ON THE SCALABLE GENERATION OF CYBER THREAT INTELLIGENCE FROM PASSIVE DNS STREAMS

Anhar Haneef

A thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science in Information System Security

Concordia University

Montréal, Québec, Canada

January 2016

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Anhar Haneef**

Entitled: **On the Scalable Generation of Cyber Threat Intelligence from Passive DNS Streams**

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

Master of Applied Science in Information System Security

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

Signed by the final examining commitee:

Dr. Amin Hammad _____ Chair

Dr. Amr Youssef _____ Examiner

Dr. Walaa Hamouda _____ Examiner

Dr. Mourad Debbabi _____ Supervisor

Approved _____

Chair of Department or Graduate Program Director

_____ 20 _____ _____

Dr. Amir Asif, Dean

Faculty of Engineering and Computer Science

Abstract

On the Scalable Generation of Cyber Threat Intelligence from Passive DNS Streams

Anhar Haneef

Domain Name System (DNS) has become an important element of recent cybercrime infrastructures. Indeed, DNS protocol is being used, for instance, to operate infected machines and transport malicious payloads. In this context, it is of paramount importance to analyze passive DNS streams in order to generate timely and relevant cyber threat intelligence that can be used to detect, prevent and attribute cyber attacks. In this thesis, we explore the analysis of the aforementioned streams in order to detect DNS anomalies that correspond to cyber incidents. By DNS anomaly, we mean any deviation from what is expected in terms of regular DNS activities (queries/responses). The identification of these anomalies leads to precious intelligence that could pinpoint domains that are involved in malicious activities (e.g., spamming, botnets, phishing, DDoS, etc.). We propose, design and implement a system that analyzes, in near-real-time, passive DNS streams and generates cyber threat intelligence in terms of: suspicious domains, DNS record abuse and passive DNS anomalies. We correlate the generated intelligence with other sources of intelligence such as our malware database. We dedicate a special care to the scalability of the proposed system. In addition to picking appropriate data structures and database technologies, we proceed with the distribution of the analysis over a cluster of computers using the so-called map/reduce paradigm with the Apache Spark framework. Our experiments show that our system is efficient and scalable while generating important, relevant and timely cyber threat intelligence.

Acknowledgments

It is my pleasure to thank all those who made this thesis possible.

First, I would like to thank Allah for helping me to complete this thesis and for guiding me through the way.

It would not have been possible to complete this thesis without the support of my supervisor and mentor Dr. Mourad Debbabi. I would like to thank him for introducing me to academic research. His excellent guidance, patience at all time, encouragement as well as academic advice played an important role to make this research possible.

I take this opportunity to thank my lab colleagues for their support and friendship. I would like to extend special thanks to Amine Boukhtouta with whom I closely collaborated in my research. He provided invaluable assistance and great support during my program.

Last but not the least, my deepest acknowledge goes to my family, my parents for their constant and invaluable support from my early age, my husband Abdulaziz Gassas for his help, patience, and encouragement during my study, and my pretty daughter Lamar. Also, I should not forget my brothers and sisters for their endless love and encouragement. I would like also to thank Saudi Arabia Ministry of Education for providing me all the necessary financial support during my research under the scholarship of Umm Al-Qura University.

Contents

List of Figures

List of Tables

List of Acronyms

2LD Second-Level Domain

ASN Autonomous System

AuthNS Authoritative Name Server

BGP Border Gateway Protocol

CDN Content Delivery Network

DDoS Distributed Denial of Service

DGA Domain Generation Algorithm

DKIM Domain-key Identified Mail

DMARK Domain-based Message Authentication, Reporting and Conformance

DNS Domain Name System

DNSBL DNS-based Blackhole List

EDNS Extension Mechanisms for DNS

HDFS Hadoop File System

IP Internet Protocol

ISP Internet Service Provider

NCFTA National Cyber-Forensics & Training Alliance

P2P Peer-to-Peer

PPM Prediction per Partial Matching

RDD Resilient Distributed Dataset

RRs Resource Records

SIE Security Information Exchange

SPF Sender Policy Framework

TLD Top Level Domain

TTL Time to Live

VMM Variable order Markov Model

XSS Cross-Site Scripting

Chapter 1

Introduction

1.1 Motivations

The Domain Name System (DNS) plays an important role in most Internet activities. It provides a core service to Internet applications by translating domain names into IP addresses. Therefore, Internet users only have to memorize domain names without the need to remember the IP addresses, which are indeed more difficult to remember. DNS protocol is critical in the operations of cyber infrastructures including those that are used for malicious purposes. In addition, there is an increasing evidence that demonstrates the abuse of DNS to conduct malicious operations. For instance, botnets use malicious domains as hosts for commands and control servers and change these domains frequently to remain stealthy. Moreover, cyber criminals use fast-flux networks to frequently change the IP addresses of the malicious domains to evade detection. In addition, DNS spoofing is another example of malicious use of DNS. It replaces the IP address of a benign domain with the IP address of the attacker's server.

By doing so, the users requesting access to a legitimate domain will be redirected to a malicious server. Another example of DNS abuse is the use of the DNS protocol to transport a malicious payload (e.g., upload of the configuration of a bot) or to infiltrate/exfiltrate sensitive information.

Nowadays, there are many attacks that are leveraging the DNS protocol. For instance, DNS resolvers can be used to amplify distributed denial of service attacks that are called Distributed Reflection Denial of Service (DRDoS) attacks. An infamous illustration of such attack is the targeting of Spamhaus in March 2013 with a DNS-amplified DRDoS [42]. Spamhaus is an anti-spam non-profit organization that provides large spam blacklists for research institutions and Internet Service Providers (ISPs). It was one of the biggest DDoS attacks in history with a speed that is up to 300 Gbps of DNS traffic. The attackers were sending `ANY` DNS requests to a large number of open DNS resolvers. They spoofed the IP address of Spamhaus as the source of the requests, which generates a large number of responses that targeted Spamhaus servers. This attack did not only affect Spamhaus but also affected the entire Internet. Another notorious illustration of cyber attacks that leverage DNS is the so-called `DNSChanger` malware [28]. It modifies the DNS settings of the infected machines by redirecting DNS queries to a malicious DNS server that takes control over all the infected computers' traffic. It hijacked more than 4 million computers in US. To stop this attack, the Federal Bureau of Investigation (FBI) took action by blocking Internet connections of all infected computers. Then, it took down the underlying DNSChanger infrastructure. These two attacks give and idea how DNS could be used and abused for malicious activities and how these activities could globally

affect Internet operations.

DNS traffic is generally considered, by most network and security administrators, to be safe, and therefore, it bypasses most network defenses. Unfortunately, this provides attackers with the opportunity to either instrument or abuse DNS to perpetrate cyber attacks. To mitigate this issue, security experts use static domain name blacklists against such abuse. However, these domain blacklists lack effectiveness in terms of detection as a huge number of new malicious domain names appears on the Internet every day. Some of these domains are even the result of Domain Generation Algorithms (DGAs).

In this context, there is a desideratum that consists of designing and implementing techniques that have the capability of fingerprinting these cyber threats/attacks, on DNS traffic, as soon as they appear. It is important to mention here that these techniques absolutely need to be scalable as they have to deal with very large volumes of DNS traffic in order to correctly identify the previously mentioned malicious activities.

1.2 Objectives

The primary objective of this thesis is to address the analysis of passive DNS streams for the purpose of generating timely, relevant and important threat intelligence on cyber crime incidents. In this respect, our work strives to answer the following questions:

- How to analyze passive DNS streams in order to generate intelligence on cyber

crime incidents?

- How to isolate suspicious/malicious domains, DNS abuse and DNS anomalies?
- What features of the DNS protocol should we rely on to fingerprint these malicious activities?
- How to make this analysis both efficient and scalable?

1.3 Contributions

The main contributions of this thesis are:

- Comparative study of the state-of-the-art approaches in terms of passive DNS analysis. These approaches are analyzed and compared against a set of objective criteria that we have compiled.
- Design and implementation of an efficient and scalable passive DNS analysis technique that provides timely and relevant threat intelligence on different types of malicious activities including: suspicious domains, DNS record abuse, and DNS anomalies.
- Correlation of the generated threat intelligence with other sources of intelligence such as our malware database. The intention is to get additional intelligence such as the identification of the underlying malware samples that are responsible for the observed domains or behaviours.
- Design and implementation of two analysis modes: The first mode, called general detection, aims at analyzing the DNS traffic underlying every observed domain

or IP. The second mode is optimized to deal with domains or IP-blocks that are of interest to the security analyst and that need additional monitoring and analysis.

- Design and implementation of a distribution technique that leverages the computational clustering paradigm using the so-called Apache Spark framework in order to achieve scalability of the analysis of DNS streams.

1.4 Thesis Structure

This thesis is organized as follows: In Chapter 2, we introduce the required background on DNS and passive DNS. Chapter 3 presents a comparative study of the existing approaches to detect malicious domains using passive DNS analysis. In Chapter 4, we present our work regarding the near-real-time detection of anomalies on passive DNS streams. Finally, we present the conclusion of this thesis in Chapter 5.

Chapter 2

Background and Related Work

2.1 Background

This chapter introduces the concepts needed to support the work developed in this thesis. In Section 2.1.1, an overview of the domain name system is provided. In Section 2.1.2, the passive DNS technique is explained.

2.1.1 Domain Name System

Domain Name System (DNS) [37,38] is a protocol designed to map domain names to their corresponding IP addresses. An Internet user initiates a DNS query to obtain the IP address of a specific domain name. For instance, the domain name `www.concordia.ca` is mapped to its IP address 132.205.23.199.

In DNS, all the domain name space information is saved on a distributed database. DNS protocol has the structure of a tree. Each node stores a file about a particular domain name (resource record). A group of nodes represents a zone. A zone is

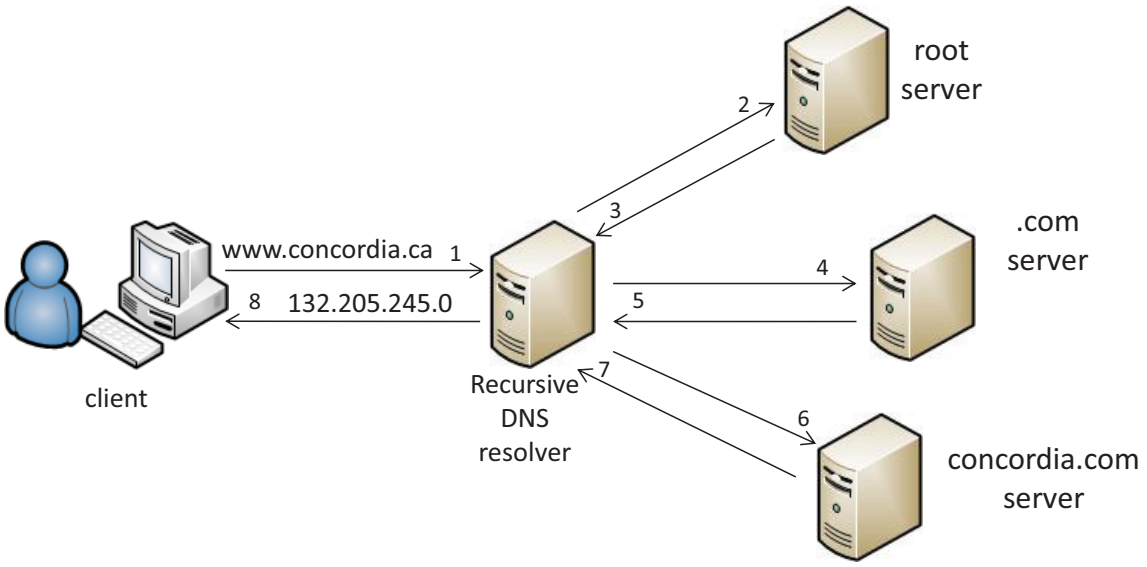

Figure 1: DNS Query Process

authorized by one or more authoritative name servers (AuthNSs). The AuthNSs have full knowledge about the zones. They provide answers about the information stored in the zones. Zone files contain multiple resource records (RRs) used for different purposes. A resource record consists of five parts: name, class, type, time to live (TTL) and data. The RR name is the name of the node of the resource record. Its length is limited to 256 bytes. The RR class defines the protocol family. It usually assigns to IN (Internet) protocol. The RR type describes the usage of the record. Table (1) presents some of RR types used in our work [1].

A DNS query is initiated from a user's machine through its stub resolver. To get the IP address of a specific domain name, the stub resolver is relying on a local DNS resolver to get the required information. The DNS resolver contacts a DNS server in case the cache is empty. The obtained IP address could be cached by the DNS resolver for a specific time (time to live). Therefore, there is no need to interact with

Type	Decimal Value	Description
A	1	IPv4 address: an IPv4 address for a domain name
NS	2	Name server: the authoritative name server for a domain name
CNAME	5	Canonical name: an alias name for a domain name
NULL	10	Null RR: placeholders in some experimental extensions of the DNS
MX	15	Mail exchanger: domain name for a mail server
TXT	16	Text information: Text information associated with a domain name
AAAA	28	IPv6 address: an IPv6 address for a domain name
SRV	33	Service locator: specific services available in the zone
OPT	41	Option: pseudo DNS record type
ANY	255	All cached records: all records types available at the name server

Table 1: DNS Record Types

the DNS server in case the answer is cached by the DNS resolver.

The typical scenario of obtaining the IP address is: The stub resolver sends the query (www.concordia.ca) to the DNS resolver (step 1). If the cache is empty, the DNS resolver sends a query to the root DNS server (step 2). The root server is responsible of all information of authoritative name servers of Top Level Domains (TLDs). The root severs send back to the DNS resolver the IP address of (.ca) TLD servers (step 3). Then, the DNS resolver re-sends the query to the (.ca) TLD server to obtain the IP address of the name server of the Second Level domain (2LD) servers (concordia.ca) and the .ca servers send back the IP address of the authoritative name server of concordia.ca (steps 4-5). In step 6, the DNS resolver queries the authoritative name server of concordia.ca and receives the IP address of concordia.ca (step 7). Finally, the DNS resolver sends the IP address back to the stub resolver (step 8).

2.1.2 Passive DNS

Passive DNS is a technique where DNS queries and responses are captured by sensors and replicated for further analysis. Passive DNS technique was invented by Florian Weimer [55] to reconstruct the structure of the DNS traffic in an easy as well as accessible format. There are many implementations of passive DNS replication technique. One of the most important passive DNS systems is a set of products/services from Farsight Security [16]. The latter provides rich and comprehensive passive DNS channels based on query/response pairs that are exchanged between recursive DNS resolvers and authoritative DNS servers [22]. The passive DNS traffic is captured by deploying sensors above recursive caching DNS servers in multiple networks around the world [22].

DNS-based Security Measures

SPF (Sender Policy Framework) [30] and DKIM (Domain-Key Identified Mail) [13] are exciting email specification frameworks used by different email providers. They protect the domains and domain owners from different kinds of threats such as email spoofing. In addition, They decrease fraud threats by making it difficult for the attacker to forge sender addresses. The domain owners publish specification information in their domain zone as TXT RR because it is flexible in terms of the size and the format.

Domain owners use SPF to identify the mail servers they use to send emails from their domains. When a mail server receives an email that claims to be from a specific domain, it can verify whether the email can be sent by that particular IP address

or not by querying the TXT RR of the 2LD of the email address domain. Domain owners also use DKIM to validate that the incoming email is sent from a specific domain. It is also used to check if the email has been changed during transit. To this end, a digital signature is added to the message's header. The digital signature can be verified through the recipient by using the public key that is stored in the DNS zone file.

DNS Tunneling

DNS tunneling is a technique proposed by Dan Kaminsky [23] to encode the data by using DNS queries and responses. DNS tunneling bypasses most network filtering mechanisms due to DNS protocol property. It embeds the data as sub-domain label in the query. It could be embeded in many sub-domain labels because of the limited size of labels. It should use only the alphabet in the encoded data in DNS query and response. The data is embedded in different types of RRs such as TXT, CNAME and NULL. TXT RR is mostly used due to its features comparing with CNAME and NULL. The format of CNAME RR is restricted while NULL RR could be excluded by DNS servers [32].

In Figure 2, the typical scenario of DNS tunneling is explained. Many requirements are needed to tunnel the data. First, the host and the server should have access to the internet. Second, the attacker should have control over the domain name and the authoritative server of the same domain. The attacker has to add an RR which is NS record to delegate the domain name to another name server. After that, the attacker needs to setup DNS tunneling tool on the host and the server. The host encodes the

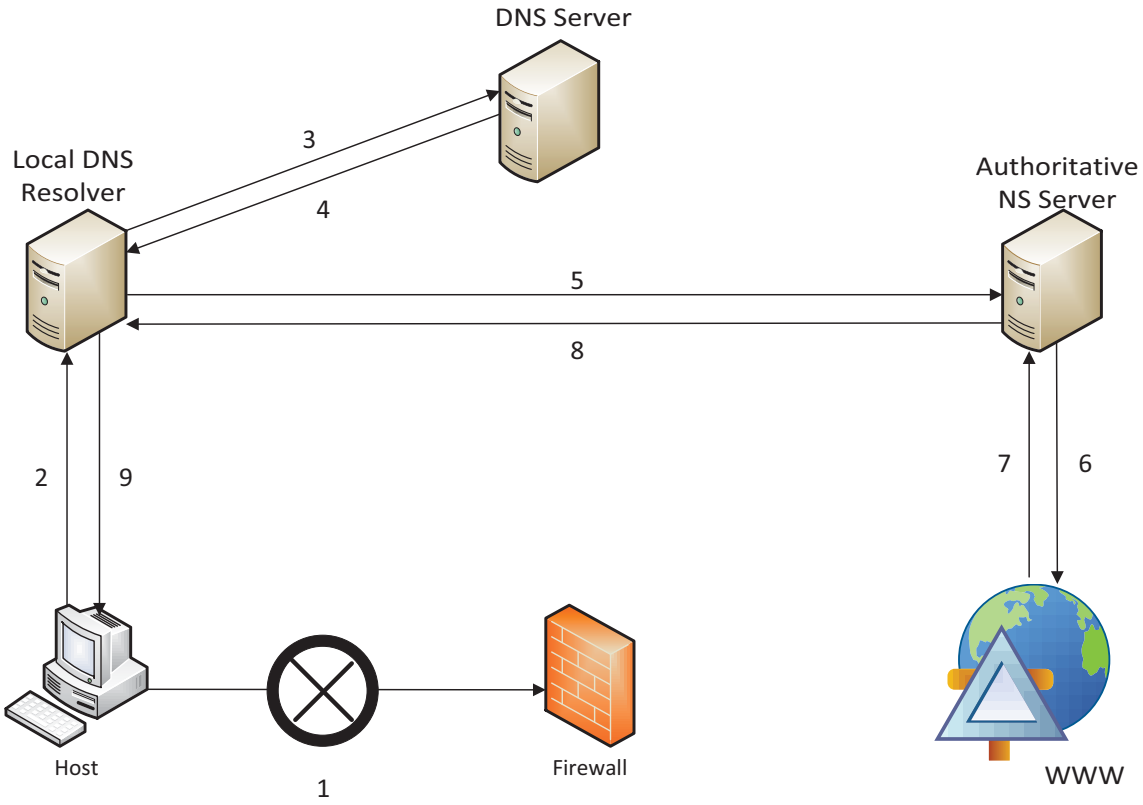

Figure 2: DNS Tunneling Scenario

data in DNS query and sends it to the domain. The server receives the query, decodes the data and sends back the DNS response.

DNS tunneling can be used to deliver any kind of information. However, it has many limitations. First, it is only capable to send a small amount of data due to the limitation of length of the RR query. Second, the size of DNS response is 215 bytes which is small to send enough information. Thus, the attacker may use Extension Mechanisms for DNS (EDNS) to increase the capacity of the DNS response up to 4096 bytes. However, some firewalls do allow bypassing any DNS response that exceeds 215 bytes.

Domain Generation Algorthim (DGA)

Domain Generation Algorithm (DGA) [11] is a technique used by different types of malware families to generate a large number of domains. The generated domains are used to locate command and control (C&C) servers in order to send data or receive commands and updates. Some malware families tend to use old ways to locate and connect C&C servers by embedding a static list of domain names with malware. When the malware is installed on the computers, it tries to connect to the domain names in the list until it finds the C&C server. However, capturing the static list of domain names by security analysts could result in C&C taking down the malware and blacklisting it from connecting with the C&C servers.

To avoid the problems of static lists, the attacker designed an algorithm that takes defined inputs and generates a list of domain names. Then, it attempts to contact each domain name in order to receive updates and commands. Conficker is one of

the earliest malware families that use DGA. Conficker.a and Conficker.b generated 250 domain names over five different TLDs per day. Conficker.D generated 50000 domain names and attempted to connect to 500 domain names per day. After that, this technique has been used by different types of malware families such as Conficker, Murofet, BankPatch, Bonnana and Bobax.

Using domain generation algorithms to generate domain names has many advantages. First, it makes shutting down the domain names difficult for law enforcement due to the large number of domain names and just in time registration. Second, it makes getting the list of all C&C domains difficult for static blacklists. Third, it helps the attacker to avoid network filtering systems [11].

2.2 Related Work

Many techniques have been proposed for detecting malicious activities and distinguishing them from legitimate domains using passive DNS traffic. Some techniques are used to detect malicious domains that are associated with specific types of malicious activities such as fast flux and spam. In [40], Perdisci et al. introduced an approach to detect malicious fast flux services through passive analysis of recursive DNS traces. Unlike the other works that are limited to extract malicious fast flux domains from spam emails [21, 26, 35, 39], Perdisci's approach has the ability to distinguish malicious fast flux domain names from legitimate domains by characterizing features that pinpoint to fast fluxing IPs. In [41], Perdisci et al. extended previous work, where they detect passively flux networks from above local recursive DNS

servers in contrast with the first work, where they used DNS traffic from below recursive DNS servers. In [3], Antonakakis et al. introduced Notos, which is a DNS dynamic reputation system. It differentiates the malicious activities from benign activities using many features. It assigns reputation scores for the new domains based on models of known benign and malicious domains. The score shows if the domain is malicious or benign. It has been deployed in a large ISP's network and be able to find domains before the public blacklist. In another work [7], Bilge et al. introduced a system to detect malicious domains, namely, EXPOSURE. They characterize passive DNS logs to segregate between malicious and benign domains. The segregation is based on 15 features. They conducted an experiment on 100 billion DNS requests and deployed their solution for two weeks in an ISP. They managed to identify malicious domains used in botnet command and control, spamming, and phishing. In another work [8], the same authors extended their initial work, where they deployed EXPOSURE for 17 months and showed the results. In [50], the authors put forward a technique to identify botnets using DNS queries. The system uses Naïve Bayesian classifier to segregate between malicious and benign domain names. The classifier achieves a positive detection rate of 82% and false positive rate of 8.30%. In [58], the authors introduced a technique, which correlates successful and failed DNS queries for the purpose of detecting DGA-based botnets based on the entropy of domain names. In [57], Yadav et al. presented a technique to detect DGA-based botnets by using distribution of unigrams and bigrams for all domains associated with the same IP address, TLD or SLD. In [10], the authors presented a system, namely, BotGAD (Botnet Group Activities Detection). They used an unsupervised approach (X-means

clustering algorithm) to group domain names into clusters. Each cluster has a binary matrix, where rows are hosts sending DNS queries and columns represent time periods. This matrix is used to compute a cosine similarity score to decide if the cluster represents a botnet group or not. In [4], Antonakakis et al. proposed another system called Kopis. It is a system that detects malware using upper DNS hierarchy. It distinguishes between legitimate and malware domains using the global DNS query resolution patterns. In addition, it has the ability to detect malware domains in the absence of IP reputation information. The eight-month experiment shows that Kopis was able to identify new malware domains before the blacklist. In [5], Antonakakis et al. presented a system that detects the DGA-generated domains by analyzing Non-Existent Domain (NXDomain) responses without the need for reverse engineering technique. It uses two algorithms, which are clustering and classification algorithms. The cluster groups similar domains names in terms of structure. The classification algorithm refers the clusters to known DGA models. If there is no model to assign, it generates a new DGA model. The system has been deployed on real time data. It was able to find new DGA families. In [46], Sharifnya et al. proposed a reputation system to detect DGA-based botnets based on the symmetric Kullback-Leibler divergence score to compute reputation of hosts mapping to a large number of suspicious domain names. Their approach marked domain names as dynamically generated if their distribution of uni-grams or bi-grams do not fit the normal distribution. In [47], the same authors proposed a negative reputation system that detects domain flux botnets. Unlike previously cited works, it relies on the history of the large number

of malicious activities to a specific IP address beside the suspicious failures. It assigns a high negative score to the suspicious domains. They performed experiments and managed to detect infected domains. In [25], Kara et al. proposed a detection mechanism for malicious payload distribution channels in DNS. The authors used a significant amount of DNS traffic to identify covert channels that abuse DNS protocol. They proposed a technique that counts the usage of resource records to detect payload distribution channels despite the fact that they have been rarely exploited.

Chapter 3

Comparative Study of Anomalies Detection Systems on Passive DNS

3.1 Introduction

In the past few years, many techniques have been proposed to monitor and analyze passive DNS traffic. These techniques focus on detecting malicious activities including spam domains, fast flux domains, phishing domains, algorithmically generated domain names and botnets using passive DNS traffic. Each approach used different features to detect malicious activities. Some approaches detect different types of suspicious activities and some of them target specific types of suspicious activities. In this chapter, we present the existing approaches that have been conducted in the state-of-the-art on the analysis of passive DNS. In Section 3.2, we present the existing approaches that analyze passive DNS traffic. Section 3.3 presents the comparative study performed on these techniques.

3.2 Presentation of Existing Techniques

The scope of this study covers many different techniques on the detection of malicious activities using passive DNS streams. This section gives a summarization of each technique.

3.2.1 Exposure

Exposure [7] is a novel technique to detect malicious domains based on passive DNS stream. It uses machine learning techniques to build detection models. It uses a set of features that are extracted from DNS traffic to identify the malicious domains and distinguish them from the benign domains. After testing the system for two weeks, it was able to detect malicious domains with high accuracy (98.4%) and low false positive rate (0.5%). Thus, the authors deployed it for 17 months. It detected more than 100K malicious domains.

Exposure has five components: (1) Data Collector, (2) Features Attribution, (3) Malicious-Benign Domains Collector, (4) Learning Module, (5) Classifier. The Data Collector collects the DNS traffic of monitored networks. Then, the Feature Attribution attributes the recorded domains with all sets of features(2). At the same time, Malicious and Benign Domains Collector collects known benign and malicious domains from different sources. Then, it labels the recorded domains as either malicious or benign using known domains. The labeled domains are the input to the Learning Module which trains the data to build the detection module. The unlabeled domains and the detection module are fed to the Classifier. The Classifier classifies

the unlabeled domains into two categories which are malicious domains or benign domains. The system should be trained for seven days before it can be ready for malicious domains detection.

Exposure system applies two filters to reduce the amount of traffic to manageable size without losing any relevant data. The first filter targets popular and well-known domains. This filter removes all queries that are related to domains that are unlikely to be associated with malicious activities such as Alexa top 1000 global sites. The second filter targets the domains that are older than one year. Most of malicious domains are detected and blacklisted after a short time of activity (several months). This means if the domain is older than one year and was not detected by any tool, it is not malicious. These filters removed almost 70% of the traffic which made it more manageable.

Exposure was deployed for two weeks in large ISPs. It was able to detect many unknown malicious domains with a detection rate of (98.4%) and false positive rate of (0.5%). Because of the good results, Exposure was deployed as a free online service for 17 months. It provides a daily list of detected malicious domains (200 malicious domains per day). It detected around 100K malicious domains in total. However, Exposure has limitations. First, it needs to be trained for seven days before it can be used for identifying malicious domains. Second, it cannot detect domains that belong to a malware family if the classifier is not trained for that family.

Feature Set	Feature Name
Time-based Features	Short life Daily similarity Repeating patterns Access ratio
DNS Answers-based Features	Number of distinct IP addresses Number of distinct countries Reverse DNS query results Number of domains share the IP with
TTL Value-based Features	Average TTL Standard Deviation of TTL Number of distinct TTL values Number of TTL change Percentage usage of specific TTL ranges
Domain Name-based Features	% of numerical characters % of the length of the LMS

Table 2: Exposure Features

3.2.2 Notos

Notos [3] is a DNS dynamic reputation system that identifies malicious domains using passive DNS stream. It assigns scores to new domains that indicate if the domain is malicious or benign. High scores are assigned to legitimate domains and low scores are assigned to malicious domains. It uses many features to classify malicious and legitimate domains correctly(3). The system was deployed for real time data and was able to detect malicious domains with high true positive rate (96.8%) and low false positive rate (0.38%).

Notos system has two modes: off-line mode and on-line mode. In the off-line mode, Notos reputation engine builds three modules. The first module is network profiles which extract network-based features of well-known domains (CDN domains, popular websites and dynamic DNS domains). Then, it trains a classifier to identify whether a new domain has similar characteristics to the popular domains or not.

The second model is domain name clusters which build clusters of domains based on network based features and zone-based features. Some clusters contain malicious domains that share similar characteristics and the other clusters contain legitimate domains. The third model is the reputation function. It is a statistical classifier that assigns a score to a new domain based on evidence-based features and the output of network profiles and domain name clusters modules. It assigns a low reputation score to malicious domains such as spam campaigns and C&C servers and assigns a high reputation score to benign domains. Now the training is completed and the system is ready for on-line mode. First, a new domain is sent to network profiles module and the output is the probability that a domain name belongs to well-known domains. Second, the domain name is sent to domains clusters module. The system extracts the features and identifies the nearest cluster. Then, the evidence features are computed to the domain name. Finally, the reputation function calculates the score based on the outputs of the other modules.

Feature Set	**Feature Name**
Network-based Features	BGP features AS features Registration features
Zone-based Features	String features Average, TLD features
Evidence-based features	Blacklist features Honeypot features

Table 3: Notos Features

Notos is able to detect new malicious domains with high accuracy. In addition, it is able to identify malicious domains before they are appearing in public blacklists. However, Notos has some limitations. First, Notos could assign inaccurate scores

to domains that don't have enough passive DNS information. Second, it requires time and large passive DNS traffic to train the system to be able to detect malicious domains correctly. Third, due to the false positive rate, it is better to use Notos in corporation with other defense systems not as a standalone system. Finally, Notos would assign low reputation scores to legitimate domains that are hosted in bad networks, which increases the false positive rates.

3.2.3 Kopis

Kopis [4] is a novel detection system to detect malware domains using passive DNS stream at the upper DNS hierarchy. Because the stream is obtained from the upper DNS hierarchy, the system has a global visibility that makes DNS operators work independently to detect malware domains without needing other data from other networks. It uses many features to detect malware domains including IP-reputation features (4). The system was deployed for 8 months with real data. The result indicates that Kopis has high detection rate (98.4%) and low false positive rate (0.3%-0.5%).

Kopis detection system consists of three components: features computation, learning module and statistical classifier. The features computation is used to compute sets of features for observed domain names in the stream. The features are computed after dividing the stream into epochs (one day). The learning module works in two modes which are training mode and operation mode. In the training mode, the module takes only the computed features from domains that exist in the knowledge base (KB). KB contains a list of known benign domains (Alexa, DNSWL) and known

malicious domains (public blacklists). The module labels each domain's feature vector with legitimate label or malicious label. The statistical classifier is used to learn the behaviors that are related to legitimate and malicious domains. In the operation mode, the statistical classifier assigns label and confidence scores to the domains that are not in the KB at the end of each epoch based on the similarity with known legitimate and malicious domains. After that, the average of the confidence scores of each domain is calculated. If the score is higher than a threshold, it is malicious.

Feature Set	**Feature Name**
Requester Diversity	Mean, standard deviation and variance of BGP prefixes of requesters Mean, standard deviation and variance of AS numbers of requesters Mean, standard deviation and variance of CCs of requesters The number of distinct IP addresses The number of distinct BGP prefixes The number of distinct AS numbers prefixes The number of distinct CCs prefixes
Requester Profile	Average, biased and unbiased standard deviation and biased and unbiased variance of number of domain names queried by specific IP during specific time Average, biased and unbiased standard deviation and biased and unbiased variance of number of domain names queried by specific IP during specific time multiply by the IP's weight
Resolved IPs Reputation	The average number of known malware-related domain names, Spamhaus Block List domains and the Alexa top 30 domain have pointed to IP addresses of given domain name during the last month The average number of known malware-related domain names, Spamhaus Block List domains and the Alexa top 30 domain have pointed to BGP prefixes IP addresses of given domain name during the last month The average number of known malware-related domain names, Spamhaus Block List domains and the Alexa top 30 domain have pointed to AS numbers IP addresses of given domain name during the last month

Table 4: Kopis Features

The features that are used by Kopis detection system fall into three sets: requester diversity (RD), requester profile (RP) and resolved IPs reputation (IPR). First, the requester diversity set of features is used to check diversity of machines that query

a given domain name. The diversity of machines that query malicious domains is different from the diversity of machines that query legitimate domains. Second, the requester profile features' set is used to check whether the revolver's IP address represents home machine, small business or large network. It assigns weights to each resolver based on the number of queries issued by the resolver. If the number is large, the weight is high and vice versa. Malicious domains usually receive queries that are issued from large number of resolvers with high weight while legitimate domains receive queries from revolvers with high weight and resolveres with low weight. Finally, resolved IPs reputation set of features are used to check the history of IP addresses that mapped to a given domain name.

3.2.4 FluxBuster

FluxBuster [41] is a novel technique to detect malicious flux networks through analyzing passive DNS traffic obtained from large numbers of local recursive DNS located in different places. The system consists of many components: message prefiltering to reduce the amount of messages by ignoring the messages that have no relation with the flux domains, domain clustering to group domain names that are mapped to the same group of IP addresses, and classifier to find out which clusters are malicious and which are not. FluxBuster was deployed on real passive DNS traffic for five months. It was able to identify malicious flux networks with a low false positive rate. The malicious flux networks were detected by FluxBuster before public blacklists.

FluxBuster starts by receiving passive DNS stream and aggregating DNS messages that are related to a domain name during a certain time. It summarizes DNS

messages into one message that contains all mapped IP addresses, average TTL and number of messages. Then, it filters the messages based on some flux domain characteristics (short TTL, large numbers of resolved IPs, resolved IPs located in different networks). However, this step does not remove all legitimate domains. Some of legitimate domains that share the characteristics of flux networks still exist. After that, domain clustering is applied to group the domains that have some similarity in terms of their IP addresses. To find the similarity, two factors are computed which are Jaccard index and sigmoidal weight. The second factor is used to measure the confidence in Jaccard index. Each cluster contains one domain at the beginning. Then, every cluster merges with the nearest cluster until one cluster is remained. It represents the merged process as dendrogram and cuts it at certain height. The domains in the connected subgraph after cutting are in the same clusters. After that, FluxBuster use a statistical classifier to detect malicious flux network. Table 5 shows the features that are used by the classifier to detect which cluster is malicious. The classifier is trained on a data set that contains known malicious flux domains and known non-flex legitimate domains. Then, it classifies clusters as malicious or benign based on the computed features set.

FluxBuster has a few limitations. First, each cluster should have at least 30 IP addresses during each epoch to perform classification on it. Second, the system needs to observe at least one DNS query and response to a flux domain to be able to detect it. Finally, FluxBuster needs 30 hours to detect malicious flux networks, which means it will take a long time before taking action and stopping the threats.

Feature Number	Feature Name
1	Number of resolved IPs
2	Number of domains
3	Average TTL per domain
4	Number of domains per network
5	IP diversity
6	IP growth ratio
7	IP last growth ratio
8	IP prefixes last growth ratio
9	Novelty

Table 5: FluxBuster Statistical Features

3.2.5 CROFlux

CROFlux [19] is a detection system used to identify fast flux domains through passive DNS system. The system depends on public knowledge about malware lists. It detects domains with flux characteristics. However, it is able to reduce false positives by discarding benign domains that have similar properties to flux domains such as hosting providers and content delivery networks.

CROFlux system has three components: (1) Prefiltering of collected domains, (2) Candidate domains clustering, (3) Detection of fast flux clusters. The system starts by monitoring DNS traffic and filtering domains. It chooses the domains that meet specific criteria which are: TTL should be less than 3 hours, the number of mapped IP addresses to a domain is more than or equal to 3 and diversity of networks of a domain is larger than 0.333. In addition, it takes a domain with less than 3 mapped IP addresses if the TTL was equal or less than 30 seconds. After that, it calculates the fluxy score for each domain based on the number of distinct IP addresses, number of dynamic IP addresses and number of distinct autonomous systems. If the fluxy score is more than 450, the domain is a fast flux candidate. Then, it clusters the

domains that have common IP addresses. Finally, it identifies fast flux clusters from other clusters by comparing the clusters' domains with private and public malware lists. If the number of malicious domains in a cluster exceeds a threshold, it is a fast flux cluster.

The system collected DNS traffic for four months and detected 265 malware domains with detection rate (91%).

3.2.6 Stalmans and Irwin

The authors in [50] introduced a system to detect fast flux and algorithmically generated domains by monitoring passive DNS traffic. The system identifies different types of features in order to classify domains. The system uses many statistical measures techniques to classify domains as malicious or not. It was able to detect malicious domains with high accuracy and low false positive rate.

The system contains two parts. The first part for detecting fast flux domains and the second part for detecting algorithmically generated domains. For fast flux domains detection, many classifiers were employed and trained based on a set of malicious and benign domains. They used a set of features which are the difference between standard domain, CDN domain and fast flux domain in terms of: (1) number of A records, (2) number of NS records, (3) number of network ranges, (4) number of unique ASNs, (5) average TTL. These features are used to distinguish between benign and malicious domains. For algorithmically generated domains detection, the classifiers were trained based on an alphanumeric character distribution. Legitimate domains are different comparing to algorithmically generated domains in terms of

alphanumeric character distribution.

The system was able to identify fast flux and CDN domains correctly. In addition, it was able to detect algorithmically generated domains with 87% accuracy rate, 82% true positive rate and 8% false positive rate. It was determined that Bayesian classifier is the best choice comparing with other classifiers because it provides high accuracy with low false positive rate. The system increases the defense of the network.

3.2.7 Kara et al.

Kara et al. [25] introduced a system to detect DNS malicious payloads distribution channels using passive DNS traffic based on access counts of RRs. The system uses a set of features that differentiate between benign and malicious domains. It was able to detect unknown malware domains in addition to malicious payload distribution channels. The system was deployed for 30 days with real passive DNS traffic and showed good results.

Each domain name has a zone file that contains different types of RRs. Each zone file has an authoritative name server that is responsible of receiving a DNS query and returning the a DNS response. Attackers should have access to the name server to deliver malicious payloads via DNS responses. To detect malicious payload distribution channels, the system monitors the DNS zone activities and the access counts of each RR. Malicious domains that used to deliver payloads are different from regular domains in terms of received query types. Malicious domains mostly receive queries for RRs that used to deliver payloads such as TXT records while regular domains receive queries for different kinds of RRs such as A, AAAA and

MX records. The system starts by receiving passive DNS traffic and extracting DNS queries that have TXT records. For each domain name, it aggregates DNS messages and forwards them to domain-based queuing. At the same time, the DNS traffic is stored in a database for the purpose of offline analysis. Then, the aggregated messages are sent to the payload distribution detection module to find the malicious payload distribution channels. The system extracts the access count of all RRs from the passive DNS database. For each domain name, it computes the ratio of access count of TXT records to access count of other record types that are used by payload distribution channels for a certain period of time. If the ratio is large, it means the domain is involved in payload distribution activities.

The system applies two filters to remove the legitimate domains that have some similarities with payload distribution channels such as SPF, DKIM, IKE and DNSBL. Because these specifications are designed for mail server, MX record should exist in the zone file. The first filter removes any domain that has defined strings in its TXT records such as SPF or DKIM. The second filter removes any domain that is associated with MX records. These two filters remove non-payload distribution channels.

The system was deployed for one month and was able to detect payload distribution channels after applying the filters. It can detect the payload distribution channels even if they changed their syntax because it is based on the access count of the RRs. However, the system has some limitations. First, if the malicious domain that is used for payload distribution is accessed for different RRs such as A record, the system will consider this domain as non-payload distribution channel. Second, malware could send the queries to an open revolver instead of caching resolver. So, the traffic cannot

be captured by the sensors.

3.2.8 BotGAD

BotGAD [10] (Botnet Group Activity Detector) is a system designed to detect botnets using group activity characteristics by monitoring passive DNS traffic. A botnets group has many activities such as receiving order, downloading updates and performing malicious activities. BotGAD was deployed using real time DNS data and was able to detect botnet domains even if they use evasion techniques such as DGA.

The system consists of five components: data collector, data mapper, correlated domain extractor, matrix generator and similarity analyzer. It starts by collecting and aggregating DNS traffic using data collector. Then, the data mapper stores the information in a hash map structure. It maps a domain name to IP addresses that query the domain name and maps the IP addresses to the time of the query. Then, the correlated domain extractor is used to cluster correlated domains based on sets of features 6. The correlated domains could be domains generated using DGA, domains from hard-coded C&C or domains from spam recipient list. The features that are used for clustering are chosen from three different aspects: (1) DNS lexicology, (2) DNS query information, (3) DNS answer information. At the same time, the system builds a matrix to find temporal similarity of group activity using conventional vector based method. The matrix contains the domain name, the IP addresses that query the domain and the timestamps. The matrix generator marks the element with 1 if the IP address queries the domain at that time and 0 if the IP address queries the domain at that time. Then, the similarity analysis measures the similarity score of

the generated matrix using the cosine similarity coefficient. Finally, hypothesis test is performed to decide whether the group is botnet or not.

BotGAD was deployed and tested on three real time DNS traces including large ISP networks. It was able to detect known and unknown botnets in large networks with more than 95% detection rate, 0.4% false negative rate and 5% false negative rate.

Feature Type	**Feature Name**
DNS lexicology features	Number of domain tokens Average length of domain tokens Longest length of domain token Blacklisted SLD presence IP diversity
DNS query features	Number of queries sent Number of distinct sender IPs Number of distinct sender ASNs Query type (A, NS, CNAME, MX, PTR) Estimated similarity of a domain
DNS answer feature	Number of distinct resolved IPs Number of distinct ASNs of resolved IPs Number of distinct countries of resolved IPs TTL value in a DNS answer packet Estimated similarity of a domain

Table 6: BotGAD Features

3.3 Comparative Study

We compare the aforementioned techniques according to the following criteria:

- **Approach**: Techniques that are used to detect DNS malicious activities such as clustering and classification.

- **Goals**: Objectives of each system that is addressed.

- **Inputs**: Description of different types of system inputs that are used for many purposes, e.g., benign domain list used to train the system.
- **Traffic Level**: Level of traffic where the system performs DNS monitoring.
- **Training Time**: Time that is required to train a system before it becomes able to produce accurate results.
- **Accuracy Rate**: It deals with accuracy of system results.
- **FP Rate**: It presents false positive rate of each system.

We have chosen these criteria since they are important aspects that one should take into consideration to compare the performance and the accuracy of systems. The comparison is presented in Table 7

3.4 Conclusion

We conclude this chapter by making the following observations.

- Most of the techniques [41], [19], [50], [25], [10] are specified to detect a certain type of DNS malicious activities (e.g., botnet or fast flux networks).
- The techniques reported in [7], [3], [4] aim at detecting malicious domains generically and are not limited to certain types of malicious activities. They use classification approaches based on a large amount of training data sets. In addition to the training set, their systems require extensive time for training.
- The training sets in question should be updated and the system should be regularly retrained. Hence, the system could keep up with the behavior of new

malicious domains and malware.

System	Approach	Goals	Inputs	Traffic Level	Training Time	Accuracy Rate	FP Rate
Exposure [7]	Training Classification	Malicious domains detection	DNS traffic Malicious domain list Benign domain list	Local recursive DNS server	Require training (7 days)	98.4	0.5
Notos [3]	Training Classification	Malicious domains detection	DNS traffic Malicious domain list Alexa domain list	Local recursive DNS server	Require training	96.8	0.38
Kopis [4]	Training Classification	Malicious domains detection	DNS traffic Malicious domain list Known legitimate domain list	Upper DNS hierarchy	Require training (5 Days)	98.4	0.5
FluxBuster [41]	Prefiltring Clustering Classification	Malicious flux network detection	DNS traffic Flux networks Nonflux networks	Local recursive DNS server	Require training	High	Low
CROFlux [19]	Prefiltring Clustering	Fastflux domains	DNS traffic Malicious blacklist	Local recursive DNS server	No training	91.0	-
Stalmans and Irwin [50]	Classification	Fastflux domains detection	DNS traffic Malicious data 1000 top visited domains	Local recursive DNS server	Require training	85.0	-
Kara et al [25]	Filtering Analysis	Malicious payload distribution channels detection	DNS traffic	Local recursive DNS server	No training	-	-
BotGAD [10]	Clustring	Botnet detection	DNS traffic Black list White list	Local recursive DNS server	No training	-	-

Table 7: Passive DNS Detection Systems

Chapter 4

Anomalies Detection in Passive DNS

4.1 Introduction

The Internet has undergone more than just being used for standard activities such as simple look-ups of general or scientific topics. Its network expanded and the number of computers and servers connected to, has increased exponentially. It results in being an essential infrastructure for finance, business, communication, research and development. Despite its popularity, the Internet has its own dark side, since it has been turned to be a nest of cyber-crimes including information theft, scams, email spams, malware infections, etc. As a part of the Internet evolution, DNS protocol [37, 38] plays the role of a phone-book for the Internet. This protocol is considered as a vital piece that allows host-names being accessible. However, it is used as an artifact for malicious activities. For instance, domain names are used as proxies and command

& control servers (C&Cs) of malicious networks enclosing infected machines. C&Cs communicate with bots through DNS queries to perpetrate malicious activities like key-logging, spamming and spreading infections through networks. DNS protocol has been also used to conduct reflection DDoS attacks. As such, there is a desideratum in the generation of cyber-threat intelligence based on DNS traffic replica. Thus, some research efforts [3,4,7,8,40] put an emphasis on using passive DNS to detect malicious activities as well as DNS abuses. They use mainly classification techniques to segregate malicious domains from benign domains or to detect fast-flux malicious services. In spite of interesting results obtained by proposed systems in aforementioned works, they have not integrated all-in-one solution to gather threat-intelligence. To this quest, the availability of a tool that detects passive DNS anomalies is of a great help for security experts since it allows to pinpoint abnormal DNS activities (e.g., frequent change of city for a given domain, abrupt changes in DNS query number, detection of new fast fluxing IPs, TTL values change, malicious abuse of DNS records). These abnormal activities are considered as good indicators to detect zero-day attacks, and can be correlated with malware analysis. We perceive our work as being different in the proposition of a system that integrates an all-in-one solution, which has the following capabilities:

- Design and implementation of an efficient and scalable passive DNS analysis technique that provides timely and relevant threat intelligence on different types of malicious activities including: suspicious domains, DNS record abuse, and DNS anomalies.

- Correlation of the generated threat intelligence with other sources of intelligence such as our malware database. The intention is to get additional intelligence such as the identification of the underlying malware samples that are responsible for the observed domains or behaviours.

- Design and implementation of two analysis modes: The first mode, called general detection, aims at analyzing the DNS traffic underlying every observed domain or IP. The second mode is optimized to deal with domains or IP-blocks that are of interest to the security analyst and that need additional monitoring and analysis.

This chapter is organized as follows. Section 4.2 presents the detection of passive DNS anomalies and abuses. The experimental results are discussed in Section 4.3. Finally, concluding remarks are provided together with discussion of future research in Section 4.4.

4.2 pDNS Anomalies and Abuse Detection

In this section, we aim to give an overview of the proposed passive DNS anomalies and abuses detection framework.

4.2.1 Overview

We aim to design a system that monitors all domains and their DNS records as observed in passive DNS stream. We intend to deploy a system that detects in near real-time DNS malicious activities including potential anomalies and abuses of DNS

protocol observed on captured DNS logs. However, we faced some challenges in the design and the implementation of our system. The main challenge is how to handle the high volume of DNS data. We observe in average $38,891.4$ DNS records per second and more than 6 million unique domains per hour. Sometimes, the traffic of DNS data reaches more than 8 millions unique domains per hour. Thus, we need scalable techniques to monitor the huge load of DNS data as well as a reliable storage system for the purpose of identifying the anomalies and storing the results.

Our system targets to: (1) Aggregate real-time DNS data for analysis purposes. (2) Segregate different types of DNS records. (3) Identify suspicious domain names and aliases. (4) Identify DNS records abuse, e.g., *TXT*, *SRV*, *CNAME*, *NULL*, *OPT* and *ANY*. (5) Extract sets of features representing time series analytics. (6) Detect DNS anomalies based on extracted features. (7) Correlate with different sources of cyber-threat intelligence. (8) Monitor and archive all DNS activities related to specific organizations of our interest.

A number of previous research efforts [29, 45, 56, 59] used Apache Spark to analyze the large amount of data. This has motivated us to use it as part of our system. Spark monitors the huge amount of passive DNS data to extract anomalies. Such framework that allows multi-purpose data processing is designed for intensive in-memory and distributed clustering computations. It emphasizes on improving the performance of applications that cannot be expressed efficiently as acyclic data flows, where a working set of data is across multiple parallel operations [60]. It includes two use-cases, namely, iterative jobs and interactive computing, where Hadoop [20] is deficient. Hadoop is a processing model of on-disk data that supports single-pass,

batch. A key asset of Spark lies in introducing the Resilient Distributed Dataset (RDD). RDD is known to ensure the abstraction of data such that large datasets can be cached effectively in memory or disk. RDDs represent immutable collection of objects grouped into partitions. Spark uses RDD to allow re-usability of memory cached objects, which significantly improves performance in work-flow execution.

In addition to in-memory and disk caching, Spark supports many data abstractions, namely, graphs, streaming logs (e.g., Twitter feeds), databases (e.g., MySQL, Cassandra) and hadoop data formats. Moreover, Spark has an elaborated programming model, which was initially integrated by SCALA programming language [14], then wrapped to other languages, i.e., object-oriented programming such as Java [36], script programming such as Python [18], and statistical computing language R [17]. Spark provides programmers with the ability to: (1) construct RDDs from files in a shared file system, (2) divide collections into slices that can be sent to multiple nodes, resulting in computation parallelism, (3) transform data smoothly from one type to another (e.g, a log to a mapping object) and, (4) alter persistence of objects through two actions, namely, cache action and save action. Cache action leaves a dataset lazy but kept in memory for re-usability, and, save action dumps data into a distributed file-system like Hadoop File System (HDFS).

Spark supports also several parallel operations, namely, *Filter*, *Collect*, *Reduce*, *Map-Reduce* and *Foreach*. *Filter* operation allows to eliminate items into collections, that do not satisfy a boolean predicate function. *Collect* operation sends all elements of a dataset to a driver for a parallel gathering of items into a collection (e.g., arrays, lists, etc.). *Reduce* operation combines dataset elements through an associative

function. *Map-Reduce*, known also as grouped reduce function, which allows to map datasets to a common mapping objects like tuples and reduce them by a key entry and an associative function. *Foreach* operation allows a streaming loop through collection to execute functions like exporting results to databases or copying them into shared variables in a program.

4.2.2 System Architecture

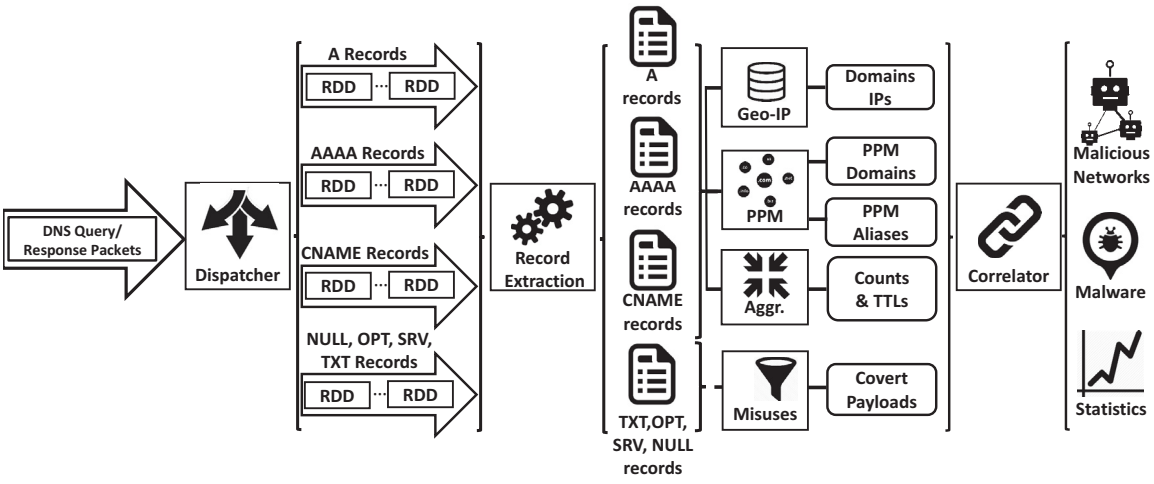

Figure 3: DNS Anomalies Detection Architecture

DNS Eco-system aims to monitor real-time DNS replica logs. These logs contain passive DNS entries enclosing information about different record types (e.g., "A", "AAAA", "CNAME", "TXT", etc). DNS anomalies detection system falls into the

following components: Dispatcher, Record Extraction, Geo-location, Prediction per Partial Matching (PPM) detection, Aggregation, Record Misuses Filter and Correlator. Figure 3 illustrates the architecture of the system.

Dispatcher

The dispatcher component receives DNS logs and transforms them into streams of RDDs. These RDDs are divided into four categories: (1) RDDs containing A records, (2) RDDs containing AAAA records, (3) RDDs containing CNAME records and, (4) RDDs containing records (e.g., "TXT", "OPT", "SRV", "NULL") that are the inputs for DNS misuses like covert channels and tunneling. The dispatcher component employs Spark filter action to check the resource record type and moves it to the RDDs that contain the same type.

Record Extraction

The record extraction component takes RDD streams of each type as inputs and extracts tuples as outputs. This component uses Spark map operation to produce many tuples and push them to the next components that detect DNS anomalies and record misuses.

Geo-location

The geo-location component uses a geo-location database called MaxMind [31] to geo-locate IP addresses that exist in DNS "A" records. Usually, malicious domains are mapped to many infected machines. Since botmasters have no control over the geographical locations of the infected machines, the IP addresses of these machines

are located in different countries, Border Gateway Protocol (BGP) and Autonomous Systems (ASNs). Legitimate domains like `google.com` and `youtube.com` are also mapped IP addresses located in many countries and cities. Therefore, we use on the fly a white-list to remove all these domains and we geo-locate the domains that do not appear in the white-list. The system takes a conservative approach, since it archives all domains changing frequently cities and countries. Then, it cross-validates with other features (IP-based features and TTL-based features) that will be explained in the following sections. We collect the changes of countries and cities and place them into immutable sets labeled by domain names. For this purpose, we use Spark filter action to check if these sets are mapped to one country or city and remove all sets that are mapped to one city or one country. Our system only geo-locate the IPv4 addresses that are collected from "A" records and we decide to do it for IPv6 addresses that are collected from "AAAA" records in the near future.

PPM Detection

The PPM detection component uses a technique proposed by Begleiter et al. [6] to domain fluxing activities and misuse of CNAME records. Domain fluxing is one of the techniques that are used by attackers where the malware change the fully qualified domain names frequently. Such domains are used as C&C that send commands, store stolen information and begin attacks. Domain generation algorithm (DGA) is the simplest way for botnets to achieve domain fluxing through the generation of random domain strings. Canonical name (CNAME) records misuse is another technique used by malware. They map malicious domains to machine generated alias domains to

appear as legitimate domains.

The technique proposed by Begleiter et al. uses Prediction per Partial Matching (PPM) algorithm, which is a type of Variable order Markov Model (VMM) algorithms. PPM algorithm predicts the symbol based on previous symbols. It has two phases, namely: training phase and prediction phase. At the training phase, it builds a tree structure (Trie, in particular), which stores the sub-sequences of the training sequences and the counts of the symbols that appear after them. At the prediction phase, it calculates the probability estimation of sub-sequences of a new sequence and compares it with the estimation of the training set. Our abnormal domains detector computes the probability of a domain name to verify whether it is benign or not. In our case, the training dataset is composed of different domain white-lists (e.g., Alexa top one million domains [2], Quantcast US domains ranking [43]). PPM algorithm has two steps to build the training sequences. First, it reverses each domain name. Second, it adds two delimiters at the beginning of each reversed domain names. For example, *google.com* is reversed to *##moc.elgoog*. The delimiters are used to separate domains, which avoids appearance of noisy context in the training sequence. The domains are reversed to let the training based on domain TLDs. Once the sequences are ready, PPM classifier stores the count of symbols that occurs after sub-sequences of a specific length (distance) in the training sequences.

Then, the classifier is used to calculate the average per symbol probability of the 50% of the training sequences, which are uniformly sampled. The computed average is considered as a pivot score. The reason behind choosing half of the training sequences lies in the fact that the pivot score should represent the average benign domain

name. To determine if a new domain is abnormal or not, we first reverse the domain name and add the delimiters. After that, we use the PPM classifier to compute the average per symbol probability. Then, we calculate the ratio of the average per symbol probability and the pivot score. If the ratio is less than a threshold (0.8), the domains is considered as abnormal. Otherwise, it is considered as benign.

Aggregation

The aggregation component monitors different features of passive DNS records to detect DNS anomalies. Since the records are received each minute, we gather the traffics every minute. After collecting the features for a specified time window (2 hours for specific domains and IPs, 1 hour for the rest of passive DNS records) and having enough samples, our system uses map-reduce Spark action to reduce tuples generated by the record extraction component (See Sub-Section 4.2.2) and groups them according to domain names. Then, we apply anomaly detection techniques to identify malicious behaviours. The aggregation is done on the following features: (1) Number of Queries, (2) Number of IP Changes per Domain and (3) TTL values.

Number of Queries: The number of queries that target a particular domain name over a specific period of time is used as indicator of malicious behaviors such as domain flux and spam campaigns. Benign domains approximately have similar number of requests over time. However, domains flux is a technique where an IP address maps to many suspicious domains. These domains often have abrupt increase in their "A" or "AAAA" queries number followed by an abrupt decrease. Similarly, when a new

domain joins spam campaigns, it exhibits abrupt increase in the number of queries over a period of time.

Chauvenet test is a method that detects outliers by identifying a probability band centered on the mean of a normal distribution with a certain standard deviation that contains all time-series points. Any time series point located outside the probability band is considered as an outlier. The algorithm has two inputs: the time series points and a significance level α (0.5). First, the mean of time series data (M) and the standard deviation (S) are calculated. Then, we create a normal distribution based on the mean and standard deviation. Using the normal distribution, we compute a cumulative probability (P) of each value in time series data from the normal distribution. Then, we multiply the inverse probability by 2 to compute the criterion value (P'). Finally, we multiply (P') by the length of data points and compare it with α. If the result is less than α, the value is considered as an outlier. Algorithm 1 summarizes different steps to identify outliers for queries number attribute.

Algorithm 1: Chauvenet Test

Input: $Data$
Input: $\alpha = 0.5$
Output: $Outliers$
  $M = Mean(Data)$;
  $S = StDev(Data)$;
  $ND = NormalDist(M, S)$;
  for all V in $Data$ **do**
    P=$CumulativeProb(ND, V)$
    $P' = 2 \times (1 - P)$;
    if $P' \times Length(Data) < \alpha$ **then**
      $Outliers.Append(V)$;
    end if
  end for
  return $Outliers$;

The reason behind using Chauvenet test rather than $\lambda_{1,t}$ linear regression test to

identify queries number anomalies lies in the fact that $\lambda_{1,t}$ linear regression test is aggressive in terms of detecting outliers. As such, any slight change is seen as an outlier. However, this is not the case for the number of queries, since we aim to detect drastic changes instead of slight changes.

Number of IP Changes per Domain: Malicious domains tend to have many IP addresses. Thus, we decide to monitor IP addresses of each domain observed in the stream. The system begins with gathering sets of IPs on 2 minute sliding window. After the 2 hours for domains and IP- blocks of our interest, 1 hour for the rest of domains, it indexes the collected IPs sets by domain names using Spark map-reduce action. Then, it converts the sets to time series by computing the number of IP addresses observed at the time $t + 1$ and not observed at time t. Using the time series of each domain as an input for the scoring function, it computes the priority of that domain. Domains having high priority are more likely to be malicious. So, it is considered for investigation. Domains having low priority are rejected. There are many legitimate domains that have many IP addresses. Thus, we use a white-list which is a set of known benign domains to decrease the false positives.

The scoring function works following these steps: (1) computing the positive shifts ($positiveShifts$) between time series values at time t and $t + 1$, (2) computing the number of time series values that are greater than or equal to average of the values ($aboveAverage$). (3) computing the number of time series values that are greater than 0 ($aboveZero$). After that, the score is computed by summing the ratio of $positiveShifts$ and the ratio of ($aboveAverage$) and ($aboveZero$). Then, we divide

by 3 to normalize it to a value between 0 and 1. Algorithm 2 illustrates how the IPs changes scoring function is computed.

Algorithm 2: IP Changes Score Function

Input: $TimeSeries$
Output: $Score$
$M = Mean(TimeSeries)$;
$S = Size(TimeSeries)$;
for all $i = 0; i < S; i++$ **do**
  if $TimeSeries[i] < TimeSeries[i+1]$ & $i \neq S-1$ **then**
    $positiveShiftse$
  end if
  if $TimeSeries[i] > M$ **then**
    $aboveAverage++$
  end if
  if $TimeSeries[i] > 0$ **then**
    $aboveZero++$
  end if
end for
$Score = \frac{1}{3} \times [(positiveShiftse/S - 1) + ((aboveAverage + aboveZero)/S)]$;
return $Score$;

TTL values: TTL value is one of the features that is used by many security experts to detect DNS malicious activities. Most of the legitimate domains assign high values to the TTL to gain caching advantages while malicious domains such as fast-flux systems tend to use small TTL values such as 3600 seconds or less as well as the round-robin technique. It provides many IP addresses for a given domain name. These techniques help malicious domains to stay available all the time and make it difficult for DNS-based Blackhole List (DNSBL) to detect them. One of the systems that uses low TTL values to detect Fast-Flux systems is Exposure by Bilge et al [7]. They claim that some malicious networks change their TTL frequently. These networks have many bots and some of them are chosen to be proxies. Proxies running on ADSL server are less dependable so, their TTL values are low. However, proxies running on university servers are more dependable thus, their TTL values are high. They

validate their assumption about TTL values with Conflicker botnet domains. Thus, the frequent changes of TTL values as well as the low TTL values are some of the characteristics that distinguish the malicious domains from the benign domains. We use these characteristics in our scoring function to detect such domains. Algorithm 3 illustrates the steps used to compute the score. The algorithm has two parts. The first part checks if the time series values are the same and less than one hour (3600 seconds). The score is assigned based on the ranges that TTL values belong to. For example, the TTL values belonging to the range $[0, 1]$ have the highest score while the ones that belong to the range $[900, 3900]$ have the lowest score. The second part computes the ratio of TTL values per range *RangesRatio* and the ratio of TTL changes *ChangesRatio*. For computing *RangesRatio*, we multiply the number of TTL value per range with a priority number. The priority number of lowest ranges $[0, 1]$ is the highest number (6) while the priority number of the highest range $[3600, \infty[$ is the lowest number. Then, we sum them and divide them by 6 to normalize the ratio. The final score is computed by summing *RangesRatio* and *ChangesRatio* and dividing them by 2.

Some benign domains such as Content Delivery Networks (CDNs) also assign low values to the TTL and use round robin technique to ensure the availability. We create a white-list of CDNs systems to remove such domains and decrease the false positive rate.

Algorithm 3: TTL Score Function

Input: $TimeSeries$
Output: $Score$
  $F = Frequency(TimeSeries, TimeSeries[0])$;
  $S = Size(TimeSeries)$;
  if $S = F$ **then**
    if $TimeSeries[0] \in [0, 1]$ **then**
      $Score = 1.0$
    else if $TimeSeries[0] \in [1, 60]$ **then**
      $Score = 0.9$
    else if $TimeSeries[0] \in [60, 300]$ **then**
      $Score = 0.8$
    else if $TimeSeries[0] \in [300, 900]$ **then**
      $Score = 0.7$
    else if $TimeSeries[0] \in [900, 3600]$ **then**
      $Score = 0.6$
    else
      $Score = 0.0$;
    end if
  else
    for $i = 0; i < S; i++$ **do**
      if $TimeSeries[i] \in [0, 1]$ **then**
        $a[6]++$
      else if $TimeSeries[0] \in [1, 60]$ **then**
        $a[5]++$
      else if $TimeSeries[0] \in [60, 300]$ **then**
        $a[4]++$
      else if $TimeSeries[0] \in [300, 900]$ **then**
        $a[3]++$
      else if $TimeSeries[0] \in [900, 3600]$ **then**
        $a[2]++$
      else
        $a[1]++$
      end if
      if $i + 1 \neq S$ & $TimeSeries[i] \neq TimeSeries[i + 1]$ **then**
        $Changes++$;
      end if
    end for
    $ChangesRatio = Changes/(S - 1)$;
    $RangesRatio = \frac{1}{6 \times S} \times \sum_{i=1}^{6} i \times a[i]$;
    $Score = \frac{1}{2} \times (RatioChanges + RatioRanges)$;
  end if
  return $Score$;

Record Misuses Filter

DNS protocol has been used by cyber-criminals as a carrier for communication between malware infected machines and remote bot-masters or proxies. Being inspired by the emergence of DNS tunneling tools (e.g., iodine [27], NSTX [51], OzymanDNS [24], Heyoka [44], etc.), attackers have been misusing DNS records, namely, "TXT","SRV","OPT", "NULL" and "ANY", to perpetrate malicious activities. In this section, we present examples of how these DNS records can be abused.

DNS tunneling: Although being considered as a benign service provided online or by customized tools, this service is a good artifact to exfiltrate data from networks that permit traffic to be sent only through a trusted server or proxy. By having a moderate bandwidth (110 Kilobytes per second) and latency (150 Milliseconds) [52], attackers consider it as a good medium to send blocked IP traffic through and conduct stealthy communication between bot masters. In [48], Ed Skoudis claimed that DNS tunneling malware is among the most dangerous attacks.

Malware covert channels: In [12], the authors explained the Modus-Operandi of Feederbot botnet. Malware belonging to Feederbot family, exfiltrates data within DNS query sub-domain labels and infiltrates attack payloads in DNS response packets. To detect DNS traffic generated by Feederbot botnet, the authors defined empirically a set of features that span over record data features and behavioral communication features. Based on these features, they adapted an hybrid approach (clustering &

classification techniques) to detect malicious DNS traffic. In [34], Mullaney introduced another malware family, namely, Morto, which uses a more resilient method to exchange communication through covert DNS channels. Morto botnet sends a limited amount of payload, which makes its payload distribution stealthier in comparison with Feederbot botnet.

DNS malicious responses: Attackers have put forward tools to send malicious responses in reply to DNS queries in order to test if DNS look-up servers are vulnerable. For instance, dnsxss [49] is a tool that returns a string containing JavaScript to "MX", "CNAME", "NS", and "TXT" requests. By looking at Passive DNS stream, we have found a lot of domains having "TXT" records containing script and frame tags. If a vulnerable server does not sanitize "TXT" record data returned by such domain, XSS attacks can be performed leading to some abnormal behavior in the server side.

Indicators of DNS DDoS attacks: In [9], the authors stipulated that "ANY" record is usually used in amplification DDoS attacks since it has an amplification factor of 52 as it replies with a response packet of $3,336$ bytes to a request packet of 64 bytes. In [15], the authors monitored Darknet for the purpose of inferring DDoS attacks. They observed that "ANY" records are in order of 52.23% of observed records involved in DDoS attacks during a period of three months. In the prevailing of these facts, we decide to monitor "ANY" records observed in passive DNS stream of data.

To extract record misuses, we use Spark filter functions on tuples. For instance, "TXT" record is used to publish email sender policies associated with domains (e.g.

SPF, DKIM and DMARK) and to identify verification of different search engines. It detects misuse of "TXT" record by creating patterns that identify benign usage of "TXT" record while the rest is considered as malicious. The "SRV" record is used to publish services for DNS. Based on observations done on "SRV" record data, we create patterns that identify if the record data is suspicious or not. In addition, we capture all passive DNS entries that have "OPT", "NULL" or "ANY" as a record type. The reason behind doing so lies in the fact that such records are rare and can be indicators of compromise.

Correlator

Our passive DNS anomaly detection system has been corroborated with a set of scripts that correlate intelligence gathered from passive DNS anomalies with other cyber-threat intelligence components, namely, dynamic malware analysis database and VirusTotal database [53]. Threat correlation falls into: (1) anomaly domains correlation, (2) anomaly IPs correlation and (3) DNS records abuse correlation. Regarding anomaly domains correlation, once we have a suspicious domain name, we check if it is related to malware activities through the dynamic malware analysis database. If so, we collect the different malware names and use VirusTotal malware naming capability to link the domain name to a set of malware families. In addition, we use VirusTotal URL scanner to check if the domain name is associated with threats identified by VirusTotal. Moreover, we use the suspicious domain database to collect the different IPs associated with the domain name. These IPs are used to gather geo-location information for the purpose of threat source identification and

profiling and be correlated with the dynamic malware analysis database and VirusTotal search engine to potentially identify other threats. Usually, anomaly domains have alpha-numerical patterns. Thus, we look into the dynamic malware analysis database to discover other domains that may have such patterns. Once these domains are identified, we repeat the same aforementioned correlation steps. Regarding anomaly IPs correlation, we monitor mainly anomalies related to the change of IPs number. For each domain responsible of triggering those types of alerts, we collect IP addresses mapped to the domain by correlating it with suspicious domains collections. In addition, we correlate the domain and its IPs with the dynamic malware analysis database and Virus-total search engine to check if it is related to threats or not. Regarding DNS records abuse, we correlate obtained DNS records' data with DNS traffic collected from the dynamic malware analysis. DNS can be used as a covert channel for malware communication or a vector for reflection DDoS attacks. Thus, we correlate any pattern found on "TXT", "NULL", "SRV", "OPT" and "ANY" data records with DNS traffic generated by malware samples.

4.3 Experiments and Results

In this section, we illustrate the performance of our application in different aspects.

4.3.1 Application Performance

Three servers have been dedicated to our DNS monitoring system. All the servers use Debian OS version 7.8. In addition, they use a document-based database (MongoDB [33]) to archive different collections of data. Each server has specific characteristics that suit its requirements. The first server is used to detect suspicious domains through PPM algorithm and misuses of different types of resource records ("TXT","SRV","OPT", "NULL", "ANY"). It is a Dell Poweredge T410 and has 64 GB memory, 24 CPU cores, and 3.8 TB space. The second server is employed to identify the change of cities, countries and IP addresses and TTL values per domain. It is a HP Proliant DL580 and has 125 GB memory, 48 CPU cores and 4.5 TB space. The third server is dedicated to monitor suspicious domains through PPM algorithm, the change of cities, countries and IP addresses, and TTL values of domains or IP-blocks that are of interest to the security experts. It is a SuperMicro and has 125 GB memory, 48 CPU cores and 4.0 TB space.

In addition, we analyze performance of servers in terms of memory consumption, CPU usage and time delay. For the time delay, we aggregate scheduling and processing time for batches. We monitor it through Spark user interface. For memory consumption and CPU usage, we observe them through JavaVisual VM [54]. We give different observations of memory consumption, CPU usage and delay time for the three servers. The calculations are done based on 18 hours logs.

The First Server: PPM Detection and Record Misuse

The first server is employed to detect PPM domains as well as misuses of passive DNS records. The system reads data in slide batches of 1 minute. We explain the system performance from various aspects (memory consumption, CPU usage and processing delay time).

Memory Consumption We observe that the memory consumption of Server 1 is high because of the large numbers of tuples received each minute. In addition, we notice that the garbage collector is used frequently (2 to 3 times every 3 hours) to avoid overhead usage of the memory (See Figure 4).

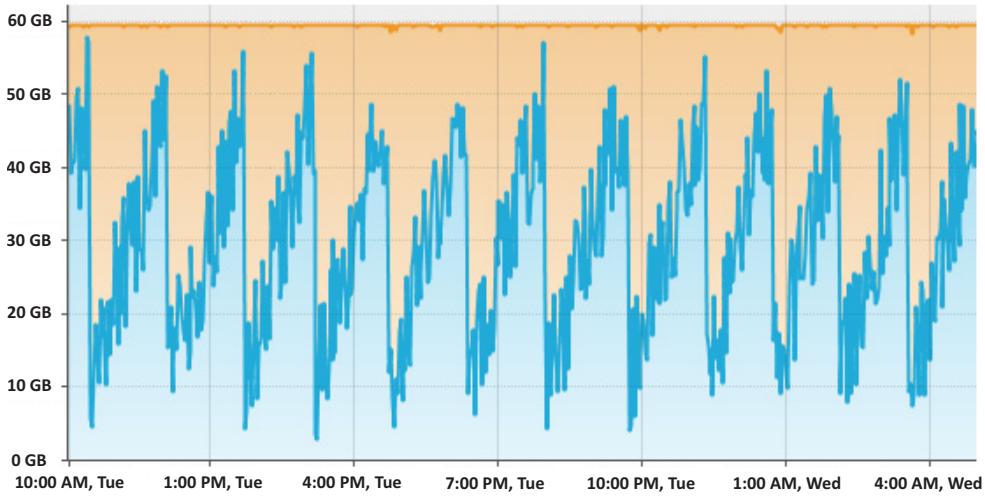

Figure 4: Memory Consumption of Server 1

CPU Usage For PPM detection and misuses of records, our system employs a filtering technique to recognize suspicious tuples collected from passive DNS stream. The batches are read each minute and no need for aggregation. We notice that the CPU

usage of Server 1 is less than 50% of CPU. Sometimes, we notice an increase of CPU usage which is due to the huge number of collected data received that minute. This is illustrated in Figure 5.

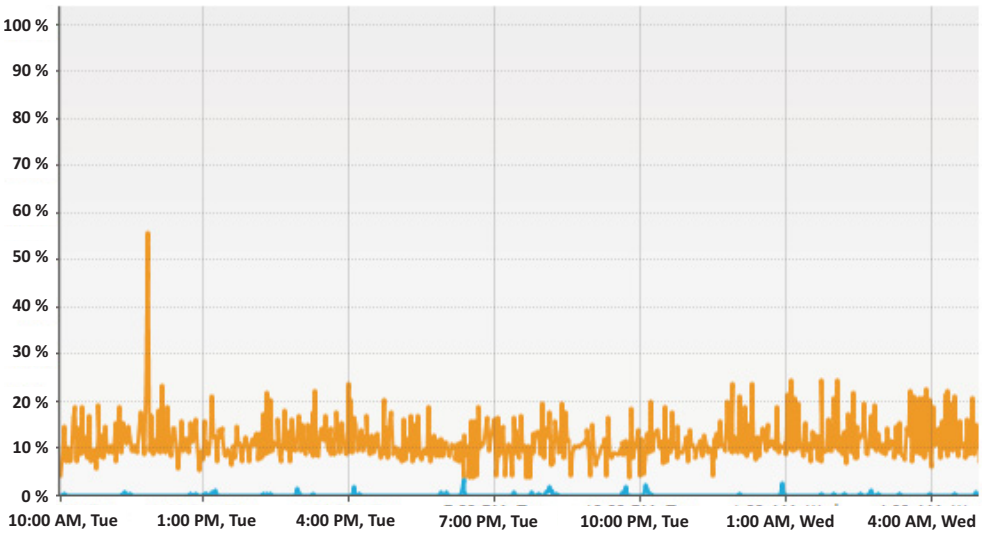

Figure 5: CPU Usage of Server 1

Delay Time Due to the size difference of data received each minute, the delay time of the application varies from 1 minute to 20 minutes (see Figure 6). The average of the delay time is 2 minutes 23 seconds which is considered to be reasonable because of the large amount of passive DNS data analyzed on-the-fly.

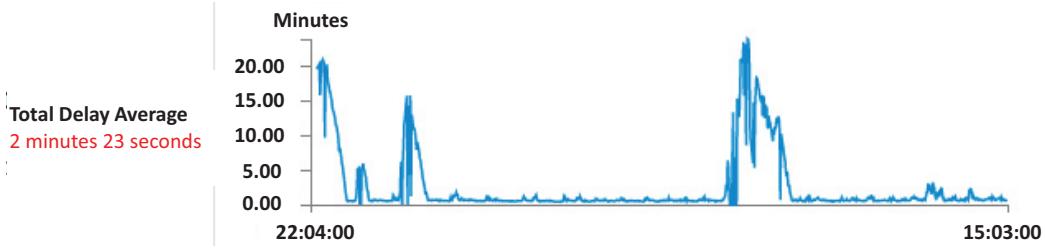

Figure 6: Delay Time of Server 1

The Second Server: IPs and Domains Features Monitoring

The second server is committed to compute different features of DNS. The features are cities changes, countries changes, IPs changes and TTL values. The slide batches of reading data is 2 minutes and the slide batches of the aggregating data is 1 hour. We describe the performance in terms of memory consumption, CPU usage and delay time.

Memory Consumption The System consumes most of the memory to compute the required features. When the window batches of 1 hour are finished and the data is ready for processing, the application uses the memory intensively due to the large amount of data being processed (see Figure 7). In addition, the garbage collector is used many times compared to the first server to release more memory for the application.

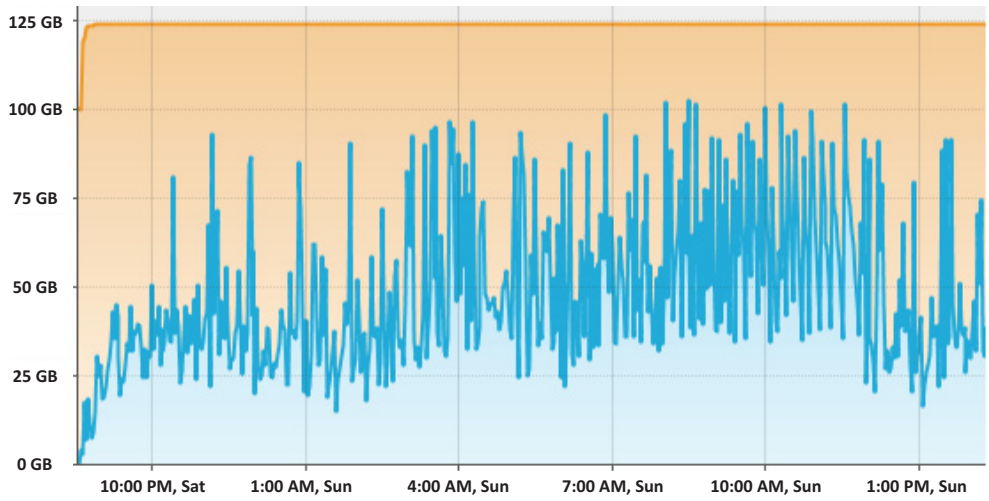

Figure 7: Memory Consumption of Server 2

CPU Usage The system uses the CPU massively every 2 minutes. These batches store different feature values including cities changes, countries changes, IPs changes and TTL values. In addition, the CPU is used intensively after 1 hour to perform map-reduce operation and insert the scoring function results to the database collections (see Figure 8).

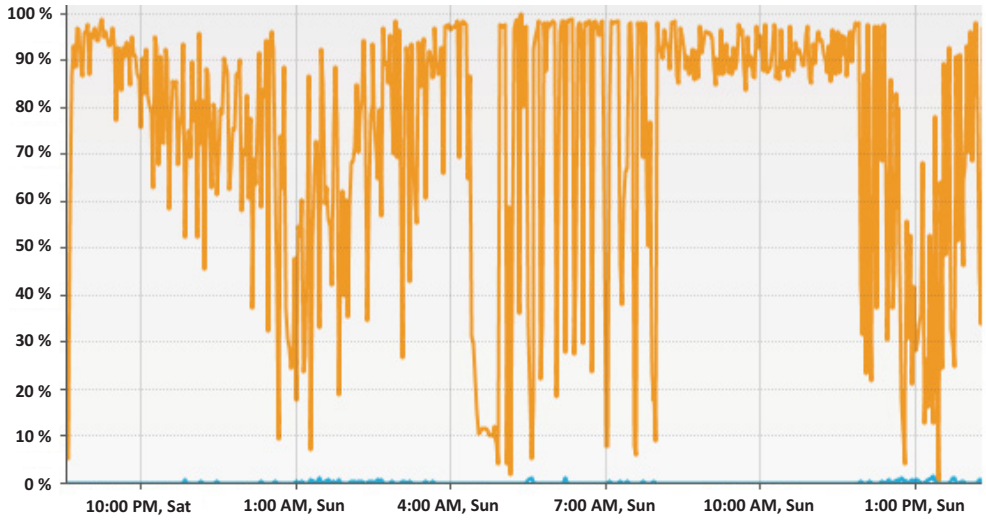

Figure 8: CPU Usage of Server 2

Delay Time The delay time of this application is high compared to the delay time of Server 1 application (see Figure 9). The average delay time for 1 hour batch window is 40 minutes and 48 seconds. However, this is considered acceptable due to the large number of features needed to be processed.

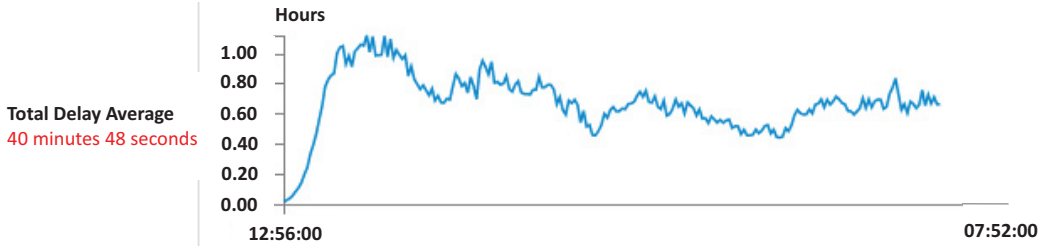

Figure 9: Delay Time of Server 2

The Third Server: Specific Domains and IP-Blocks Monitoring

This server is set up to monitor all domains or IP-blocks that are of interest to the security analyst. This application combines all capabilities of Server 1 and Server 2. So, we monitor all the PPM domains, DNS record misuses and DNS features statistics. Spark filter is used to distinguish the required domains and IP-blocks. Due to the fact that the domains and IP-blocks are just a small part of collected DNS records, we use 2 hours batch window rather than 1 hour to collect more data for the features.

Memory Consumption We notice that the memory consumption of Server 3 is low compared to the previous servers. However, after windows batches of 2 hours, the application consumes more memory to compute the features statistics (see Figure 10). The garbage collector of this application is used few times since the application is not using the memory intensively like Server 1 and Server 2.

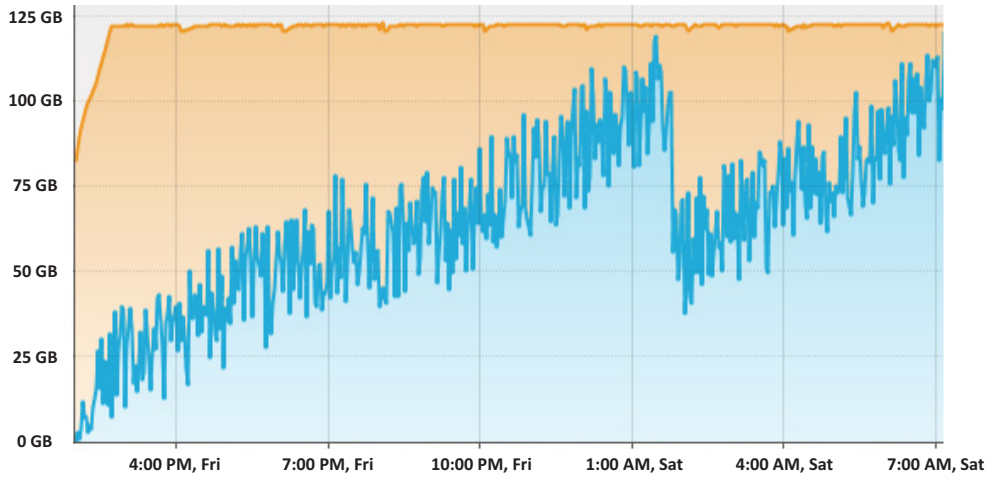

Figure 10: Memory Consumption of Server 3

CPU Usage We observe from Figure 11 that the CPU usage of Server 3 is low during 2 hours of slide windows batches. During the aggregation period, we notice an increase of the CPU usage due to Spark map-reduce operation.

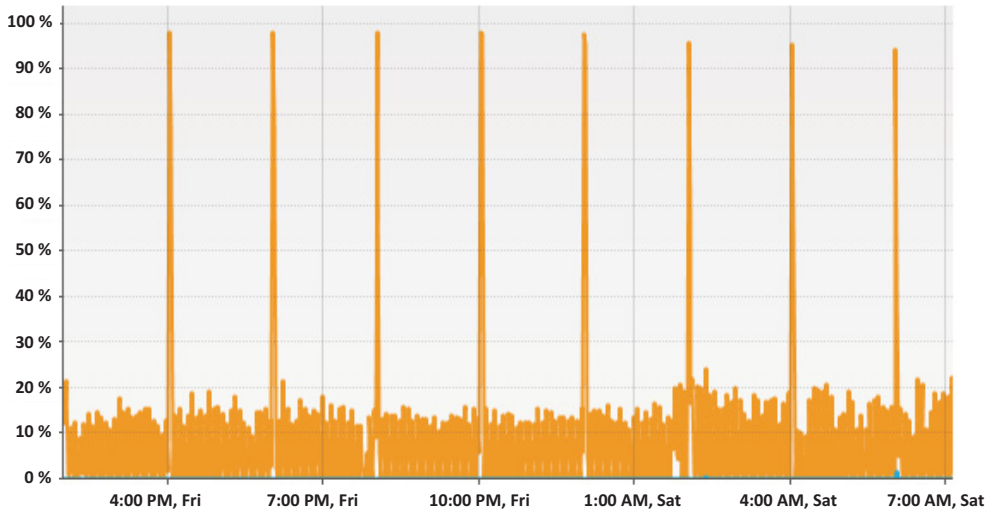

Figure 11: CPU Usage of Server 3

Delay Time From Figure 12, we observe that the delay time of application 3 is less than 1 minute before the aggregation period (2 hours). However, the delay time is increased at the end of window batch (2 hours) due to the computation of features. The total delay average is 23 seconds and 658 milliseconds that is considered to be very low comparing with application 1 and application 2.

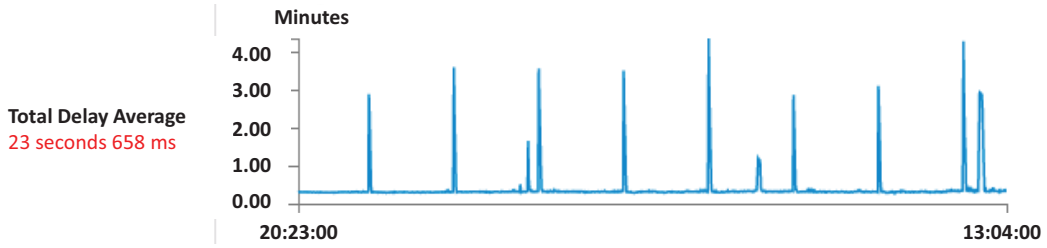

Figure 12: Delay Time of Server 3

4.4 Conclusion

In this chapter, we introduced a system that detects in near real-time different malicious activities including suspicious domains, DNS records misuses and DNS anomalies. Our system monitors all domains and their DNS records within passive DNS traffic. However, due to the large volume of DNS records, we implemented Apache Spark as part of our system. It is a framework that has the ability to monitor the huge load in addition to store the large amount of results. Our experiments on real-time passive DNS traffic show that our system can detect several types of DNS malicious domains such as spamming domains, bitcoining domains and phishing domains. Our system is also able to detect suspicious domains that are associated with several types of malware families. In addition, we identified different DNS record misuses such as CNAME and TXT records. We believe that the results of our system can help security experts to identify potential cyber threats.

Chapter 5

Conclusion

The Domain Name System (DNS) is a cornerstone component that intervenes in the operations of cyber threat infrastructures. As such, it is also being extensively used for malicious purposes. Moreover, there is an increasing evidence that demonstrates the abuse of DNS to conduct a wide variety of malicious activities such as operating botnets, spamming, phishing, fast-fluxing networks to evade detection, transporting malicious payloads, and infiltrating/exfiltrating sensitive information from organizational networks. In this setting, it becomes essential to subject DNS traffic to extensive analysis for the purpose of detecting the aforementioned cyber crime activities.

In this thesis, we addressed the problem of the analysis of passive DNS streams for the purpose of generating timely, relevant and important threat intelligence on cyber crime activities. Accordingly, we investigated the identification of suspicious/-malicious domains, the detection of DNS abuse as well as the fingerprinting of DNS anomalies. An important subgoal of the thesis was to cross-correlate the generated

intelligence with other sources of cyber threat intelligence. A special care was dedicated to make the elaborated analysis both efficient and scalable. In essence, the contributions of the thesis are:

- Comparative study of the state-of-the-art approaches in terms of passive DNS analysis. These approaches are analyzed and compared against a set of objective criteria that we have compiled.
- Design and implementation of an efficient and scalable passive DNS analysis technique that provides timely and relevant threat intelligence on different types of malicious activities including: suspicious domains, DNS record abuse, and DNS anomalies.
- Correlation of the generated threat intelligence with other sources of intelligence such as our malware database. The intention is to get additional intelligence such as the identification of the underlying malware samples that are responsible for the observed domains or behaviours.
- Design and implementation of two modes of analysis: The first mode, called general detection, aims at analyzing the DNS traffic underlying every observed domain or IP. The second mode is optimized to deal with domains or IP-blocks that are of interest to the security analyst and that need additional monitoring and analysis.
- Design and implementation of a distribution technique that leverages the computational clustering paradigm using the so-called Apache Spark framework in order to achieve scalability of the analysis of DNS streams.

Our experiments show that our proposed system is capable of uncovering important intelligence about cyber crime infrastructures. Indeed, we have been able to map the elements of several infrastructures that are perpetrating several attacks using botnets, fast-fluxing networks and malware distribution. In addition, our system demonstrated high efficiency in monitoring domains and IP-blocks that are of interest. Furthermore, our system is able to identify suspicious/malicious domains that are associated with different malware families. Although some of the detected domains have not been correlated with any malware families, this could be an indication that the underlying domains and IPs are elements of zero-day attacks. A very nice feature of our system, compared to existing techniques, is that it does not require any training and does not focus on specific types of threats.

As future work, we intend to experiment with additional features and time series prediction algorithms and compare their performance and efficiency to the ones used in this work. We envision also to conduct correlations with additional sources of cyber threat intelligence.

Bibliography

[1] R. Aitchison. *Pro DNS and BIND 10*. Apress, Berkely, CA, USA, 1st edition, 2011.

[2] Alexa. Alexa top sites. Available at `http://www.alexa.com/topsites`. Last visited: August 2015.

[3] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *USENIX security symposium*, pages 273–290, 2010.

[4] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, II, and D. Dagon. Detecting malware domains at the upper dns hierarchy. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 27–27, 2011.

[5] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of DGA-based malware. In *USENIX security symposium*, pages 491–506, 2012.

[6] R. Begleiter, Y. Elovici, Y. Hollander, O. Mendelson, L. Rokach, and R. Saltzman. A fast and scalable method for threat detection in large-scale DNS logs. In

Big Data, 2013 IEEE International Conference on, pages 738–741. IEEE, 2013.

[7] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE : Finding malicious domains using passive DNS analysis. In *NDSS 2011, 18th Annual Network and Distributed System Security Symposium, 6-9 February 2011, San Diego, CA, USA*, 2011.

[8] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur.*, 16(4):14:1–14:28, 2014.

[9] T. Cai, J. Yang, and X. Jin. Inferring DNS flooding attack through passive data analysis.

[10] H. Choi and H. Lee. Identifying botnets by capturing group activities in DNS traffic. *Computer Networks*, 56(1):20–33, 2012.

[11] Damballa. DGAs in the hands of cyber-criminals. Available at `http://tinyurl.com/zlql6jm`, 2012. Last visited: August 2015.

[12] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. van Steen, and N. Pohlmann. On botnets that use DNS for command and control. In *Proceedings of Seventh European Conference on Computer Network Defense*, pages 9–16. IEEE, 2011.

[13] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, M. Thomas. Domainkeys identified mail (dkim) signatures draft-ietf-dkim-base-10. Available at `https://tools.ietf.org/html/draft-ietf-dkim-base-10`, Feb. 2007. Last visited: August 2015.

[14] École Polytechnique Fédérale de Lausanne. The scala programming language. Available at `http://www.scala-lang.org`. Last visited: October 2015.

[15] C. Fachkha, E. Bou-Harb, and M. Debbabi. Fingerprinting internet dns amplification ddos activities. In *Proceedings of the IEEE Sixth IFIP International Conference on New Technologies, Mobility and Security (NTMS'2014), March 30-April 2nd, 2014, Dubai, UAE*, pages 1–5. IEEE, 2014.

[16] Farsight Security, Inc. Farsight Security. Available at `https://www.farsightsecurity.com`. Last visited: August 2015.

[17] R. Foundation. The R project for statistical computing. Available at `https://www.r-project.org`. Last visited: September 2015.

[18] T. P. S. Foundation. Welcome to Python.org. Available at `https://www.python.org`. Last visited: September 2015.

[19] T. Grzinic, D. Perhoc, M. Maric, F. Vlasic, and T. Kulcsar. Croflux - passive DNS method for detecting fast-flux domains. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014, Opatija, Croatia, May 26-30, 2014*, pages 1376–1380, 2014.

[20] Hadoop. Apache Open Source Project. Available at `https://hadoop.apache.org/`. Last visited: October 2015.

[21] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[22] F. S. Inc. Passive DNS architecture. Available at `https://archive.farsightsecurity.com/Passive_DNS/passive-dns-architecture.pdf`. Last visited: November 2015.

[23] D. Kaminsky. Black ops of DNS. *Black Hat Briefings*, 2004.

[24] Kaminsky, Dan. Dan kaminsky's Blog. Available at `http://dankaminsky.com/2004/07/29/51/`. Last visited: August 2015.

[25] A. M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi. Detection of malicious payload distribution channels in DNS. In *Communications (ICC), 2014 IEEE International Conference on*, pages 853–858. IEEE, 2014.

[26] M. Konte, N. Feamster, and J. Jung. Dynamics of online scam hosting infrastructure. In *Passive and Active Network Measurement*, volume 5448, pages 219–228. Springer, 2009.

[27] Kryo. iodine tool. Available at `http://code.kryo.se/iodine/`. Last visited: August 2015.

[28] Lance Whitney. FBI kills dnschanger network, but how many will be affected? Available at `http://tinyurl.com/hg6oolp/`, 2012. Last visited: August 2015.

[29] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin. Large-scale logistic regression and linear support vector machines using spark. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 519–528. IEEE, 2014.

[30] M. Wong, W. Schlitt. Sender policy framework (spf) for authorizing use of domains in e-mail, version 1. Available at `https://www.ietf.org/rfc/rfc4408.txt`, 2006. Last visited: August 2015.

[31] Maxmind. Geolocation and Online Fraud Prevention. Available at `http://www.maxmind.com`. Last visited: September 2015.

[32] P. V. Mockapetris. Domain names-concepts and facilities. rfc 1034. 1987.

[33] MongoDB, Inc. Mongodb. Available at `https://www.mongodb.org/`. Last visited: October 2015.

[34] C. Mullaney. Morto worm sets a DNS record. Available at `http://www.symantec.com/connect/blogs/morto-worm-sets-dns-record`, 2011. Last visited: December 2015.

[35] J. Nazario and T. Holz. As the net churns: Fast-flux botnet observations. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 24–31. IEEE, 2008.

[36] Oracle. Java. Available at`https://www.java.com`. Last visited: September 2015.

[37] P. Mockapetris. Domain names - concepts and facilities. Available at `http://www.ietf.org/rfc/rfc1034.txt`, Nov. 1987. Last visited: August 2015.

[38] P. Mockapetris. Domain names - implementation and specification. Available at `http://www.ietf.org/rfc/rfc1035.txt`, Nov. 1987. Last visited: August 2015.

[39] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. Fluxor: Detecting and monitoring fast-flux service networks. In *Detection of intrusions and malware, and vulnerability assessment*, pages 186–206. Springer, 2008.

[40] R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting malicious flux service networks through passive analysis of recursive DNS traces. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 311–320. IEEE, 2009.

[41] R. Perdisci, I. Corona, and G. Giacinto. Early detection of malicious flux networks via large-scale passive DNS traffic analysis. *Dependable and Secure Computing, IEEE Transactions on*, 9(5):714–726, 2012.

[42] M. Prince. The ddos that knocked spamhaus offline (and how we mitigated it). *Cloudflare blog*, 20(3), 2013.

[43] Quantcast. Rankings. Available at `https://www.quantcast.com/top-sites`. Last visited: August 2015.

[44] Revelli, Alberto and Leidecker, Nico. Heyoka: your fast&spoofed DNS tunnel. Available at`http://heyoka.sourceforge.net/`. Last visited: August 2015.

[45] J. G. Shanahan and L. Dai. Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2323–2324. ACM, 2015.

[46] R. Sharifnya and M. Abadi. A novel reputation system to detect DGA-based botnets. In *Computer and Knowledge Engineering (ICCKE), 2013 3rd International eConference on*, pages 417–423. IEEE, 2013.

[47] R. Sharifnya and M. Abadi. Dfbotkiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic. *Digital Investigation*, 12:15–26, 2015.

[48] E. Skoudis. The six most dangerous new attack techniques and what's coming next. In *RSA Conference (RSA'12)*, 2012.

[49] SkullSecurity. Dnsxss tool. Available at `https://wiki.skullsecurity.org/Dnsxss`. Last visited: October 2015.

[50] E. Stalmans and B. Irwin. A framework for DNS based detection and mitigation of malware infections on a network. In *Information Security South Africa (ISSA), 2011*, pages 1–8. IEEE, 2011.

[51] Tamas Szerb. NSTX tunneling network-packets over DNS. Available at `http://savannah.nongnu.org/projects/nstx/`. Last visited: August 2015.

[52] T. van Leijenhorst, K.-W. Chin, and D. Lowe. On the viability and performance of DNS tunneling. *Conference on Information Technology and Applications*, 2008.

[53] Virustotal. Free online virus, malware and url scanner. Available at `https://www.virustotal.com`. Last visited: December 2015.

[54] VisualVM. All-in-One Java Troubleshooting Tool. Available at `https://visualvm.java.net/`. Last visited: October 2015.

[55] F. Weimer. Passive DNS replication. In *Proceedings of the FIRST Conference on Computer Security Incident*, page 98, 2005.

[56] M. S. Wiewiórka, A. Messina, A. Pacholewska, S. Maffioletti, P. Gawrysiak, and M. J. Okoniewski. Sparkseq: fast, scalable, cloud-ready tool for the interactive genomic data analysis with nucleotide precision. *Bioinformatics*, pages 2652–2653, 2014.

[57] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *Networking, IEEE/ACM Transactions on*, 20(5):1663–1677, 2012.

[58] S. Yadav and A. N. Reddy. Winning with DNS failures: Strategies for faster botnet detection. In *Security and privacy in communication networks*, pages 446–459. Springer, 2012.

[59] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. Franklin, S. Shenker, and I. Stoica. Fast and interactive analytics over hadoop data with spark. *USENIX; login*, 37(4):45–51, 2012.

[60] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, volume 10, page 10, 2010.