/`. Accessed: 2014-07-30.

[24] Nicolas Falliere and Eric Chien. Zeus: King of the bots, 2009. URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf`. Accessed: 2014-07-30.

[25] Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of Peer-to-Peer based botnets: a case study on storm worm. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[26] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: detecting the rise of DGA-based malware. In *USENIX Security Symposium*, pages 491–506, Bellevue, WA, USA, August 2012.

[27] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries.

In *ACM Computer and Communications Security*, pages 211–222, Alexandria, VA, USA, October 2008. doi: http://dx.doi.org/10.1145/1455770.1455798.

[28] David Dagon, Niels Provos, Christopher P Lee, and Wenke Lee. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Network and Distributed System Security Symposium*, San Diego, California, USA, February 2008.

[29] Tom van Leijenhorst, Darryn Lowe, and KW Chin. On the viability and performance of DNS tunneling. In *Conference on Information Technology and Applications*, Cairns, Queensland, Australia, 2008.

[30] Elias Bou-Harb, Nour-Eddine Lakhdari, Hamad Binsalleeh, and Mourad Debbabi. Multidimensional investigation of source port 0 probing. *Digital Investigation*, 11: S114–S123, 2014.

[31] Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320. ACM, 2011.

[32] Ned Moran and Nart Villeneuve. Hand me downs: Exploit and infrastructure reuse among APT campaigns, 09 2003. URL `http://www.fireeye.com/blog/technical/cyber-exploits/2013/09/hand-me-downs-exploit-and-infrastructure-reuse-among-apt-campaigns.html`. Accessed: 2014-07-30.

[33] Security Information Exchange (SIE), Farsight Security Inc. URL `https://www.farsightsecurity.com`. Accessed: 2014-07-30.

[34] Hamad Binsalleeh, Thomas Ormerod, Amine Boukhtouta, Prosenjit Sinha, Amr Youssef, Mourad Debbabi, and Lingyu Wang. On the analysis of the Zeus botnet

crimeware toolkit. In *Conference on Privacy Security and Trust*, pages 31–38, Ottawa, Ontario, Canada, August 2010.

[35] Hamad Binsalleeh, A Mert Kara, Amr Youssef, and Mourad Debbabi. Characterization of covert channels in DNS. In *New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on*, pages 1–5. IEEE, 2014.

[36] A. Mert Kara, Hamad Binsalleeh, Mohammad Mannan, Amr Youssef, and Mourad Debbabi. Detection of malicious payload distribution channels in DNS. In *Communication and Information Systems Security Symposium (ICC), 2014 IEEE International Conference on*. IEEE, 2014.

[37] Farkhund Iqbal, Hamad Binsalleeh, Benjamin Fung, and Mourad Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *digital investigation*, 7(1):56–64, 2010.

[38] Farkhund Iqbal, Hamad Binsalleeh, Benjamin Fung, and Mourad Debbabi. A unified data mining solution for authorship analysis in anonymous textual communications. *Information Sciences*, 231:98–112, 2013.

[39] Thomas Ormerod, Lingyu Wang, Mourad Debbabi, Amr Youssef, Hamad Binsalleeh, Amine Boukhtouta, and Prosenjit Sinha. Defaming botnet toolkits: A bottom-up approach to mitigating the threat. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 195–200. IEEE, 2010.

[40] Hamad Binsalleeh, Noman Mohammed, Parminder S Sandhu, Feras Aljumah, and

Benjamin CM Fung. Using RFID tags to improve pilgrimage management. In *Innovations in Information Technology, 2009. IIT'09. International Conference on*, pages 1–5. IEEE, 2009.

[41] Provos Niels, Mavrommatis Panayiotis, Rajab Moheeb Abu, and Monrose Fabian. All your iframes point to us. In *SS'08: Proceedings of the 17th conference on Security symposium*, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

[42] Brian Krebs. Storm worm Dwarfs world's top supercomputers. URL `http://blog.washingtonpost.com/securityfix/2007/08/storm_worm_dwarfs_worlds_top_s_1.html`. Accessed: 2014-07-30.

[43] CRM Today. Financial insights evaluates impact of phishing on retail financial institutions worldwide, 2004. URL `http://www.crm2day.com/content/t6_librarynews_1.php?news_id=EplAlZlEVFjAwhYlkt`. Accessed: 2014-07-30.

[44] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-Peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–8, Berkeley, CA, USA, 2007. USENIX Association.

[45] Robert Lemos. Bot software looks to improve peerage. URL `http://www.securityfocus.com/news/11390`. Accessed: 2014-07-30.

[46] Ryan Vogt, John Aycock, and Jr. Michael J. Jacobson. Army of botnets. In *14th Annual Network and Distributed System Security Symposium*, pages 111–123, 2007.

[47] Paul Mockapetris. Domain names: concepts and facilities. RFC 1034, November 1987.

[48] Wesley Hardaker. Requirements for management of name servers for the DNS. RFC 6168, May 2011.

[49] Paul Mockapetris. Domain names: implementation and specification. RFC 1035, Internet standard, November 1987.

[50] Florian Weimer. Passive DNS replication. In *17th FIRST Conference on Computer Security Incident*, Singapore, June 2005.

[51] Rich Rosenbaum. Using the domain name system to store arbitrary string attributes. RFC 1464, May 1993.

[52] Jianda Li, Bharath Kumar Chandrasekhar, and Kong Yew Chan. Updating of malicious code patterns using public DNS servers. Patent no. US8171467 B1, Filed July 3th, 2007, Issued May 1st, 2012.

[53] John Gordon. Systems and methods for identifying a network. Patent no. US8353007 B2, Filed October 13th, 2009, Issued January 8th, 2013.

[54] Lars Rasmusson and Sverker Jansson. Simulated social control for secure internet commerce. In *Proceedings of the 1996 workshop on New security paradigms*, pages 18–25. ACM, 1996.

[55] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[56] Audun Jøsang and Jochen Haller. Dirichlet reputation systems. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 112–119. IEEE, 2007.

[57] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[58] Jose Nazario. Blackenergy DDoS bot analysis. Technical report, Arbor Networks, 2007.

[59] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 10–18, Berkeley, CA, USA, 2007. USENIX Association.

[60] Neil Daswani and Michael Stoppelman. The anatomy of clickbot.A. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 11–22, Berkeley, CA, USA, 2007. USENIX Association.

[61] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran. A multi-perspective analysis of the storm (peacomm)worm. Technical report, Computer Science Laboratory, SRI International, 2007.

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

[63] David Dittrich and Sven Dietrich. P2P as botnet command and control: a deeper insight. In *3rd International Conference on Malicious and Unwanted Software (MALWARE)*, pages 41–48, Piscataway, NJ, USA, 7-8 Oct. 2008 2008. Appl. Phys. Lab., Univ. of Washington, Washington, DC, USA, IEEE.

[64] David Dittrich and Sven Dietrich. Discovery techniques for P2P botnets. *Stevens Institute of Technology CS Technical Report 2008*, 4, 2008.

[65] Maria Konte, Nick Feamster, and Jaeyeon Jung. Dynamics of online scam hosting infrastructure. In *Passive and Active Network Measurement*, pages 219–228. Springer, 2009.

[66] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *USENIX Security Symposium*, pages 95–110, 2010.

[67] Luca Invernizzi, Sung-Ju Lee, Stanislav Miskovic, Marco Mellia, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, and Giovanni Vigna. Nazca: Detecting malware distribution in large-scale networks. pages 1–16, 2014.

[68] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Connected colors: Unveiling the structure of criminal networks. In *Research in Attacks, Intrusions, and Defenses*, pages 390–410. Springer, 2013.

[69] DNSBL. DNS Blacklists Information. URL `http://www.dnsbl.info/`. Accessed: 2014-07-30.

[70] SORBS. Spam and open relay blocking system. URL `http://www.sorbs.net/`. Accessed: 2014-07-30.

[71] SpamHaus. The spamhaus project, Feb 2014. URL `http://www.spamhaus.org/`. Accessed: 2014-07-30.

[72] SpamCop. Spam reporting service. URL `http://www.spamcop.net/`. Accessed: 2014-07-30.

[73] Christian J. Dietrich and Christian Rossow. Empirical research of ip blacklists. In Norbert Pohlmann, Helmut Reimer, and Wolfgang Schneider, editors, *ISSE 2008 Securing Electronic Business Processes*, pages 163–171. Vieweg+Teubner, 2009. ISBN

978-3-8348-9283-6. URL http://dx.doi.org/10.1007/978-3-8348-9283-6_17.

[74] Jian Zhang, Phillip A Porras, and Johannes Ullrich. Highly predictive blacklisting. In *USENIX Security Symposium*, pages 107–122, 2008.

[75] J. Ullrich. Dshield global worst offender list. URL http://dshield.org/. Accessed: 2014-07-30.

[76] Fabio Soldo, Anh Le, and Athina Markopoulou. Predictive blacklisting as an implicit recommendation system. In *proceedings of INFOCOM 2010, IEEE*, pages 1–9. IEEE, 2010.

[77] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, pages 6–14, 2010.

[78] Kazumichi Sato, Keisuke Ishibashi, Tsuyoshi Toyono, Haruhisa Hasegawa, and Hideaki Yoshino. Extending black domain name list by using co-occurrence relation between dns queries. *IEICE transactions on communications*, 95(3):794–802, 2012.

[79] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *USENIX Security Symposium*, pages 273–290, Washington, DC, USA, August 2010.

[80] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2011.

[81] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou II, and David Dagon. Detecting malware domains at the upper DNS hierarchy. In *USENIX Security Symposium*, pages 27–43, 2011.

[82] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Understanding the prevalence and use of alternative plans in malware with network games. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 1–10. ACM, 2011.

[83] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 231 –240, December 2009. doi: 10.1109/ACSAC.2009.29.

[84] A.J. Kalafut, C.A. Shue, and M. Gupta. Malicious hubs: Detecting abnormally malicious autonomous systems. In *proceedings of INFOCOM 2010, IEEE*, pages 1 –5, March 2010. doi: 10.1109/INFCOM.2010.5462220.

[85] M Patrick Collins, Timothy J Shimeall, Sidney Faber, Jeff Janies, Rhiannon Weaver, Markus De Shon, and Joseph Kadane. Using uncleanliness to predict future botnet addresses. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104. ACM, 2007.

[86] Francesco Roveta, Giorgio Caviglia, Luca Di Mario, Stefano Zanero, Federico Maggi, and Paolo Ciuccarelli. Burn: Baring unknown rogue networks. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pages 6–16. ACM, 2011.

[87] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting Fast-Flux service networks. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, pages 1–12, 2008.

[88] P. Mockapetris. Domain names - implementation and specification. Technical report, November 1987.

[89] Jose Nazario and Thorsten Holz. As the net churns: Fast-Flux botnet observations. pages 24 – 31, Alexandria, VA, United states, 2008.

[90] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. FluXOR: Detecting and monitoring Fast-Flux service networks. In *Detection of intrusions and malware, and vulnerability assessment*, pages 186–206. Springer, 2008.

[91] Alper Caglayan, Mike Toothaker, Dan Drapeau, Dustin Burke, and Gerry Eaton. Real-time detection of Fast Flux service networks. *Proceedings - Cybersecurity Applications and Technology Conference for Homeland Security, CATCH 2009*, pages 285 – 292, 2009.

[92] Kui Xu, Patrick Butler, Sudip Saha, and Danfeng Yao. DNS for massive-scale command and control. *IEEE Transactions on Dependable and Secure Computing*, 10(3): 143–153, 2013. doi: http://dx.doi.org/10.1109/TDSC.2013.10.

[93] Daan Raman, Bjorn De Sutter, Bart Coppens, Stijn Volckaert, Koen De, Pieter Danhieux Bosschere, and Erik Van Buggenhout. DNS tunneling for network penetration. In *International Conference on Information Security and Cryptology (ICISC)*, pages 65–77, Seoul, Korea, November 2012.

[94] Dušan Bernát. Domain name system as a memory and communication medium. In *SOFSEM 2008: Theory and Practice of Computer Science*, volume 4910, pages

560–571. Springer, 2008. ISBN 978-3-540-77565-2. doi: http://dx.doi.org/10.1007/978-3-540-77566-9_49.

[95] Seth Bromberger. DNS as a covert channel within protected networks, 2011. URL `http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/DNS_Exfiltration_2011-01-01_v1.1.pdf`. Accessed: 2014-07-30.

[96] Alessio Merlo, Gianluca Papaleo, Stefano Veneziano, and Maurizio Aiello. A comparative performance evaluation of DNS tunneling tools. In *Conference on Computational Intelligence in Security for Information Systems*, pages 84–91, Torremolinos-Málaga, Spain, 2011.

[97] Lucas Nussbaum, Pierre Neyron, and Olivier Richard. On robust covert channels inside DNS. In *Information Security Conference*, volume 297, pages 51–62. Pafos, Cyprus, May 2009. doi: http://dx.doi.org/10.1007/978-3-642-01244-0_5.

[98] Kenton Born and David Gustafson. Detecting DNS tunnels using character frequency analysis. In *Annual Security Conference*, Las Vegas, NV, USA, April 2010.

[99] Cheng Qia, Xiaojun Chenb, Cui Xud, Jinqiao Shia, and Peipeng Liub. A bigram based real time DNS tunnel detection approach. In *Information Technology and Quantitative Management (ITQM)*, pages 852–860, Suzhou, China, May 2013.

[100] Hyunsang Choi, Hanwoo Lee, Heejo Lee, and Hyogon Kim. Botnet detection by monitoring group activities in DNS traffic. In *Conference on Computer and Information Technology*, pages 715–720, Aizu-Wakamatsu, Fukushima, Japan, October 2007. doi: http://dx.doi.org/10.1109/CIT.2007.90.

[101] S. Marchal, J. Francois, C. Wagner, R. State, A. Dulaunoy, T. Engel, and O. Festor. DNSSM: A large scale passive DNS security monitoring framework. In *Network*

Operations and Management Symposium (NOMS '12), pages 988–993, Maui, HI, USA, April 2012. doi: http://dx.doi.org/10.1109/NOMS.2012.6212019.

[102] Thorsten Holz, Markus Engelberth, and Felix Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. *Computer Security ESORICS 2009*, pages 1–18, 2009.

[103] Top-10 botnet outbreaks in 2009. URL `http://blog.damballa.com/?p=569`. Accessed: 2014-07-30.

[104] Banking malware Zeus sucessfully bypasses anti-virus detection. URL `http://www.ecommerce-journal.com/news/18221_zeus_increasingly_avoids_pcs_detection`. Accessed: 2014-07-30.

[105] Zeus, king of the underground crimeware toolkits. URL `http://www.symantec.com/connect/blogs/zeus-king-underground-crimeware-toolkits`. Accessed: 2014-07-30.

[106] IDAPro - Multi-processor disassembler and debugger. URL `http://www.hex-rays.com/idapro/`. Accessed: 2014-07-30.

[107] PaiMei - a reverse engineering framework. URL `http://code.google.com/p/paimei/`. Accessed: 2014-07-30.

[108] OUTPOST firewal from agnitum. URL `http://www.agnitum.com/`. Accessed: 2014-07-30.

[109] ZoneAlarm personal firewal. URL `http://www.zonealarm.com/`. Accessed: 2014-07-30.

[110] IDAPython: an IDA Pro plugin. URL `http://d-dome.net/idapython/`. Accessed: 2014-07-30.

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

[112] Richard Ford and Sarah Gordon. Cent, five cent, ten cent, dollar: hitting botnets where it really hurts. In *NSPW '06: Proceedings of the 2006 workshop on New security paradigms*, pages 3–10, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-923-4. doi: http://doi.acm.org/10.1145/1278940.1278942.

[113] Vesselin Bontchev. Current status of the caro malware naming scheme. In *15th Virus Bulletin Conference*, October 2005.

[114] Nick FitzGerald. A virus by any other name: Towards the revised caro naming convention. *Proc. AVAR*, pages 141–166, 2002.

[115] Virus Total. URL `http://www.virustotal.com/`. Accessed: 2014-07-30.

[116] Ben Stock, Jan Gobel, Markus Engelberth, Felix C Freiling, and Thorsten Holz. Walowdac-analysis of a peer-to-peer botnet. In *2009 European conference on Computer Network Defense (EC2ND)*, pages 13–20. IEEE, 2009.

[117] Alberto Dainotti, Alistair King, Ferdinando Papale, Antonio Pescape, et al. Analysis of a/0 stealth scan from a botnet. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 1–14. ACM, 2012.

[118] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 30(1):31–48, 2008.

[119] MG Everett and SP Borgatti. The dual-projection approach for two-mode networks. *Social Networks*, 35(2):204–210, 2013.

[120] Ravi Kumar, Andrew Tomkins, and Erik Vee. Connectivity structure of bipartite graphs via the KNC-plot. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 129–138. ACM, 2008.

[121] Stanley Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[122] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, 2010.

[123] S Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[124] ThreatTrack Security Inc. URL `http://www.threattracksecurity.com/`. Accessed: 2014-07-30.

[125] Neo4j - the world's leading graph database. URL `http://www.neo4j.org/`. Accessed: 2014-07-30.

[126] Geolocation and online fraud prevention from maxmind. URL `http://www.maxmind.com`. Accessed: 2014-07-30.

[127] WHOIS Domain Tools. URL `http://whois.domaintools.com/`. Accessed: 2014-07-30.

[128] Web of Trust (WOT). URL `https://www.mywot.com/`. Accessed: 2014-07-30.

[129] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[130] Zeus Tracker. Zeus Tracker. URL `https://zeustracker.abuse.ch/`. Accessed: 2014-07-30.

[131] Malware Domain List. Malware Domain List. URL `http://www.malwaredomainlist.com/hostslist/mdl.xml`. Accessed: 2014-07-30.

[132] SiteAdvisor software for safety ratings. URL `http://www.siteadvisor.ca/`. Accessed: 2014-07-30.

[133] Paul Royal. Maliciousness in top ranked Alexa domains. Barracuda labs, March 2012. URL `http://barracudalabs.com/2012/03/maliciousness-in-top-ranked-alexa-domains/`. Accessed: 2014-07-30.

[134] Alexa the web information company. URL `http://www.alexa.com/`. Accessed: 2014-07-30.

[135] Quantcast. URL `https://www.quantcast.com/`. Accessed: 2014-07-30.

[136] Moheeb Abu Rajab, Lucas Ballard, Noé Lutz, Panayiotis Mavrommatis, and Niels Provos. CAMP: Content-Agnostic Malware Protection. In *Proceedings of Annual Network and Distributed System Security Symposium, NDSS*, 2013.

[137] Duen Horng Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. Polonium: Tera-scale graph mining and inference for malware detection. In *SIAM International Conference on Data Mining (SDM)*, pages 131–142, 2011.

[138] Paul Vixie. Extension mechanisms for DNS (EDNS0). RFC 2671, August 1999.

[139] M. Wong and Wayne Schlitt. Sender Policy Framework (SPF) for authorizing use of domains in e-mail, version 1. RFC 4408, experimental, April 2006.

[140] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. Domainkeys identified mail (DKIM) signatures. RFC 4871, May 2007.

[141] Michael Richardson and D.H. Redelmeier. Opportunistic encryption using the internet key exchange (IKE). RFC 4322, December 2005.

[142] John Levine. DNS blacklists and whitelists. RFC 5782, February 2010.

[143] Rod Rasmussen and Paul Vixie. Surveying the DNS threat landscape, 2013. URL http://www.internetidentity.com/white-papers. Accessed: 2014-07-30.

[144] Spyeye and Zeus malware: Married or living separately?, 2011. URL http://threatpost.com/spyeye-and-zeus-malware-married-or-living-separately-101411. Accessed: 2014-07-30.

[145] Nijhof. Element53: DNS tunneling application for android. URL http://www.nijhof.biz/pages/project/171/Element53. Accessed: 2014-07-30.

[146] Botnet intelligence reviews: Rustock, March 2011. URL http://www.microsoft.com/security/sir/story/default.aspx#!rustock. Accessed: 2014-07-30.

[147] Microsoft, financial services and others join forces to combat massive cybercrime ring. Microsoft. URL http://www.microsoft.com/en-us/news/press/2013/jun13/06-05dcupr.aspx. Accessed: 2014-07-30.

[148] Brian Foster. Three reasons why botnet takedowns are ineffective, May 2014. URL https://blog.damballa.com/archives/2195. Accessed: 2014-07-30.