DNS Traffic Based Classifiers for the Automatic Classification of Botnet Domains

Submitted in fulfilment
of the requirements of the degree of

Master of Science

of Rhodes University

Etienne Raymond Stalmans

Grahamstown, South Africa
June 2013

Abstract

Networks of maliciously compromised computers, known as botnets, consisting of thousands of hosts have emerged as a serious threat to Internet security in recent years. These compromised systems, under the control of an operator are used to steal data, distribute malware and spam, launch phishing attacks and in Distributed Denial-of-Service (DDoS) attacks. The operators of these botnets use Command and Control (C2) servers to communicate with the members of the botnet and send commands. The communications channels between the C2 nodes and endpoints have employed numerous detection avoidance mechanisms to prevent the shutdown of the C2 servers. Two prevalent detection avoidance techniques used by current botnets are algorithmically generated domain names and DNS Fast-Flux. The use of these mechanisms can however be observed and used to create distinct signatures that in turn can be used to detect DNS domains being used for C2 operation. This report details research conducted into the implementation of three classes of classification techniques that exploit these signatures in order to accurately detect botnet traffic. The techniques described make use of the traffic from DNS query responses created when members of a botnet try to contact the C2 servers. Traffic observation and categorisation is passive from the perspective of the communicating nodes. The first set of classifiers explored employ frequency analysis to detect the algorithmically generated domain names used by botnets. These were found to have a high degree of accuracy with a low false positive rate. The characteristics of Fast-Flux domains are used in the second set of classifiers. It is shown that using these characteristics Fast-Flux domains can be accurately identified and differentiated from legitimate domains (such as Content Distribution Networks exhibit similar behaviour). The final set of classifiers use spatial autocorrelation to detect Fast-Flux domains based on the geographic distribution of the botnet C2 servers to which the detected domains resolve. It is shown that botnet C2 servers can be detected solely based on their geographic location. This technique is shown to clearly distinguish between malicious and legitimate domains. The implemented classifiers are lightweight and use existing network traffic to detect botnets and thus do not require major architectural changes to the network. The performance impact of implementing classification of DNS traffic is examined and it is shown that the performance impact is at an acceptable level.

Acknowledgements

I would like to thank my family for all their support and constant encouragement throughout the writing of this thesis. I also want to thank my supervisor Dr. Barry Irwin for all his support and guidance. And finally I would like to thank John Richter for the hours he put into proof reading and correcting my mistakes, any remaining errors are mine alone.

This work was performed in and funded by the Centre of Excellence in Distributed Multimedia at Rhodes University, with financial support from Telkom SA, Comverse, Verso Technologies, Tellabs, StorTech, EastTel and THRIP. Financial assistance towards the completion of my MSc was provided by Telkom SA and I would like to thank them for the support without which this research would not have been possible.

ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System (1998 version, valid through 2012)

C.2.0 [General] Security and protection

C.2.3 [Network Operations] Network monitoring

C.2.3 [Network Operations] Public networks

I.2.7 [Natural Language Processing] Text analysis

Contents

1 Introduction **1**

1.1 Problem Statement 2

1.2 Research Objectives 3

1.3 Research Method 4

1.4 Document Structure 5

2 Background **7**

2.1 Domain Name System 7

2.2 Botnets 11

2.2.1 Structure of Botnets 11

2.2.2 DNS Fast-Flux 14

2.2.3 Algorithmically Generated Domain Names 15

2.3 Related Work 17

2.3.1 URL Analysis 17

2.3.2 DNS Fast-Flux Detection 19

2.4 Summary 20

3 Techniques for Botnet Identification **22**

3.1 Data Description 22

3.1.1 Domain Name Data Sources 23

3.1.2 Fast-Flux Data Sources 24

3.1.3 Live Test Data 25

3.2 Lexical Features 25

3.2.1 Frequency Analysis 26

3.3 Lexical Classifiers 27

3.3.1 Probability Distribution 29

3.3.2 Total Variation Distance 29

3.3.3 Bayesian 30

3.3.4 Naïve Bayesian Classifier 31

3.4 Domain Name Query Features Classifiers 32

3.4.1 Modified Holz Classifier 33

3.4.2 Rule-based Classifier 34

3.4.3 Naïve Bayesian Classifier 34

3.5 Geographic Features Classifiers 35

3.5.1 GeoIP Database 37

3.5.2 Geographic Value 40

3.6 Spatial Autocorrelation 42

3.6.1 Moran's Index 43

3.6.2 Geary's Coefficient 44

3.7 Summary 44

4 Results **46**

4.1 Data Metrics . . . 46

4.1.1 Labelling Data . . . 46

4.1.2 Results Evaluation . . . 47

4.2 Algorithmically Generated Domain Name Detection . . . 48

4.2.1 Unigram Classifiers . . . 49

4.2.2 Bigram Classifiers . . . 56

4.2.3 Results Summary . . . 59

4.3 DNS Fast-Flux Detection . . . 59

4.3.1 Modified Holz Classifier . . . 61

4.3.2 Rule-Based Classifier . . . 61

4.3.3 NaïveBayesian Classifier . . . 63

4.3.4 Combined Classifier . . . 63

4.3.5 Summary . . . 65

4.4 Spatial Autocorrelation . . . 65

4.4.1 Moran's Index . . . 66

4.4.2 Geary's Coefficient . . . 70

4.4.3 Spatial Autocorrelation Summary . . . 71

4.5 Performance Analysis . . . 72

4.5.1 Test Platform . . . 73

4.5.2 Lexical Classifiers Performance Analysis . . . 73

4.5.3 Fast-Flux Classifiers Performance Analysis . . . 74

4.6 Summary . . . 77

5 Discussion **79**

5.1 Algorithmically Generated Domain Names 79

5.1.1 Observed Character Distribution 80

5.1.2 Unigram Versus Bigram Classifier Accuracy 83

5.1.3 Summary 84

5.2 DNS Fast-Flux 85

5.2.1 Classifier Results 85

5.2.2 Legitimate Fast-Flux Domains 87

5.3 Spatial Autocorrelation 89

5.3.1 Observed Distribution Patterns 90

5.3.2 Moran's Index and Geary's Coefficient 91

5.3.3 Open Source Intelligence 93

5.4 Performance Analysis 94

5.4.1 Lexical Classifiers Performance 94

5.4.2 Fast-Flux Classifiers Performance 95

5.5 Real-World Application 95

5.5.1 Log Analysis 96

5.5.2 DNS Traffic Monitoring 97

5.6 Summary 97

6 Conclusion **99**

6.1 Future Work 101

6.1.1 Anti-Malware Protection 101

6.1.2 DDoS Detection 102

References 103

A Fast-Flux Domain Query Responses 113

B Algorithmically Generated Domain Names 116

C Geographic Distribution of C2 Servers 118

D Published Works 120

List of Tables

2.1 Fast-Flux Domain Query Result 14

2.2 Content Distribution Network DNS Query Result (*fbcdn.com*) 14

2.3 Sample Algorithmically Generated Domain Names 16

3.1 Domain Name Data Sources 23

3.2 Fast-Flux Data Sources 25

3.3 Live Test Data 25

3.4 Geographic Data for a Fast-Flux Domain (*cjjasjjikooppfkja.ru*) 37

3.5 Input Values for the IP Address 59.146.177.153 40

4.1 Results of the Unigram Classifiers 49

4.2 Results of the Bigram Classifiers 56

4.3 Accuracy Rates of Lexical Analysing Classifiers 59

4.4 Results For Fast-Flux Classifiers 60

4.5 Mean values of DNS features for Fast-flux and legitimate domains 60

4.6 Fast-Flux Classifier Outputs 62

4.7 Naive Bayesian Classifier Results for Fast-Flux and CDN Domain Queries . 64

4.8 Summary of Fast-Flux Classifier Accuracy Rates 65

4.9 Moran's Index Classifier Performance 67

4.10 Geary's Coefficient Classifier Performance 70

4.11 Summary of Spatial Autocorrelation Classifier Accuracy 71

4.12 Real-World Performance of Lexical Classifiers 74

4.13 Processing Time Per DNS Response Packet 75

4.14 Projected Performance Impact of DNS Query Response Classification . . . 77

5.1 Observed Domains 88

5.2 Observed Mean Values for Moran's Index 92

5.3 Observed Mean Values for Geary's Coefficient 92

B.1 Conficker-C Domains 116

B.2 Kraken Domains 117

B.3 Bobax Domains 117

C.1 Geographic Distribution of C2 Servers 118

List of Figures

2.1 DNS Hierarchical Tree Structure 8

2.2 DNS Resolution Process 9

2.3 Output for the DNS Resolution Process Using the *dig* Command 10

2.4 Botnet Structure 12

3.1 Frequency Distribution of Unigrams. 27

3.2 Frequency Distribution of Bigrams. 28

3.3 Histograms of Mean Nearest Neighbour Distances for Legitimate Domains and Fast-Flux 38

3.4 Geographic Distribution of Hosts for a Botnet and a Legitimate Domain . 39

4.1 Total Variation Classifier Output for Unigrams 51

4.2 Density Distribution of Unigram Naive Bayesian Classifier Output. 52

4.3 Naive Bayes Classifier Output for Legitimate Domains 54

4.4 Naive Bayes Classifier Output for Algorithmically Generated Domain Names 55

4.5 Density Distribution of Bigram Naive Bayesian Classifier Output. 58

4.6 ROC of Fast-Flux Classifiers Accuracy 66

4.7 Kernel Density Comparison of Moran's Index Using UTM. 68

4.8 Kernel Density Comparison of Moran's Index Using MGRS. 69

4.9 ROC for Moran's I Classifiers 69

4.10 Kernel Density Distribution for Geary's Coefficient. 71

4.11 ROC for Geary's Coefficient Classifiers 72

4.12 Processing Time for Lexical Classifiers 73

4.13 Processing Time of the Various Fast-Flux Classifiers 75

4.14 Projected Performance Impact of DNS Query Response Classification . . . 76

5.1 Comparison of Vowel to Consonant Ratios 81

5.2 Geographic Distribution of Botnet C2 Server IP Addresses 91

List of Algorithms

3.1 Total Probability 29
3.2 Total Variation Distance 30
3.3 Bayesian Classifier 31
3.4 Naïve Bayesian Classifier 32
3.5 Modified Holz Heuristic Classifier 33
3.6 Rule-based Fast-Flux Classifier 35
3.7 Naïve Bayes Probability of Continuous Attributes. 36
3.8 Naïve Bayesian Fast-Flux Classifier 36
3.9 Timezone Value Calculation 40
3.10 The Military Grid Reference System Numeric Value 41
3.11 Moran's Index 43
3.12 Geary's Coefficient 44
4.1 Classifier Performance Calculations 47

1 Introduction

Botnets have emerged as a serious threat to Internet security and are commonly used to conduct malicious and illegal activities. These botnets may consist of thousands of infected corporate and household hosts spread around the world. Botnet operators use Command and Control (C2) servers to distribute commands to and manage the hosts of the botnet and thus need to ensure that these servers are resistant to being shutdown. The distributed nature of these botnets makes mitigation and remediation difficult. This distributed nature of the host belonging to the botnet means a central call home point is required for the hosts to receive instructions. Therefore the hosts require a means of determining the C2 server's addresses and call home for instructions. To add to the difficulties posed by the homogeneous nature of botnets, botnet operators employ numerous detection and shut-down evasion techniques. These techniques predominantly rely on the Domain Name System (DNS) to provide a means for infected hosts to contact the C2 servers, while also providing effective anti-detection and shutdown protection.

This report describes a number of techniques for detecting botnet traffic from DNS query responses. These detection techniques exploit the signatures created by the detection avoidance mechanisms employed by botnets. The first set of classifiers described employ frequency analysis to detect algorithmically generated domain names with a high degree of accuracy with a low number of false positives. The characteristics of Fast-Flux domains are used in the second set of classifiers. The final set of classifiers use spatial autocorrelation to detect Fast-Flux domains from the geographic distribution the botnet C2 servers. It is shown that Fast-Flux domains can be accurately identified and differentiated from legitimate domains such as Content Distribution Networks. The proposed classifiers are lightweight and use existing network traffic to detect botnets and thus do not require major architectural changes to the network.

1.1 Problem Statement

The growth in the number of Internet connected devices has seen a related rapid growth in the number of hosts infected by malicious software. This software is known as malware (Vanier, 2011; Arbor Networks, 2012a). Internet connected devices expose multiple vectors by which malware can infect the system. Current malware protection schemes are largely failing as systems are becoming more exposed to external threats (Wilson, 2008). Once a system has been infected, the malware can be used to steal data, sending spam, phishing, distributing malware and Distributed Denial-of-Service (DDoS) attacks. Current malware detection methods predominantly rely on host-based malware detection mechanisms that are based on pattern matching and heuristics. These traditional detection techniques are easily bypassed by zero-day attacks and polymorphic code (Ollmann, 2008). Current network-based solutions often focus on preventing malware from entering the system through the use of firewalls, Intrusion Detection Systems and blacklists. These systems are blind to malware that enters the system through other attack vectors, such as mobile Internet connections or removable devices, thus highlighting the need for detection systems that also focus on traffic leaving the network.

The current generation of malware is largely focused on the creation of large networks of infected hosts known as botnets. Botnets consist of thousands of infected hosts, referred to as bots, that receive instructions from C2 servers operated by an individual. Traditionally Internet Relay Chat (IRC) servers have been used as C2 servers and have communicated with the botnet through IRC channels (Lee, Jeong, Park, Kim, and Noh, 2008). This has led to network administrators often blocking IRC traffic on the network. Recent trends in botnet development have seen the use of alternative communication channels, such as DNS-tunnelling and HTTP requests, between the C2 servers and infected hosts (Pereira, Fucs, and de Barros, 2007; Lee et al., 2008).

The use of alternative communication channels has allowed botnet traffic to bypass common network filters (Gu, Zhang, and Lee, 2008). These channels cannot simply be blocked as IRC traffic has been due to many of the underlying protocols being essential for normal network activity. An example of this is DNS, where almost all network communication is reliant on DNS for address translation to aid the establishing of communication between hosts. While most web traffic relies on HTTP or HTTPs for reliable communication between servers and hosts.

An emergent trend shown by recent botnets such as ZeuS, Kehlios and Citadal is the use of new detection avoidance techniques. One of these avoidance techniques, known as DNS

Fast-Flux, allow botnets to avoid detection and to reduce the ability of researchers to find and shut-down the C2 servers. Fast-Flux relies on rapidly changing domain name records to mask the location of C2 servers and to ensure domains are defended against common shutdown techniques such as IP address blacklisting.

A further detection avoidance technique is algorithmically generated domain names. In this method each infected host employs a Domain Generation Algorithm (DGA) to generate a large set of domain names to query (Yadav, Reddy, Reddy, and Ranjan, 2010). These generated domain names are queried until a live C2 server is found. Botnets such as Conficker, Kraken and Torpig have successfully employed algorithmically generated domain names to ensure the longevity of the botnet C2 servers. Trends in algorithmic name generation have seen bots, such as those infected with Conficker-C, generating upwards of 50000 domain names an hour (Porras, Saidi, and Yegneswaran, 2009). This large volume of generated domain names makes it nearly impossible for researchers to block or pre-register all domains associated with these botnets, as was done with earlier Conficker variants (Leder and Werner, 2009; Porras et al., 2009). Furthermore, the massive amount of generated names makes the maintenance and use of domain blacklists slow, cumbersome and ultimately largely ineffective.

1.2 Research Objectives

This research has been conducted with the the aim of detecting botnet domains and communication using features contained in DNS query responses. The research objectives can be formally defined as the following:

- The detection of algorithmically generated domain names such as those used by the Torpig, Kraken and Conficker botnets. These botnets employ DGAs to create a large pool of domain names which are queried by infected hosts to query when attempting to contact the C2 servers. These generated domain names tend to display a different frequency distribution of characters than those observed in legitimate domain names. The aim of this research is to identify these algorithmically generated domain names using statistical classifiers, providing a lightweight, learning system capable of accurately differentiating between algorithmically generated and semantically correct domain names.

- The detection and identification of Fast-Flux domains used to host the C2 servers by using the information contained in the DNS query response. The distinct characteristics of Fast-Flux domains were examined and used to create a signature, which could be used to identify and differentiate between Fast-Flux domains and legitimate domains such as Content Distribution Networks (CDNs).
- The geographic location of servers were used with the aim of detecting Fast-Flux domains. The characteristic displayed by Fast-Flux domains where the C2 infrastructure usually consists of geographically widely dispersed hosts was employed as an identifying feature that could be used statistical classifiers. The research aims to formally define this geographic dispersion and hence classify domains as either Fast-Flux or legitimate.
- Finally the possible performance impact the proposed classifiers would have on existing network traffic was considered. The classifiers are all passive and thus the research aim was to keep classification time as close to zero as possible to match the zero interaction required with the botnet hosts.

1.3 Research Method

The research was conducted to identify techniques that could be used to identify domains from DNS query response network traffic that could be linked to potential botnet activity. The proposed detection techniques were used to construct prototype classifiers capable of detecting known botnet domains. These classifiers were then evaluated to determine their accuracy as well as the rate at which false positives and false negatives occur. Finally the performance of the classifiers was evaluated to determine the feasibility of deploying the classifiers on a real world network.

The proposed classifiers were derived by examining current detection evasion techniques employed by botnets: namely algorithmically generated domain names and DNS Fast-Flux. This led to the identification of features which could be used in the construction of classifiers. Once these features were known, statistical techniques were researched to identify techniques which could be used to produce classifiers. The proposed classifiers were trained using known, legitimate botnet domains. The trained classifiers were then tested against real world samples to determine their accuracy when exposed to data gathered from botnets operating at the time. Real world performance testing was conducted to

evaluate the performance impact the proposed classifiers would have on existing network traffic.

1.4 Document Structure

The remainder of the document consists of five chapters as follows:

- Chapter 2 provides background information about the Domain Name System, botnets and the evasion techniques used by botnets. Related work is outlined and it is explained how this research aims to extend and improve on previous work, as well as introducing novel classification techniques.
- Chapter 3 describes the datasources used and how data was divided into training and test sets. Lexical analysis is explained and the lexical features of English words, domain names and algorithmically generated domain names are examined. Following this, the techniques used to construct the classifiers for the classification of domain names are outlined. The techniques for detecting Fast-Flux domains are described along with the spatial autocorrelation and how this can be applied to Fast-Flux domain detection.
- Chapter 4 presents the results for the classifiers described in Chapter 3. First the means of measuring classifier performance are outlined, followed by the results for the lexical analysis classifiers. The results for classifiers using DNS query features to detect Fast-Flux domains are presented. Finally the results obtained through spatial autocorrelation are presented.
- Chapter 5 discusses the results obtained in testing. The performance in terms of accuracy of the classifiers are compared to related works and the implications of the research results are discussed. The weaknesses of the classifiers are discussed along with possible bypass techniques and how these could be countered.
- Chapter 6 provides a conclusion to the research and suggests possible future research based on the findings and work presented in this document.

The appendices provide additional information that may be useful in understanding the data used in this document. Sample Fast-Flux botnet DNS queries are provided in Appendix A. Algorithmically generated domain names for the Conficker-C, Kraken and

Bobax botnets are provided in Appendix B. Appendix C provides a sample of the geographic distributions of Fast-Flux C2 servers.

Elements of this document have been published in short format and links to these are provided in Appendix D.

2 Background

This chapter provides background information relevant to the research presented in the remainder of this document. The domain name system (DNS) and how it is used ubiquitously in modern Internet communication is described in section 2.1. Botnets are described in detail in section 2.2 along with two of the evasion techniques employed by these. These techniques are discussed separately as DNS Fast-Flux (subsection 2.2.2) and algorithmically generated domain names (subsection 2.2.3). Furthermore, the relationship these evasion mechanisms have with DNS are discussed in detail. Related work in botnet detection and the use of DNS in the detection of botnets is outlined in section 2.3. This includes work in the identification of algorithmically generated domain names and Fast-Flux detection. The chapter concludes with section 2.4 providing a summary of the information presented.

2.1 Domain Name System

All devices connected to the internet have a globally unique address used to identify the device. This unique address is known as the device's Internet Protocol (IP) address and consists of 32-bits in IP version 4 (IPv4). In IPv4 this is typically written as a series of four binary octets (A.B.C.D) known as dotted notation (example 192.168.0.1). IP version 6 (IPv6) consists of 8 octets and thus allows for a larger set of address to be represented than in IPv4. IPv6 is the official replacement for IPv4 and has slowly been integrated into existing networks (Colitti, Gunderson, Kline, and Refice, 2010). For the purpose of this research IPv6 is out of scope, though all the methods described should be portable to IPv6. These numerical IP addresses are easy for computing devices to

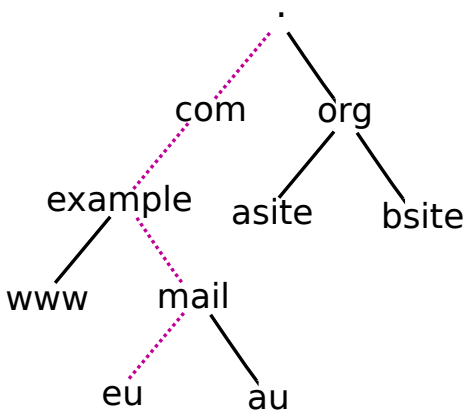


Figure 2.1: DNS Hierarchical Tree Structure

use but harder for humans to remember. The Domain Name System (DNS) was created to provide a mapping between the user-friendly domain names and machine friendly IP addresses. DNS is a distributed, hierarchical naming system that ensures the names used to identify resources on the Internet can remain constant despite changes to the underlying IP address or physical location. A domain name may consist of up to 253 characters, with an individual domain label not allowed to exceed 63 characters (Mockapetris, 1987). Characters which are allowed to be present in domain labels are alphanumerical ([a-z][A-Z][0-9]) as well as the hyphen. All other characters are invalid in domain names (Mockapetris, 1987). International domain name servers (iDNS) allow domain names to fall outside these limits, allowing for characters from languages such as Chinese to be used (Tan, Seng, Tan, Leong, De Silva, Lim, Tay, Subbiah, et al., 2002).

Domain names are constructed using a hierarchical tree structure, where a domain name may consist of multiple domain labels separated by a dot (‘.’) such that eu.mail.example.com. constitutes a fully qualified domain name. A fully qualified domain name ends in a ‘.’ which is known as the root node. The root node is taken as implied and is commonly left off. Each domain name identifies a path from the root node, identified by the rightmost ‘.’, to the node representing the Internet endpoint resource. This hierarchical structure can be represented as shown in Figure 2.1. The domain eu.mail.example.com. can be mapped by following the branches of the tree structure as shown. The path starts at the root node progressing downwards through each child node, known as a resource record (RR), along the path until eventually terminating at the final node, which would typically resolve the domain address and other records. The progression down the tree is shown as purple dashes, starting at the root node ‘.’ and progressing down to the lowest level domain ‘eu’. The depth of a node in the tree is known as the domain level, with each domain name consisting of at least two domain levels, such that for the domain example.com,

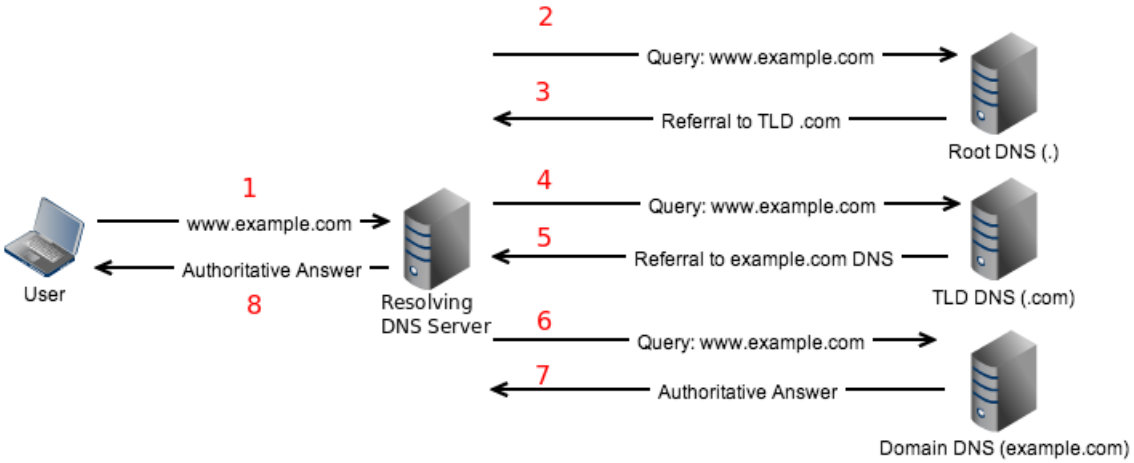


Figure 2.2: DNS Resolution Process

the Top-Level Domain (TLD) is 'com' and the Second-Level Domain (SLD) is 'example'. All the child nodes of 'example' indicate sub-domains controlled by the 'example.com' domain. Available TLDs are controlled by Internet Corporation for Assigned Names and Numbers (ICANN) and can be divided into two subgroups, Generic TLDs (gTLD) and Country Code TLD (ccTLD). Generic TLDs represent domains such as .com, .edu, .gov and .net, while ccTLDs consist of a two-character country code such as .uk, .ru and .za (Aitchison, 2011).

When a user wishes to contact a host on the Internet the domain name first needs to be mapped to the IP address of that host. This is done through the DNS resolution process, as can be seen in the simplified process depicted by Figure 2.2, where a query for the domain name www.example.com returns an authoritative answer from the DNS server for example.com. A DNS query is done to the host's local DNS resolver (1), usually internal to the organisation or the host's Internet Service Provider. This DNS server performs a lookup of the domain name in it's local cache and if the address record is available returns an answer to the host (8). When the address record is not available the DNS server performs either an iterative or recursive DNS lookup. This referral process shown by steps 2 through 7 consists of a series of lookups to the DNS servers for each node in the DNS hierarchy until a valid DNS record is located. If no DNS record is found a MX record is returned to indicate that the domain name could not be resolved. This process is however outside the scope of this research. Additional information on iterative or recursive DNS lookup can be found in the literature Aitchison (2011). The result of this resolution process is a DNS query response consisting of numerous fields containing values pertinent to the domain in question. A typical DNS query response can be seen in

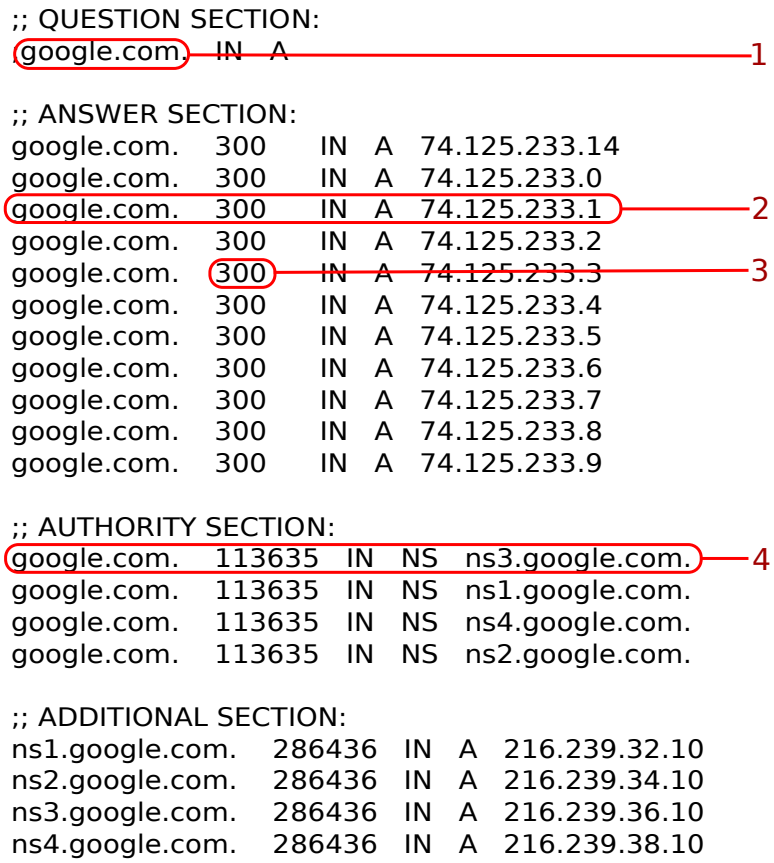


Figure 2.3: Output for the DNS Resolution Process Using the *dig* Command

Figure 2.3, where the resolution of the domain google.com has been done using the dig[1] command.

Examining the query response in Figure 2.3, the value at 1 represents the ORIGIN directive, which is the name of the domain that has been queried. This will be extracted from DNS query responses and used in the lexical analysis of domain names as outlined later in section 3.2. The next value of concern is the A Resource Record, as identified by 2. This value defines the IPv4 address of a particular host in the domain. A related field to this is the AAAA Resource Record, which has an identical structure to a standard A Resource Record. It is used to identify a host using it's IPv6 address. Each IP address associated with an A record has an Autonomous System Number (ASN), which is used in the routing of traffic on the Internet (Dragon Research Group, 2011). Each ASN is usually associated with a single organisation and is thus a good indicator of the owner of an IP address block. The value at 3 is the Time-to-Live (TTL) directive for the domain.

[1]dig (domain information groper) is a network administration command-line tool for querying DNS name servers. Dig is a built-in tool in many Linux distributions.

This value specifies the amount of time, in seconds, that a domain name should be cached by another name server. Once the TTL expires the DNS record should be renewed by performing a new DNS query. RFC 1912 recommends minimum TTL values of 1-5 days, allowing clients to benefit from the effects of DNS caching (Barr, 1996). The final value of interest is labelled as 4. This is the NS Resource Record and is used to identify the authoritative name servers for the domain. The authoritative name server is used to host the DNS records for a domain and answer queries for resolution.

2.2 Botnets

The term 'botnet' is used to describe a collection of compromised hosts that are networked together and are under the control of a third party, known as a botmaster (Pereira et al., 2007; Shadowserver Organisation, 2012). These compromised hosts may consist of any computing device capable of accessing the Internet, including private home computer systems, corporate computer systems and even mobile devices (Xiang, Binxing, Lihua, Xiaoyi, and Tianning, 2011). Botnets are used by botmasters to commit multiple cybercrimes such as spam distribution, Distributed-Denial-of-Service (DDoS) attacks and malicious software distribution. Furthermore, the compromised hosts may report back to the botmaster with user details such as online banking passwords and credit card information. Once a botnet has been created, the botmasters require a means of communicating instructions to all the hosts in the botnet. A common strategy for this is the use of Command and Control (C2) servers. These C2 servers provide a central location for botnet members to receive instructions as well as a location to report back with stolen information. As C2 servers provide a central location for hosts to call-back to, they also create a central weak-point which security researchers can use to shutdown the entire botnet. For this reason botmasters have employed many detection and shutdown techniques. The structure of botnets and the evasion techniques employed by botmasters are discussed in greater detail in the subsequent sections.

2.2.1 Structure of Botnets

The structure of a typical botnet is presented in Figure 2.4, where the botnet consists of all the components outlined above. The botmaster is able to submit commands to the bots through the C2 servers, while using the C2 servers as relays for data being returned by the bots. The use of C2 servers as proxies makes locating the botmasters location

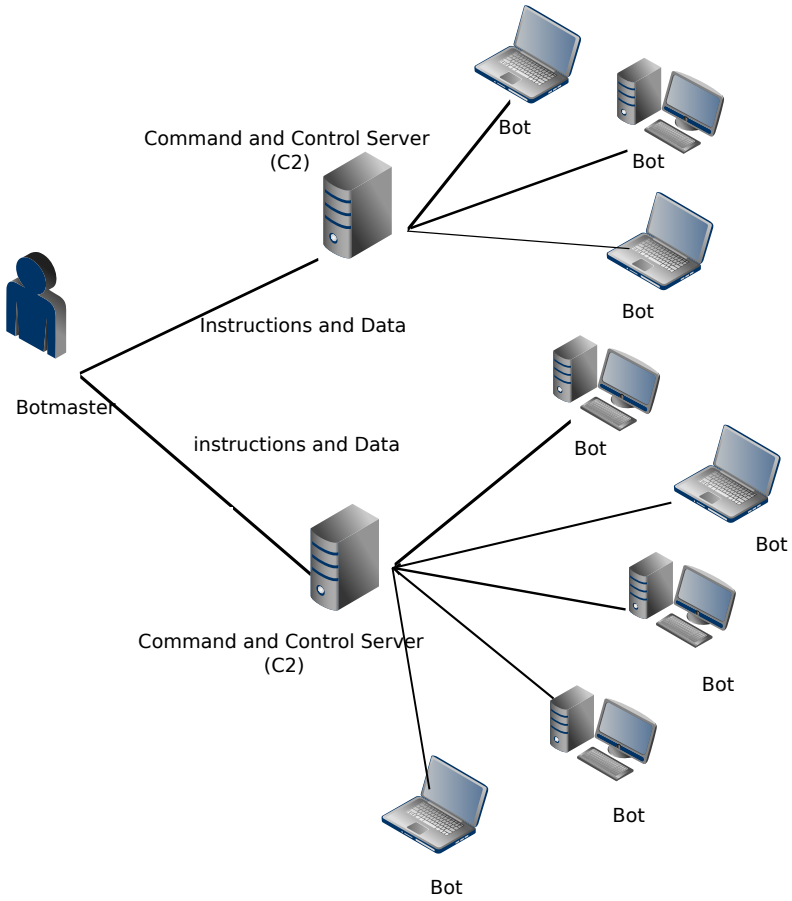


Figure 2.4: Botnet Structure

nearly impossible and provides anonymity to the botmasters. As the botmaster relies on C2 servers for communicating with the bots of the botnet, C2 servers are the weak point of any botnet as taking down a C2 server denies a botmaster access to the bots of the botnet (Silva, Silva, Pinto, and Salles, 2012). To slow down locating and shutting down of C2 servers, botmasters have employed multiple strategies. Two of these strategies, DNS Fast-Flux and algorithmically generated domain names, are discussed in the next sections.

Botnets consist of multiple parts, independent of the botnet's size or the architecture employed. These parts each play different roles in the operation of the botnet and provide a different target for managing and taking down botnets. This structure can be broken down into four main components as follows:

- Botmaster: An individual or group that controls the operation of the botnet. The botmaster may not be the creator of a botnet and may have purchased the botnet from another botmaster or rent the botnet temporarily. The botmaster

issues commands to the bots belonging to the botnet through the C2 servers using the C2 channel.

- Bot: A bots in the botnet may consists of any computing device that has been compromised and is under the control of the botmaster. The bots in a botnet receive instructions from the botmaster through the C2 channel and will perform periodic checks with the C2 servers for new instructions. The botmaster may also directly control the bot through a remote access trojan (RAT). Bots may be instructed to harvest user information on compromised hosts. This information may include keystrokes, credit card information and online banking logins. Botmasters may compromise hosts in multiple ways including malicious software, drive-by-downloads and hacking. Bots will typically perform DNS queries, just as legitimate hosts would, to find the IP addresses of C2 servers to contact (Morales, Al-Bataineh, and Sandhu, 2009).

- C2 Server: C2 servers are used to relay instructions and data between the botmaster and the bots of the botnet. In many cases the botmaster will never have direct access to individual bots but rather uses the C2 servers as proxies for controlling these hosts (Silva et al., 2012). The C2 servers may be bots on the botnet, server infrastructure such as web-servers which have been compromised, or so called bullet-proof hosting. Bullet-proof hosting consists of networks that are known to tolerate hostile network activity and ignore take-down notices for malicious hosts.

- C2 Channel: The C2 channel employed defines how communication is performed between bots and the C2 servers. Early botnets employed Internet Relay Chat (IRC) as a communication channel but in recent years there has been a shift towards the use of standard HTTP communication channels. This has largely been due to stricter firewall policies, limiting the types of traffic which may egress from a network. With HTTP being ubiquitous in the functioning of the web traffic, it is nearly always allowed on a network (Lee et al., 2008). Furthermore, botnet HTTP traffic is not as noticeable with legitimate HTTP traffic already present on most networks. Botnets have steadily been shifting towards the use of encrypted communication channels, preventing Man-in-The-Middle (MiTM) attacks by security researchers. This shift towards encrypted C2 channels has made botnet take down more difficult and has made it near impossible to submit rogue commands to bots belonging to a botnet.

Table 2.1: Fast-Flux Domain Query Result

IP Address	AS Number	Organisation	Country
222.106.31.123	AS4766	Korea Telecom	KR
95.139.78.214	AS48400	CJSC "Comstar-Regions"	RU
110.133.1.126	AS9824	Technology Networks Inc	JP
80.54.192.197	AS5617	Telekomunikacja Polska S.A.	PL
211.125.152.16	AS10019	Matsusaka Cable-TV Station Inc	JP

Table 2.2: Content Distribution Network DNS Query Result (*fbcdn.com*)

IP Address	AS Number	Organisation	Country
66.220.149.88	AS32934	Facebook, Inc.	US
66.220.152.16	AS32934	Facebook, Inc.	US
66.220.158.70	AS32934	Facebook, Inc.	US
69.171.234.21	AS32934	Facebook, Inc.	US
69.171.237.16	AS32934	Facebook, Inc.	US
69.171.247.21	AS32934	Facebook, Inc.	US

2.2.2 DNS Fast-Flux

DNS Fast-Flux is another method used by botmasters to build resilient and robust botnet control infrastructure. Fast-Flux uses rapidly, repeatedly changing DNS records to provide constantly changing IP addresses to which a domain name resolves. The use of rapidly changing DNS records is not in itself malicious and has been used by legitimate services to provide load balancing for high availability and high volume web sites (Holz, Gorecki, Rieck, and Freiling, 2008). Fast-Flux domains rely on a short TTL for the resource records, ensuring that each subsequent DNS query will request a new resource record and not use the cached version. With each new DNS query, a new set of resources records is returned with new resources records mapping to different IP addresses from the previous query. This rapid flux in DNS records provides a means of concealing the C2 servers as no two DNS queries map back to the same hosts, with many of the resource records returned linking to proxy hosts which relay instructions back to the C2 servers. A further benefit of DNS Fast-Flux is that as long as a single address returned is available, the whole service remains online.

As has been stated, Fast-Flux can be used as a legitimate means of providing load balancing for web sites such as those hosted on CDNs. There is, however, a noticeable difference between legitimate CDNs and Fast-Flux domains (Holz et al., 2008). Botmasters are not free to choose the hardware and network location of individual nodes, resulting in diverse IP ranges being returned with each DNS query response. This can be seen in Table 2.1,

where five widely dispersed IP ranges are returned for a single DNS query. Furthermore, each of these IP ranges were registered to a different organisation and belong to different ASNs. The geographic locations of the hosts also appear randomly distributed globally, with four different countries represented. This can be compared to the results for CDNs' DNS query result in Table 2.2, where - despite two IP ranges being returned - they are from the same ASN and both ranges belong to Facebook Inc. Furthermore, all the hosts mapped to these IP addresses are located in the United States and are not widely dispersed geographically, as is the case with the Fast-Flux domain presented in Table 2.1. It has been noted that employing Fast-Flux as a defensive measure to mitigate DDoS attacks has a high success rate (Lua and Yow, 2011). For this reason Fast-Flux presents a double-edge sword, as it provides an effective means for both legitimate and malicious domains to maintain availability and throughput (Lua and Yow, 2011).

2.2.3 Algorithmically Generated Domain Names

Algorithmically generating domain names is a technique employed by various botnet families to increase the lifespan of botnet C2 domains. These domain names are generated using a Domain Generation Algorithm (DGA), ensuring that the botmaster knows in advance which domain names will be generated and thus knowing which domain names to register. By using a DGA botmasters are able to have clients generate a large number of domain names that could possibly be used by C2 servers (Damballa, 2012). Due to the large number of domains being generated, security researchers are unable to determine which domain names will actually be used. Furthermore, the sheer volume of domain names makes it near impossible to preregister all the domain names that have been generated. The technique of preregistering domain names had been effective in preventing communication between bots and the C2 servers for early variants of botnets such as Conficker-A and Conficker-B, where only 250 domain names were generated in a day. The release of the Conficker-C variant saw the generation of 50000 domain names a day. This was an extremely difficult number of domains to preregister, in both monetary and logistic terms (Leder and Werner, 2009). As the botmasters require only one domain to be reachable, researchers have to ensure a one hundred percent success rate in preregistering all domains if they wish to effectively prevent communication between the botmaster and the botnet. The botmasters are able to pre-generate a list of domain names that will be generated by the botnet hosts in the future and register a selection of these domains in advance.

The success of the Conficker-C botnet has led to the use of DGAs by multiple botnets,

Table 2.3: Sample Algorithmically Generated Domain Names

Torpig	Kraken	Conficker-C
wghfqlmwtwe.com	aafsyt.dyndns.org	nzfpt.ir
oderkayotwe.com	aetqzvfzub.dyndns.org	gxigcsiv.sh
ghplzwgtwe.com	aifuunhoomc.dyndns.org	motied.sk
wdecfjiwtwe.com	gmqaeoeudhd.dyndns.org	zuwomto.com.fj
bcplcwytwe.com	kpjobheecz.dyndns.org	dtofeqdih.in
aefjchpatwe.com	likkhxbl.dyndns.org	bgwmdt.hu
mhjqxaxmtwe.com	bnbnpqkagr.dyndns.org	hdwcgwvr.co.za
sfgtilbstwe.com	danssxjpgqh.dyndns.org	udjgefanm.com.ag
aefnmvuatwe.com	ggdcnsp.dyndns.org	slkvruja.am
jbcfqmmjtwe.com	baqydcdnusq.dyndns.org	rozikf.com.gt
ocdvjxdotwe.com	uresesbfsb.dyndns.org	mcptvhezs.com.hn
ajicfjiatwe.com	gbsszmdkuq.dynserv.com	jkoo.com.do
qefswxaqtwe.com	zpuxycznd.dynserv.com	hfhvue.vn

with the five largest DGA based botnets being Conficker, Murofet, BankPatch, Bonnana and Bobax (Damballa, 2012). The DGAs use a seed value that is consistent across all the bots of the botnet, such as the system time. The Torpig botnet used the Twitter API[2] to generate domain names from the most popular terms being discussed on the platform. The pseudo random nature of these domain names mean that characters in the generated domain names will appear to have a uniformly random distribution. Closer investigation reveals that certain characters tend to appear more frequently than others. The frequency distribution of characters is discussed in greater detail in section 3.2. Samples of algorithmically generated domain names for the Torpig, Kraken and Conficker-C botnets are shown in Table 2.3. It can clearly be seen that the domain names generated do not look the same as legitimate domain names such as facebook.com, youtube.com and myspace.com, which all consist of a combination of English words. Furthermore, it can be noted that all the Torpig domain names end in 'twe', indicating they were generated in December Unmask Parasites (2009). This is due to the DGA used to generated these domains, where a suffix relating the current month is appended to each generated domain name (Unmask Parasites, 2009).

[2]https://www.twitter.com/api

2.3 Related Work

The emergence of botnets as a serious threat in the modern Internet landscape has led to numerous researchers to examine ways of detecting botnets and the traffic associated with botnets. Two areas of botnet research related to this work were examined. Work in the analysis of URLs is discussed in subsection 2.3.1, where previous attempts to detect algorithmically generated domains are examined. Research in spam filtering is also discussed as this relates to the Bayesian techniques used in this research. The second subset of botnet research examined was DNS Fast-Flux detection and this is discussed in subsection 2.3.2. Multiple techniques of botnet detection based on DNS query analysis are outlined.

2.3.1 URL Analysis

Signal theory and signal processing methods have been proposed as an approach for detecting algorithmically generated domain names (Yadav et al., 2010). Analysis of the distribution of alphanumeric characters as well as the distribution of bigrams within domain names was conducted in Yadav et al. (2010). The frequency distribution of characters were found for legitimate and algorithmically generated domain names. The K-L divergence[3] was used as a statistical measure of how closely the frequency distributions for the test domain relate to the known distributions for legitimate and algorithmically generated domain names. A domain was classified based on which distribution it diverged from the most. They extended their analysis to examine the distribution of bigrams (also known as character pairs) in an attempt to identify domain names that had been generated using algorithms that attempt to match the frequency distribution of characters in natural language. The techniques employed required domain grouping to increase accuracy, with observed domains being primarily grouped according to domain and secondly according to IP address. Domain name grouping was done on sets of 50, 100, 200 and 500 test words, with at least 50 domain names in a group required to positively identify malicious domains, while the researchers noted the best results were achieved once 500 domain names had been analysed. The results achieved showed a 100% detection rate when 500 domain labels were used, with only 5-7% false positives, with a true positive rate (TPR) of 93%. Bigram analysis resulted in more domains being detected with a lower false positive rate of 2%. When only 50 domain names were present in a grouping, the detection rate dropped to 80% with a 20% false positive rate (FPR).

[3]Kullback–Leibler divergence: A measure of the difference between two probability distributions

The analysis of domain names based on the character distribution offer the benefit of not having to do costly lookups of known 'safe words', such as suggested in other works such as Alienvault Labs (2012). The work by Alienvault Labs (2012) used syntax heuristics, where domains were classified according to the number of consonants contained in the domain name after common English words such as 'or', 'and', 'page', 'free' had been removed. The results from the AlienVault research showed a 61% TPR and a low FPR of 10%. These results were obtained using only Conficker-C domains as the malicious dataset. While the low FPR and relatively high TPR are commendable, the system is easily bypassed by simply reducing the number of consonants in the domain name. This can be achieved by altering the DGA and thus it is hypothesised that the AlienVault solution would not be as effective in detecting algorithmically generated domain names for previously unseen malware.

The move towards social services such as Twitter, where users are limited in the number of characters which they can post, 140 character limit, has seen the emergence of URL shortening services (Lee and Kim, 2012). These shortening services provide an intermediary service between the user and the shortened domain, where a mapping between a unique short domain and a fully qualified domain is created. The popularity and sheer number of URL shortening services available has seen a move by botnet creators towards using these services as a means of fluxing between botnet C2 servers (Lee and Kim, 2012).

The URLs used for phishing and advertising spam were analysed by Ma, Saul, Savage, and Voelker (2009). They identified that malicious URLs exhibit different alphanumeric distributions than legitimate URLs. Statistical learning techniques were employed to identify malicious URLs from lexical features such as domain name length, number of dots in the URL and host names. The proposed system aimed to identify single URLs as malicious, whereas Yadav et al. (2010) looked at the grouping of domain names (Ma et al., 2009). Work performed by Xie, Yu, Achan, Panigrahy, Hulten, and Osipkov (2008) led to the development of regular expression based signatures for detection of spam URLs. The solution proposed in this research is intended to surpass the accuracy of the regular expression based solution. Furthermore, the solution should be harder to bypass and will avoid the need to constantly update signatures to match new attacks as they develop.

Classifiers based on Bayesian statistics have been used successfully in the classification of binary problems. One area of information security where this has been particularly successful is in the detection of spam emails (Seewald, 2007; Sahami, Dumais, Heckerman, and Horvitz, 1998). Bayesian statistics allow for the creation of classifiers that are content-based as well as self-learning, allowing for probabilistic problem solving (Androut-

sopoulos, Koutsias, Chandrinos, Ch, Paliouras, and Spyropoulos, 2000). The two main types of classifiers that have been used in the detection of spam emails are Bayesian classifiers and Naive Bayesian classifiers. The Bayesian classifiers assume dependence between the attributes being examined, while the Naive Bayesian classifiers assume independence between attributes. Assuming independence allows for smaller, less complicated training sets to be used. Furthermore, the probabilities of attributes in the Naive Bayesian classifier are easier to calculate as the dependence on other attributes do not have to be taken into account. It has been shown that both Naive and standard Bayesian classifiers allow for highly accurate classification of spam mail (Androutsopoulos et al., 2000; Seewald, 2007; Zhang, Zhu, and Yao, 2004), leading to the notion that they may easily be adapted to create accurate classifiers for URL classification.

2.3.2 DNS Fast-Flux Detection

A number of approaches for detecting malicious network activity through DNS traffic monitoring were studied. The system implemented by Perdisci, Corona, Dagon, and Lee (2009) for the detection of malicious Fast-Flux service networks through the passive analysis of recursive DNS traffic traces identified common features in Fast-Flux DNS query results (Perdisci et al., 2009). Common features identified were a short time-to-live (TTL), multiple Address (A) records and multiple ASNs. It was shown that the IP addresses resolved to the domain name were often from dissociated networks and changed rapidly, matching the operation of Fast-Flux as described in subsection 2.2.2. The definitive work in Fast-Flux detection was done by Holz et al. (2008), who identified the same key DNS features. These DNS features were used in the creation of heuristic classification models for the detection of Fast-Flux botnets. Their primary observation was that Fast-Flux botnets could be detected using the number of distinct A records returned and the number of different ASNs associated with the domain. Results showed botnet creators attempt to mimic the structure of CDNs. This behaviour masks botnet activity and hinders the automatic classification of domains (Holz et al., 2008). It was, however, identified that the inherent distributed structure of botnets could be used as a distinguishing factor. The authors Holz et al. (2008) noted that botmasters have limited control over the hardware and network location of individual nodes, where CDNs have full control over the locations of nodes. Furthermore, botmasters could not easily obfuscate these features. These features were used in a Fast-Flux detection system. The proposed system required secondary DNS queries once the original queries TTL expired, increasing the time required to classify domains.

The distributed nature of botnet C2's is a well established fact and researchers have attempted detection and classification of botnets using the geographic locations of botnet nodes (Caglayan, Toothaker, Drapaeau, Burke, and Eaton, 2009; Hu, Knysz, and Shin, 2011; Huang, Mao, and Lee, 2010). The research conducted by Huang et al. (2010) proposed a method for delay-free detection of Fast-Flux service networks. The solution relied on spatial distribution estimation and spatial service relationship evaluation. Timezones were used to distinguish between different geographic system spaces and were combined with information entropy to measure how uniformly nodes were distributed. The authors noted that benign domains tend to be distributed in the same timezone, while Fast-Flux nodes are widely distributed across multiple timezones. The authors further noted that if all the hosts of a botnet were to be located in the same timezone, timezone based entropy would not be an effective measure for detecting if the hosts belonged to a benign or Fast-Flux domain. The work performed by Caglayan et al. (2009) aimed to model the behavioural patterns of Fast-Flux botnets. Using DNS records, they showed Fast-Flux botnets exhibit common characteristics: that botnets form clusters based on botnet size, growth and operation. It was further shown that the majority of Fast-Flux botnets operate in more than five countries at a time, averaging between 20 and 40 countries. In Hu et al. (2011) the global IP usage patterns of Fast-Flux botnets were studied. Their research benefited from a global perspective, with 240 nodes on four continents monitoring DNS traffic. Hu et al. (2011) found that Fast-Flux botnets advertise IP addresses from multiple countries, irrespective of where the DNS query came from, where as CDNs advertise IPs in a geographically aware manner. This observation provides valuable insight into the operation of Fast-Flux botnets, and further helps determine how a classifier that is capable of differentiating between Fast-Flux botnets and CDNs may be constructed.

2.4 Summary

This chapter presented background information pertinent to the research presented in the rest of this document. The Domain Name System was described in section 2.1, where the DNS hierarchy was explained along with the mapping between easy to remember domain names and the numerical addresses used by hosts on the Internet. The structure of DNS query responses were outlined along with explanation of the records relevant to this research.

section 2.2 provided background information on botnets and their structure. The relationship between C2 servers and the hosts of a botnet was described as well as the

communication channels used by botnets. Two detection evasion techniques employed by modern botnets were described. subsection 2.2.3 described the use of DGAs by hosts on a botnet to generate a list of domain names to query when attempting to contact the C2 servers. A brief description of the Torpig domain name generation algorithm was provided to represent common techniques used by botnet creators in setting up DGAs. The second detection avoidance technique described was DNS Fast-Flux. subsection 2.2.2 described the operation of Fast-Flux botnets and how DNS records are used to mask the addresses of the C2 servers and to increase the botnets resilience against shutdown.

Finally related works in the detection of algorithmically generated domain names were presented in subsection 2.3.1, along with works focusing on the detection of malicious domain names. Bayesian spam filtering techniques were discussed, leading to the premise that these techniques may be applied to the detection of algorithmically generated domain names. Research into Fast-Flux domains and the detection there-of was described in subsection 2.3.2. Research into the use of geographic dispersion in the detection of botnets was presented and briefly discussed.It was shown that methods for detecting Fast-Flux botnets have been successful and that there is, however, room for improvement.

The techniques used in related works are expanded upon and improved in the chapter 3. New and novel means of detecting algorithmically generated domain names are presented, along with the adaption of Bayesian spam filtering techniques to detect algorithmically generated domain names. Three techniques expanding on the research discussed in subsection 2.2.2 are described. These aim to detect Fast-Flux domains from DNS query responses. Spatial autocorrelation is presented as a new, novel technique for detecting Fast-Flux based on the geographic distribution of C2 servers.

3 Techniques for Botnet Identification

This research aims to identify botnet traffic on a network before actual communication is established between the infected host and the Command and Control (C2) servers. Botnet creators have developed a variety of means of allowing the infected hosts to contact the correct C2 server. These techniques, like most connections on the Internet, rely on the Domain Name System (DNS). This chapter describes multiple techniques used in the identification of DNS queries used by infected hosts attempting to communicate with C2 servers. By detecting queries for C2 servers, it is possible to block traffic to these host preemptively ensuring hosts are unable to establish a communication channel with the botnet controller and thus prevent infected hosts from receiving instructions. Infected hosts are identified in the process allowing network administrators to clean infected hosts.

The data used in the training and testing of the classifiers is described in section 3.1. Techniques for identifying algorithmically generated domain names are described in section 3.2. The DNS query responses are examined in more depth in section 3.4, where multiple techniques for detection based on these features are detailed. In section 3.5 the geographic features extracted from the DNS query are detailed as another means of accurately identifying Fast-Flux domain queries.

3.1 Data Description

Due to the nature of botnets and the multiple detection techniques examined, the data used in this study was collected from various sources. Furthermore the data was standardised and divided into two distinct datasets: training data and test data. This was done to allow classifiers to be trained using one set of data while the testing of classifier

accuracy was performed on a separate set of data. The use of two datasets ensured an accurate representation of classifier accuracy as the classifier results were not influenced by testing known data.

3.1.1 Domain Name Data Sources

The data used for the lexical analysis of domain names was taken from sources known to contain accurate and clean data. These sources are outlined below and a summary of the datasets are presented in Table 3.1.

- Dictionary words: 10 000 words taken from the Oxford English Dictionary (Oxford University Press, 2011), based on words longer than 6 characters. The dataset is labelled as AD1,
- Legitimate domain names: 10 000 domain names from a combination of the *Google Doubleclick Ad Planner Top-1000 Most Visited Sites* (Google, 2012) and the *Alexa Top 10 000 Global Sites* list (Alexa, 2012). The combined dataset is labelled as AD2,
- Algorithmically generated domain names: 10 000 sample domain names generated by Kraken (Royal, 2008), Torpig (Unmask Parasites, 2009) and Conficker-C (Leder and Werner, 2009; Porras et al., 2009). The combined dataset is labelled as AD3.

The training and test data were extracted from these datasets at random with a random selection of 3000 dictionary words, legitimate domain names and algorithmically generated domain names being selected from each dataset. The remaining 7000 samples from each dataset were then used as the test data for measuring classifier accuracy.

Table 3.1: Domain Name Data Sources

Label	Description
AD1	English Dictionary Words
AD2	Legitimate Domain Names
AD3	Algorithmically Generated Domain Names

3.1.2 Fast-Flux Data Sources

Acquiring the data used in the construction and testing of Fast-Flux classifiers was particularly difficult. This was due to the nature of Fast-Flux botnets, which dictates rapidly changing variables within the datasets as well as domains frequently being taken offline. Due to this constant flux the values for each domain feature were captured over the period that the domains were active and stored for further examination at a later stage. Classifier construction was done using this historical data while testing was performed on live domains when possible and historical data for reference. The collection of data was further hampered by the take-down of multiple botnets during the research period. The re-emergence of the *Hlux2* (Garnaeva, 2012) botnet for a limited period of time allowed for the capture of valuable data, however the rapid take-down of this botnet made live testing impossible. The take-down of the ZeuS botnet by Microsoft (Microsoft Corporation, 2012) hampered live testing of the classifiers during the later stages of the research, while the emergence of the Citidale variant of ZeuS has led to a recent increase in the number of Fast-Flux botnet domains. A set of 500 domains were randomly selected as training domains, while the remaining 1500 domains were used for testing the classifiers accuracy.

The sources of Fast-Flux botnet data were as follows:

- ZeuS Tracker (abuse.ch, 2012) produced a set of 238 Fast-Flux domains for the period March 2011 - October 2012. Labelled as FD1.
- Spyeye Tracker (abuse.ch, 2011) produced a set of 28 Fast-Flux domains for the period March 2011 - September 2011. Labelled as FD2.
- The Hlux2/Kelihos botnet domains acquired from a private source at a large European Internet Service Provider consisted of 507 Fast-Flux domains that had been manually classified for the period 23 January 2012 - 1 March 2012. Labelled as FD3.
- Arbor ATLAS summary report (Arbor Networks, 2012b) consisted of 674 Fast-Flux domains for the period January 2012 - March 2012. Labelled as FD4.

The total number of Fast-Flux botnet domains in the dataset was 1447, with the combined dataset being divided at random into a set of 400 training domains and 1047 test domains. The dataset of legitimate domains consisted of the top 2000 domains taken from a merged dataset of the Google Doubleclick Ad Planner Top-1000 Most Visited Sites (Google, 2012)

and the Alexa Top 10000 Global Sites list (Alexa, 2012). The datsets used were labelled with the prefix FD with each datasource labelled as FD1, FD2, FD3 and FD4 respectively. This can be seen in Table 3.2.

Table 3.2: Fast-Flux Data Sources

Label	Sample Size (domains)	Botnet Family
FD1	238	ZeuS
FD2	28	Spyeye
FD3	507	Hlux2/Kelihos
FD4	674	Unknown
FD5	2000	Legitimate

3.1.3 Live Test Data

Testing of the classifiers was performed using DNS traffic logged at a large South African university and a local schools network to determine if the classifiers were capable of identifying malicious domains not seen anywhere else. The datasets consisted of a pcap dump containing 40 910 498 raw DNS packets. This dataset was labelled as LD1. Accompanying this raw pcap dump, a secondary dump of 33 261 575 visited URLs along with timestamps as seen by the web proxy. This dataset was labelled as LD2. To simulate blacklist testing, a set of test data was obtained from MalwareURL (MalwareURL, 2011), containing a listing 254 214 malicious URLs. This dataset has been labelled as LD3. A summary of the datasets used is shown in Table 3.3.

Table 3.3: Live Test Data

LABEL	SIZE
LD1	40 910 493 packets
LD2	33 261 575 domains
LD3	254 214 domains

3.2 Lexical Features

The Domain Name System relies on the mapping of alphanumeric, human-understandable and -memorable addresses to numeric IP addresses. These domain names usually consist of dictionary words or a combination of dictionary words in a memorable sequence. Randomly generated domain names are a technique used by botnet creators for ensuring C2

longevity and to protect against the shutdown of C2 domains. These randomly generated domain names need be consistent across the botnet to ensure that all infected hosts contact the correct C2 servers. To this end, botnet operators use a class algorithm, known as a Domain Generation Algorithm (DGA), to determine how these domain names are generated. These algorithmically generated domain names tend to exhibit a bias toward certain character distributions depending on how the algorithm was constructed. These character distributions vary greatly from the character distributions found in legitimate dictionary words and thus legitimate domain names (Crawford and Aycock, 2008). Using the lexical features of a domain name, frequency analysis can be performed to determine the distribution patterns for letters in dictionary words, domain names and algorithmically generated domain names. The DGAs have been reverse engineered for numerous botnet families including Torpig (Unmask Parasites, 2009), Conficker (Conficker Working Group, 2010) and Kraken (Royal, 2008). This process of reverse engineering the DGA is, however, time consuming and tedious (Thomas Barabosch, 2012), hence the need to detect this algorithmically. Frequency analysis can be based on single character distribution or on the distribution of character combinations such as *ie*, *ch* and *qu*, known as bigrams.

3.2.1 Frequency Analysis

Establishing a baseline for all comparison is essential in creating effective classifiers. Frequency analysis was performed on datasets of dictionary words, known legitimate domain names and known algorithmically generated domain names. Frequency analysis was done on single character distributions (unigrams), character combinations (bigrams) and on the distribution of vowels and consonants. The frequency distribution for unigrams for dictionary words, legitimate domain names and algorithmically domain names are shown in Figure 3.1.

As seen in the figure, there is a clear distinction between unigram distribution from legitimate and algorithmically generated domain names. Legitimate domain names tend to follow the same distribution pattern for unigrams as English words. This can be attributed to the fact that the domain names examined are largely taken from the 'English Web'. Algorithmically generated domain names follow a similar distribution pattern as a random distribution of unigrams. There is, however, a bias towards certain unigrams, which can be attributed to the pseudorandom nature of DGAs.

Bigram analysis focused on all possible two character groupings for alphanumeric characters allowed in domain names. The frequency distribution of these bigrams can be

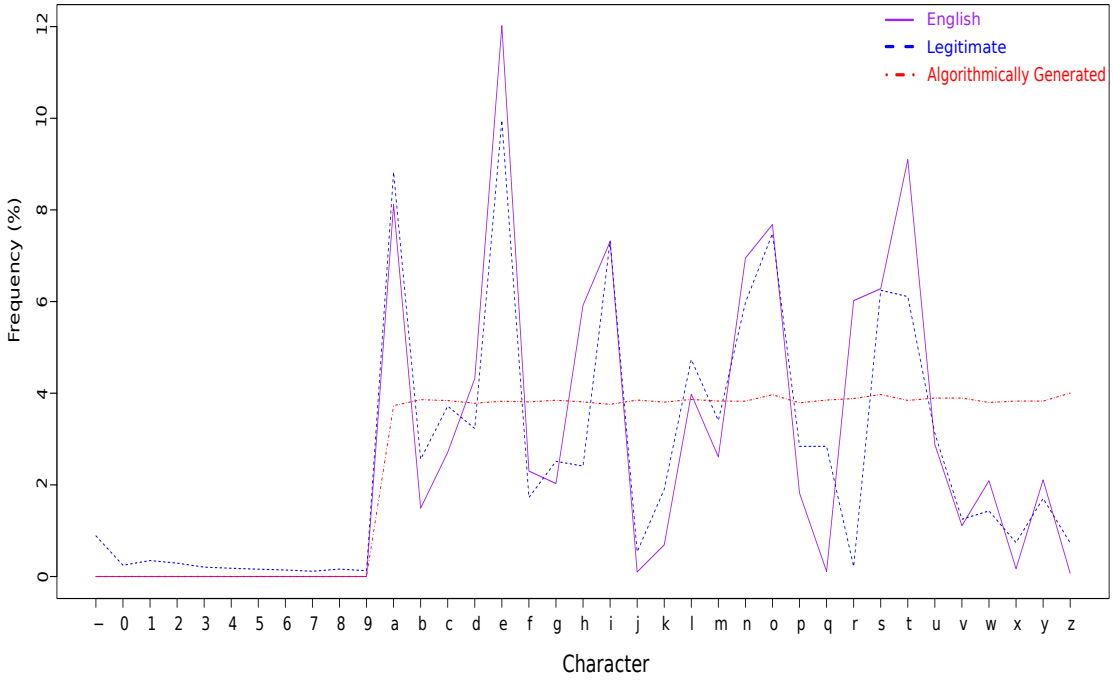


Figure 3.1: Frequency Distribution of Unigrams.

seen in Figure 3.2. The figure clearly shows that the bigram frequency for algorithmically generated domain names never exceed 0.3%, while common bigrams from the English language display the same frequency distribution in legitimate domain names. The inset shows the bigram distributions for all combinations of the regular expression [0-9][a-z]. The frequency distributions of the combinations *ah*, *aj*, *al* and others have the identical frequency distributions for legitimate domain names and English words.

3.3 Lexical Classifiers

The clear distinction between the frequency distribution of both unigram and bigram characters in algorithmically generated domain names and legitimate domain names lends itself to the idea that highly accurate classifiers can be constructed based on the known frequency of character distributions. The problem of identifying a domain name as legitimate or algorithmically generated can be seen as a binary problem and thus binary classifiers and likelihood ratios were investigated. For the remainder of this section unigrams and bigrams are collectively referred to as tokens.

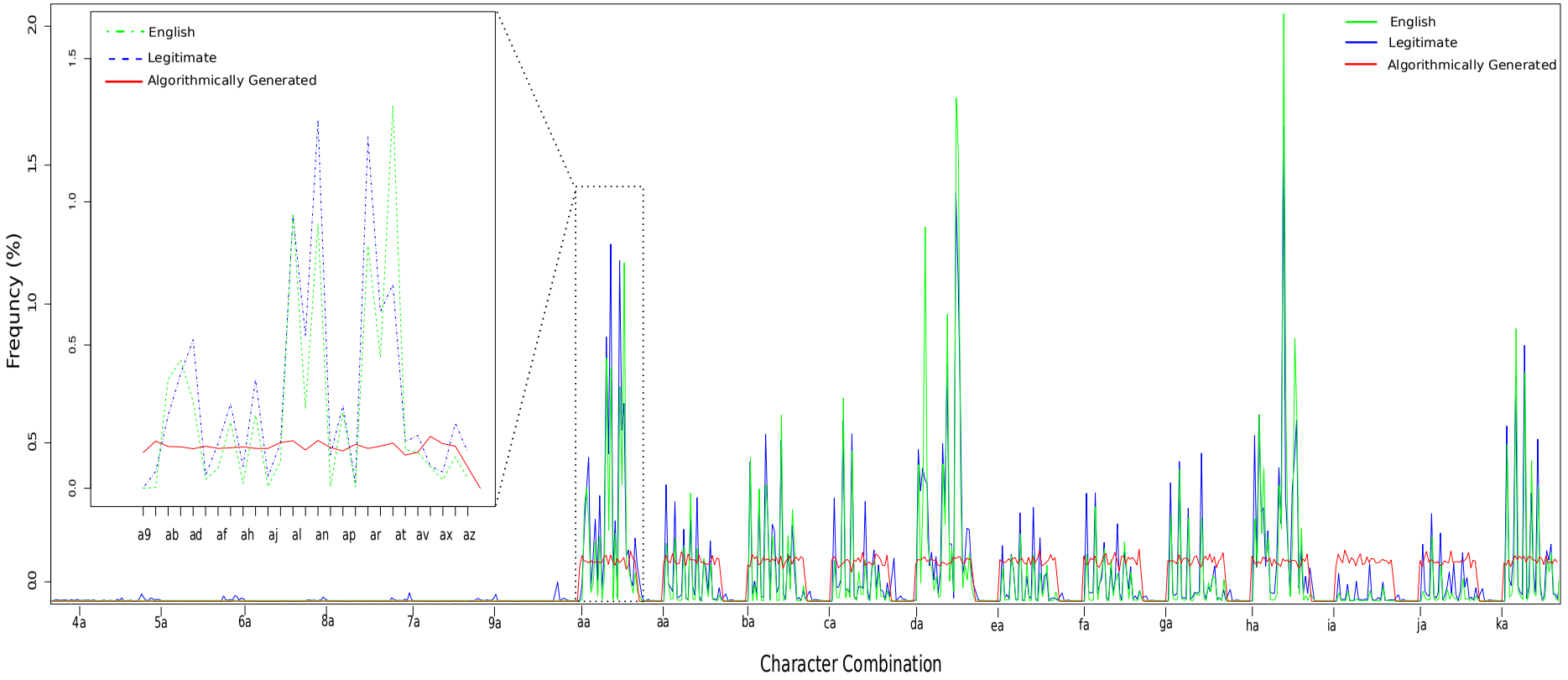


Figure 3.2: Frequency Distribution of Bigrams.

Algorithm 3.1 Total Probability

$$P(D|C) = \prod_i p(x_i|C)$$

Where:

- $P(D|C)$ is the probability that a domain name D belongs to a class given all letters x_i
- x_i is the token at i; for P('google'): x_0 ='g'

3.3.1 Probability Distribution

Frequency analysis results in the frequency distributions of tokens for the target domain space. These frequencies can be represented as the probability of a token occurring in a domain name. The probability of a token occurring is treated as an independent event, with preceding characters having no influence on the probability of a token occurring. The probability of a token occurring in one of the two domain spaces, legitimate or algorithmically generated, is denoted as $P(x|B)$ where x is the token and B represents the legitimate class.

The most basic means of calculating the likelihood of a domain name being legitimate or algorithmically generated is to calculate the product of probabilities for each token occurring in each class. This can be represented by the formula shown in Algorithm 3.1.

The product of probabilities are calculated for both legitimate and algorithmically generated domain names. These are then compared and the larger of the two probabilities is used as an indicator of which domain space the domain name belongs to.

3.3.2 Total Variation Distance

The total variation distance is the maximum possible difference between two probability distributions that can be assigned to a single event (Ehm, 1991). The variation distance is used to gauge the difference between the probability of a domain name being legitimate or algorithmically generated. The total variation distance measure used relies on a finite alphabet being defined. It is possible to generate a finite alphabet with frequency analysis and the knowledge that the characters used in domain names remain constant. The total variation distance can be formally defined as seen in Algorithm 3.2.

Algorithm 3.2 Total Variation Distance

$$\sigma(P, Q) = \frac{1}{2} \sum_{i=1}^{n} \mid P(x_i) - Q(x_i) \mid$$

Where:

- x_i is the token being examined (either a unigram or bigram)
- $P(x_i)$ is the probability of x_i occurring in a legitimate domain name
- $Q(x_i)$ is the probability of x_i occurring in an algorithmically generated domain name

3.3.3 Bayesian

Bayesian inference is a statistical technique that is useful in the classification of problem domains which have binary outcomes (Sahami et al., 1998). Bayesian inference is based on Bayes' rule and is used to update the probability estimate for a hypothesis as additional evidence. The outcome of this probability estimate is a likelihood ratio that compares the likelihood that an observation belongs to a specific domain space. Bayesian classifiers have been used to solve other computer security related problems and have become a well established means of creating self-updating classifiers for binary classification problems. The use of Bayesian classifiers have been particularly effective in email spam-filtering where words found in emails are given probabilities of occurring in spam email and legitimate email (Sahami et al., 1998). Probabilities are calculated from training data that had been manually classified as either spam or legitimate. Once a classifier had been trained it could be used for classification (Seewald, 2007; Zhang et al., 2004).

During frequency analysis the particular probabilities of tokens occurring in legitimate and algorithmically generated domain names are calculated. Legitimate domain names tend to follow the token frequency distribution in line with the distribution of tokens in the English language, while algorithmically generated domains display frequency distribution similar to a random distribution of tokens. The classifier doesn't know these probabilities in advance and is thus trained to build them up. Training of the classifier was done using known legitimate domain names and known algorithmically generated domain names. As each token is encountered in the training domains, the classifier adjusts the probability of that token occurring in legitimate and algorithmically generated domain names. Once the classifier has been trained, the token probabilities are used to compute the probability that a domain belongs to either category. The classification of a domain relies on the probabilities of each token found in the domain name. This reliance on each

Algorithm 3.3 Bayesian Classifier

$$P(F \mid t) = \frac{P(t \mid F).P(F)}{P(t \mid F).P(F) + P(t \mid \neg F).P(\neg F)}$$

Where:

- $P(F \mid t)$ is the probability that a domain name is algorithmically generated, if the token t is in the domain name
- $P(F)$is the overall probability that a domain name is algorithmically generated
- $P(t \mid F)$ is the probability that the token appears in an algorithmically generated domain name
- $P(\neg F)$ is the overall probability that a domain name is legitimate
- $P(t \mid \neg F)$ is the probability that the token appears in a legitimate domain name

token is known as the posterior probability. The overall likelihood is calculated over the posterior probability and if the result exceeds a threshold, the domain will be classified as algorithmically generated. The formula for calculating the likelihood ratio using the Bayesian Classifier is formally defined in Algorithm 3.3.

The classic Bayesian classifier treats all events as dependent on previous events, this makes it ideal for analysing tokens in natural language text. This assumption of dependence presents a problem in the analysis of domain names, as these names may consist of multiple English words combined together creating unusual character combinations. An example of this is 'dailymotion.com' where the combination of daily and motion results in the character combination *ym* that has a low probability of occurring in standard English text. As a result, a second Bayesian classifier was created, where each token is treated as an independent event and is discussed in subsection 3.3.4.

3.3.4 Naïve Bayesian Classifier

The naïve Bayesian classifier assumes that all events are independent, where the presence of any one token is not affected by the presence of any other token. The assumption of independence has been shown to result in highly trainable classifiers that only require a small training dataset (Sahami et al., 1998; Seewald, 2007). Due to the independence of tokens being assumed, only the variance of each token is required and not the covariance matrix of how variables are related. The construction of the naïve Bayesian classifier assumes tokens are randomly distributed in the domain name and that tokens are not

Algorithm 3.4 Naïve Bayesian Classifier

$$ln\frac{P(F \mid D)}{P(\neg F \mid D)} = ln\frac{P(F)}{P(\neg F)} + \sum_i ln\frac{P(t_i \mid F)}{P(t_i \mid \neg F)}$$

Where:

- $ln\frac{P(F|D)}{P(\neg F|D)}$ is the logarithmic probability ratio that a domain name is algorithmically generated ($P(F \mid D)$) or legitimate ($P(\neg F \mid D)$)
- $P(F)$ is the overall probability that a domain name is algorithmically generated
- $P(\neg F)$ is the overall probability that a domain is legitimate
- $P(t_i \mid F)$ is the probability that the token appears in an algorithmically generated domain name
- $P(t_i \mid \neg F)$ is the probability that the token appears in a legitimate domain name

dependent on the length of the domain name, position within the domain name with relation to other tokens, or other domain name contexts. This ensures that training datasets can be kept small, making training easier and faster. Each token is taken as a categorical attribute where the conditional probability for $P('a' = Yes|No)$ is equal to the probability that the letter a is present in domain name.

The formula described in Algorithm 3.4 produces a log-likelihood ratio. This ratio described the relationship $P(F|D) > P(\neg F|D)$ where $ln\frac{P(F|D)}{P(\neg F|D)} > 0$ indicates a legitimate domain name. The resultant classifier is robust and insensitive to isolated points of noise as these are averaged out. Furthermore, the classifier is not significantly effected by irrelevant attributes as each token is considered to be independent (Seewald, 2007).

3.4 Domain Name Query Features Classifiers

The contents of a DNS query response were extracted and used to create classifiers for DNS Fast-Flux domain detection. As discussed in subsection 2.2.2 all DNS query responses are expected to return certain values such as the Answer section and Authoritative section. The values for these sections were analysed for known legitimate domains and known Fast-Flux domains to identify attributes that are shared between these domains, while also identifying attributes that are unique to either domain class.

3.4.1 Modified Holz Classifier

A heuristic classifier classifies a record based on the record meeting predefined heuristics or characteristics. The work of Holz et al. (2008) identified key DNS query response features that could be used in the identification of Fast-Flux domains. Their observations noted the same patterns as identified earlier in subsection 2.2.2. They noted that Fast-Flux domains displayed the characteristics:

- Numerous A records.
- Multiple network ranges.
- A low TTL.

These observations were combined with observations made while examining known Fast-Flux and legitimate domains to construct a heuristic classifier.

Algorithm 3.5 Modified Holz Heuristic Classifier

$$fs = (1.32 * a_{count} + 18.54 * asn_{count} + ttl_{score} * 5) - 50$$

Where:

- a_{count} is the number of A records in the DNS query
- asn_{count} is the number of unique ASN linked to the A records
- ttl_{score} is 0 if the lowest TTL returned was greater than 300, otherwise 1

The original Holz classifier was proposed in 2008 and, after initial testing, it was noted that the weights proposed by Holz et al. (2008) had to be modified to match current trends in Fast-Flux botnets. The most significant change made to the Holz classifier was the introduction of a score associated with the domain's TTL. It was noted the botnets observed had higher TTLs (mean 595) and returned more A records per DNS query. This was in contrast to the botnets observed by Holz, where Fast-Flux domains had low TTLs (0 and 2), returning a single A record with each query.

A modified Holz classifier is proposed, where a weighted TTL is used in the calculation of the Fast-Flux score along with a new fixed constant value of 50. The multipliers used in the heuristic classifier were derived from the values calculated by Holz et al. (2008) (1.32*a_{count} and 18.54*asn_{count}) along with a custom multiplier for the TTL derived from

observations of known Fast-Flux domains. This helped eliminate the need for additional DNS queries once the TTL of the first DNS query expired introducing an unwanted delay before classification could be performed. The modified classifier is formally defined in Algorithm 3.5. The score represented by fs, was indicative of a Fast-Flux domain if greater than zero.

3.4.2 Rule-based Classifier

The rule-based classifier was constructed from observations of Fast-Flux domains active at the time and based on the performance of the modified Holz classifier when a variation from the norm occurred in domain query attributes. It was noted that the modified Holz classifier did not take into account the number of different countries IP addresses were from. This was incorporated into the rule-based classifier to add a further unique identifier of Fast-Flux domains. It had been noted that legitimate domains tended to be hosted in a single country while Fast-Flux domains were hosted in multiple countries. This is discussed further in section 3.5. The effect of large numbers of A records was also taken into account, with the total number of A records returned receiving a lower weighting than is given in the modified Holz classifier. The rule-based classifier is described algorithmically in Algorithm 3.6. The classification of a domain was broken down into two steps, with Equation 3.1 used to calculate the Fast-Flux score. This score was used in Equation 3.2 to determine if a domain is Fast-Flux, where a value of ten or greater for Y indicated a Fast-Flux domain.

3.4.3 Naïve Bayesian Classifier

The Naïve Bayesian classifier, developed to classify whether domains are Fast-Flux, differs from the classifier discussed in subsection 3.3.4. The properties of the DNS query response used in the classifier are treated as continuous attributes, where the attributes have a numerical value. This property required the calculation of the Naïve Bayes to be altered to work with continuous attributes and not categorical attributes. The formula for calculating the Naïve Bayes for continuous attributes is shown in Algorithm 3.7. The classifier is constructed on the assumption that a probability distribution for the continuous attribute exists from the training data.

The calculated continuous probability for each DNS query feature is then used to calculate the total probability that a domain is either Fast-Flux or legitimate. The calculation

Algorithm 3.6 Rule-based Fast-Flux Classifier

$$Y = (0.1 * a + 1.5 * b + 1.5 * c + d + 2 * e) \tag{3.1}$$

$$Y(x) = \begin{cases} Y \geq 10 & \text{if x is Fast-Flux domain} \\ Y < 10 & \text{if x is Benign domain} \end{cases} \tag{3.2}$$

Where:

- Y is the rule-based score
- $[Y \geq 10]$ indicates a Fast-Flux domain
- a is the number of A records in the DNS query
- b is the number of different IP ranges
- c is the number of unique ASNs
- d is the TTL score, where a TTL < 300 is 1 otherwise 0
- e is the number of different countries

of the overall probability is performed using the formulas described in Equation 3.3 for legitimate domains and Equation 3.4 for Fast-Flux domains. The probabilities derived in these calculations are compared and the domain is classified according to the largest probability.

The classifier is trained using known legitimate and Fast-Flux domains with the sample mean ($\bar{x}$) and sample variance (s^2) calculated for each attribute. Using the Naïve Bayes formula as a classifier creates a more robust classifier than simple rule-based and heuristic based classifiers. This is due to any isolated points of noise being averaged out when estimating conditional probabilities during the training phase. Furthermore, the probability distribution obtained during the training phase can be used to determine the ideal decision boundary for each classification feature. These decision boundaries can then be used in constructing rule-based classifiers specific to the observed data.

3.5 Geographic Features Classifiers

The Domain Name System is used to resolve one or more network addresses to a central domain name. Each of these network addresses can be mapped back to a physical geographic location (Padmanabhan and Subramanian, 2001). Domain name lookups allow

Algorithm 3.7 Naïve Bayes Probability of Continuous Attributes.

$$P(X_i = x_i|Y = y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}}} \exp^{-\frac{(x_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

Where:

- X_i is an attribute such as TTL or number of A records
- x_i is the value of the attribute in the domain record being examined
- y_i is the class being tested: Fast-Flux or legitimate
- $\mu_{ij}$is the sample mean ($\bar{x}$)
- σ_{ij}^2 is the sample variance (s^2)

Algorithm 3.8 Naïve Bayesian Fast-Flux Classifier

$$p(D \mid S) = \prod_i p(w_i \mid S) \tag{3.3}$$

and

$$p(D \mid \neg S) = \prod_i p(w_i \mid \neg S) \tag{3.4}$$

Where:

- Equation 3.3 is used to calculate the probability that a domain is legitimate $p(D \mid S)$
- Equation 3.4 is used to calculate the probability that a domain is Fast-Flux $p(D \mid \neg S)$
- w_i is the domain feature being examined (Number of A Records, TTL, ect.)

infected hosts in the botnet to look up the address of C2 servers from which they need to receive instructions. Nazario and Holz (2008) and Holz et al. (2008) noted that hosts used as C2 servers for a botnet need to meet specific criteria. These include a globally accessible, globally unique IP address (Nazario and Holz, 2008). In further work Holz et al. (2008) identified the inherent distributed structure of botnets as a distinguishing factor. To contrast, legitimate domains tend to be set up with geographic location in mind, with all servers for the domain hosted in a central location, such as a data-centre. It is hypothesised that the principles behind animal population statistics and distribution modelling can be applied to the geographic distribution patterns of Fast-Flux botnets. Using data collected in datasets FD1, FD3 and FD4, the mean nearest neighbour distances were calculated in earlier work Stalmans, Hunter, and Irwin (2012). These nearest

Table 3.4: Geographic Data for a Fast-Flux Domain (*cjjasjjikooppfkja.ru*)

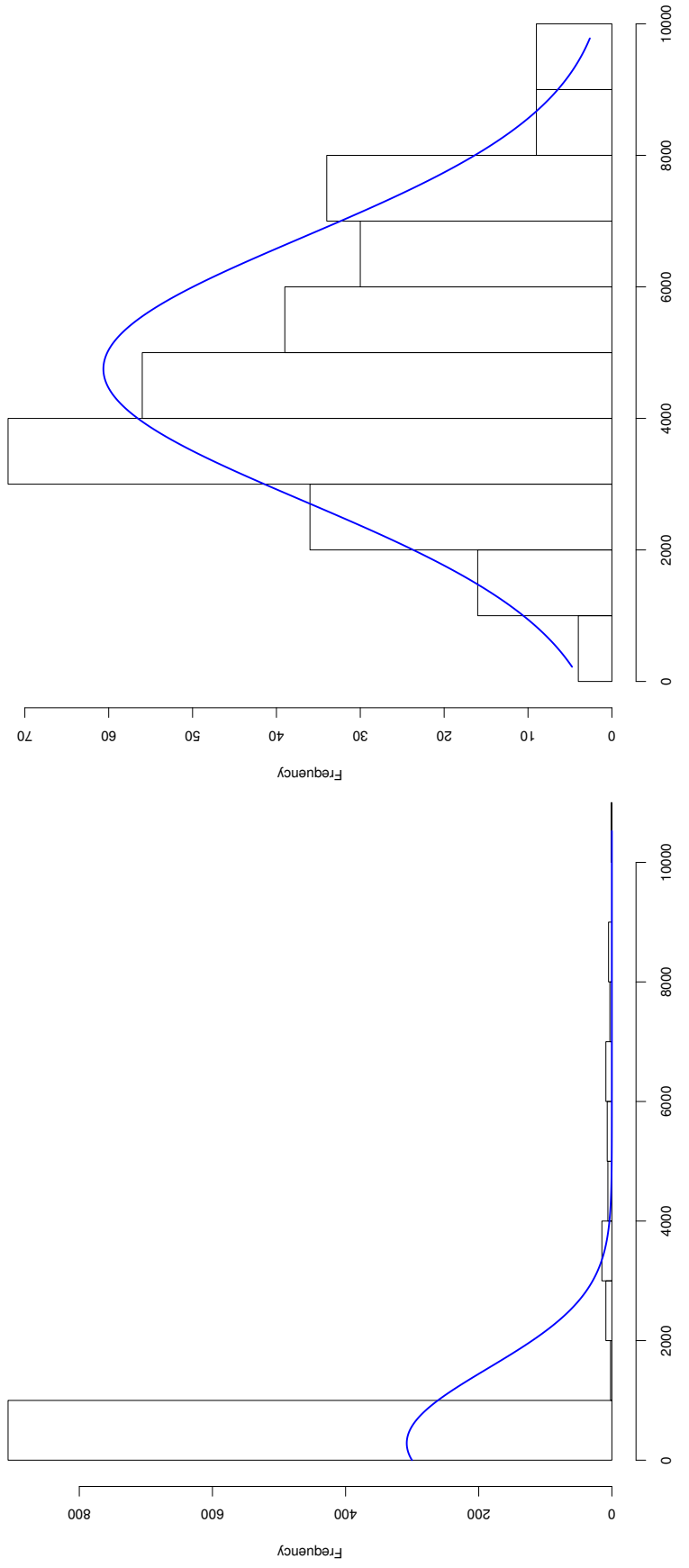
IP Address	Latitude:Longitude	UTM	MGRS
79.108.149.71	38.25:-0.7	37M	30SYH0125936055
79.139.110.20	49.7833:22.7833	39Q	34UFA2837416063
31.45.148.102	38.0:-97.0	37Z	14SPH7560307702
88.132.63.164	47.0333:19.7833	38Q	34TDT0755809583
124.6.3.225	22.6333:120.35	34Z	51QTF2762705352
89.229.214.126	53.7333:18.9167	39Q	34UCE6257855864

neighbour distances are shown in Figure 3.3 where it clearly be seen that legitimate domain servers (Figure 3.3a) tend to be closer together, with the majority of servers being in the same location, while the C2 servers of Fast-Flux domains (Figure 3.3b) tend to be far apart with a binomial distribution centred around a mean nearest neighbour distance of 5000km (Stalmans et al., 2012).

Figure 3.4 maps the locations of C2 servers for the domain 'cjjasjjikooppfkja.ru' which was listed as a Fast-Flux domain hosting C2 servers for the ZeuS botnet on 14 March 2012 (abuse.ch, 2012). It is noticeable that the C2 servers are widely dispersed globally, in the case of ZeuS, they were distributed across 11 different servers in 11 different countries across three continents for a single Fast-Flux domain. In contrast a legitimate domain such as 'google.com' has all six servers returned by a DNS query result located in one central location. The timezone in which a server is located is also of use as noted by (Huang et al., 2010). The geographic distribution can be further analysed on a finer grained level than timezone using co-ordinate systems such as the Universal Transverse Mercator system (UTM) and the Military Grid Reference System (MGRS). Table 3.4 provides the translation of IP addresses to geographic locations for a known Fast-Flux domain using the three different co-ordinate systems. These co-ordinate systems and how they were adapted to provide a numerical value for the use in the classifiers are described in greater detail in section 3.5.2 and section 3.5.2.

3.5.1 GeoIP Database

MaxMind and other organisations have developed IP Intelligence databases that contains geographic information for IP addresses throughout the world (MaxMind, 2012). MaxMind states that their database provides information for 3,467,581,993 IP addresses, mapping to 250 countries. The GeoIP City database, used by this research, allows for the country, city, latitude, longitude and other information pertaining to an IP address to



(a) Histogram for Legitimate Domains

(b) Histogram for Fast-Flux Domains

Figure 3.3: Histograms of Mean Nearest Neighbour Distances for Legitimate Domains and Fast-Flux

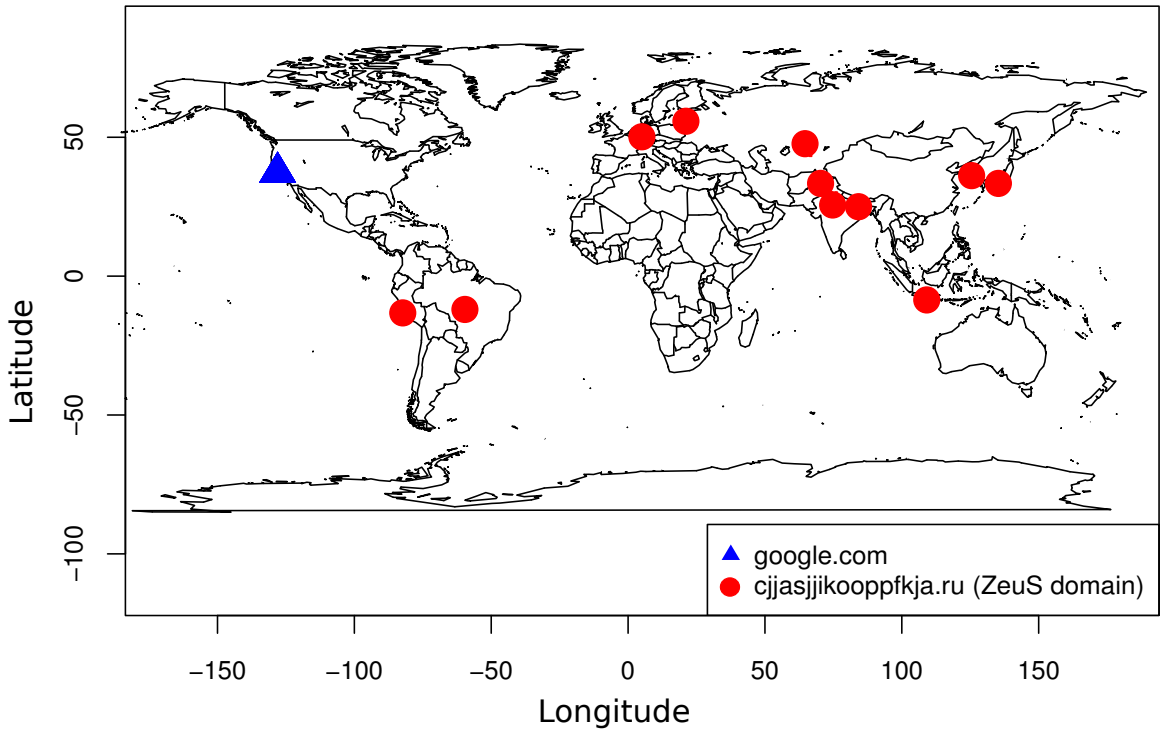


Figure 3.4: Geographic Distribution of Hosts for a Botnet and a Legitimate Domain

be retrieved. The information contained within the MaxMind database is collected using Open Source intelligence where members of the community submit the data to be used in the database. This community submitted data is then verified and augmented by other community members, much in the same way as articles on wikipedia.com are submitted, verified and added. This system allows large amounts of data to be collected and verified. Alternative solutions investigated were hostip.info which operated on a similar model to MaxMind where community submissions were used to build up a database of IP to geolocations (hostip.info, 2012). Commercial options such as wipmania.com[1], ip2location.com[2] and ipligenceMax[3] were all investigated and found to be too limited in either the data available or the number of queries which could be performed in the free versions. Thus numerous alternatives exist, the MaxMind database was found to contain the most accurate data while still being free to use. Furthermore, the MaxMind database allowed for offline lookup of geolocations, eliminating the need for queries to external services that may introduce a classification delay or may not be available when required.

[1]http://www.wipmania.com/en/
[2]http://www.ip2location.com/
[3]http://www.ipligence.com/products#max

Table 3.5: Input Values for the IP Address 59.146.177.153

Method	Representive Value	Input value
Latitude:Longitude	35.685^{o}:139.7514^{o}	-
Timezone	Asia/Tokyo (GMT+10)	1000
UTM	36Z	3240
MGRS	54SUE8701849729	111366272831742

3.5.2 Geographic Value

The geographic locations for each server needed to be assigned a numerical value to be used in classifier calculations. These values were obtained using three different means as outlined in the following sections. Table 3.5 shows the numerical values as calculated from a servers timezone, UTM grid location and MGRS grid location (Stalmans et al., 2012).

Timezone

The MaxMind database used for determining the geographic location of an IP address contains the timezone in which the IP address is located. The timezone provides an easily convertible numeric input value which uses a similar technique to that used for calculating time. The Greenwich Meridian Time (GMT) was used as a base value of zero, with each timezone getting assigned a positive value based on it's distance from GMT. This calculation was trivial and was performed using the following conversion: GMT+1 was assigned the value of 100, GMT+2 the value 200 and so forth. This was repeated for the timezones GMT-(1...n) where the value was converted to a positive value to be used as input to the classifiers. Calculating the input value associated with a timezone is formally defined in Algorithm 3.9.

Algorithm 3.9 Timezone Value Calculation

$$v = 0+ \mid 100 * n \mid$$

Where:

- GMT is indicated by the value 0
- n is numerical value indicating the timezone (GMT+1)
- v is a positive value to be used in the classifiers

Universal Transverse Mercator Coordinate System

The Universal Transverse Mercator (UTM) coordinate system is an alternative to the standard latitude and longitude coordinate system. Developed for use by the United States Military, UTM is based on an ellipsoidal grid model of the Earth (U.S. Geological Survey, 2012). The UTM system allows the earth to be divided into sixty distinct zones, each zone representing a six-degree band of longitude. The zone is identified by a numeric value followed by an alphabetic character value known as the grid designator. This accounts for a total of 36 grid designations. For UTM to be used in the classifier calculations the UTM value needed to be converted to a fully numeric value. This was achieved by multiplying the numeric grid designator with the ordinal value of the alphabetic grid designator. A sample value produced using the UTM system is shown in Table 3.5.

Military Grid Reference System

The Military Grid Reference System (MGRS) was developed to standardise geo-co-ordinatation between NATO militaries and is based on the UTM grid system and the similar Universal Polar Stereographic grid system (Hostert, 1997). The benefit of MGRS is that it allows fine grained grid designation of a geographic point down to one square meter. A MGRS grid point is identified by a grid zone designation, followed by a 100000-meter square identification (Hostert, 1997). For example: using MGRS, the latitude (26.12°) and longitude (28.04°) for Johannesburg, South Africa can be represented as 35RPJ3997589726, where 35R is the grid zone designation, PJ the 100000-meter square identifier and 3997589726 is the numerical location within the grid. The MGRS value provides a grid location for each C2 server which needs to be converted to a numeric value before it can be used in a classifier. The calculation can be performed using the formula shown in Algorithm 3.10. A result of this calculation can be seen in Table 3.5.

Algorithm 3.10 The Military Grid Reference System Numeric Value

$$m = V1 * V2$$

Where:

- $V1 = v_1 * (v_2 + v_3 + v_4)$ where v_1 is the numerical value and $(v_2 + v_3 + v_4)$ are the ordinal values of the characters
- $V2$ is the integer of the numerical location

3.6 Spatial Autocorrelation

Spatial autocorrelation is the process of correlating values of a single variable, strictly according to the proximity of those values in geographic space. Traditionally autocorrelation has been used in areas such as signal processing, where correlation is done between values in serial. Observations of a variable are arranged according to a measure of ordering such as time. Spatial statistics can be characterised by the fact that it violates the statistical assumption of independence. Patterns produced in space result from spatial patterns, where the value is one of numerous possibilities from the same spatial process.

In statistics autocorrelation refers to the process of finding the correlation between points of a random process at different points in time. Autocorrelation is achieved by cross-correlating a signal with itself, effectively removing noise and revealing any obscured patterns hidden in the signal. Correlation is used to measure the dependence or statistical relationship between any two points in a distribution. This correlation can refer to any characteristic that the points share such as geographic location, value or dependence on other points. The benefits offered by autocorrelation have led to its use in different fields of study such as signal processing, astrophysics and music recording. While autocorrelation measures the dependence of points in one dimension, time, spatial autocorrelation was developed to measure the dependence of points in two-dimensional space. Spatial autocorrelation allows for the correlation of points in time and space, along with multi-directional points. Spatial autocorrelation has mainly been used to measure the spatial dependence of locations within a geographic area. This measure of dependence is based on the *First Law of Geography* which states, 'Everything is related to everything else, but near things are more related than distant things' (Tobler, 1970). Spatial autocorrelation has largely been used in animal population statistics and disease modelling to find common features that relate dispersed populations to each other (Schabenberger and Gotway, 2004).

The results of spatial autocorrelation can either be positive, negative or no correlation. Positive spatial autocorrelation indicates that geographically nearby values tend to be similar. Similarity will show high values located near other high values while low values are located near other low values. As noted by Lea and Giffith (2001) demographic and socioeconomic characteristics such as population density and household income will likely exhibit positive spatial autocorrelation. It is hypothesised by this research that Internet infrastructure will exhibit positive spatial autocorrelation, based on variables such as network speed and available bandwidth. Negative spatial autocorrelation indicates the

inverse of positive spatial autocorrelation, where nearby values will be dissimilar to other nearby values.

3.6.1 Moran's Index

Moran's Index (MI) provides a test for spatial autocorrelation in a set of continuous data (Moran, 1950). The MI is a weighted correlation coefficient using the distance between dispersed points as weights. The MI is based on the observation that points spatially closer together are more likely to be similar than points far apart (Cliff and Ord, 1973). Moran's coefficient is calculated using the formula:

Algorithm 3.11 Moran's Index

$$I = \frac{N}{\sum_i \sum_j w_{ij}} \frac{\sum_i \sum_j w_{ij}(X_i - \bar{X})(X_j - \bar{X})}{\sum_i (X_i - \bar{X})^2}$$

Where:

- I is the Moran Index
- N is the number of locations returned by the DNS query
- $X_n$is the value of the n^{th} variable of interest (timezone value, UTM value, MGRS value)
- $\bar{X}$ is the average of all values of N
- w_{ij} is the weight (distance) between two spatial points i and j

The weight (w_{ij}) between each C2 server location was calculated using the Haversine formula (Robusto, 1957) and a matrix of weights was constructed. Due to the large distances between hosts on a global scale, the inverse weights ($\frac{1}{w_{ij}}$) were used. Output values for the MI range from -1 to 1 where negative spatial correlation is indicated by values less than zero, with -1.0 indicating perfect negative spatial autocorrelation. Positive spatial correlation is indicated by values greater than zero, where 1.0 indicates perfect positive spatial correlation. An index value of zero represents a perfectly random spatial pattern. Values outside the range -1 to +1 indicate spatial autocorrelation that is significant at the 5% level.

The MI allows for the measuring of global spatial autocorrelation and is not severely influenced by large amounts of whitespace, making it ideal for using in the classifying of Fast-Flux C2 server distribution, where large distances exists between the servers.

Inversely, the global nature of MI decreases it's effectiveness for measuring localised spatial correlation.

3.6.2 Geary's Coefficient

Similarly to Moran's coefficient, Geary's coefficient (GC) is used to measure spatial autocorrelation. The value of GC lies in the range [0-2]. Values between 0.0 and 1.0 indicates positive spatial autocorrelation while values between 1.0 and 2.0 indicate negative spatial autocorrelation. A value of 1.0 for GC indicates no spatial autocorrelation. The GC is calculated using the formula:

Algorithm 3.12 Geary's Coefficient

$$C = \frac{(N-1)\sum_i \sum_j w_{ij}(X_i - X_j)^2}{2W\sum_i (X_i - \bar{X})^2}$$

Where:

- C is the Geary Coefficient
- N is the number of locations returned by the DNS query
- X_n is the value of the n^{th} variable of interest (timezone value, UTM value, MGRS value)
- $\bar{X}$ is the average of all values of N
- w_{ij} is the weight (distance) between two spatial points i and j
- W is the sum of all w_{ij}

The GC tends to be more sensitive to localised correlation and has been shown to be a good indicator of differences in small neighbourhoods. The GC is influenced far more than the MI by skewed distribution of numbers of neighbours and by outliers. The Moran's Index and Geary's Coefficient tend to give similar results and thus may be used in a two factor classifier. The MI and GC are negatively related.

3.7 Summary

This chapter presented multiple techniques to be used to detect botnet traffic from DNS query responses. The techniques described use features such as the domain name, the

A records, the TTL and NS records extracted from the DNS query response as inputs. The first set of techniques presented make use of lexical analysis to determine whether the domain name being queried was algorithmically generated. Four techniques were identified for classifying domain names. Each technique made use of Bayesian statistics to train a classifier from sample data.

The second set of techniques described are aimed at identifying botnet C2 domains, specifically domains using Fast-Flux as a avoidance technique. Three techniques were described, resulting in a classifier based on modifying previous research into Fast-Flux domain detection using DNS. A new rule-based classifier was described next, which takes into account the geographic dispersion of servers. Finally a novel Naïve Bayesian classifier was described.

Spatial autocorrelation was described as a technique for detecting Fast-Flux domains based on the geographic distribution of domain hosts. Two statistical methods known as Moran's Index and Geary's Coefficient were described, along with different co-ordinate systems used to quantify the geographic location of a server.

The techniques presented in this chapter were applied to the datasets outlined in section 3.1 to measure the feasibility of these techniques as first pass classifiers. The results are presented in the chapter 4, with the results presented in separate subsections for each detection type. chapter 4 extends the observations made in this chapter and evaluates the overall accuracy of the proposed techniques.

4 Results

This chapter presents the results obtained from evaluating the classification techniques outlined in chapter 3. Each classifier was tested using known legitimate and malicious data which had been manually verified and labelled. This process is discussed in section 4.1 along with the means used to measure the performance of each classifier. The results for the detection of algorithmically generated domain names are presented in section 4.2. The results for the detection of Fast-Flux domains from DNS query response attributes are provided in section 4.3. Finally the results for the spatial autocorrelation classifiers are presented in section 4.4.

4.1 Data Metrics

Multiple data sources were used in testing the classifiers and thus data needed to be standardised and sanitised through reformatting and the removal of duplicate values. Once all the data had been standardised each entry was labelled to allow for the correct training of classifiers. The labelling of data is discussed in detail in subsection 4.1.1. The results obtained from testing the classifiers had to be rated to allow for measurement and comparison the performance of the different classifiers. The criteria used for measuring classifier performance is outlined in subsection 4.1.2.

4.1.1 Labelling Data

The data used in this thesis was divided into two sets, a training set and a test set. The training set of data was used in creating the classifiers that will be used in determining

Algorithm 4.1 Classifier Performance Calculations

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{4.1}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{4.2}$$

$$ACC = \frac{TP + TN}{P + N} \tag{4.3}$$

Where:

- P is the total number of positive samples
- N is the total number of negative samples
- TN is the total number of samples correctly been identified as negative
- TP is the total number of samples correctly identified as positive
- FN is the total number of samples incorrectly identified as negative
- FP is the total number of samples incorrectly identified as positive

whether domains are legitimate or belong to a botnet. It was essential that the validation of the classifier's performance was done on a different dataset to the dataset used to train the classifiers. This was done to avoid over-training the classifiers, as well as to avoid skewing the performance of the classifiers due to bias developing towards the training data. Data in these datasets had to be manually labeled to facilitate training and, later, performance measurement. The labelling of data for the training set was required as classifiers had to learn to classify each type of domain, legitimate or malicious. Labeled test cases were needed for the evaluation of classifier performance to allow the researcher to evaluate whether a test case had been correctly classified. The classification problem was a binary problem and thus values were labeled with either 0 or 1 to indicate the class to which they belong. Data from legitimate domains was labeled with a 1, while malicious domains were labeled as 0.

4.1.2 Results Evaluation

Measuring the performance of each classifier and being able to compare the performance of multiple classifiers in a fair and effective manner is essential to this research. Each classifier produces a binary result of either true (the domain is malicious) or false (the

domain is safe). Using known data, it is possible to calculate the accuracy of the classifier. From the known data the total number of Positive (P) and Negative (N) domains are calculated, where Positive domains are the domains which have been manually classified as malicious. Furthermore, incorrectly classifying benign domains as malicious is deemed more costly than classifying malicious domains as legitimate, as this may have a greater impact on the overall user experience. Incorrectly classifying benign domains as malicious is labeled a False Positive (FP). Correctly classifying a malicious domain as malicious is labeled as a True Positive (TP) and a correct classification of a benign domain as a True Negative (TN). Incorrectly classifying a malicious domain as benign gets labeled as a False Negative (FP). Through the labelling of classifications it is possible to determine the rate at which domains are being correctly classified. The True Positive Rate (TPR) defines how many correct positive classifications occur among all positive samples, while the False Positive Rate (FPR) defines how many incorrect classifications occur among all the negative samples. These rates can be calculated as shown in Algorithm 4.1, with the calculation of the TPR shown in Equation 4.1 and the FPR calculation in Equation 4.2.

The Accuracy (ACC) is used to measure the number of TP versus the number of TN, allowing for classifier performance measurements and comparisons in terms of correctly classifying each type of domain. The accuracy will be higher the more accurately a classifier classifies domains, where a high number of TP and TN is considered desirable. The accuracy is calculated as shown in Equation 4.3.

The overall performance of a classifier is measured using the Area Under the Curve (AUC) of the Receiver Operator Characteristic curve (ROC) and is also known as the ROC space (Schabenberger and Gotway, 2004). The ROC provides a graphical representation of the performance of a binary classifier, plotting the TPR versus the FPR. The AUC is calculated to provide a measure of the trade-off between the TPR and the FPR. The optimal classifier will have an AUC approaching 1, where a high TPR and low FPR is desired.

4.2 Algorithmically Generated Domain Name Detection

Classifiers were developed to identify domain names that have been algorithmically generated. These domain names could be indicative of malware presence on the network. A

Table 4.1: Results of the Unigram Classifiers

Classifier	Accuracy (%)	TPR (%)	FPR (%)
Total Variation	82	80	17
Probability	84	86	17
Bayesian	85	81	11
Naive Bayesian	87	82	8
Combined	89	89	11

well-documented technique used by malware authors for evading detection and increasing the longevity of malicious domains is algorithmically generating domain names. The classifiers employed use multiple statistical methods for determining the likelihood that a domain name is either algorithmically generated or legitimate. The classifiers were trained using frequency analysis of character distributions in known algorithmically generated domain names, such as those employed by the Conficker and Torpig malware. They were also trained using known legitimate domain names. The likelihoods were compared and domains were classified according to the higher likelihood. In the case of the Naïve Bayesian classifier the output was a likelihood ratio. This ratio was examined and a decision boundary was determined and used for classifying domain names based on the values observed during first pass training of the classifiers. Results showed that likelihood ratio classifiers based on the Bayes formula as discussed in subsection 3.3.3, resulted in the highest accuracy rates, with the high true positive rates (TPR) and low false positive rates (FPR).

The classification of domains using single character frequencies (unigrams) and paired character frequencies (bigrams) were examined separately. These are discussed in detail in the following sections.

4.2.1 Unigram Classifiers

The unigram based classifiers examined domain names based on the frequency distribution of single alpha-numeric characters. The summary of the results of the classifiers using unigrams are shown in Table 4.1, these results were obtained on a test dataset of 7 000 legitimate domain names and 7 000 algorithmically generated domain names, taken from datasets AD2 and AD3 respectively, which were described in subsection 3.1.1. It was noted that the accuracy rates of the tested classifiers were all similar, with the Naive Bayesian classifier producing the best accuracy rate, 5% higher than the worst performing

classifier, the total variation distance classifier. The FPR of the Naive Bayesian classifier was the lowest at 8%, significantly lower than the 17% FPR of the total variation distance and probability classifiers. The results for each classifier are examined in more detail in the following subsections.

Total Variation Distance Classifier

The total variation distances for 5 000 algorithmically generated domain names and 5 000 legitimate domain names are shown in Figure 4.1. It was noted that the total variation distance for legitimate domains, shown in blue, increased as the length of the domain name increased, with a clear upward trend towards a variation distance of 0.2 developing. Algorithmically generated domain names, shown in purple, displayed a trend line fluctuating around zero, with a downward slope towards the end as domain name length increased beyond 15. The available datasets of algorithmically generated domain names did not contain any entries with lengths less than 4, while multiple legitimate domain names of lengths 1 to 3 were present. It was noted that there was a larger overlap of variation distance for domain names with short domain name lengths than domain names with longer lengths. The overall accuracy of the total variation classifier was 82% with a TPR of 80% and a high FPR of 17% as noted in Table 4.1. Only classifying domains of length 6 or greater increased the TPR to 95% and decreased the FPR to 13%.

Probability Classifier

The output of the probability classifier showed similar trends to those seen in the total variation distance classifier section 4.2.1, with domain names of longer length showing greater variation between the probability outputs for algorithmically generated domain names and legitimate domain names. It was noted that domain names such as *xnxn.com* and *fbcdn*.com with few or no vowels were classified as algorithmically generated although they actually belonged to legitimate domains found in the Alexa Top 10000 list. The accuracy of the probability classifier was higher than that of the total variation classifier at 84% as result of the higher TPR of 86%. The FPR for both classifiers were the same at 17%. As seen with the total variation classifier, adjusting the classifier to only examine domains of length 6 or greater increased the TPR to 89% and dramatically decreased the FPR to 8%.

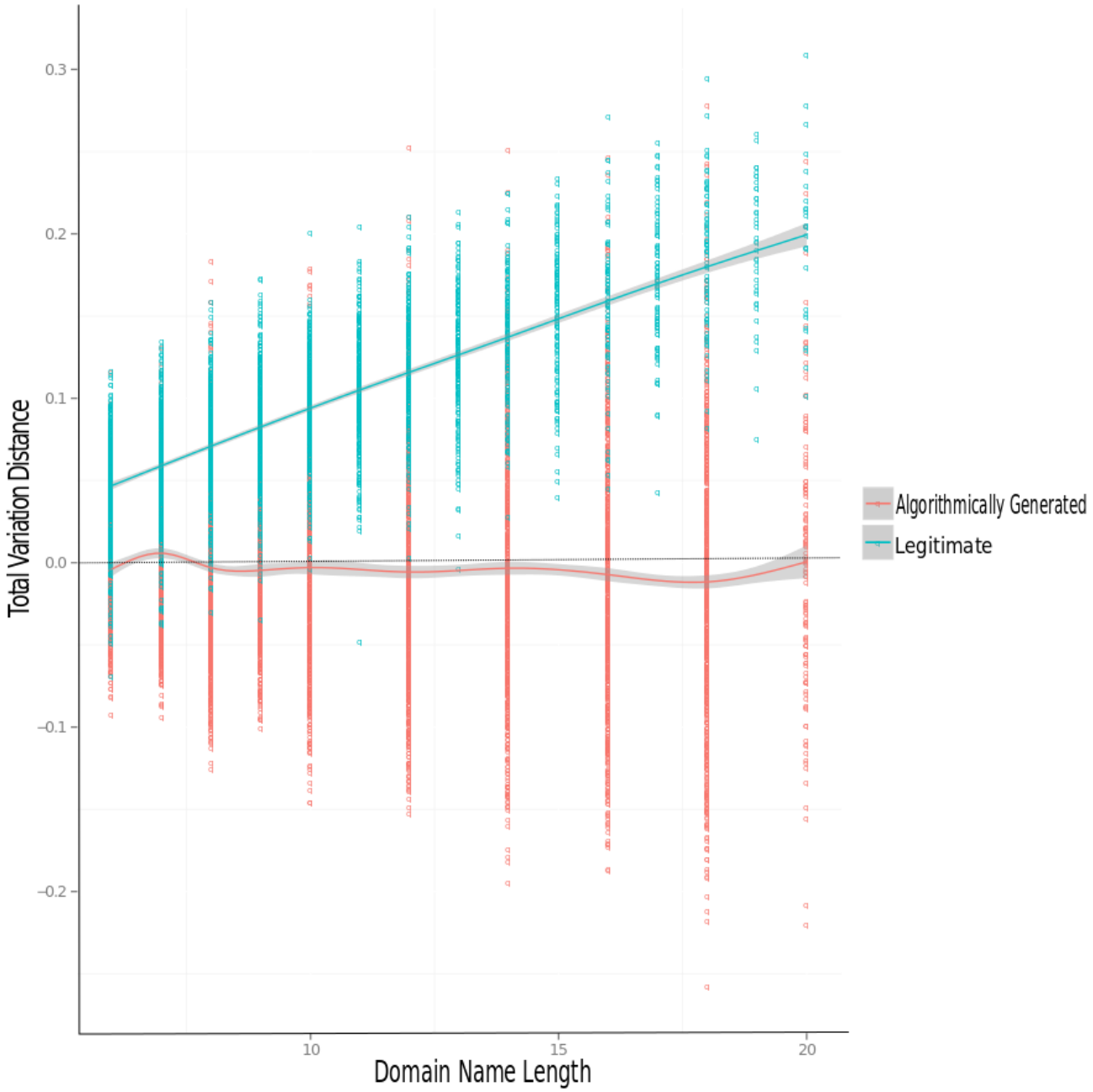


Figure 4.1: Total Variation Classifier Output for Unigrams

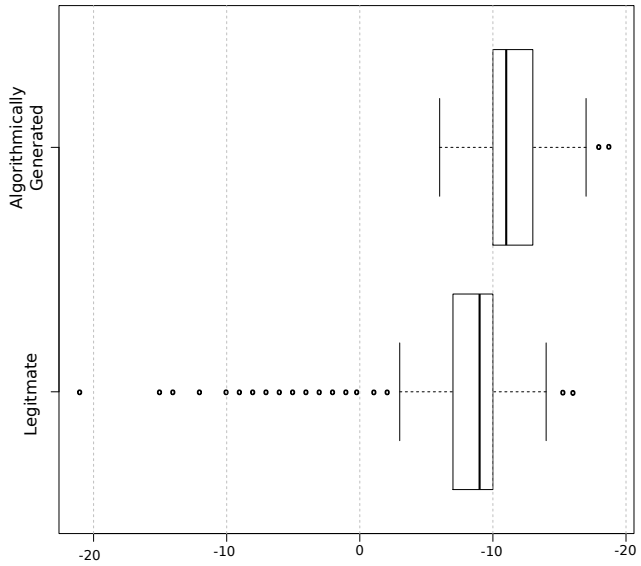


Figure 4.2: Density Distribution of Unigram Naive Bayesian Classifier Output.

Bayesian Classifier

The Bayesian Classifier produced a likelihood ratio which could be used for classifying domain names. The produced likelihood ratios for legitimate and algorithmically generated domains were compared across the training dataset to determine a decision boundary. It was found that the mean likelihood ratio for legitimate domains was 0.263260, while algorithmically generated domain names had a mean likelihood ratio of 0.789548. This resulted in a decision boundary of 0.53, where values greater than 0.53 indicated algorithmically generated domain names. The results showed trends seen in both the total variation distance and probability classifiers where domain names of longer length produced higher likelihood ratios. The Bayesian classifier had an accuracy of 85%, while having a lower TPR (85%) than the probability classifier. The FPR of 11% was significantly lower than both the Total Variation Distance and Probability classifiers. The classifier was adjusted to only examine domains of length 6 or greater and it was found that the TPR increased by 8% to 89% while there was a 1% decrease in the number of false positives, with the FPR dropping to 10%.

Naïve Bayesian Classifier

The likelihood ratio output from the Naïve Bayesian classifier was examined and a decision boundary determined. The box plot in Figure 4.2 shows the distribution of likelihood ratios for legitimate and algorithmically generated domain names. It was noted that a positive likelihood ratio with a median of 1 was produced by the classifier for algorithmically generated names, while the processing of legitimate domain names resulted in a negative likelihood ratio with a median of -1. The minimum value for algorithmically generated domain names fell below the lower quartile of outputs for legitimate domain names, indicating that an overlap of results existed. The heatmap of likelihood values for legitimate domains shown in Figure 4.3 displays a higher density of values greater than the legitimate domain name median of -1 occurring for domain names with a length . It was observed that for the vast majority of values fall below zero, with numerous values much smaller than the median of -1. Furthermore, it was observed that the average value of the output decreased as the domain name length increased. Figure 4.4 shows a heatmap of the output from algorithmically generated domain names, where a more uniform distribution of results can be seen. In contrast to the results seen in Figure 4.3, the output is extensively positive with most values greater than one. Furthermore, it was noted that the number of domains with a value greater than two increases with domain name length. These observations led to the construction of a decision boundary around zero, with values greater than zero indicating algorithmically generated domain names while negative values indicating legitimate domain names. The Naïve Bayesian classifier had an accuracy of 87% with a TPR of 82% and a low FPR of 8%, as seen in Table 4.1. Limiting classification to only domain names of length 6 or greater increased the TPR to 89% and while the FPR stayed the same at 8%.

Combined Unigram Classifier

A feedforward neural network was constructed to create a classifier using the output from the four above-mentioned classifiers as inputs. The network consisted of four inputs, a single hidden layer using a sigmoid function as a combiner and had a single output. A dataset of 10000 classified domain names was constructed used the output from the other classifiers as results and using known legitimate and algorithmically generated domain names. The dataset was divided into a training set of 7500 domain names while the remaining 2500 domain names were used for testing. Once the network had been trained the test domain names were fed into the network and the classifications were recorded. Six

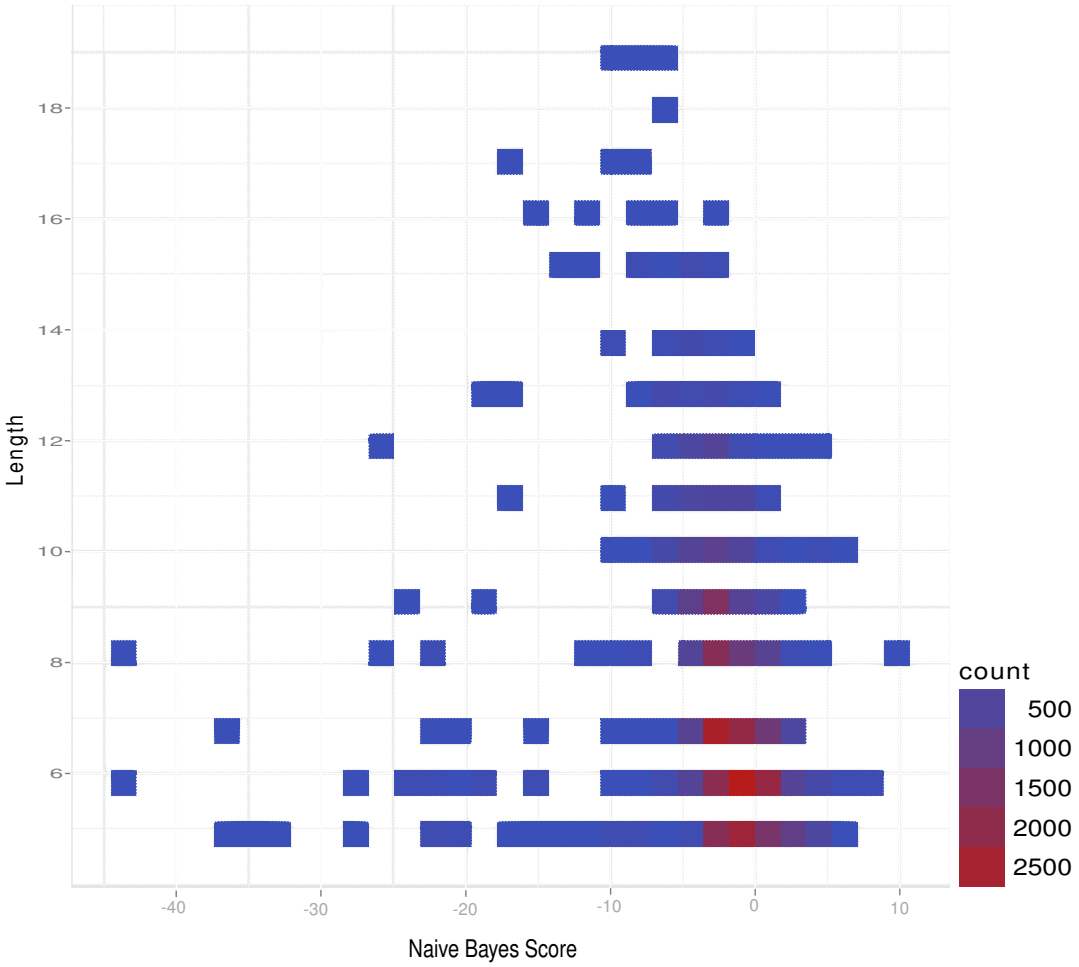


Figure 4.3: Naive Bayes Classifier Output for Legitimate Domains

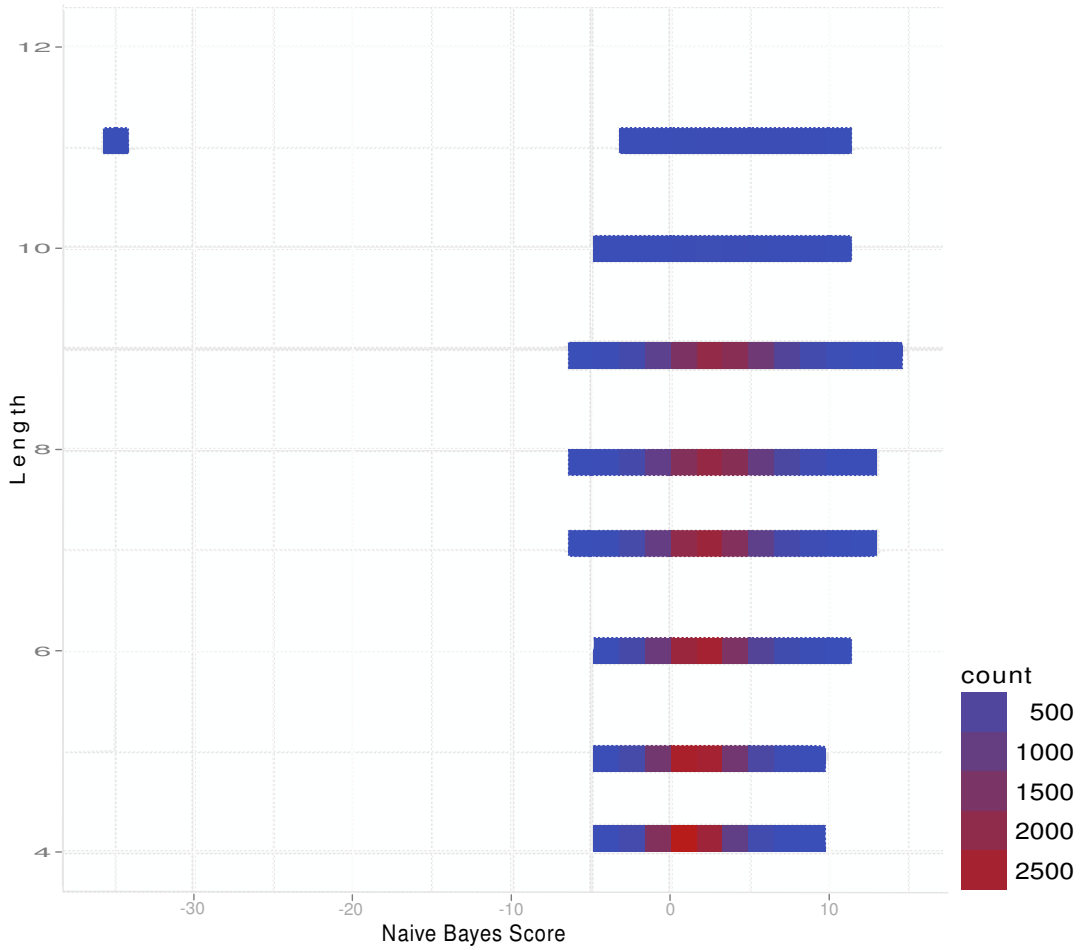


Figure 4.4: Naive Bayes Classifier Output for Algorithmically Generated Domain Names

Table 4.2: Results of the Bigram Classifiers

Classifier	Accuracy(%)	TPR (%)	FPR (%)
Total Variation	79	84	27
Probability	89	91	14
Bayesian	90	88	8
Naive Bayesian	90	91	9
Combined	88	89	12

hundred iterations were performed and the accuracy, TPR and FPR were averaged across these iterations and used as the overall values of the classifier. The resulting accuracy for the classifier was observably better than the best performing stand alone classifier. The accuracy of the classifier for the test dataset was 89% with a TPR of 89% and a FPR of 14%. The neural network was reset and trained with only the domain names in the dataset that had a length greater or equal to six. The test dataset was modified in the same manner and tested with again, with 600 iterations being performed again to calculate the average accuracy, TPR and FPR. The resultant accuracy was 90% with a decrease in the TPR to 89% and an improvement in the FPR to 10%.

4.2.2 Bigram Classifiers

The bigram-based classifiers were used to examine domain names based on the frequency distribution of character pairs of alphanumeric characters. The test dataset consisted of the same 7000 legitimate domain names and 7000 algorithmically generated domain names as previously used with the unigram classifiers. There was an overall improvement in classifier accuracy except for the total variation distance classifier faring 3% worse than the same classifier using unigrams. The largest improvement was seen in the Bayesian classifier with accuracy increasing by 5% and improvements in both the TPR and FPR. Similar results were seen for the Naive Bayesian classifier, with the largest improvement of 9% in the TPR and a minor increase in the FPR of 1%. The results from the classifiers are shown in Table 4.2.

Total Variation Classifier

The bigram-based total variation classifier performed similarly to the unigram based total variation classifier, with a 3% lower accuracy of 79%. There was, however, a marked

increase in the number of false positives with the FPR increasing from 17% to 27%. There was a minimal change in the TPR to 84% (an increase of 4%). The FPR of the total variation classifier decreased to 9% when only domains with a length of 6 or greater were examined, which was 14% lower than the FPR of total variation classifier based on unigram probabilities. Furthermore, the TPR increased to 88% (a total increase of 8% over the TPR of the unigram classifier).

Probability Classifier

There was a noticeable improvement in results for the probability classifier when using bigrams. The classifier's accuracy increased to 89% with improvements in both the TPR (91%) and FPR (14%). The probability classifier had a slight improvement in it's TPR to 92% when domains of length 6 or greater were examined, with a 10% decrease in the number of false positives with a FPR of 4%.

Bayesian Classifier

The Bayesian classifier showed near identical improvements to the probability classifier. The increase in accuracy of 5% was identical to the improvement in accuracy seen by the probability classifier, while both classifiers had a decrease in the FPR of 3%. The Bayesian classifier had a slightly larger increase in it's TPR, from 81% to 88%. Modifying the classifier to only classify domains of length 6 and greater resulted in the classifier accuracy increasing to 94% with only 5% of legitimate domains being incorrectly classified.

NaïveBayesian Classifier

The box plot in Figure 4.5 shows the distribution of likelihood ratios for algorithmically generated and legitimate domain names. It was noted that the values showed similar distributions to those seen for the Naïve Bayesian classifier based on unigram frequency distribution, with legitimate domain values largely being less than zero and algorithmically generated domain names values being positive. The likelihood ratio distribution for legitimate domain names did, however, lie between negative one and negative seven, where the mean for the unigram based classifier was negative one. The mean also shifted lower to negative five. The same decision boundary of zero was used to classify domains. The accuracy of the Naïve Bayesian classifier improved to 90% as shown in Table 4.2.

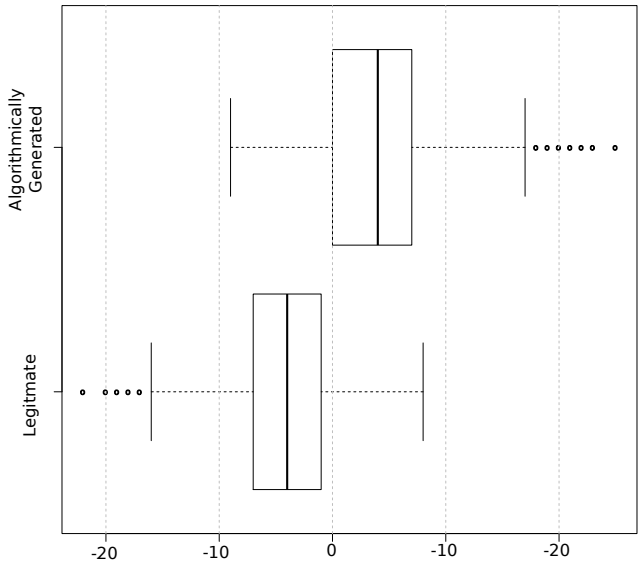


Figure 4.5: Density Distribution of Bigram Naive Bayesian Classifier Output.

There was a large increase of 9% in the TPR to 91% compared to the unigram Naïve Bayesian classifier, while there was a minor increase of 1% in the FPR to 9%. Adjusting the classifier to only examine domains of length 6 or greater had the same effect as with the other classifiers, with the TPR increasing to 92% and the FPR decreasing to 3%.

Combined Bigram Classifier

The same process of constructing a dataset for training and testing was followed as described in section 4.2.1. The testing data was applied to the trained neural network and again 600 iterations were completed with the average of the results across the iterations being recorded. The accuracy from the combined classifier was slightly worse than the best performing stand-alone classifier with an average accuracy of 88% being observed. Both the TPR and FPR were lower than expected with an average TPR of 89% and a relatively high FPR of 12%. The tests were repeated using only domain names of length 6 or greater and the average results across 600 iterations were recorded. The observed accuracy for the combined classifier using only domains of length 6 or greater was 93%, with a higher TPR of 91% and a lower FPR of 5%.

Table 4.3: Accuracy Rates of Lexical Analysing Classifiers

	ACCURACY (%)	ACCURACY (%) (> 6 CHARACTERS)
UNIGRAMS		
Total Variation	82	88
Probability	84	90
Bayesian	85	89
Naive Bayesian	87	90
Combined	89	90
BIGRAMS		
Total Variation	79	89
Probability	89	94
Bayesian	90	94
Naive Bayesian	90	93
Combined	88	93

4.2.3 Results Summary

This section presented the results obtained for evaluating the accuracy of the proposed classifier techniques. The section was divided into two subsections, with the results from evaluating the unigram distributions in domain names presented in subsection 4.2.1. The second subsection 4.2.2, presented the results from evaluating the classifiers using the bigram character distributions. A summary of the classifier accuracies is presented in Table 4.3. It was observed that the use of bigrams produced higher accuracy rates than the use of unigrams, however the largest effect on accuracy was domain name length. Classifying domain names of length six or greater resulted in an average increase in accuracy of 4% in unigram classifiers and 5.4% in bigram classifiers. Classifying only domains of length six or greater saw the largest improvement in classifier accuracy for the total variation distance classifier with a 10% increase. The results from these classifier evaluations are discussed in detail in section 5.1, where possible reasons for the observed results are discussed along with possible means of increasing classifier accuracy further.

4.3 DNS Fast-Flux Detection

DNS Fast-Flux is a means of evading detection and preventing shutdown employed by malware authors. The principles behind Fast-Flux and the characteristics of Fast-Flux domains are discussed in subsection 2.2.2. Classifiers were constructed to examine DNS

Table 4.4: Results For Fast-Flux Classifiers

Classifier	Accuracy (%)	TPR (%)	FPR (%)
Modified Holz	93	86	2
Rule-Based	92	86	3
Naive Bayesian	92	86	3
Combined	95	89	2

Table 4.5: Mean values of DNS features for Fast-flux and legitimate domains

	A Records	NS Records	IP Ranges	Unique ASNs	TTL
Fast-Flux	4.09	3.92	3.89	3.70	595
Legitimate	1.73	3.88	1.15	1.09	14885

query responses with the aim of classifying whether of not a domain was Fast-Flux. Furthermore, the classifiers aimed to identify Fast-Flux domains from a single DNS query response, as opposed to previous work by Holz et al. (2008) which required a second DNS query once the TTL of the original query had expired. Reducing the amount of time required to classify a domain while maintaining high detection rate of Fast-Flux domains was essential. The classifiers were constructed to use only the DNS query response thus minimising interaction with the suspect domain and reducing the number of network resources required. Three classifiers were constructed: a modified version of the Holz classifier, a rule-based classifier and a Naïve Bayesian statistical classifier. The classifiers were tested using a dataset of 1 047 known Fast-Flux domains and 1 500 known legitimate domains.

The properties of DNS Fast-Flux domains were compared to those of legitimate domains and CDNs and are recorded in Table 4.5. It was observed that Fast-Flux domains had a larger number of associated A-Records with a mean of 4.09 compared to legitimate domains with a mean of 1.73. The number of Nameservers (NS Records) returned by Fast-Flux and legitimate domain name queries were similar with a difference of 0.04 in the mean values. A close correlation between the number of different IP ranges and the number of ASNs was observed, with legitimate domains usually having a single IP range and single ASN, while Fast-Flux domains had multiple IP ranges (a mean of 3.89) associated with the domain. Furthermore each of these Fast-Flux domains had IP ranges from numerous different ASNs. On average Fast-Flux domains had hosts in 3.70 ASNs. The largest difference in mean values observed was in the mean TTL values, where Fast-Flux domains displayed a low mean TTL value of 595, while legitimate domains generally had more persistent domain records with high TTLs (mean 14885).

4.3.1 Modified Holz Classifier

The modified Holz classifier (subsection 3.4.1) outputs a flux-score used to indicate the confidence in the classification of a domain as Fast-Flux. Domains with a score above 0 were deemed to be Fast-Flux. The classifier outputs for a selection of standard, CDN and Fast-Flux domains are shown in Table 4.6.The lowest flux-score was -25.14 for the *wikipedia.com* domain which is not Fast-Flux and only returns a single A-record with a high TTL of 3600. The Fast-Flux domains *girlsmeetclub.com*, *hookupdatingsite.net* and *lovenewgirl.com* all received a flux-score of 54.3 indicating a high confidence that these domains are Fast-Flux. It was noted that all three of these domains returned A-records from different IP ranges and unique ASNs, while all have a low TTL set. The domain *sergnovgorod23.narod2.ru* had been identified by the ZeuS tracker abuse.ch (2012) as a Fast-Flux domain used for hosting the C2 servers of the ZeuS botnet. The modified Holz classifier, however, classified this domain as a standard domain. The work produced by Holz et al. (2008) showed a detection accuracy of 99.98% with a standard deviation of 0.05%. During initial testing it was not possible to achieve the same results as the original Holz work with the traditional Holz classifier. The modified Holz classifier was tested against the set of known Fast-Flux domains and an accuracy of 93% was achieved. A low FPR of 2% was observed, with a TPR of 86%.

4.3.2 Rule-Based Classifier

The rule-based classifier produced results similar to the modified Holz classifier. The results closely matched those from the modified Holz classifier, with the vast majority of domains being classified similarly to those classified by the modified Holz classifier. The Fast-Flux domain *sergnovgorod23.narod2.ru* that had been misclassified by the modified Holz classifier was correctly identified as Fast-Flux if rounding was applied to the classifier results. Without rounding the classification fell just within the standard domain boundary and thus was misclassified. The same was observation was noted for the domain *lovenewgirl.com* which again fell just below the Fast-Flux domain decision boundary and was misclassified as standard. This was the same classification as made by the modified Holz classifier. The classifiers had an identical TPR of 86% while the rule-based classifier had a slightly higher FPR of 3%. This resulted in an overall classifier accuracy of 92% for the rule-based classifier.

Table 4.6: Fast-Flux Classifier Outputs

Domain	Domain Type	A-Records	IP Ranges	ASNs	Country Codes	TTL	Modified Holz	Rule Based
google.com	S	11	1	1	1	300	-11.94	6.1
facebook.com	S	6	3	1	1	1650	-18.54	8.6
cnn.com	S	4	1	1	1	240	-21.18	5.4
wikipedia.com	S	1	1	1	1	3600	-25.14	5.1
zynga.com	CDN	18	2	1	1	120	-2.7	8.3
cloudflare.com	CDN	5	1	1	1	170	-19.86	5.5
yahoo.com	CDN	3	0	3	1	3380	-27.5	7.3
girlsmeetclub.com	FF	5	5	5	4	350	54.3	20.5
nomorelala.com	FF	3	3	3	3	300	14.58	13.3
sergnovgorod23.narod2.ru	FF	4	4	1	1	90	-21.18	9.9
hookupdatingsite.net	FF	5	5	5	5	600	54.3	21.5
xdateslocal.com	FF	5	4	4	4	600	35.76	17.5
lovenewgirl.com	FF	5	5	5	5	500	54.3	21.5
girlsinsidecity.com	FF	4	6	4	4	300	34.44	20.4
loveschemes.com	FF	2	2	2	2	300	-5.28	9.2
runnewlove.com	FF	4	4	4	3	600	34.44	16.4
ocicinaka.com	FF	4	4	1	1	300	-21.18	9.9

4.3.3 NaïveBayesian Classifier

The Naive Bayesian classifier assumes that a domain has equal initial probabilities of being Fast-Flux or legitimate. It was hoped that the Naïve Bayes classifier would be able to establish fuzzy decision boundaries, where domains mis-classified by the modified Holz and Rule-Based classifiers would be correctly classified. The output obtained from the classifier is shown in Table 4.7 and both the Fast-Flux and benign scores are listed for comparison. The results from the classifier were examined to determine if there was an improved accuracy compared to the existing rule-based type classifiers. Furthermore, an investigation was made to determine whether the similarity between Fast-Flux and CDN domains could be discerned by the classifier. It was noted that the classifier correctly identified the CDNs of *cloudflare.com* and *yahoo.com* as legitimate, while misclassifying the CDN for *zynga.com* as Fast-Flux. All but a single Fast-Flux domain was classified correctly, with the domain *sergnovgorod23.narod2.ru* correctly being classified as Fast-Flux, in contrast to both the modified Holz and rule-based classifiers, which had classified it as legitimate. The domain *loveschemes.com* was misclassified by all three classifiers as legitimate: it was actually Fast-Flux. It was noted that the output value of the classifier widely varied for each domain with no convergence towards a point for either legitimate or Fast-Flux domains. The classifier achieved the same TPR (86%) as both the modified Holz and the rule-based classifiers. The FPR was slightly higher than the modified Holz classifier but was the same as the rule-based classifier, resulting in all three classifiers having near-identical accuracy rates as previously summarised in Table 4.4.

4.3.4 Combined Classifier

A combined classifier was constructed to use the outputs from the three aforementioned classifiers in order to produce a better classification. The classifier consisted of a feed-forward neural network with three inputs and a single hidden layer. The hidden layer used a sigmoid function as a linear combiner and the result was passed to the output layer where the final classification was done. The output from all three classifiers were recorded for 2547 known legitimate and Fast-Flux domains and divided into a training and test datasets. The training dataset consisted of 1800 domains while the remaining 747 domains were used as test cases. The test data was applied to the trained neural network (training error 0.072) and 600 iterations were performed with the average results recorded as the classifiers performance. The trained neural network produced results similar to the

Table 4.7: Naive Bayesian Classifier Results for Fast-Flux and CDN Domain Queries

Domain	Type	Benign Score	Fast-Flux Score	Classifier Result
google.com	S	5.700×10^{-8}	3.478×10^{-8}	Legitimate
facebook.com	S	130.960×10^{-8}	550.480×10^{-8}	Fast-Flux
cnn.com	S	1566.200×10^{-8}	86.598×10^{-8}	Legitimate
wikipedia.com	S	2804.800×10^{-8}	66.272×10^{-8}	Legitimate
zynga.com	CDN	$0.000\,027 \times 10^{-8}$	$0.000\,652\,39 \times 10^{-8}$	Fast-Flux
cloudflare.com	CDN	1804.500×10^{-8}	38.235×10^{-8}	Legitimate
yahoo.com	CDN	26.425×10^{-8}	5.2744×10^{-8}	Legitimate
girlsmeetclub.com	FF	$0.000\,000\,155\,37 \times 10^{-8}$	1449.700×10^{-8}	Fast-Flux
nomorelala.com	FF	16.172×10^{-8}	847.070×10^{-8}	Fast-Flux
sergnovgorod23.narod2.ru	FF	11.698×10^{-8}	502.280×10^{-8}	Fast-Flux
hookupdatingsite.net	FF	$0.000\,000\,103 \times 10^{-8}$	492.58×10^{-8}	Fast-Flux
xdateslocal.com	FF	$0.000\,595 \times 10^{-8}$	1539.200×10^{-8}	Fast-Flux
lovenewgirl.com	FF	$0.000\,000\,132 \times 10^{-8}$	1232.800×10^{-8}	Fast-Flux
girlsinsidecity.com	FF	$0.000\,000\,324 \times 10^{-8}$	1232.800×10^{-8}	Fast-Flux
loveschemes.com	FF	933.52×10^{-8}	278.470×10^{-8}	Legitimate
runnewlove.com	FF	$0.000\,916 \times 10^{-8}$	1895.000×10^{-8}	Fast-Flux
ocicinaka.com	FF	2.635×10^{-8}	19.154×10^{-8}	Fast-Flux

Table 4.8: Summary of Fast-Flux Classifier Accuracy Rates

Classifier	Accuracy (%)
Modified Holz	93
Rule-Based	92
Naive Bayesian	92
Combined	95

three stand-alone classifiers, with an overall accuracy of 95%. The total TPR was 86% and the false positives were recorded at a FPR of 2%.

4.3.5 Summary

The results showed near identical accuracy for the three classifiers examined, with all three classifiers achieving accuracy rates of 92%-93%. The hand-crafted classifiers adapted from Holz and the rule-based classifier both had TPRs of 86% and low FPRs of 2-3%, exactly matching the performance of the Naïve Bayesian classifier. This closely matched performance resulted in near-identical performance curves for all three classifiers, as seen in Figure 4.6. The AUC of for all three classifiers was 0.94 indicating a high degree of performance by the classifiers, with a very low trade off between the TPR and FPR. Combining the results of the classifiers and using them as inputs for a neural network based classifier resulted in an increase in the number of Fast-Flux domains correctly identified, with a TPR of 86%, while the number of legitimate domains being incorrectly classified dropped to 2%. The combined classifier produced the best degree of accuracy 95%. The accuracy rates of the three classifiers as well as the combined classifiers are summarised in Table 4.8.

4.4 Spatial Autocorrelation

Classifiers based on spatial autocorrelation were constructed with the aim of classifying domains as either Fast-Flux or legitimate based on the the geographic distribution of the domain's servers. These servers were identified using the A records contained in the DNS query response and the geographic location determined using the MaxMind database (subsection 3.5.1). Different methods of quantifying the geographic position were used, including the Latitude/Longitude, the UTM grid position and the MGRS grid position. The distance between geographic points was measured using the Haversine

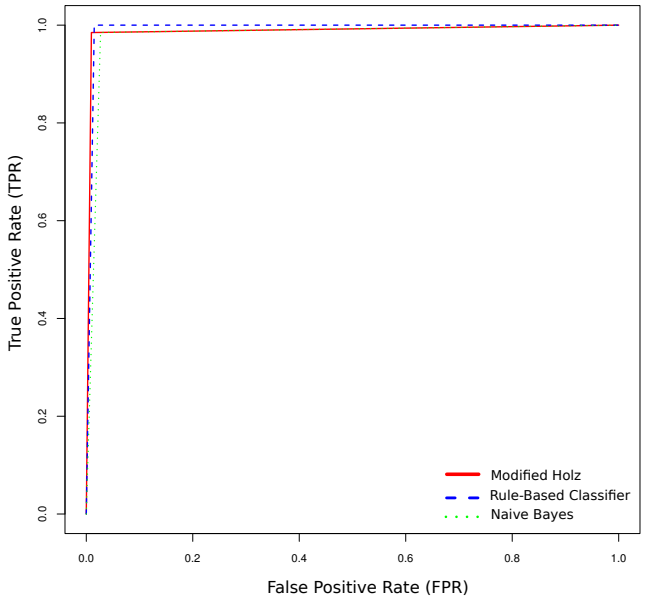


Figure 4.6: ROC of Fast-Flux Classifiers Accuracy

formula (Robusto, 1957), which measures the distance between two points on a curved surface. The classifiers were constructed using known Fast-Flux and legitimate servers with the output from each classifier being used to determine a decision boundary for classifying domains. The classification problem was treated as a binary decision with the classifier output indicating whether the geographic distributions identify Fast-Flux hosting or legitimate server hosting, including CDNs and standard domains. Two well known statistical techniques for measuring spatial autocorrelation were used: Moran's Index (subsection 3.6.1) and Geary's Coefficient (subsection 3.6.2). Each classifier was examined separately, with the three means of quantifying geographic position used with each classifier to find the classifier with the greatest accuracy.

4.4.1 Moran's Index

Moran's Index (MI) relies on the observation that points closer together in geographic space tend to have more similarities in their attributes than points far apart. The values for I were calculated separately for legitimate and Fast-Flux domains. Once these values had been calculated they were compared to see if there were any distinguishing values which could be used for accurately classifying the domains. The results for the MI classifier are shown in Table 4.9 where it can be seen that the classifier produced high TPRs using all three geographic position measures, while maintaining low FPRs. The overall accuracy for

Table 4.9: Moran's Index Classifier Performance

	Accuracy (%)	TPR (%)	FPR (%)
Timezones	97	97	3
UTM	95	99	6
MGRS	95	99	6

the classifiers was high with the lowest observed accuracy rate of 95%. The MI classifiers were tested using 1047 known Fast-Flux domains and 1500 known legitimate domains.

Using the timezones in which servers are located, results displayed index values for legitimate domains with a mean of 0, with 97% of observed legitimate domains having an index value of zero. A small cluster of domains produced a value of -1 and accounted for 2% of the total observed index values. The remaining 1% of domains having an index value between -0.2 and 0. The opposite holds true for Fast-Flux domains, with only 3% of observed domains had an index value of zero, while most Fast-Flux domains displayed index values distributed between -1 and 0. A small sample of domains had index values greater than 1 and accounted for 1.5% of all the observed domains. Using these observations as a classifier decision boundary, domains were labeled as Fast-Flux if the returned index value was not equal to zero. Results for this classifier can be seen in Table 4.9, where the classifier has a high true positive rate of 97%, a low false positive rate of 3% and an overall accuracy of 97%.

The UTM grid location of a server was used to provide a more fine grained location designation than timezones as the grid area identified by UTM represents a smaller surface area. Furthermore, unlike timezones the UTM grid location takes into account the hemisphere in which a server is located. The kernel density distribution of the Moran Index values for legitimate (Alexa Top 1000) domains and Fast-Flux domains are compared in Figure 4.7. It was observed that the index values for legitimate domains were zero, with a few outliers with index values of 1 accounting for 1% of all domains in the training set. Fast-Flux domains tended to have an index value of one or greater with a peak just below zero and at 1. A classifier decision boundary was set at a value of zero, where any index value not equal to zero indicating a Fast-Flux domain. A TPR of 99% was achieved as seen in Table 4.9, while a low false positive rate of 6% was achieved. The classifier achieved an overall accuracy of 95%.

The MGRS co-ordinate system provided a grid system with smaller grids than the UTM co-ordinate system, allowing for even finer grained representation of server locations.

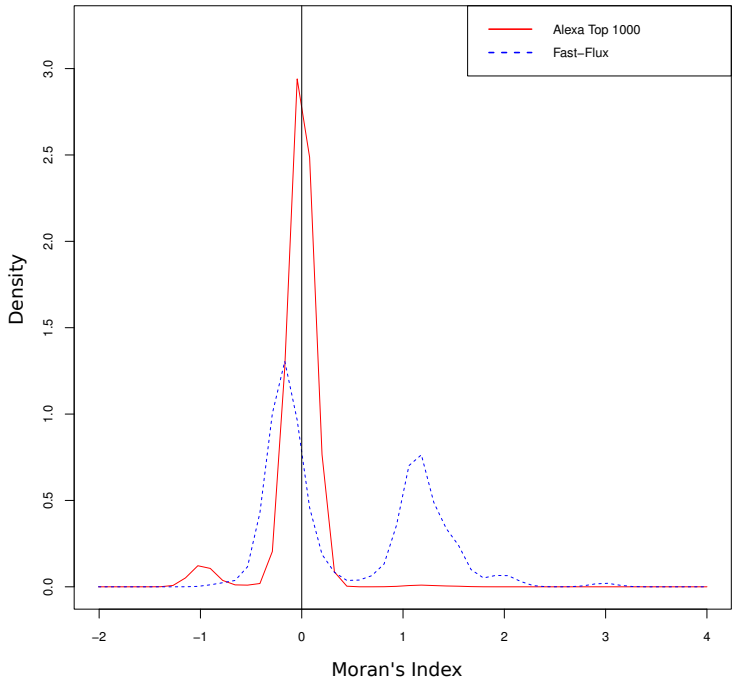


Figure 4.7: Kernel Density Comparison of Moran's Index Using UTM.

Using MGRS produced interesting results as the index value for Fast-Flux domains was distributed between -0.5 and 0. While the index value for legitimate domains was mostly zero, a smaller cluster formed around an index value of -1. Basing the classifier on the same logic as was used for the timezone and UTM classifier, where an index value of zero indicated a legitimate domain, a TPR of 99% was achieved, and a lower FPR of 6% was achieved. As seen in Figure 4.8, the Moran's I for numerous legitimate domains is -1, with no Fast-Flux domains having a Moran's I of -1. Thus modifying the classifier to classify any value of -1 or 0 as legitimate led to an improved FPR of 1%, increasing the classifiers accuracy to 99%.

All three classifiers performed with a high degree of accuracy and thus a high AUC was observed for each one. The Figure 4.9 shows the performance curves for all three classifiers, where it can be seen that there was a low trade-off between the TPR and FPR for each classifier.

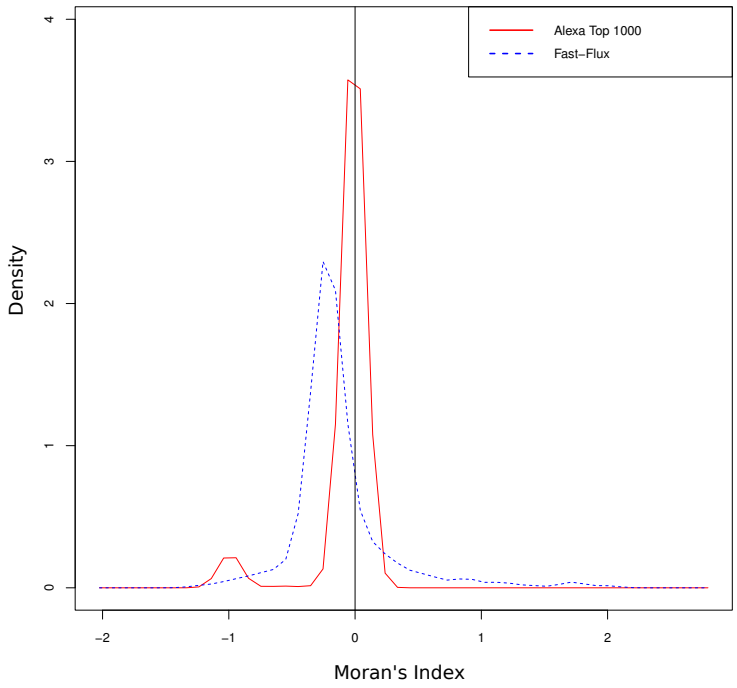


Figure 4.8: Kernel Density Comparison of Moran's Index Using MGRS.

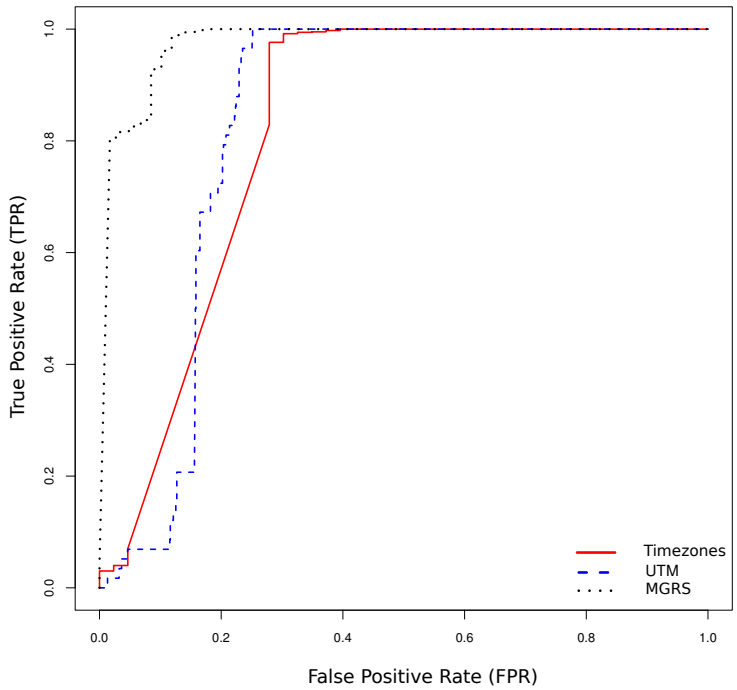


Figure 4.9: ROC for Moran's I Classifiers

Table 4.10: Geary's Coefficient Classifier Performance

	Accuracy (%)	TPR (%)	FPR (%)
Timezones	95	92	3
UTM	96	98	5
MGRS	95	99	6

4.4.2 Geary's Coefficient

Geary's Coefficient (GC) is used for spatial autocorrelation, but is more sensitive to localisation than MI. The use of GC should allow for classification of spatial clusters in instances where MI might not be as accurate, which may occur when the geographic location of servers are closer together. The same dataset of 1047 Fast-Flux and 1500 legitimate domains were used for testing the GC classifier's performance compared to the MI classifiers.

The value produced by Geary's formula was used as the basis for classifying a domain as legitimate or Fast-Flux. The GC value was returned as zero for 97% of all legitimate domains, while returning a value greater than zero for 92% of Fast-Flux domains. Using this as the classification criteria, domains that returned a GC value greater than zero were all classified as Fast-Flux. The resulting classifier had an accuracy of 95% while the TPR was high with 92% of Fast-Flux domains correctly identified. A small portion of legitimate domains were incorrectly identified resulting in a FPR of 3%.

As seen with the MI classifier, UTM provided a higher degree of certainty when classifying domains due to the more fine grained nature. The GC value for legitimate domains was clustered around zero, a similar distribution to that observed with the equivalent MI classifier. Two clusters were observed for Fast-Flux domains: one cluster at 0.5 and a second cluster around 1. This led to the use of the same classifying criteria as before, with a value of zero indicating a legitimate domain while a value greater than zero indicated a Fast-Flux domain. The results from this classifier produced a 98% TPR with a slightly higher FPR than the timezone based classifier at 5%. The overall accuracy achieved by this classifier was 96%.

The MGRS was used to provide an input value for the GC classifier with the aim of allowing for a finer grained analysis of server distribution. The values for legitimate domains were clustered at zero, as can be seen in Figure 4.10. It was further observed that the Fast-Flux domains were clustered around 1. There were no negative values for

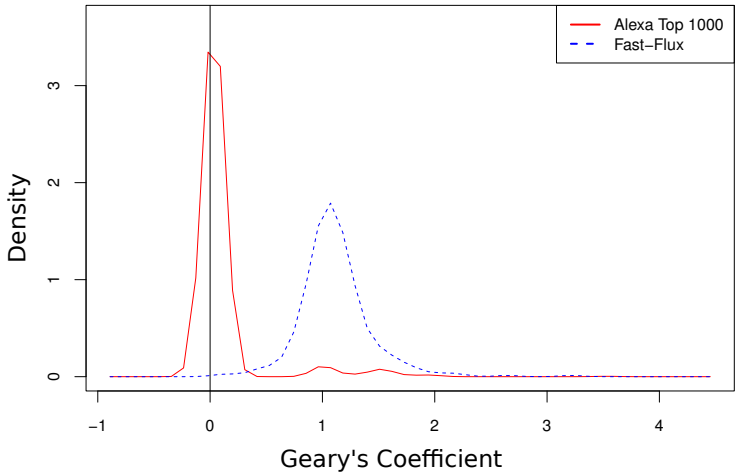


Figure 4.10: Kernel Density Distribution for Geary's Coefficient.

Table 4.11: Summary of Spatial Autocorrelation Classifier Accuracy

Classifier	Moran's Index Accuracy (%)	Geary's Coefficient Accuracy (%)
Timezone	97	95
UTM	95	96
MGRS	95	95

GC as expected, due to the definition of GC. The classifier was constructed with a value of zero indicating a legitimate domain and any value above zero indicating a Fast-Flux domain. The performance of the classifier was in line with the performance of the MGRS classifier based on MI, with a similar TPR of 99%. The FPR of the classifier remained low at 6% and resulted in a classifier accuracy of 95%. The accuracy of the MGRS based classifier was equal to that of the timezone based classifier and marginally less accurate than the UTM classifier. The AUC for all three classifiers was above 0.9 as can be seen in Figure 4.11.

4.4.3 Spatial Autocorrelation Summary

Two means of measuring spatial autocorrelation were examined as classifiers, while three different measures of spatial value were used for each of these classifiers. It was found that classifiers based on timezones produced the lowest number of false positives, while the use of the MGRS produced the best TPR. The best performing classifier was the Moran's Index classifier using timezones as a spatial measure, resulting in a classifier accuracy of 97% with a TPR of 97% and a FPR of 3%. The worst performing classifier was the GC classifier using timezones, with an accuracy of 95% and a TPR of 92%, with a 3%

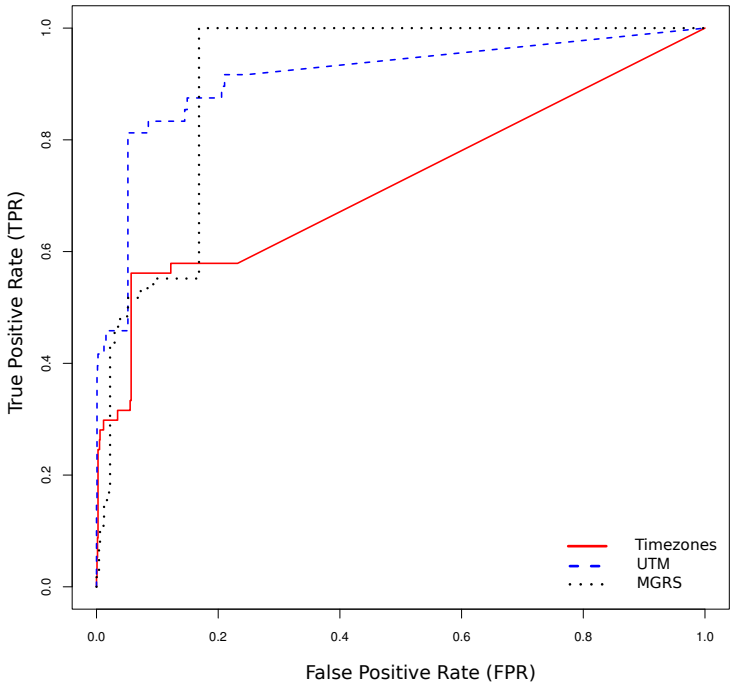


Figure 4.11: ROC for Geary's Coefficient Classifiers

FPR. Overall it was found that the classifiers are highly accurate and it was possible to detect up to 99% of all Fast-Flux domains with a high degree of confidence. There was a minimal trade-off between the TPR and FPR with the worst FPR observed being 6% but this was traded-off against a TPR of 99%. The similarities between the performance of the Moran's Index and Geary's Coefficient classifiers can be seen summarised in Table 4.11.

4.5 Performance Analysis

The accuracy of the classifiers has been described in the preceding sections, however another critical aspect of evaluating the classifiers was the performance of the classifiers in terms of speed and resource consumption. These results are particularly pertinent to establishing the feasibility of using the proposed classifiers outside of the research environment and the potential for future implementation on live networks.

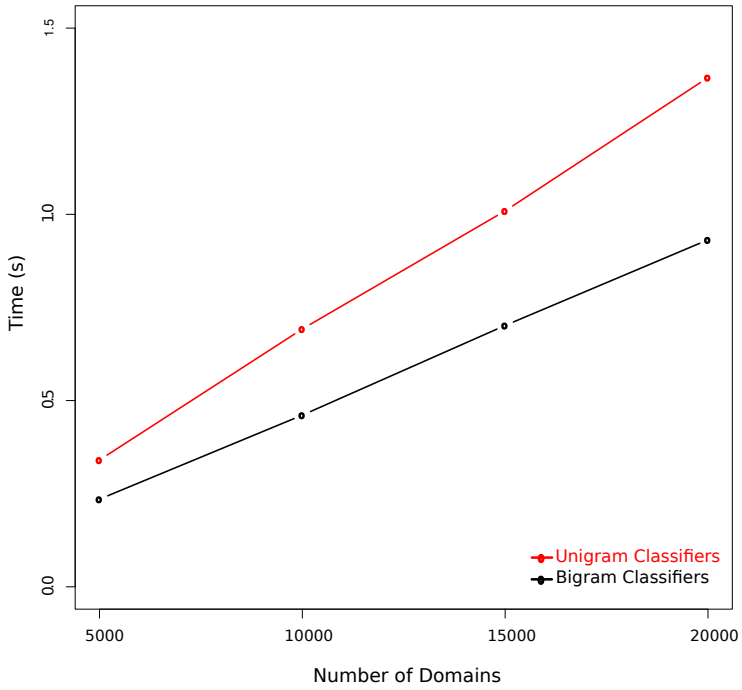


Figure 4.12: Processing Time for Lexical Classifiers

4.5.1 Test Platform

Testing was performed on an Intel Core i5-2410M 2.30GHz Ubuntu 12.04 x64 desktop PC, with 8 GB of DDR3 1600MHZ RAM. The classification system was coded in Python 2.7 and used a single-threaded execution model. The live traffic dump file used, dataset LD1, was read into memory in 20 000 packet batches and then passed to the classifiers. This was done to try avoid delays introduced when reading from disk, and also to minimise the effect of the inefficiency of reading .pcap files (Nottingham, 2011).

4.5.2 Lexical Classifiers Performance Analysis

Initial performance testing of the lexical classifiers was performed using a flat file consisting of the combined datasets AD2 and AD3. This performance test was designed to simulate the process of log parsing, such as might occur on a daily basis or during post-incident analysis. The tests were performed using all the classifiers described in section 3.3 together. The time to execute was recorded for each test over 600 iterations and the average processing time was used as the final performance measure. The time to classify

Table 4.12: Real-World Performance of Lexical Classifiers

	DOMAINS	TIME (s)	TIME PER DOMAIN (s)
Unigram	254,214	9.864	3.800×10^{-5}
Bigram	254,214	14.005	5.500×10^{-5}

5 000, 10 000, 15 000 and 20 000 domains is shown in Figure 4.12. It was seen that the unigram classifiers executed 1.5 times faster than the bigram classifiers. A linear increase in processing time was observed with both unigram and bigram classifier processing time increasing at a uniform rate as the number of domain names increased. The time taken to process a single domain name was calculated using the results obtained when 20 000 domains were classified. The combined unigram classifiers took approximately 0.046ms per domain name, while the combined bigram classifiers required 0.068ms per domain name. The tests were repeated using datasets LD2 and LD3 as real-world examples of proxy log parsing and blacklist parsing. The results for 600 iterations of LD3 are presented in Table 4.12. The current solution has no form of caching, thus reading from disk does have a slight performance impact, though this is negligible as it took 0.06s to read the total LD3 dataset.

4.5.3 Fast-Flux Classifiers Performance Analysis

The Fast-Flux detection classifiers were tested using a real world dump of network traffic as contained in the LD1 dataset described in subsection 3.1.3. The dataset consists of raw DNS query responses, thus giving a good representation of a real world scenario where the classifiers are used to process a network capture post-incident. The use of raw DNS query response captures would give a good indication of the processing duration of a DNS query response in realtime. The time it took to process a set number of packets was measured for each classifier, with each test being repeated 600 times. The average processing time across these 600 iterations was used as an indicator of the classifier's overall performance. The performance of the various Fast-Flux classifiers is shown in Figure 4.13; the performance of each classifier was tested separately. Finally the performance of all the classifiers employed simultaneously.

From these performance measures the estimated time to process a single DNS query response was estimated. Table 4.13 shows the estimated processing time per DNS packet for each classifier. These values were estimated across 600 iterations, processing 20000 domains during each iteration. It was noted that the average time to parse a packet was

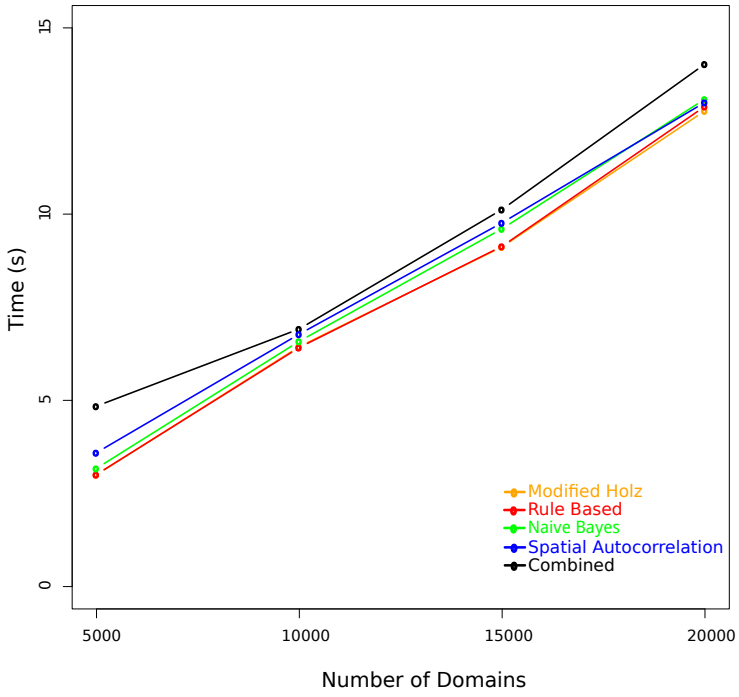


Figure 4.13: Processing Time of the Various Fast-Flux Classifiers

Table 4.13: Processing Time Per DNS Response Packet

Classifier	Total Time (s) (N=20,000)	Processing Time (s)
Modified Holz	12.775	6.387×10^{-4}
Rule Based	12.893	6.447×10^{-4}
Naive Bayes	13.088	6.544×10^{-4}
Spatial Autocorrelation	13.002	6.501×10^{-4}
Combined	14.030	7.051×10^{-4}

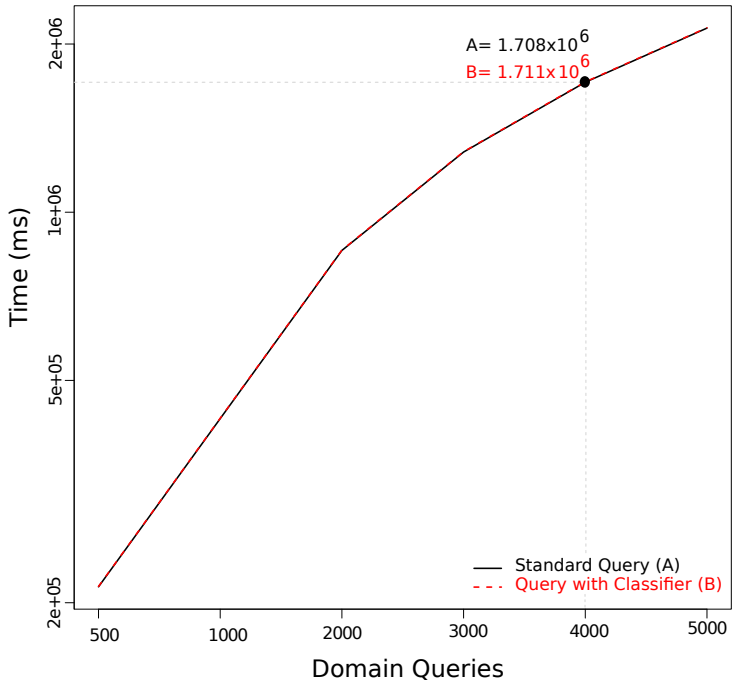


Figure 4.14: Projected Performance Impact of DNS Query Response Classification

6.5×10^{-4} s across the classifiers. This translates to 6.5×10^{-1} ms per packet. The average time per DNS query was calculated by querying the top 500 domains in the AD2 dataset (subsection 3.1.1) and was it was determined that a DNS query takes approximately 427.132ms to process. The expected performance for classifying DNS query responses, compared to the calculated performance for standard DNS queries without classification are shown in Figure 4.14. It was noted that there was an indecernable increase in DNS query completion time, as can be seen in Figure 4.14, where 4000 standard DNS queries took 1.708×10^6 ms to complete and 4000 queries with the classifiers took 1.711×10^6 ms. There was an approximately increase of 0.152% in the time taken for a DNS query to be completed.

The projected times used in Figure 4.14 are shown in Table 4.14, where the minor difference between the projected processing time of standard DNS queries and DNS queries that are filtered by the classifiers can be seen.

Table 4.14: Projected Performance Impact of DNS Query Response Classification

Domains	Standard Query (ms)	Query With Classification (ms)
500	2.135×10^5	2.138×10^5
1000	4.271×10^5	4.277×10^5
2000	8.542×10^5	8.555×10^5
3000	1.281×10^6	1.283×10^6
4000	1.708×10^6	1.711×10^6
5000	2.136×10^6	2.139×10^6

4.6 Summary

This chapter presented the results obtained from three different types of classifiers. The results for classifiers used to detect algorithmically generated domain names were presented in section 4.2, where the performance of the classifiers were compared when using the unigram distribution and bigram distribution of characters in domain names. It was shown that the use of bigrams for classification resulted in a higher accuracy rate, with the unigram classifiers having an average accuracy rate of 85%, while bigram classifiers having an average accuracy of 87%. Combining the classifiers produced accuracy rates of 89% and 88% respectively, with the unigram classifier edging out the bigram classifier due to a 1% lower FPR.

The second set of results presented in section 4.3 showed the performance of classifiers used to detect Fast-Flux domains using the features of DNS query responses. Three classifiers were compared: a modified Holz classifier, a rule-based classifier and a Naive Bayesian classifier. It was shown that Fast-Flux domains could be detected with a 93% accuracy rate, with false positive rates between 2 and 5%. The Holz classifier had the lowest FPR while the Naive Bayesian classifier had the highest TPR with only a marginally higher FPR. Combining the three classifiers using a neural network increased the overall accuracy to 95% with a only a 2% FPR where 86% of the observed Fast-Flux domains were correctly identified.

section 4.4 presented the results of the spatial autocorrelation classifiers. These used the geographic location of C2 nodes to determine whether a domain was Fast-Flux. It was shown that Fast-Flux domains could be detected with an average accuracy of 96% across the proposed classifiers. Using the geographic location, it was possible to correctly detect up to 99.89% of Fast-Flux domains with a FPR of only 6%.

In section 4.5 a performance analysis was conducted to determine the feasibility of deploying the proposed classifiers in a real-world environment. It was shown that the proposed

classifiers had a minor performance drawback in terms of time taken to process each domain. This performance impact was particularly negligible in light the time taken to perform a standard DNS query, with the classifiers adding only 0.152% to the execution time.

These results show that it is possible to accurately identify two types of botnet domains, namely DGA and Fast-Flux domains, from the contents of a DNS query response. The results are discussed in greater detail in chapter 5. The possible weaknesses of the classifiers are discussed along with the means to improve the classifier results. The projected real-world performance of the classifiers are discussed with possible means of increasing the performance presented.

5 Discussion

The primary aim of this research was to accurately classify domains used for botnet Command and Control servers (C2) by performing passive analysis of DNS traffic. Three techniques for detecting botnet domains using the contents of an observed DNS query response were proposed. These techniques focused on the detection of botnets that made use of two well known obfuscation techniques, Domain Generation Algorithms (DGAs) and DNS Fast-Flux. The first classifier used lexical analysis to detect algorithmically generated domain names such as those used by botnets employing DGAs to mask the C2 nodes. The second classifier used the fields of DNS Fast-Flux query responses to identify the properties of Fast-Flux domains and hence detect queries for Fast-Flux domains. The final detection technique made use of spatial autocorrelation and the geographic dispersion of C2 servers returned in DNS query responses to detect Fast-Flux domains.

The results showed that it was possible to achieve a high degree of accuracy in classifying botnet domains based on the outlined techniques. The implications of these results and their shortcomings are discussed in the following section. The results of lexical analysis and the detection of algorithmically generated domain names are discussed in section 5.1. Results from the different Fast-Flux classifiers are discussed in section 5.2, while the results of the spatial autocorrelation classifiers are discussed in section 5.3.

5.1 Algorithmically Generated Domain Names

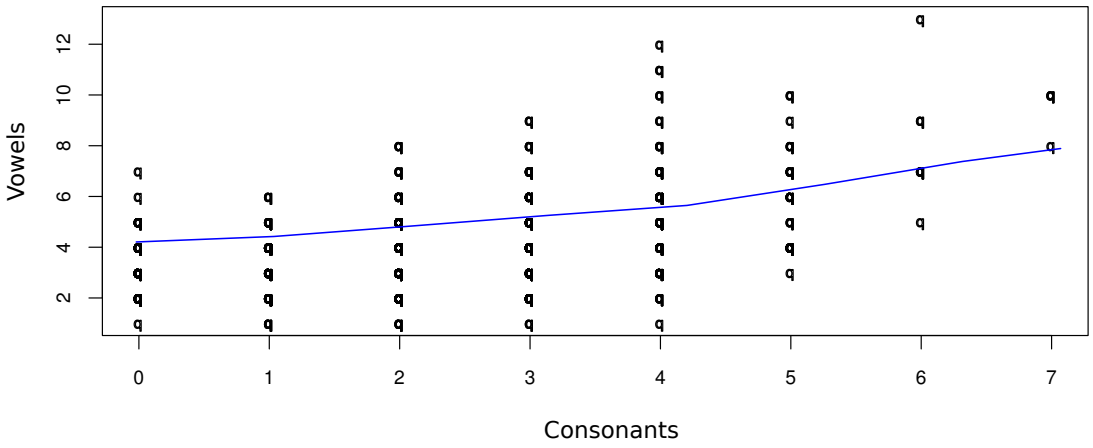
The detection of algorithmically generated domain names, such as those employed by the Kraken, Torpig and Conficker botnets, was based on the lexical analysis of known

legitimate domain names and known algorithmically generated domain names. Domain names were extracted from DNS queries and multiple classifiers were developed using different statistical techniques which examined single letter frequencies (unigram analysis) along with letter combinations (bigram analysis). These techniques all showed a high degree of accuracy, with the majority of algorithmically generated domain names being detected and a low number of legitimate domain names being misclassified. The results achieved were comparable to those seen in previous work Yadav et al. (2010), which showed that the detection of algorithmically generated domain names could be achieved through K-Means clustering.

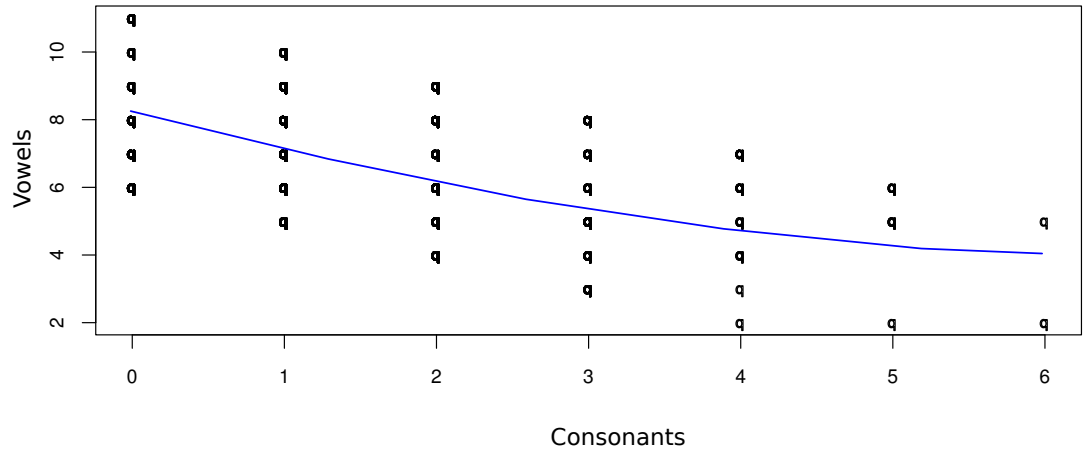
The work presented in section 4.2, however, was a novel solution where single algorithmically generated domain names could be detected with a high degree of accuracy, whereas the work of Yadav et al. (2010) suggested that a domain-fluxing botnet could be accurately characterised by the time it had generated around 500 names. While previous work achieved high detection rates with a low false positive rate, the large number of observed domains to achieve this resulted in a delay between the observation of the first algorithmically generated domain and the classification of this domain. In the case of a botnet such as Conficker-C, where upwards of 50 000 domain names were generated in a 24 hour period (Fitzgibbon and Wood, 2009; Porras et al., 2009), this delay would be negligible. Botnets generating fewer than 500 domain names would not necessarily be detected though, as was the case with Conficker-A which only generated 250 domains a day. Therefore our work aimed to produce a classifier capable of detecting algorithmically generated domains with a high degree of accuracy, even when only a single domain was observed. Previous works in the field of algorithmically generated text detection focused on larger key-spaces than this body of work, usually looking at whole bodies of text as opposed to this work that only examines only single words as seen in domain names.

5.1.1 Observed Character Distribution

Analysis of the algorithmically generated domain names showed that depending on the DGA used, the generated domains displayed characteristics which could help identify which DGA was used to generate a given domain name. This was particularly easy with the Torpig domains as inspection of the last three characters could help identify whether a domain name was a possible Torpig domain. This was possible due to the Torpig DGA using a known suffix of three characters depending on the current month (Unmask Parasites, 2009). Similar patterns were noticed in the analysis of Conficker-C



(a) Vowels to Consonants Ratio for Legitimate Domain Names



(b) Vowels to Consonants Ratio for Conficker-C and Kraken Domain Names

Figure 5.1: Comparison of Vowel to Consonant Ratios

and Kraken domains. Both DGAs shared the characteristic of producing domain names with fewer vowels as the domain name length increased. This was the opposite from what was observed in legitimate domain names, where the number of vowels increased with the domain name length and a steady ratio between vowels and consonants was maintained. A comparison of the ratios are shown in Figure 5.1, where 5.1a shows how the number of vowels seen in the sample AD2 dataset, steadily increase to match the increase in consonants. 5.1b shows how the number of vowels steadily increase as the number of consonants increases. From these observations it is clear that future work can be done in investigating other features present in domain names that have been algorithmically generated.

The classifiers were developed on the assumption that most domain names follow similar distribution patterns to latin based languages such as English as described in section 3.2,

however numerous domain names were observed that did not conform to this assumption. Examples of these are *qq.com*[1] and *fbcdn.net*[2]. These valid domains do not contain any vowels and contain letters that have a low probability of occurring in English words (q), as well as letter combinations not commonly seen in English words (qq,fb,cd,dn), thus being incorrectly classified as algorithmically generated domain names. Furthermore, during testing it was noted that services such as the Google API generate algorithmic domain names such as *cdsj1ojsf7281.google.com*. This may be overcome by ensuring that the Second Level Domain (SLD) name is always examined by the classifier.

The obvious weakness of the above mentioned classifiers is that they assume that domain names tend to follow a probability distribution similar to the letter frequency distribution in English, while DGAs produce a probability distribution akin to a randomly distributed sample of letter frequencies. Research has been conducted into the algorithmic generation of pronounceable words, particularly for the use in password generators (Gasser, 1975). These systems use the frequency distribution of characters in the English language as the basis for generating pronounceable and English looking words. These password generators tend to use the tri-graph probabilities of characters as this leads to more pronounceable words (Gasser, 1975). Due to the use of tri-graph probabilities these generated words may still be detected using the unigram distribution of characters. Applying the classifiers to these English looking words that had been generated using the tri-graph probabilities of characters showed that there was a marked increase in the number of false positives. The majority of these words were, however, detected, with the unigram classifiers proving to be more effective than bigram classifiers by up to 10%. Another possible weakness of the proposed system is the assumption that domain names are constructed from the English language. This may hold true for the domains observed in this research, but if the system is to be deployed in a 'real world' network, the domains visited by users on that network will depend heavily on the native language of those users. As a result the system will need to be trained to work with the letter frequencies of that language, as these vary depending on the language (MacKenzie and Soukoreff, 2003). Determining the domains which are most visited by the users of the network will allow for the construction for a network specific training set to be used in construction of the classifiers.

[1]Chinese Internet service portal owned by Tencent Inc.

[2]The Content Distribution Network used to host the Facebook.com photo service

5.1.2 Unigram Versus Bigram Classifier Accuracy

The unigram classifiers employed all achieved similar detection rates with the main variant being the number of false positive results. A high detection rate was observed when the probability of characters occurring in domain name were calculated. The probability of a character occurring in a domain names was taken as an independent event, meaning that the characters position within the domain name had no effect on the probability of that character occurring. Furthermore, each character was treated as independent of the other characters in the domain name. The assumption of independence did result in the length of the domain name influencing the accuracy of the classifier. This can be observed in the results for the probability classifier in section 4.2.1. The results showed that shorter domain names resulted in a higher number of false positives, as can be seen in the results of the probability classifier, section 4.2.1, where the FPR decreased from 17% to 8% when only domain names longer than six characters were examined. This was attributed to the fact that a single character with a high probability of occurring in either a legitimate or algorithmically generated domain name could sway the results. The minimum domain name length of domains from the sample botnet datasets was four, thus indicating the possibility of constructing classifiers which classify domains with domain name length of less than four as benign by default. This works because of the assumption that all one, two and three character domain names have already been registered. By limiting the length of domain names examined to length of five or greater, it was possible to get the FPR down to as low as 2% with a TPR of 92% using the bigram Bayesian classifier. This technique would work for DGAs such as those employed by the earlier variants of Conficker (Conficker-A and Conficker-B) which generated domain names of length 8 to 11 characters. The Conficker-C variant would not be detected as accurately as this generated domain names with length 4 to 9 characters (Fitzgibbon and Wood, 2009).

Classifiers adjusted to examine bigrams (as seen in subsection 4.2.2) were constructed with the aim of increasing the TPR and decreasing the FPR. The same training and test datasets were used and the results compared with those from the unigram classifiers. The results showed, on average, an increased TPR but it was noted that the FPR remained the same or only decreased slightly. These results are similar to those seen in the work by Yadav et al. (2010) where only a minor change in classifier accuracy was noted in their research when bigrams and trigrams were used. The observed change in classifier accuracy was, however, a positive change in accuracy rate compared to Yadav et al. (2010) who noted a decrease in classifier accuracy. The higher accuracy was expected but was less significant than initially predicted. This diminished improvement could be attributed to

the number of short domain names with a length of less than six. Adjusting the classifiers to only classify domain names with longer lengths resulted in a greater increase in accuracy rates and a lower number of false positives. The increased accuracy when examining domain names of length 8 or greater was attributed to the larger sample space and the normalising effects of having a greater number of probabilities. The increased number of bigrams in longer domain names minimise the effect of bigrams with large probabilities influencing the overall result of the classifier output. This was seen as a non-ideal solution as the aim of this research was to construct classifiers capable of identifying domain names of any length. The detection rate based solely on character probabilities performed better than other proposed solutions that focused on the distribution of consonants and vowels in the domain name, such as suggested by Alienvault Labs (Alienvault Labs, 2012). The results of the classifiers surpass those obtained in the Alienvault Labs study when applied to only Conficker-C domains. When applying the detection algorithm to domains generated by Torpig and Kraken, which the Alienvault Labs work had not been exposed to previously, the proposed classifier produced far superior results.

5.1.3 Summary

The research showed that it was possible to detect algorithmically generated domain names with a high degree of accuracy. These classifiers have the benefit of being lightweight, accurate and fast. Furthermore, the proposed classifiers provide the added benefit of being able to detect algorithmically generated domain names from a single query unlike previous work which required large numbers of samples before detection was possible. This allows for earlier detection of communication attempts by infected hosts and subsequently allows for this traffic to be blocked before any communication can be established. As the domain name is contained in the initial DNS query it is possible to detect whether a DNS query is to a algorithmically generated domain name even before a DNS query response has been received. This early detection means that communication between infected hosts and C2 servers could be prevented even before an attempt to establish connection has been made.

During the development of the classifiers it was observed that the size of the training dataset had a marked influence on the accuracy of the classifiers. Furthermore, it was noted that the use of a single malicious dataset (such as Conficker-C domains) produced classifiers capable of identifying algorithmically generated domain names from other botnet families such as Kraken and Torpig. This observation showed the benefit of creating classifiers capable of detecting algorithmically generated domain names as the classifiers

would be able to identify algorithmically generated domain names that have never been seen before.

5.2 DNS Fast-Flux

Botnets have historically employed numerous techniques to prevent or delay the shut-down of the C2 servers used to distribute commands to nodes of the botnet. One of these techniques -DNS Fast-Flux- relies on rapidly changing DNS query responses, with various different servers' IP addresses being returned in a short period of time. This means of shut-down avoidance has led to the creation of a variety of detection strategies by researchers as highlighted in subsection 2.3.2. These detection strategies predominantly rely on predefined detection metrics or detection rules based on observed Fast-Flux characteristics such as a short TTL. A modified classifier was developed to extend these current Fast-Flux detection metrics to match the observed behaviour of modern Fast-Flux domains. Furthermore, two new classifiers were constructed one rule-based classifier and a novel statistics based classifier. The results of the three proposed classifiers were used as inputs to a neural network, resulting in the creation of a fourth classifier which increased the overall detection rate of Fast-Flux domains.

5.2.1 Classifier Results

The results of the developed classifiers showed that the inherent unreliability of servers used to host C2 domains could be used as a metric for reliably detecting Fast-Flux domains. Botnet controllers are unable to fully control the uptime of C2 servers due to their lack of physical access to these hosts, therefore they make use of multiple servers to ensure that the botnet control infrastructure is more robust. Using multiple servers limits the ability of security researchers to shutdown the botnet infrastructure even better (Barsamian, 2009). The use of multiple servers to host the C2 infrastructure mimics the behaviour of legitimate Content Distribution Networks (CDNs), which use multiple servers to host content and these hosts are rapidly swapped out to allow for load balancing. During the analysis of Fast-Flux domain behaviour it was noted that multiple A records would be returned with each DNS query. These A records were observed to be from multiple IP ranges and ASNs. On average Fast-Flux domains were found to be spread across three IP ranges and three ASNs.

The analysis of Fast-Flux domains and legitimate domains in section 4.3 showed that Fast-Flux domains had distinct features that could be used to differentiate Fast-Flux domains from both standard and domains used by CDNs. The most noticeable difference between Fast-Flux domains and CDN domains was that the A records returned for Fast-Flux domain queries mapped back to multiple IP ranges, multiple ASNs and multiple countries, while the A records from a CDN regularly mapped back to a single IP range, single ASN and one country. These observations matched those of other researchers (Hu et al., 2011; Caglayan et al., 2009; Bilge, Kirda, Kruegel, and Balduzzi, 2011), who all noted the distributed nature of Fast-Flux domains as a defining feature.

The initial stages of research involved constructing a classifier based on the work done by Holz et al. (2008). This work had produced classification techniques capable of detecting Fast-Flux domains with a high degree of confidence. Initial testing on active Fast-Flux domains did not produce the same results as those stated in the past research, and no apparent means were available to replicate these results. Investigation into active botnets showed that this failure could largely be attributed to the constant changing nature of botnets (Hunt, 2010) and the changing techniques used to evade detection. The domains observed during this research period exhibited longer TTL's than those observed in earlier work with a mean TTL of 595 seconds, while Passerini, Paleari, Martignoni, and Bruschi (2008) identified a mean TTL of 291 seconds and the metrics used by Arbor ATLAS to classify Fast-Flux service networks identified a TTL below 900 seconds as indicative of a Fast-Flux domain (Nazario and Holz, 2008). The large variation in observed mean TTL times highlights the difficulty in creating an all-encompassing classifier capable of detecting all known and emerging Fast-Flux botnet domains. Furthermore, it was seen that not all Fast-Flux domains could be detected using a single query response, such was the case with the Hlux2 botnet where TTLs were set between zero and two seconds with only a single IP address being returned with each subsequent query. Despite it not being possible to classify these domains in a single query using the observed behaviour of other Fast-Flux domains, it was noted that a domain could still be identified with three DNS queries on average by combining the data collected from each query response. Though this detection of these domains did not occur within a single query, the low TTL on the DNS records meant that classifying a domain with a TTL of two seconds took, on average, eight seconds on average. It was observed that it was uncommon for legitimate domain records to display a TTL below ten seconds and thus it is proposed that domains with a TTL below this threshold are deemed highly suspicious. This is the same approach taken in (Nazario and Holz, 2008), where domains with a TTL of two seconds or lower are repeatedly queried to search for distinct replies that could be indicative of Fast-Flux

behaviour. The changing nature of Fast-Flux domains posed the problem of finding a means of classifying existing Fast-Flux domains with the ability to adjust to the changes in Fast-Flux networks over time. To this end the original Holz classifier was first modified to match current observations in Fast-Flux botnets and to be used as a reference point. The rule-based classifier was created to incorporate additional Fast-Flux features not present in the original Holz classifier or any other classifiers. The third classifier proposed was based on Bayesian statistics with the aim of creating a classifier capable of learning which features constitute a Fast-Flux domain and the ability to adjust to changing features over time.

The modified Holz classifier achieved a lower accuracy than the reported accuracy in the original Holz work (Holz et al., 2008), with the modified classifier achieving a 93% accuracy while the Holz classifier was reported to achieve a 99.98% accuracy. The accuracy of this classifier was, however, 4.6% better than the original Holz classifier when applied to our test dataset. It was further noted that this accuracy rate was achieved with a single DNS query as opposed to the Holz classifier which required two or more DNS queries to be effective. This modified classifier was able to identify Fast-Flux domains that resemble CDNs with a low number of false positives. The rule-based classifier was constructed with the aim of identifying domains that might be missed by the modified Holz classifier. It was hoped that by using the country in which the servers are located, it would introduce an additional detection metric, increasing the accuracy of the classifier and leading to fewer false positives. The accuracy of this classifier was, however, identical to that of the modified Holz classifier, though it identified a small number of Fast-Flux domains not identified by the modified Holz classifier. The difference in the true positive rates where, however, not significant enough to set the rule-based classifier apart as an improved alternative to the modified Holz classifier.

5.2.2 Legitimate Fast-Flux Domains

Legitimate domains may also make use of Fast-Flux as a defensive mechanism against denial of service attacks (Lua and Yow, 2011). The domain *webmoney.ru* has an Alexa (Alexa, 2012) rating of 574 globally and 23 in Russia and can be seen as a legitimate site. It can, however, be seen in Table 5.1 that the DNS query response for this domain matches the features of Fast-Flux domains. The servers hosting *webmoney.ru* are located in three different countries, are all from different IP ranges and belong to five different ASNs. To further complicate the classification of *webmoney.ru* as either legitimate or not,

Table 5.1: Observed Domains

		Properties				
Domain	Label	A Record	IP Ranges	ASNs	Country Codes	TTL
maxsidelnikov31.narod2.ru	FF	4	4	1	1	3000
yahoo.com	CDN	3	3	1	1	3000
megasuperzx.com	FF	4	4	4	4	300
webmoney.ru	S	5	5	5	3	6600

it has been documented that *webmoney.ru* is widely used as a payment system on the Internet underground or blackmarkets (Wehinger, 2011). Due to its popularity as a black-market payment system, it is highly probable that *webmoney.ru* is frequently subjected to denial-of-service attacks. Thus it is speculated that *webmoney.ru* has employed Fast-Flux techniques to help mitigate the dangers posed by denial of service attacks against it. The problem of correctly classifying *webmoney.ru* highlights how easy it is to misclassify domains and thus strategies for dealing with false positives needs to be developed. These are discussed further in the future works section in section 6.1.

The increase in distributed denial-of-service (DDoS) attacks has led to the emergence of services such as CloudFlare that use techniques similar to those employed by Fast-Flux service networks (CloudFlare, 2012). The main benefit of Fast-Flux is that it provides botmasters with a more robust infrastructure, as their service is spread across multiple networks, thus making shutting down the entire network more difficult. The ability to provide a robust service make Fast-Flux domains ideal for creating robust networks more resistant to DDoS attacks and thus have been employed as one of the strategies used by companies such as CloudFlare. It remains possible to differentiate between Fast-Flux domains used by botnets and Fast-Flux type domains used by CloudFlare, Amazon and other companies despite the numerous A records returned being spread across multiple network ranges. This is due to legitimate services having full control over the IP ranges available to them and these IP ranges are usually registered back to a single ASN and a single country. Furthermore, if doubt remains about a domain it is possible to perform more heavyweight analysis. This heavyweight analysis involves doing reverse DNS resolution on the A records returned to determine if they map to domains other than the Fast-Flux domain. Once it has been established which domains A records are associated with, a WHOIS query can be performed to determine the organisation or individual the domain is registered to. With legitimate services, such as CloudFlare and Amazon, all the domains queried will have the same registration information, while Fast-Flux domains will have different WHOIS entries. As the number of legitimate services using Fast-Flux type

architecture remains small, whitelists can be employed where known legitimate services are always classified as legitimate. The negative aspect of this strategy is the increased use of legitimate services such as CloudFlare by illegitimate parties. During the Lulzsec campaign the hackers used the Cloud Flare service to host their web domain to prevent DDoS attacks (Rashid, 2012). In response to this malware authors may start moving their C2 nodes into these cloud services (such as Amazon EC2), making automatic classification increasingly difficult and requiring manual verification of domains. This highlights that no matter how good automated classifiers are, there will always be a need for manual intervention in the border cases.

A problem was identified while researching Fast-Flux domains, as it was noted that numerous domains would be labelled as Fast-Flux by sources such as the ZeuS and SpyEye trackers (abuse.ch, 2011, 2012) and other domain blacklists (MalwareURL, 2011; Arbor Networks, 2012b) despite these domains not matching the characteristics of the average Fast-Flux domain. This largely occurred in cases where a domain would have numerous A records and a short TTL, however all the observed A records were from a single IP range and ASN, displaying identical behaviour to CDNs. Domains such as these were still used in the testing of the classifiers, though it was noted that they were frequently misclassified as not being Fast-Flux. An example of this can be seen in Table 5.1, where the domain *maxsidelnikov31.narod2.ru* had been listed as a Fast-Flux domain used by the Citadel botnet (abuse.ch, 2012) and at the time of writing had been active for four months. This domain has near identical features to the legitimate CDN used by *yahoo.com*. Furthermore, it can be seen that the domain's properties differ greatly from that of *megasuperzx.com* another Fast-Flux domain used by the Citadel botnet. These discrepancies in the definition of what constitutes a Fast-Flux domain and the similarities between Fast-Flux domains and CDNs makes it near impossible to create a classifier that could classify domains with complete accuracy, but the error margin could be kept to acceptable levels. The classifiers created manage to achieve a high accuracy rate while minimising the number of CDN and standard domains that were misclassified as Fast-Flux and minimised the number of Fast-Flux domains that were misclassified.

5.3 Spatial Autocorrelation

The geographic dispersion of C2 servers was investigated as a possible means of detecting Fast-Flux domains. It has been noted by numerous researchers that the servers used by botnet controllers to host their C2 infrastructure are widely dispersed geographically

(Caglayan et al., 2009). This wide geographic dispersion can be attributed to the lack of physical control botnet controllers have over infected hosts. Hosts used as C2 infrastructure often consist of hosts that have either been hacked or taken over by malware. This leaves the botnet controllers with access to the hosts but no control over the uptime, bandwidth available and other physical aspects of the hosts. It was hypothesised that being able to detect Fast-Flux domains from the servers geographic locations would create a hard to bypass detection method, as the botnet controllers are not able to physically manipulate the geographic locations of hosts. As botnet controllers cannot alter the location of the infected hosts and the difficulty of spoofing the geographic location of a hosts, botnet controllers would not be able to easily disguise their Fast-Flux domains as CDNs. Most CDNs consists of servers under the control of a single organisation, where the organisation is capable of controlling the geographic location of all the hosts in the CDN. Furthermore, one of the defining features of CDNs is that they are designed to speed up the distribution of content based on the geographic location of the host requesting the content (CloudFlare, 2012). An example of this is the caching of multimedia content hosted on YouTube: visitors based in South Africa will be directed to the Cape Town-based YouTube CDN and only be directed to CDNs in other locations if the content requested is not available on the South African CDN. Fast-Flux botnets do not display this same behaviour and the list of C2 servers returned by a DNS query will resolve to widely geographically dispersed hosts.

5.3.1 Observed Distribution Patterns

During the analysis of Fast-Flux and legitimate domains it was observed that Fast-Flux domains have a mean nearest neighbour distance of 5 000km, while legitimate domains are clustered together with a short distance between server locations as discussed in section 3.5. While it could be said that a simple rule could be constructed where domains with a mean nearest neighbour distance greater than 5000km are classified as Fast-Flux domains, this solution would not work effectively. This solution is ineffective due to a large number of actual Fast-Flux domains being missed, while numerous CDNs would be classified as Fast-Flux, particularly when a CDN has been set up to serve content to a specific country with numerous content servers distributed around the country. Thus spatial autocorrelation was proposed as a means of formally defining domains based on the clustering of servers. This wide dispersion can be seen in Figure 5.2 where each country is coloured according to the number of botnet C2 servers that were observed in that country. The botnet observed in this instance consisted of 21 194 observed hosts

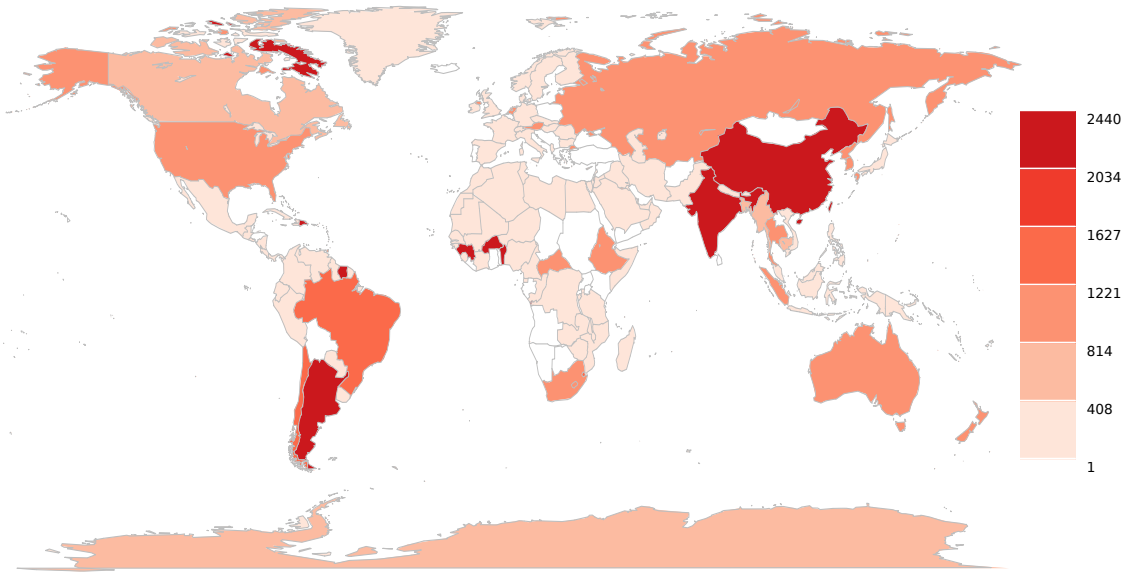


Figure 5.2: Geographic Distribution of Botnet C2 Server IP Addresses

(Arbor Networks, 2012b). As spatial autocorrelation was designed to identify distribution patterns in populations, such as disease outbreaks and animal species distribution Lea and Giffith (2001); Schabenberger and Gotway (2004); Cliff and Ord (1973), it was theorised that spatial autocorrelation would be capable of identifying whether domain servers are clustered or randomly distributed. Furthermore, as spatial autocorrelation is based on statistics, it would be possible to construct a system capable of learning as the distribution patterns of Fast-Flux domains change over time, thus creating a more robust system.

5.3.2 Moran's Index and Geary's Coefficient

Two methods of performing spatial autocorrelation were investigated: Moran's Index (MI) and Geary's Coefficient (GC), while both these methods are relatively similar they both have strengths and weaknesses. These two methods were tested using multiple means of quantifying the geographic location of servers to identify the method which provides the highest degree of accuracy, keeping the number of false positives to an absolute minimum. Work in identifying Fast-Flux domains using the geographic location of servers was done by (Huang et al., 2010). While this method proved accurate, numerous problems were identified. As noted by the authors, the main issue identified was that the detection technique relied on the timezones in which hosts were located. As timezones span both hemispheres and multiple countries, it was very possible that Fast-Flux domains with hosts far apart could be classified as legitimate if these hosts were all located in the same

Table 5.2: Observed Mean Values for Moran's Index

	Timezones	UTM	MGRS
Legitimate(N=964)	-0.0276	0.0609	-0.0536
Fast-Flux(N=305)	0.0930	1.1423	-0.1473

Table 5.3: Observed Mean Values for Geary's Coefficient

	Timezones	UTM	MGRS
Legitimate (N=964)	0.0460	0.0570	0.0803
Fast-Flux (N=305)	0.7681	0.5638	1.1160

timezone. To overcome this problem geographic co-ordinate systems other than timezones were investigated. These included the use of the UTM and MGRS co-ordinate systems. The results of the proposed classifiers were comparable with the previous work done by Huang et al. (2010) (accuracy 98%) with the classifiers using timezones as geographic measures achieving a 97% accuracy, while the FPR was lower at only 3% versus the other works 4% FPR. The results of the proposed classifiers did, however, remain consistent across datasets, where previous work showed varying results based on the test dataset used (accuracy = 98%, 96%, 92%). This again highlights the problem with attempting to fairly evaluate the accuracy of classifiers across different works, with the constantly changing nature of Fast-Flux domains making the dataset used as the time a factor in the results obtained.

Typically the values obtained for the MI are used to determine the type of correlation that exists between points, as discussed in subsection 3.6.1 the values for MI fall in the range [-1,1] where negative spatial autocorrelation is indicated by values [-1,0) with -1.0 indicating perfect negative spatial autocorrelation. Positive spatial correlation is indicated by values (0,+1], where 1.0 indicates perfect positive spatial correlation. A value of zero for MI represents a perfectly random spatial pattern. The mean MI values obtained for Fast-Flux and legitimate domains are shown in Table 5.2 for the three MI classifiers used.

It can be seen that the mean MI obtained for legitimate domains using timezones (-0.0276) is in the range [-1,0) which is used to indicate negative spatial autocorrelation, while the value for Fast-Flux domains (0.093) is in the range (0,1] indicating positive spatial autocorrelation. As negative spatial autocorrelation indicates that neighbouring values are dissimilar, it was predicted that Fast-Flux domains would display negative spatial autocorrelation due to hosts being widely dispersed. The MI value for Fast-Flux domains measured using UTM was 1.1423 which is greater than the range for MI values. This indicates positive spatial autocorrelation significant at the 5% level. Despite the values

obtained for the MI classifier being the opposite of those expected, they do not have an effect on the usability of the MI as a classifier for detecting Fast-Flux domains. The overall classifier performance is good with the classifier achieving a high accuracy of 97% with only 3% of domains being incorrectly classified. These results are achieved because the classifier does not use the type of spatial autocorrelation as a detection metric, but rather the output value of the MI classifier. As shown in subsection 4.4.1 and highlighted in Figure 4.7 and Figure 4.8 the MI values for legitimate and Fast-Flux domains are clustered accordingly and a clear boundary is distinguishable between the two clusters. The results from the classifiers clearly show that the use of geographic locations is an effective means of detecting Fast-Flux domains.

The values returned for GC are in the range [0,2] where values between [0,1) indicates positive spatial autocorrelation while values between (1,2] indicate negative spatial autocorrelation. A value of 1.0 for GC indicates no spatial autocorrelation. The mean observed values for GC are shown in Table 5.3, where it can be seen that the mean values for legitimate domains are all below 0.1 for all three spatial measures used, indicating positive spatial autocorrelation. This is as expected, with legitimate domains consisting of servers close together, all with similar values. The values approach zero, especially when using the timezone, indicative of perfect positive spatial autocorrelation. The mean GC value for Fast-Flux domains fall in the range [0,1) for both the timezone and UTM spatial measures, unexpectedly indicating positive spatial autocorrelation. These values are closer to 1. A perfectly random distribution is indicated by a value of 1, matching the prediction that Fast-Flux domains would show a random distribution of C2 server locations. The mean GC value for Fast-Flux domains when using MGRS as a spatial measure is 1.116 which indicates negative spatial autocorrelation. This value is again closer to 1, indicating a distribution pattern more random than negatively correlated. When rounding is applied to all the mean values, legitimate domains have a value of zero while Fast-Flux domains have a mean value of one, matching expectations that legitimate domains would show positive spatial autocorrelation and Fast-Flux domains would show either a perfectly random distribution or negative spatial autocorrelation. The overall classifier accuracy for the GC classifier is high at 96% with the MGRS based classifier detecting close to a 100% of all examined Fast-Flux domains.

5.3.3 Open Source Intelligence

The mapping of IP addresses to geographic locations relied on the Open Source Intelligence (OSINT) database offered by MaxMind (MaxMind, 2012) and was discussed in

detail in subsection 3.5.1. Though this database is comprehensive, the issue of missing data did arise. MaxMind claims to cover all of the IPv4 address space, though it was found that full location data was not available for a small number of addresses observed. It was found that while these addresses were present in the database, certain data points (such as the timezone) were missing from the database. The missing data points raised the problem of how in the case of missing timezone data these should be handled. The locations timezone was set to GMT+0. Setting these points to GMT+0 did not have a negative effect on any of the domains classified as there were enough other data points to use in the classification. It is, however, postulated that it is possible for a domain to exist that consists of IP addresses without full geographic information available. In this case an incorrect classification could occur. Another issue associated with the use of OSINT is the reliability of the data (Hulnick, 2002). Due to its well known nature and trusted reputation, the MaxMind database was specifically chosen to overcome the issue of reliability. This database has been used by multiple researchers when geolocation information needed to be mapped to IP addresses (Caglayan et al., 2009; Cremonini and Riccardi, 2009; Huang et al., 2010).

5.4 Performance Analysis

The performance of the classifiers was evaluated in section 4.5. Each classifier was tested in a simulated real world situation to determine the feasibility of using the classifiers on a live network. It was shown that the classifiers were able to rapidly classify domains without incurring additional network traffic. Furthermore, it was noted that the high performance of the classifiers was achieved using unoptimised code being run on a standard desktop environment.

5.4.1 Lexical Classifiers Performance

The lexical classifiers were able to quickly classify a large number of domains, as seen in subsection 4.5.2. The test scenario evaluated the performance of the lexical classifiers when used to process log files or blacklists. It was shown that a large log file could be rapidly processed and it could be determined if any network connections were made to algorithmically generated domain names. This would be a good indicator of a possible malware infection on the internal network. The benefit presented by this approach is that network logs can be used to identify hosts which may have a malware infection that has

not been detected by the on-host anti-virus solution. As the process of parsing the log files occurs quickly (250 000 log entries can be checked in under ten seconds) regular parsing of the network log files can be performed without the introduction of a long period between the parsing of the log files and a classification result being produced.

5.4.2 Fast-Flux Classifiers Performance

Performance of the DNS query response based classifiers was examined in subsection 4.5.3 and it was found that the proposed classifiers were capable of classifying domains in 0.65ms. A standard DNS query was measured taking 400ms to complete, indicating that the additional delay of performing classification would not be noticeable. The increase in time taken to complete the average DNS query was only 0.15% using unoptimised code and standard desktop computing power. Thus it is highly likely that implementing the proposed classifiers will not have a negative effect if deployed in a live network environment, once optimised and run on high end network servers. The similarity between the performance of the different classifiers indicates that the main delay in classification is the actual parsing of the DNS query response. It is postulated that by improving the DNS query response parsing routine, greater classifier performance could be achieved, which would have a lesser impact on overall performance.

Another area of concern is the time taken to do lookups for extra information not present in the DNS query response packet. Data such as geographic location and the ASN are queried from the MaxMind database located on the local system. Benchmarking by MaxMind has shown that the native C libraries are capable of 400 000 IP address lookups per second when memory caching is not used (MaxMind, 2012). The C implementation is capable of more than 1 million lookups when memory caching is used. This provides a good indication that the current solution could be dramatically improved to process more DNS query responses than is currently possible as the bottleneck is not the local database lookup but rather the actual parsing of the packet.

5.5 Real-World Application

It has been shown that the proposed classifiers are capable of detecting and classifying botnets that use algorithmically generated domain names and DNS Fast-Flux as detection avoidance techniques. It has furthermore been shown that the classifiers are lightweight

and have a minimal performance impact on current DNS query times. These classifiers can be adapted for real-world application with minimal effort. Two possible areas of application are network traffic log analysis and DNS traffic monitoring.

5.5.1 Log Analysis

The classifiers used to identify algorithmically generated domain names are suited for log analysis. It was shown in subsection 4.5.2 that these classifiers can process a large number of log entries in a short period of time. It was also shown that these classifiers are able to adapt to changing trends in algorithmically generated domain names and thus allows for earlier detection of emerging botnets. Furthermore, as the classifiers use probabilities to identify domains that match known algorithmically generated domain names, the need to maintain large blacklists is diminished. The large volume of generated names makes the maintenance and use of domain blacklists slow and cumbersome. As the classifiers do not require constant updating of blacklists, the detection is more robust and capable of detecting botnet domains that have not been seen before. This is evident in the case of Conficker-C, where attempting to either pre-register or blacklist 50 000 domains a day is a near impossible task. Once enough domains have been observed for a given DGA, it is possible to train the classifiers to specifically identify these domains. It is thus possible to have multiple classifiers each capable of identifying specific botnet domains, while a generalised classifier would be able to identify any domains that appear to be algorithmically generated. The low false positive rate also means that a weekly test of the classifiers against the top 100 000 domains, as listed by Alexa, would allow one to create a whitelist of known legitimate domains detected by the system and thus have exceptions for these domains if an attempt is made to access them.

The ability to rapidly parse log files and gain insight into possible malicious activity on the network is invaluable. The proposed classifiers offer both accuracy and performance benefits, making them suitable first pass classifiers. Traffic logs taken from network proxies such as Squid[3] and Websense[4] can be parsed offline using the proposed classifiers. Possible algorithmically generated domains may be identified for further manual inspection, helping speed up the detection of possible malware infections on the network. The classifiers may be trained using the traffic logs of the target network. This will ultimately in time lead to the detection of abnormal domain names in the network traffic logs.

[3] http://www.squid-cache.org/
[4] https://www.websense.com/content/microsoftproxyserver.aspx

5.5.2 DNS Traffic Monitoring

The proposed classifiers passively monitor DNS query responses to identify Fast-Flux botnet C2 server domains. As there is no direct interaction with the domains being queried the classifiers can be used as sensors anywhere on the network. The proposed classifiers can thus be used as another means of host based malware detection or be used for network based detection of malware on an organisation's network. It is proposed that the classifiers are placed between the network DNS server and the network hosts. This allows query responses to be monitored and classified before they are passed back to the relevant host. If a DNS query for a suspected DNS Fast-Flux domain is detected, the query response can be dropped, thus preventing the host from contacting the botnet C2 server. A further solution would be to generate a new DNS query response that will direct the hosts traffic to an address under the organisations control. This solution may be used in conjunction with a honeypot such as Dionaea (Levy, 2005; Baecher, Koetter, Holz, Dornseif, and Freiling, 2006), where suspected botnet traffic can be monitored and analysed.

As the Moran's Index and Geary's Coefficient classifiers retrieve the geographic location information about the domain servers, patterns in the geographic locations of domains visited by hosts on the network may be created. These patterns can be used to detect abnormal behaviour on the network. An example of this is a network where traffic patterns have shown that the majority of domains visited are located in Germany, a sudden increase in traffic to domains hosted in China could indicate a breach of the network or hosts infected with malware.

5.6 Summary

The results from chapter 4 clearly show that the techniques described in this thesis can accurately identify botnet domains from the contents of DNS query responses. The techniques are able to identify botnet domains that use both DGAs and Fast-Flux to avoid detection. Furthermore, the techniques outlined make it difficult for botnet controllers to bypass these detection techniques and when used in conjunction with existing techniques a robust anti-botnet system can be developed. The techniques outlined rely on the contents of DNS query responses which are already present on the network. This means no additional network traffic needs to be generated for detection to be possible. The classifiers are capable of identifying botnet domains that have not been seen before and

thus provide an added benefit over blacklists and whitelists. Blocking of botnet domains through the use of blacklists and whitelists introduces a need for on operators to keep these lists up to date. The proposed classifiers are adaptable and easily modified, lending themselves to numerous possible applications in network traffic monitoring and malicious activity detection.

6 Conclusion

This document detailed classification techniques for DNS based detection of botnet domains. These classification techniques were divided into three sub-fields focused on detecting specific evasion techniques employed by modern botnet families. These classifiers can be summarised as follows.

- The detection of algorithmically generated domain names such as those used by the Torpig, Kraken and Conficker botnets. Multiple classifiers capable of detecting these algorithmically generated domains were developed. The classifiers were based on the analysis of the frequency distribution of characters contained in domain names.
- Detecting Fast-Flux domains used to mask C2 servers by using the information contained in DNS query response. Classifiers were developed using the distinct characteristics of Fast-Flux domains to differentiate between Fast-Flux domains and legitimate domains such as CDNs.
- Finally the geographic location of servers was used to detect Fast-Flux domains. The classifiers developed relied on Fast-Flux domains consisting of widely geographically dispersed hosts and formally defined is geographic dispersion, resulting in the classification of domains as either Fast-Flux or legitimate.

The proposed techniques were evaluated to determine the feasibility of detecting botnet traffic solely from the contents of a DNS query response. The proposed techniques were tested against current active botnet domains which were using detection evasion techniques. Results from the evaluation of these techniques showed that it is possible to detect botnet domains with a high degree of accuracy based on the contents of DNS query response packets.

Theoretical background information of the Domain Name System (DNS) was provided in chapter 2 along with information regarding modern botnets. The functioning of DNS was explained, with examples of the contents of DNS query responses, identifying the sections pertinent to this document. The structure of botnets was discussed, making the reader aware of the different components of a botnet and how these components function together as a system. Common detection evasion techniques, such as algorithmically generated domain names and DNS Fast-Flux, were described, focusing on the defining features of these evasion techniques and how they could be used to identify botnet domains. Finally related works in the field of botnet detection were discussed, outlining how these works could be extended and improved. Spam filtering techniques and spatial autocorrelation techniques were discussed as non-botnet related works which could be altered to be applicable to botnet detection.

In chapter 3 the data used in this document was introduced, the sources of this data and how the data was sanitised for use with the proposed classifiers. The rest of the chapter discussed the techniques used in the construction of the classifiers used for botnet detection. The statistical foundations of lexical analysis were outlined along with the techniques used to calculate the probabilities of domains being algorithmically generated. An existing Fast-Flux detection technique was discussed, as well as how it was modified to match changes in Fast-Flux botnet structures. This modified classifier was extended into a new rule-based classifier that introduced additional data points to increase accuracy and to aid in differentiating between Fast-Flux and Content Distribution Network domains. Finally a Naive Bayesian classification technique was presented, allowing for the construction of robust and self-learning classifiers, capable of adapting to the changing nature of Fast-Flux domains. The observed geographic distribution of botnet C2 servers was discussed in the final part of the chapter. Multiple means of measuring the geographic location of hosts were described and used as inputs to the proposed classifiers. Techniques taken from plant and animal distribution statistics were modified for use with Fast-Flux botnet domains. The identified techniques allowed for the formal definition of the distribution patterns of C2 servers and could be used to differentiate between Fast-Flux and legitimate domains.

Having discussed the techniques used to construct botnet detection classifiers, testing was performed to evaluate how well these classifiers would function when exposed to real world samples. The results from these tests were presented in chapter 4. It was shown that all the classifiers performed remarkably well in detecting botnet domains, with high accuracy rates and a minimum number of false positives. The results showed that statistics-based, self-learning classifiers were feasible and performed better than manual classifiers. These

results were discussed in detail in chapter 5, where possible weaknesses were identified and their counters were discussed. The benefits of self-learning classifiers over manual blacklists were discussed and it was shown that the development of self-learning classifiers should be explored further. The DNS features most relevant to botnet domain detection were identified as an area of future research.

This document showed that it is possible to identify botnet domains by examining the features of DNS query responses. It was shown that using DNS in classifiers allowed for early detection of botnet traffic and thus allows for preventing communication disruption between botmasters and bots. The performance analysis showed that the classifiers are lightweight and introduce a small performance overhead, as low as 0.152%.

6.1 Future Work

The proposed classifiers have been shown to work well in an academic and well-structured environment. Future work will be aimed at providing a thorough evaluation of the classifiers on a real world network. The proposed classifiers only presented a means for detecting botnet domains, however no methods for reaction or remediation are presented. Thus it is proposed that future work should include an analysis of possible reactive steps that can be taken once a botnet domain has been detected. Furthermore, the proposed techniques may be applied to other areas of botnet defence, such as Distributed Denial-of-Service (DDoS) attack detection.

6.1.1 Anti-Malware Protection

Mitigation and remediation of botnet infections are an active area of research that could be applied to the proposed classifiers. Deploying the classifiers on a network will allow for early detection of malware infections when hosts attempt to contact domains associated with botnets. Infected hosts may perform a DNS query to enable communication with the botnet C2 domains. Numerous possible reactive steps exist at this point and include.

- Sinkhole - if a DNS query in association with a botnet domain is detected, the DNS server can return a modified DNS query response which directs the host to a sinkhole or null routed address. This will prevent the infected host from establishing communication with the C2 servers, while also allowing researchers to examine the

communication protocol of the botnet (Asrigo, Litty, and Lie, 2006). To enable protocol examination, infected hosts can be directed to a honeypot system such as Dionaea (Levy, 2005).

- Domain Registration Scanning - it has been shown that proactive scanning of domain registrations may form an effective means of detecting phishing domains before an actual phishing campaign is launched (Marchal, François, State, and Engel, 2012). A similar approach is proposed for the detection of Fast-Flux and algorithmically generated domain names, where new domain registrations are scanned on a daily basis to identify suspicious domains. The domains identified during this proactive scanning phase may then be incorporated into domain blacklists.

6.1.2 DDoS Detection

The spatial autocorrelation techniques proposed could be adapted to aid in the detection of DDoS. As DDoS attacks rely on large numbers of widely distributed hosts to attack a central location. Legitimate traffic should show clustering according to geographic region or timezone (Lamm, Reed, and Scullin, 1996; Padmanabhan and Subramanian, 2001). Thus it should be possible to differentiate between legitimate web traffic and DDoS traffic based on the type of spatial clustering observed. By using spatial autocorrelation, DDoS traffic could be identified earlier, before a domain has been severely affected by the large traffic volumes.

References

abuse.ch. Spyeye monitor. https://spyeyetracker.abuse.ch/monitor.php?filter=level5, March 2011. Last Accessed: 21 September 2011.

abuse.ch. Zeus tracker. https://zeustracker.abuse.ch/monitor.php?filter=level5, February 2012. Last Accessed: 5 November 2012.

R. Aitchison. *Pro DNS and BIND 10*. Apress, Berkely, CA, USA, 2011. ISBN 978-1-4302-3048-9.

Alexa. Alexa top sites. http://www.alexa.com/topsites, March 2012. Last Accessed: 5 March 2012.

Alienvault Labs. Detecting malware domains by syntax heuristics. http://labs.alienvault.com/labs/index.php/2012/detecting-malware-domains-by-syntax-heuristics, February 2012. Last Accessed: 10 February 2012.

I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, K. V. Ch, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. pages 9–17, 2000.

Arbor Networks. Anatomy of a botnet. Whitepaper, Arbor Networks, 2012a. Available at http://www.arbornetworks.com/component/docman/doc_download/494-anatomy-of-a-botnet-white-paper?Itemid=442, Last Accessed: 12 October 2012.

Arbor Networks. Atlas global fast flux report. https://atlas.arbor.net/summary/fastflux, March 2012b. Last Accessed: 18 November 2012.

K. Asrigo, L. Litty, and D. Lie. Using VMM-based sensors to monitor honeypots. In *Proceedings of the 2nd international conference on Virtual execution environments*, VEE '06, pages 13–23. ACM, 2006.

P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes platform: An efficient approach to collect malware. In *Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

D. Barr. RFC 1912: Common DNS operational configuration errors. `http://www.ietf.org/rfc/rfc1912.txt`, February 1996. Last Accessed: 14 October 2012.

A. Barsamian. *Network Characterization for Botnet Detection Using Statistical-behaviour Methods*. PhD thesis, Dartmouth College, 2009.

L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *NDSS*. The Internet Society, 2011. URL `http://dblp.uni-trier.de/db/conf/ndss/ndss2011.html#BilgeKKB11`.

A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, and G. Eaton. Behavioral analysis of fast flux service networks. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, CSIIRW '09, pages 48:1–48:4, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-518-5. doi: 10.1145/1558607.1558662. URL `http://doi.acm.org/10.1145/1558607.1558662`.

A. Cliff and K. Ord. *Spatial autocorrelation*. Pion, London, England, 1973.

CloudFlare. CloudFlare Overview. `http://www.cloudflare.com/overview`, September 2012. Last Accessed: 25 Semptember 2012.

L. Colitti, S. Gunderson, E. Kline, and T. Refice. Evaluating ipv6 adoption in the internet. In A. Krishnamurthy and B. Plattner, editors, *Passive and Active Measurement*, volume 6032 of *Lecture Notes in Computer Science*, pages 141–150. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-12333-7. URL `http://dx.doi.org/10.1007/978-3-642-12334-4_15`.

Conficker Working Group. Conficker working group: Lessons learned. `http://www.confickerworkinggroup.org/wiki/uploads/Conficker_Working_Group_Lessons_Learned_17_June_2010_final.pdf`, June 2010. Last Accessed: 7 August 2012.

H. Crawford and J. Aycock. Kwyjibo: automatic domain name generation. *Software: Practice and Experience*, 38(14):1561–1567, 2008.

M. Cremonini and M. Riccardi. The Dorothy project: An open botnet analysis framework for automatic tracking and activity visualization. In *Computer Network Defense (EC2ND), 2009 European Conference on*, pages 52–54. IEEE, 2009.

Damballa. DGAs in the hands of cyber-criminals. `https://www.damballa.com/downloads/r_pubs/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf`, February 2012. Last Accessed: 12 November 2012.

Dragon Research Group. Team cymru. `http://www.team-cymru.org/Services/ip-to-asn.html`, March 2011. Last Accessed: 15 March 2011.

W. Ehm. Binomial approximation to the poisson binomial distribution. *Statistics & Probability Letters*, 11(1):7–16, 1991.

N. Fitzgibbon and M. Wood. Conficker. C: A technical analysis. *SophosLabs, Sophon Inc*, April 2009. Available at: `http://www.sophos.com/sophos/docs/eng/marketing_material/conficker-analysis.pdf`, Last Accessed: 13 October 2012.

M. Garnaeva. Kelihos/Hlux botnet returns with new techniques. `https://www.securelist.com/en/blog/655/Kelihos_Hlux_botnet_returns_with_new_techniques`, January 2012. Last Accessed: 1 February 2012.

M. Gasser. A random word generator for pronounceable passwords, mtr-3006. Technical report, ESD-TR-75-97, AD-A017676, MITRE Corp., Bedford, Mass, 1975.

Google. Top 1000 sites - doubleclick ad planner. `http://www.google.com/adplanner/static/top1000/`, April 2012. Last Accessed: 12 April 2012.

G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *MALWARE 2008. 3rd International Conference on Malicious and Unwanted Software, 2008*, pages 24 – 31, 2008.

T. Hostert. Military map reading 201. Technical report, National Geospatial-Intelligence Agency, 1997. Available at `http://earth-info.nga.mil/GandG/coordsys/mmr201.pdf`, Last Accessed: 12 October 2012.

hostip.info. About the database. http://www.hostip.info/about.html, September 2012. Last Accessed: 18 September 2012.

X. Hu, M. Knysz, and K. Shin. Measurement and analysis of global IP-usage patterns of fast-flux botnets. In *INFOCOM, 2011 Proceedings IEEE*, pages 2633–2641. IEEE, 2011.

S. Huang, C. Mao, and H. Lee. Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection. In D. Feng, D. A. Basin, and P. Liu, editors, *ASIACCS*, pages 101–111. ACM, 2010. ISBN 978-1-60558-936-7.

A. S. Hulnick. The downside of Open Source Intelligence. *International Journal of Intelligence and CounterIntelligence*, 15(4):565–579, 2002.

A. Hunt. Visualizing the hosting patterns of modern cybercriminals. http://pen-testing.sans.org/resources/papers/gcih/visualizing-hosting-patterns-modern-cybercriminals-117542, September 2010. Last Accessed: 18 September 2012.

S. E. Lamm, D. A. Reed, and W. H. Scullin. Real-time geographic visualization of world wide web traffic. In *In Proceedings of Fifth Internation World Wide Web Conference (WWW5)*, pages 1457–1468, 1996.

T. Lea and D. Giffith. Opposites don't attract in spatial autocorrelation. *GEO-World*, (14):42–44, November 2001.

F. Leder and T. Werner. Know your enemy: Containing conficker. http://www.honeynet.org/files/KYE-Conficker.pdf, April 2009. Last Accessed: 14 September 2012.

J. Lee, H. Jeong, J. Park, M. Kim, and B. Noh. The activity analysis of malicious http-based botnets using degree of periodic repeatability. In *Security Technology, 2008. SECTECH'08. International Conference on*, pages 83–86. IEEE, 2008.

S. Lee and J. Kim. Fluxing botnet command and control channels with URL shortening services. *Computer Communications*, 2012. ISSN 0140-3664. doi: 10.1016/j.comcom.2012.10.003.

E. Levy. Dionaea: On the automatic collection of malicious code samples through honey pot farms. Invited talk at the CASCON 2005 Workshop on Cybersecurity, 2005.

R. Lua and K. C. Yow. Mitigating ddos attacks with transparent and intelligent fast-flux swarm network. *IEEE Network*, 25(4):28–33, 2011.

J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. pages 1245–1254, 2009. doi: 10.1145/1557019.1557153. URL `http://doi.acm.org/10.1145/1557019.1557153`.

I. MacKenzie and R. Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 754–755. ACM, 2003.

MalwareURL. Malwareurl. `http://www.malwareurl.com/`, April 2011. Last Accessed: 5 April 2011.

S. Marchal, J. François, R. State, and T. Engel. Proactive discovery of phishing related domain names. In D. Balzarotti, S. Stolfo, and M. Cova, editors, *Research in Attacks, Intrusions, and Defenses*, volume 7462, pages 190–209. Springer Berlin / Heidelberg, 2012.

MaxMind. Maxmind geoip. `http://www.maxmind.com/app/ip-location`, February 2012. Last Accessed: 18 September 2012.

Microsoft Corporation. Microsoft and financial services industry leaders target cybercriminal operations from zeus botnets. `https://blogs.technet.com/b/microsoft%5C_blog/archive/2012/03/25/microsoft-and-financial-services-industry-leaders-target-cybercriminal-operations.aspx`, March 2012. Last Accessed: 2 April 2012.

P. Mockapetris. RFC 1035: Domain names - implementation and specification. `http://tools.ietf.org/html/rfc1035`, November 1987. Last Accessed: 12 October 2012.

J. A. Morales, A. Al-Bataineh, and R. Sandhu. Analyzing DNS activities of bot processes. In *2009 4th International Conference on Malicious and Unwanted Software (MALWARE)*, pages 98–103. IEEE, October 2009.

P. A. P. Moran. Notes on Continuous Stochastic Phenomena. *Biometrika*, 37(1/2):17–23, June 1950. ISSN 00063444. doi: 10.2307/2332142.

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

A. Nottingham. GPF: A framework for the general packet classification on GPU co-processors. Master's thesis, Rhodes University, Grahamstown, South Africa, October 2011.

G. Ollmann. The evolution of commercial malware development kits and colour-by-numbers custom malware. *Computer Fraud & Security*, 2008(9):4–7, September 2008.

Oxford University Press. Oxford english dictionary online, 2nd edition. `http://www.oed.com/`, July 2011. Last Accessed: 5 July 2012.

V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '01, pages 173–185, New York, NY, USA, 2001. ACM.

E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. Fluxor : Detecting and monitoring fast-flux service networks. In *Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 186–206, Berlin, Heidelberg, 2008. Springer-Verlag.

R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting malicious flux service networks through passive analysis of recursive DNS traces. *2009 Annual Computer Security Applications Conference*, pages 311–320, December 2009.

V. Pereira, A. Fucs, and A. P. de Barros. New botnets trends and threats. Whitepaper, 2007. Available at `http://www.blackhat.com/presentations/bh-europe-07/Fucs-Paes-de-Barros-Pereira/Whitepaper/bh-eu-07-barros-WP.pdf`.

P. Porras, H. Saidi, and V. Yegneswaran. Conficker c analysis. *SRI International*, March 2009.

F. Y. Rashid. Cloud computing - RSA 2012: LulzSec as a CloudFlare customer - eWeek Security Watch. `http://www.eweek.com/security-watch/rsa-2012-lulzsec-as-a-cloudflare-customer.html`, February 2012. Last Accessed: 21 October 2012.

C. C. Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64 (1):38–40, 1957.

P. Royal. On the Kraken and Bobax botnets. `http://bandwidthco.com/whitepapers/compforensics/malware/bots/On%20the%20Kraken%20and%20BoBax%20Botnets.pdf`, April 2008. Last Accessed: 5 October 2012.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin: AAAI Technical Report WS-98-05, 1998.

O. Schabenberger and C. A. Gotway. *Statistical methods for spatial data analysis*, volume 64. Chapman and Hall, 1 edition, December 2004.

A. Seewald. An evaluation of Naive Bayes variants in content-based learning for spam filtering. *Intelligent Data Analysis*, 11(5):497–524, October 2007.

Shadowserver Organisation. Botnet statistics. `http://www.shadowserver.org/wiki/mpwiki.php/Stats/BotnetCharts`, February 2012. Last Accessed: 8 March 2012.

S. Silva, R. Silva, R. Pinto, and R. Salles. Botnets: A survey. *Computer Networks*, 2012. ISSN 1389-1286. doi: 10.1016/j.comnet.2012.07.021. URL `http://www.sciencedirect.com/science/article/pii/S1389128612003568`.

E. Stalmans, S. Hunter, and B. Irwin. Geo-spatial autocorrelation as a metric for the detection of Fast-Flux botnet domains. In *Information Security for South Africa (ISSA), 2012*, pages 1–7. IEEE, 2012.

T. Tan, C. Seng, J. Tan, K. Leong, D. De Silva, K. Lim, E. Tay, S. Subbiah, et al. Multi-language domain name service, Sept. 3 2002. US Patent 6,446,133.

F. L. E. G.-P. Thomas Barabosch, Andre Wichmann. Automatic extraction of domain name generation algorithms from current malware. In *NATO Symposium IST-111 on Information Assurance and Cyber Defence*, 2012.

W. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46(2):234–240, 1970.

Unmask Parasites. Twitter API still attracts hackers. `http://blog.unmaskparasites.com/2009/12/09/twitter-api-still-attracts-hackers/`, December 2009. Last Accessed: 21 October 2012.

U.S. Geological Survey. The universal transverse mercator (utm) grid. `http://egsc.usgs.gov/isb/pubs/factsheets/fs07701.html`, September 2012. Last Accessed: 18 September 2012.

F. Vanier. World broadband statistics: Short report Q4 2010, 2011. `broadband.cti.gr/el/download/broadbandshortreport2010.pdf`, Last Accessed: 29 October 2012.

F. Wehinger. The dark net: Self-regulation dynamics of illegal online markets for identities and related services. In *Proceedings of the 2011 European Intelligence and Security Informatics Conference*, EISIC '11, pages 209–213, Washington, DC, USA, 2011. IEEE Computer Society.

C. Wilson. Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress. Technical Report RL32114, CRS Report for Congress, January 2008.

C. Xiang, F. Binxing, Y. Lihua, L. Xiaoyi, and Z. Tianning. Andbot: towards advanced mobile botnets. In *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, LEET'11. USENIX Association, 2011.

Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: signatures and characteristics. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 171–182. ACM, 2008.

S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting algorithmically generated malicious domain names. In *Proceedings of the 10th annual conference on Internet measurement - IMC '10*, page 48, New York, New York, USA, November 2010.

L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. 3 (4):243–269, Dec. 2004. ISSN 1530-0226. doi: 10.1145/1039621.1039625.

Glossary

ACC Accuracy

ASN Autonomous System Number

Bot Host belonging to a botnet

Botmaster Person or group in control of a botnet

C2 Command and Control Server of a botnet

CDN Content Distribution Network

DDoS Distributed Denial of Service

DGA Domain Generation Algorithm

DNS Domain Name System

FN False Negative

FP False Positive

FPR False Positive Rate

GC Geary's Coefficient

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

IRC Internet Relay Chat

MGRS Military Grid Reference System

MI Moran's Index

MiTM Man in the Middle

OSINT Open Source Intelligence

TTL Time to Live

TN True Negative

TP True Positive

TPR True Positive Rate

URL Universal Resource Locator, the address of a World Wide Web page

UTM Universal Transverse Mercator

A

Fast-Flux Domain Query Responses

A sample of DNS query responses for Fast-Flux domains listed by Arbor Networks[1] between January 2012 and November 2012. Output obtained using the dig[2] tool.

```
;; QUESTION SECTION:
;allamur.info. IN A
;; ANSWER SECTION:
allamur.info. 600 IN A 82.131.50.191
allamur.info. 600 IN A 174.54.63.133
allamur.info. 600 IN A 85.180.225.47
allamur.info. 600 IN A 219.19.188.152
allamur.info. 600 IN A 24.113.216.70
;; AUTHORITY SECTION:
allamur.info. 86400 IN NS ns1.vseprokote.info.
allamur.info. 86400 IN NS ns2.vseprokote.info.
allamur.info. 86400 IN NS ns3.vseprokote.info.
allamur.info. 86400 IN NS ns4.vseprokote.info.
allamur.info. 86400 IN NS ns5.vseprokote.info.
;; Query time: 50 msec
;; SERVER: 109.74.193.20#53(109.74.193.20)
;; WHEN: Sat Oct 13 10:19:13 2012
;; MSG SIZE rcvd: 211
```

[1] http://atlas.arbor.net/summary/fastflux
[2] dig (domain information groper) http://linux.die.net/man/1/dig

```
;; QUESTION SECTION:
;residencelovers.com. IN A
;; ANSWER SECTION:
residencelovers.com. 600 IN A 87.247.33.207
residencelovers.com. 600 IN A 95.57.246.222
residencelovers.com. 600 IN A 98.212.50.228
residencelovers.com. 600 IN A 186.156.9.36
residencelovers.com. 600 IN A 37.229.153.244
;; AUTHORITY SECTION:
residencelovers.com. 86400 IN NS ns1.teachfashion.com.
residencelovers.com. 86400 IN NS ns2.teachfashion.com.
residencelovers.com. 86400 IN NS ns3.teachfashion.com.
residencelovers.com. 86400 IN NS ns4.teachfashion.com.
residencelovers.com. 86400 IN NS ns5.teachfashion.com.
;; Query time: 284 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sun Nov 18 23:16:10 2012
;; MSG SIZE rcvd: 220
```

```
;; QUESTION SECTION:
;fitoteafclope.pl. IN A
;; ANSWER SECTION:
fitoteafclope.pl. 300 IN A 189.8.252.12
fitoteafclope.pl. 300 IN A 221.13.79.26
fitoteafclope.pl. 300 IN A 119.60.6.254
;; AUTHORITY SECTION:
fitoteafclope.pl. 300 IN NS ns1.luisianacarwash.pl.
fitoteafclope.pl. 300 IN NS ns1.crisscrossingamendment.com.
;; Query time: 608 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sun Nov 18 23:13:36 2012
;; MSG SIZE rcvd: 160
```

```
;; QUESTION SECTION:
;funnypoets.com. IN A
;; ANSWER SECTION:
funnypoets.com. 90 IN A 107.22.28.131
funnypoets.com. 90 IN A 107.22.103.111
funnypoets.com. 90 IN A 184.72.148.193
funnypoets.com. 90 IN A 23.22.99.144
funnypoets.com. 90 IN A 72.44.43.106
;; AUTHORITY SECTION:
funnypoets.com. 14400 IN NS ns1.scalr.net.
funnypoets.com. 14400 IN NS ns2.scalr.net.
funnypoets.com. 14400 IN NS ns4.scalr.net.
funnypoets.com. 14400 IN NS ns3.scalr.net.
;; Query time: 236 msec
;; SERVER: 109.74.193.20#53(109.74.193.20)
;; WHEN: Sat Oct 13 10:20:03 2012
;; MSG SIZE rcvd: 193
```

```
;; QUESTION SECTION:
;storuofginezi.com. IN A
;; ANSWER SECTION:
storuofginezi.com. 300 IN A 83.69.139.19
storuofginezi.com. 300 IN A 123.178.150.174
storuofginezi.com. 300 IN A 99.88.223.211
storuofginezi.com. 300 IN A 189.8.252.12
;; AUTHORITY SECTION:
storuofginezi.com. 300 IN NS ns1.satsun-weekends.pl.
storuofginezi.com. 300 IN NS ns1.steppinglegalzoom.com.
;; Query time: 205 msec
;; SERVER: 109.74.193.20#53(109.74.193.20)
;; WHEN: Sat Oct 13 10:22:14 2012
;; MSG SIZE rcvd: 171
```

B

Algorithmically Generated Domain Names

A sample of algorithmically generated domains for Conficker-C in April 2009 (Table B.1).

Table B.1: Conficker-C Domains

fpkxd.tn	bupk.tl	dsjd.pe
acklirhal.co.za	clgxssep.com.pa	wlltihffk.ac
gtzo.com.tr	frrup.co.za	aini.tn
ptoww.dj	ruqp.com.co	xpoj.bz
uolodcz.dk	izeo.pk	umoee.co.cr
hcxrjl.com.gl	aqhb.lv	dprcz.com.uy
ossuw.com.gl	sdis.com.ua	aluolzo.to
yxqoc.cx	tlmozybi.tc	pwjglcfb.la
mzeqmimuk.pe	qxmdmx.mn	sfvrt.ly
hcqdy.com.jm	ywsienzog.md	bnlimlhzf.lu
sbwb.ae	mxzuxiehu.com.gt	yxbrzbs.kn
prauvdjv.ch	ptihuuel.pk	xlubugj.co.za
hbhutqj.ly	qemhvphpn.com.tr	juanfhuhx.ec
gzmw.dk	lkjsfdt.ly	vpdwsv.dj

Samples from the domain names generated by the Kraken (Table B.2) and Bobax (Table B.3) botnets, samples taken from Royal (2008).

Table B.2: Kraken Domains

bjjdhgpby.dyndns.org	ezqqddbkx.yi.org	ulssrxrzu.dynserv.com
bnbnpqkagr.dyndns.org	fcnhysydw.yi.org	uokvzojl.dynserv.com
bqzzqwwi.dyndns.org	gtyeywobh.yi.org	ucxibbeenwz.dynserv.com
bvvliba.dyndns.org	iogrdedv.yi.org	iikctrpa.dynserv.com
cipaxqmcgfz.dyndns.org	kpxvrvde.yi.org	cazrsihs.dynserv.com
cvvhgffch.dyndns.org	kpxvrvdefs.yi.org	tvzggexcvfv.dynserv.com
dkflxkqecdf.dyndns.org	orugtuapnzu.yi.org	orhsnoiv.dynserv.com
dmaciltbek.dyndns.org	ospknhemqt.yi.org	xoskcy.dynserv.com
doqsstt.dyndns.org	rdjqleu.yi.org	koaqnn.dynserv.com
dqovzm.dyndns.org	tapdcm.yi.org	ddrqyggw.dynserv.com
dvguqvob.dyndns.org	yeaigapqs.yi.org	bodrxb.dynserv.com
dztxvpt.dyndns.org	znvibonyf.yi.org	jpbytzo.dynserv.com

Table B.3: Bobax Domains

dlivmg.1dumb.com	eniaaknrxb.3-a.net	ipbjty.afraid.org
eivysjix.1dumb.com	gxjitrjifgp.3-a.net	jqevnl.afraid.org
fndvrix.1dumb.com	ihhyzby.3-a.net	mhnyavmf.afraid.org
glilepv.1dumb.com	imtoey.3-a.net	gypzmaudtlv.hn.org
kvuznwxmfoj.1dumb.com	ksfvgfrf.3-a.net	ichyig.hn.org
qeqfsvxousx.1dumb.com	kyfabyzf.3-a.net	ipurfbqpsdj.hn.org
rjjuyi.1dumb.com	mcduii.3-a.net	nttstzi.hn.org
vfpqyv.1dumb.com	mlxvdl.3-a.net	nttstziinpa.hn.org
wbghid.1dumb.com	neytteybbo.3-a.net	tsyunetwmi.hn.org
aazuxmmqqkq.3-a.net	qstffsupgu.3-a.net	xatzjf.hn.org
amjcud.3-a.net	ryhszzinxss.3-a.net	fcnhysydw.yi.org
cnntzas.3-a.net	bhlnklify.afraid.org	kpxvrvde.yi.org

C Geographic Distribution of C2 Servers

Table C.1: Geographic Distribution of C2 Servers

DOMAIN	IP ADDRESS	LATITUDE	LONGITUDE	COUNTRY
0dnotraxniki.ru	188.32.46.167	55.7522	37.6156	RU
0dnotraxniki.ru	117.199.233.10	17.3753	78.4744	IN
0dnotraxniki.ru	86.122.154.165	44.3167	23.8	RO
uthdrugs.com	78.229.170.28	49.4167	2.8333	FR
uthdrugs.com	117.215.148.207	28.6	77.2	IN
uthdrugs.com	84.109.168.59	32.0114	34.7722	IL
uthdrugs.com	202.88.76.10	15.1819	145.7567	MP
uthdrugs.com	58.94.154.22	35.685	139.7514	JP
uthdrugs.com	175.136.74.201	2.5	112.5	MY
warpills.ru	99.245.97.167	43.75	-79.2	CA
warpills.ru	113.162.74.179	16.0	106.0	VN
warpills.ru	116.71.58.158	24.8667	67.05	PK
warpills.ru	122.168.108.212	22.0833	79.5333	IN
warpills.ru	189.157.102.251	9.43421	-99.1386	MX
waydontsupface.com	4.97.38.46	24.6408	46.7728	SA
waydontsupface.com	220.50.10.127	35.685	139.7514	JP
waydontsupface.com	81.100.211.250	51.5002	-0.1262	GB
zildoctor.ru	115.64.35.231	-33.9667	151.1	AU
zildoctor.ru	183.82.242.39	20.0	77.0	IN
zildoctor.ru	19.155.28.90	33.7	73.1667	PK
zildoctor.ru	77.250.226.213	52.3667	5.15	NL

Table C.1 lists the geographic distribution of C2 servers for the Citadel botnet (abuse.ch, 2012) and a Fast-Flux botnet monitored by Arbor Networks in October 2012.

D

Published Works

E. Stalmans and B. Irwin. A framework for DNS based detection and mitigation of malware infections on a network. In *Information Security for South Africa (ISSA), 2011*, pages 1–8. IEEE, 2011. Available at: IEEE Explore Digital Library

E. Stalmans and B. Irwin. A framework for DNS based detection of botnets at an ISP level. In *South African Telecommunication Networks and Application Conference (SATNAC), 2011.* Available at: www.satnac.org.za

E. Stalmans and B. Irwin. An exploratory framework for extrusion detection. In *South African Telecommunication Networks and Application Conference (SATNAC), 2012.* Available at: www.satnac.org.za

E. Stalmans, B. Irwin and S. Hunter. Geo-spatial autocorrelation as a metric for the detection of Fast-Flux botnet domains. In *Information Security for South Africa (ISSA), 2012*, pages 1–7. IEEE, 2012. Available at: IEEE Explore Digital Library