Impact Study of Length in Detecting Algorithmically Generated Domains

by

Aashna Ahluwalia

B.Eng, University of Mumbai, 2015

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Applied Science

in the Department of Electrical and Computer Engineering

Supervisory Committee

Impact Study of Length in Detecting Algorithmically Generated Domains

by

Aashna Ahluwalia

B.Eng, University of Mumbai, 2015

Supervisory Committee

Dr. Issa Traore, Department of Electrical and Computer Engineering
Supervisor

Dr. Fayez Gebali, Department of Electrical and Computer Engineering
Departmental member

Abstract

Domain generation algorithm (DGA) is a popular technique for evading detection used by many sophisticated malware families. Since the DGA domains are randomly generated, they tend to exhibit properties that are different from legitimate domain names. It is observed that shorter DGA domains used in emerging malware are more difficult to detect, in contrast to regular DGA domains that are unusually long. While length was considered as a contributing feature in earlier approaches, there has not been a systematic focus on how to leverage its impact on DGA domains detection accuracy. Through our study, we present a new detection model based on semantic and information theory features. The research applies concept of domain length threshold to detect DGA domains regardless of their lengths. The experimental evaluation of the proposed approach, using public datasets, yield a detection rate (DR) of 98.96% and a false positive rate (FPR) of 2.1%, when using random forests classification technique

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Issa Traore for his continuous support and motivation for me to pursue my studies and research at the University of Victoria.

I am grateful to Dr. Fayez Gebali for providing the initial motivation towards the field of research. I am greatly indebted to Dr. Issa Traore who provided me an opportunity to join his team as a Research assistant and gave me access to the ISOT laboratory and research facilities. Without his precious support and encouragement, it would not be possible to conduct this research. Dr. Nainesh Agarwal has been an excellent mentor and guiding force throughout my Master's journey.

I would like to thank Dr. Fayez Gebali and Dr. Yvonne Coady, for serving on my supervisory committee.

My loving and supportive parents, Ravi and Vanita Ahluwalia and sister Ahana have been the pillars of my strength. I also want to thank my extended family and friends for their endless love and encouragement throughout my academic journey

Dedication

To my Pillars of Strength,
Mom, Dad and Ahana

Chapter 1

Introduction

1.1 Context

Botnets are considered to be one of the biggest online threats today [11]. Botnet, as the name suggests, is a network of compromised computers infected with malware. This collection of zombie machines is remotely controlled by the attacker, known as Bot-master or Bot-herder. Cybercriminals are controlling remotely malware infected networks through command–and–control servers (C&C). Figure 1 illustrates a typical Botnet Operation scenario.

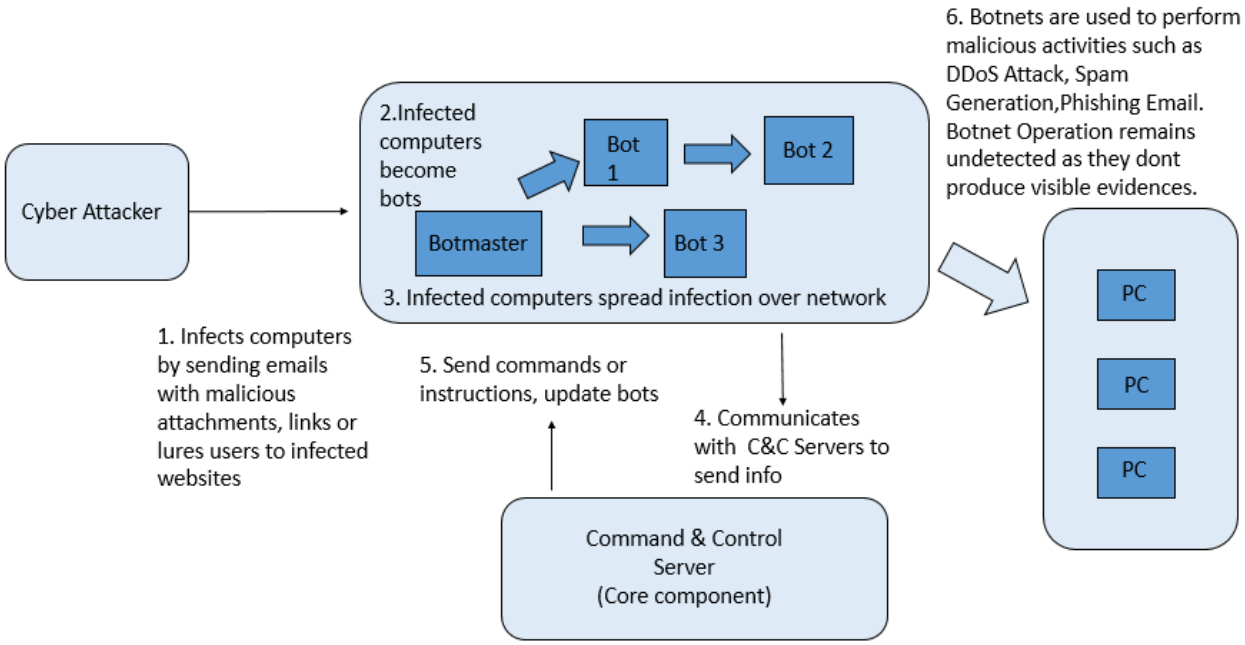

Figure 1. Botnet Operation

Malware can spread through the network by user accessing dangerous sites, unsecured digital devices, bypassing weak security protocols, ineffective firewalls etc. Victims include naïve home computer users or users that have opened email attachments that are malware-infected. Mostly the users are unaware of the fact that their devices have been compromised, as infectious programs silently get installed and remain hidden until they receive instructions from the C&C servers. Once the malware has been installed on the device, the bots contact their bot-master through C&C communication channels, which use different protocols such as IRC (Internet Relay Chat), HTTP, or various P2P (Peer-to-Peer) schemes. A Bot-master who has access to the C&C server and has collected enough bots on the network for the intended attack can send out a single command to the infected devices now distributed over the world.

Botnets are used to perform malicious operations throughout the network by sending spam mails, causing data exfiltration, and Distributed Denial of Service (DDoS) attacks. DDoS attack is carried out by overwhelming servers with large amounts of packets, as a result denying services to legitimate users. Recently these bots have been seen delivering ransomware attacks on individuals and businesses as well. In a ransomware attack, the malware on infected computers, sends out massive spam emails over its network. Access to any such spam attachment can cause the data on the host to be encrypted and locked down which can only be reversed once the ransom has been paid within given time frame. Apart from creating a ransom situation, botnets can steal this information and sell it illegally for more profit. Bot networks are being traded or leased out on temporary basis, which allows hackers across the globe to take advantage of this situation to spread fear and damage in society. Often specific organizations are targeted for ransom and this adversely affects business and customer trust.

Botnet behavior has been a very interesting topic for research among academia and industry for years now. The highlight of botnets is the ability to provide anonymity through the use of a multitier command and control (C&C) architecture [13]. It is quite challenging to capture bot behavior due to its dynamic nature and fluxing IP addresses [12].

In order to avoid detection, botnets adopt various hiding mechanisms, one of them is Fast Flux. Fast Fluxing is a DNS technique to evade detection and hide communication through an ever-evolving network of compromised hosts acting as proxies. Communication between malware and its C&C server is made more resistant to discovery by deploying Fast Flux mechanism.
Domain fluxing is when a botnet operates by continuously changing the domain name of the Bot-herder's Command and Control (C&C) server. Typically, bots are producing a large number of random domain names generated algorithmically, also known as algorithmically generated domains (AGD). Out of which one of the domains will be registered by the botnet owner. DNS queries are sent out by bots to the random domains until one of them resolves to the address of the C&C server. The biggest advantage of using DNS Fast Fluxing is that a different IP address is returned every time a DNS query is made against a domain name. So if one C&C server is taken down, the rest of the IP addresses are yet functional and malware can continue its operation without any downtime. In most cases, if the servers were taken down, they will be replaced soon by new servers by attackers in charge.

Organizations and security researchers face immense difficulty in blocking instructions and shut ting down a bot operation due to fluxing nature of botnets. Currently, botnet researchers have to reverse-engineer the bot malware and monitor domains that are generated on a regular basis in order to detect the C&C server. Recently, domain fluxing has been implemented by botnets such as Conficker C [25], Kraken [23] and Torpig [24]. Kraken employs a random word generator and

constructs words that sound like English, whereas Torpig uses a seed based on the most popular trending topic on Twitter [8].

1.2 Research Problem

Domain Generation Algorithms (DGAs) is a class of algorithms that take an input seed and append a top level domain (TLD) to produce a newer random domain name. The top level domain (TLD) could be .com, .co, .in, .uk, .ca etc. The seed is a shared entity between the bot owner and the malware infected machine. Every botnet has its own mechanism of choosing a seed. The seed can either be a static integer value or in most cases the current timestamp. Figure 2 reflects the common concept of DGA, i.e. transformation of an input domain name to an obfuscated string usually difficult for humans to understand.

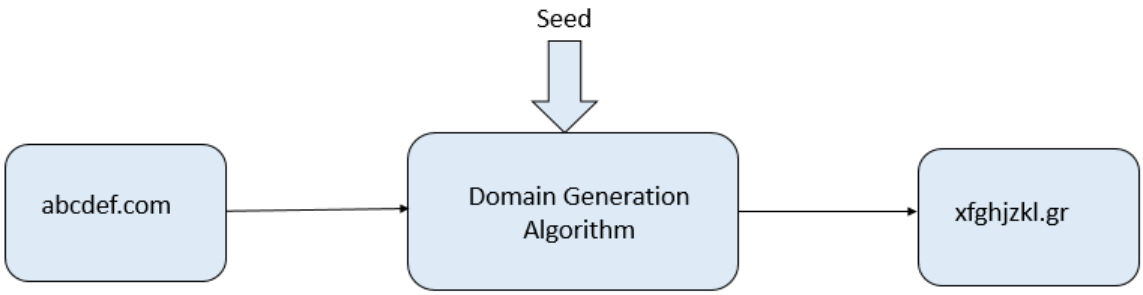

Figure 2. Algorithmically generated domain (AGD) names process

Adversaries often use domain names to connect malware to C&C servers. Earlier approaches used hardcoded list of IP addresses and domain names which were easily blacklisted once detected.

DGAs provide flexibility in terms of assigning IP addresses to C&C server and can easily update the domain name to point to a new IP address, if the domain in use is blacklisted.

To evade blacklisting, modern botnets, such as HTTP botnets, are taking advantage of various domain generation algorithms (DGAs). The DGAs embedded in the malware generate on a regular basis large amounts of pseudo-random domain names in a short period of time.

Pseudo-random means that the sequences of strings appear to be randomly generated but are actually repeatable given some initial seed or state. The algorithm runs on the victim device as well as remotely on the Bot-master device. The domains are generated based on seeds known only by the bots and the C&C servers, while the bot-master will typically register and activate only one or a handful of these domains. On the victim side, malware runs the DGA and checks by sending out DNS queries if the domain name is live or not. If a domain is registered, the malware uses this domain as its C&C server; if not, it checks another one. This process keeps repeating until the C&C server is detected or shutdown. This process is described in detail Figure 3.

Existing research has been trying to guess the seed by reverse engineering a small sample of DGA with the aim to pre generate the domains that could be used by the bots [1]. This process can be very tedious with no guarantee of desired results since thousands of domains are generated by a DGA on a given day. Reverse engineering such large volumes of non-existent algorithmically generated domain names is extremely time-consuming and resource intensive with low success rates, hence making detection harder.

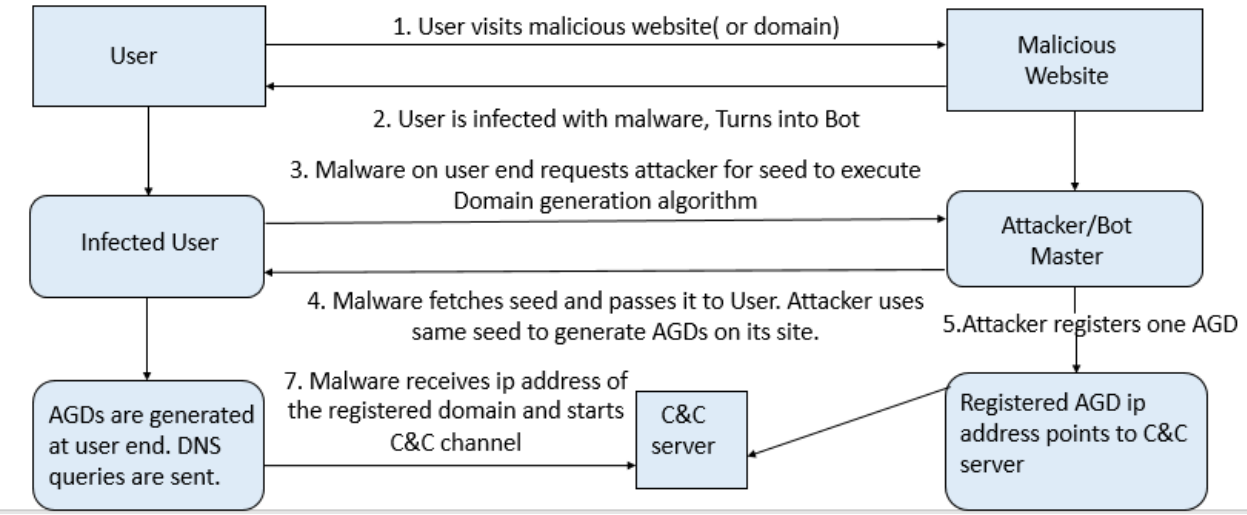

Figure 3. DGA execution and process

Alternatively, several research proposals have been published that extract various features from the AGDs (i.e. the generated domains) and use machine learning algorithms to detect the presence of DGAs, with varying degree of success [1][2][4][8].

A key characteristic of Algorithmically Generated Domains (AGD) is their length, which is unusually greater than Human Generated Domains (HGD). As a result, most existing DGA detection schemes have been centered on length detection [4, 10]. However, several recent DGA strains have been using shorter domains lengths, which are indistinguishable in length from legitimate domains, making their detection much harder.

The purpose of the research conducted in this thesis is to address this gap by developing a detection approach for AGDs that will perform well on both short and long domain names.
We study in this work the impact of length on DGA domains detection performance, and propose a simple adaptive model, which uses different feature models based on length threshold value.

1.3 Approach Outline

The main idea behind our approach is that algorithmically generated domains exhibit properties that greatly differ from legitimate domain names. Initial steps included studying two different small sample datasets. One set belonged to benign domains and the other consisted of sample AGDs. The non-DGA domain dataset helped to recognise the typical patterns of domains and build knowledge and insights about linguistic models that can be generated from them. The domains that exhibited different patterns were considered to be automatically generated.

Our proposed approach involves studying significant features from domain names that can help correctly distinguish between AGDs and HGDs. The features are extracted based on lexical and linguistic characteristics of the domain names. Information theoretic measures of domain characteristics are computed and used to detect broad range of DGAs regardless of the domain length. The extracted features are classified using machine learning. We investigated in this thesis two different classification techniques, namely, Random Forests and Decision Tree.

1.4 Thesis Contribution

The main contribution of this thesis is a unified detection model for AGDs that enables detecting various domains, including both and short domains. This is an important step toward detecting emerging malware and botnets that attempt to evade detection using short domain names. Experimental evaluation on public DGA datasets and (legitimate) Alexa top domains, yields for random forests, the best performing of both algorithms, detection rate (DR) of 98.96% and false positive rate (FPR) of 2.1%.

1.5 Thesis Outline

The outline of the thesis is as follows. Chapter 1 gives an outline of the context of the research, by highlighting the different terminology and techniques involved in Botnet operations, and discusses the challenges posed by Domain Fluxing and Domain Generation algorithms.

Chapter 2 summarizes and discusses the related works. Chapter 3 introduces our proposed approach and models. Chapter 4 presents the experiments conducted to study the impact of length on performance, and evaluates our proposed detection scheme. Chapter 5 makes concluding remarks and discusses future work.

Chapter 2

Related Works

Several works have been published in the literature on DGA detection. We review and discuss a sample of closely related work in the following.

Antonakakis et al. [1] developed Pleiades, a DGA-detection system that uses few important statistical features to translate DNS NXDomains subsets to feature vectors. Such features include splitting domain name into various levels and distribution of n-grams. Few structural features like number of domain levels, length of domains and number of unique top-level domains are calculated. They also discuss character entropy calculation across various domain levels as a feature. The approach was evaluated experimentally using a dataset consisting of 30,000 DGA domains from the malware families Conficker, Murofet, Bobax and Sinowal. Also, 20,000 normal domain names from Alexa were used for training and testing of the DGA Classifier. The Hidden Markov Model (HMM) was used for classification and the results obtained were fairly good, with TPR (True Positive Rate) around 91% and FPR of 3%.

Ma et al. [2] introduced a lightweight approach to classify URLs using lexical as well as host-based features. They consider lexical features of the URL such as length, number of dots, special characters in the URL path, etc., and label them as tokens. These tokens are collected and preserved for different categories like hostname, URL path and Top-level-domains (TLD). The extracted features are processed using machine learning classification. Three different classifiers were studied including Naïve Bayes, Logistic regression and Support Vector machine (SVM). Experimental evaluation was conducted across 20,000 to 30,000 URLs drawn from different sources, achieving prediction accuracy of 95–99%.

Wang and Shirley proposed in [9] an approach to detect malicious domains using word segmentation to derive tokens from domain names. The proposed feature space includes the number of characters, digits, and hyphens. Each set of features were calculated for 291,623 domains from a dataset collected over cellular network. Experimentally, they were able to compile a list of 30 most referenced words and created a probability model for the words based on the Google N corpus by Peter Norvig [5]. This is an interesting approach as it helps understand what words in the malicious domain can attract users. Hence, this system can be used to monitor and analyse new words being used by malicious domains. Classification was done using logistic regression models, providing more meaningful associations between the features and domain maliciousness, thus improving prediction accuracy.

Schiavoni et al. [7] proposed the Phoenix model, which attempts to discover previously unknown DGAs and helps tracking DGA-based command–and-control centers over time, using linguistic features like meaningful character ratio and n-gram normality score. The Phoenix model was evaluated on 1,153,516 domains and resulted in prediction accuracy of about 94.8%.

McGrath and Gupta highlighted in [3] that Phishing URLs and domains possess different characteristics than normal domains and URLs. They found, for instance, that phishing domains used fewer vowels and fewer unique characters compared to normal ones. They proposed a detection model for malicious domains focused on comparison based on domain lengths (excluding Top-level-domains) and character frequencies of English language alphabets. The proposed model was studied experimentally using a dataset consisting of over 9 million unique URLs and 2.7 million unique effective second-level domain names. The focus of the work was essentially on identifying the characteristics of phishing. No concrete detection model was

proposed. However, the characteristics can serve as useful basis for constructing adequate feature space for DGA domains detection.

Yadav et al. [8] have developed their detection methodology by studying patterns of lexical characteristics of domain names. Their study involves analysing distribution of alphanumeric characters to distinguish between normal and malicious domains. They believe unigram or single characters are not enough, hence bigrams are taken into account. They employ statistical measures like KL-divergence, Jaccard index and Levenshtein edit distance for classifying a group of domains. These domains included a set of legitimate domain names obtained via a crawl of IPv4 address space as well as DNS traffic from a Tier-1 ISP in Asia. They used malicious dataset consisting of domain names generated by recent botnets and domains generated by a tool that produces English pronounceable words not present in the dictionary. The main contribution of their work is to have shown empirically that in detecting AGDs, metrics such as Jaccard index performs best, followed by Edit distance metrics, and then KL divergence.

Mowbray and Hagen [4] designed a procedure to detect unusual length distribution of second-level domains (2LDs). Features used by their approach are counts of individual character, character n-grams, dots, hyphens, digits, uppercase, lowercase and length. The paper also addresses the need to identify newer DGAs than just using machine learning to classify based on pre-existing malicious samples. By running the proposed model on DNS query dataset collected over 5 days at an enterprise network, they were able to detect 19 different Domain Generation Algorithms, including nine new ones.

Sharifnya and Abadi [6] developed a reputation system to classify hosts as malicious or non malicious. They built a system that can be used for real-time analysis of hosts producing large number of suspicious domain names or numerous failed DNS queries. These characteristics help

build a suspicious matrix and assign a reputation score. Hosts with higher negative score are considered malicious. Evaluation using DNS query data indicate improving detection and false positive rates, over time.

Summary

In this chapter, we summarized and discussed related work on DGA analysis and detection. Most of the papers covered discuss the various semantic and word segmentation techniques that could be applied to correctly distinguish normal domain dataset and DGA dataset.

While length was considered in one way or another in the existing literature, there has not been a systematic focus on how to alleviate its impact on DGA domains detection accuracy. Our work tackles this challenge. We introduce, in the next chapter, a new approach for broad-length DGA detection.

Chapter 3

Proposed Approach and Models

3.1 Approach Overview

Every domain name hierarchy is composed of a Top level domain (TLD) and a Second level domain (SLD) directly below it. TLD is the part of the domain name located to the right of the dot (“. "). The most common TLDs are .com, .net, .org, .gov, .biz, .info and .ws. A domain name is a sequence of labels separated by dots. For example, in the domain *www.isot.ece.uvic.ca*, *.ca* is the TLD, *uvic* is the second-level domain (SLD) of the *.ca* TLD, while *ece* is a third-level domain (3LD).

In the design of our proposed DGA detection model, our main focus has been on studying the distribution of second-level domain names (SLDs). The SLDs with any unusual distribution of characters or syntactic or structural features were of highest interest. Simplicity is the key to performing efficient feature selection. Precautions need to be taken while shortlisting and testing features as adding too many of them might cause overfitting the data.

We have made a reasonable assumption that domains that are generated by humans (HGDs) have a degree of familiarity as compared to algorithmically generated domain names. Degree of familiarity makes domain names more readable, understandable and can be easily recollected for daily usage by humans. The essence of domain names, originally, include memorability and friendliness. As such, the overwhelming majority of legitimate domain names convey specific information, which makes sense to humans, and makes it easy for them to grasp or remember the underlying concept. In contrast, AGDs are generated for the sole purpose of evading detection. As they are used for the consumption of bots, which are automated processes, the aforementioned

(human) expectations are irrelevant. As such the conveyance of any meaningful information, beyond hosts referencing, is unnecessary. As a result, not only typical AGDs lack friendliness, the amount of information involved in their structures and semantics tend to be minimal.

An interesting observation is that the distribution of characters across HGDS and AGDs vary significantly. The characters here could be either the English alphabets (a-z) or digits or both. Our approach consists of measuring the amount of information carried by the name, and then treating the lack of meaningful information as suspicious. The amount of information or lack thereof can be captured through the Information Entropy and can be used to compute metrics that characterize the distribution of features such as alphanumeric characters, digits or groups of characters (n-grams) within the set of domain names. These features have to be developed specifically with an aim to quickly distinguish a set of human generated domains from algorithmically generated domain names and vice-versa.

Since most domain names are unique, it is extremely difficult to study the domains if they are treated as a single token. There was a need for a technique to break down the domain names into smaller substrings in order to study the unusual distributions across HGDs and AGDs. We measure the amount of information conveyed by the domain by analyzing character n-grams and computing the corresponding entropy. First step in this task was to devise segmentation patterns based on n-grams. Various n-gram features such as unigram (*n=1*), bigram (*n=2*) and trigram (*n=3*) were extracted from our training corpus. The accuracy of the n-gram derived patterns depend on factors such as the value of *n* and dataset size. Our initial experiments attempt to implement a basic feature model with a small dataset of 20,000 domain names. The domain dataset comprises of 10,000 legitimate domain names extracted from Alexa top 1 million domain names [16]. Remaining domains belong to AGD dataset [17]. It was designed to measure basic features such as character

count, number of digits, length of domain names etc. All the above mentioned features are calculated from the second-level domain names.

At the same time, we leverage basic lexical and linguistic characteristics of the domain names which have proven effective at detecting DGAs, and are also a by-product of the underlying random structure. Further experimentation led to calculating other features such as individual character frequency and studying occurrence patterns, n-gram analysis, correlating information entropy etc.

One such characteristic which has proven effective, while at some time being elusive is the domain length. We study the impact of length on detection accuracy, and leverage such knowledge to develop a simple detection model that uses the notion of length threshold to alleviate its impact on accuracy.

3.2 Basic Features Model

A common characteristic of early AGDs was their length. According to Weymes [10], in general, legitimate domain names are below 12 characters long, while AGDs are often much longer. In this perspective, the *Length* can be considered as a key feature in identifying DGA domains. However, as shown in Table 1, some of the recent malware have started generated shorter DGA domain names, which makes it difficult to distinguish them from legitimate domains (see Table 2 for samples).

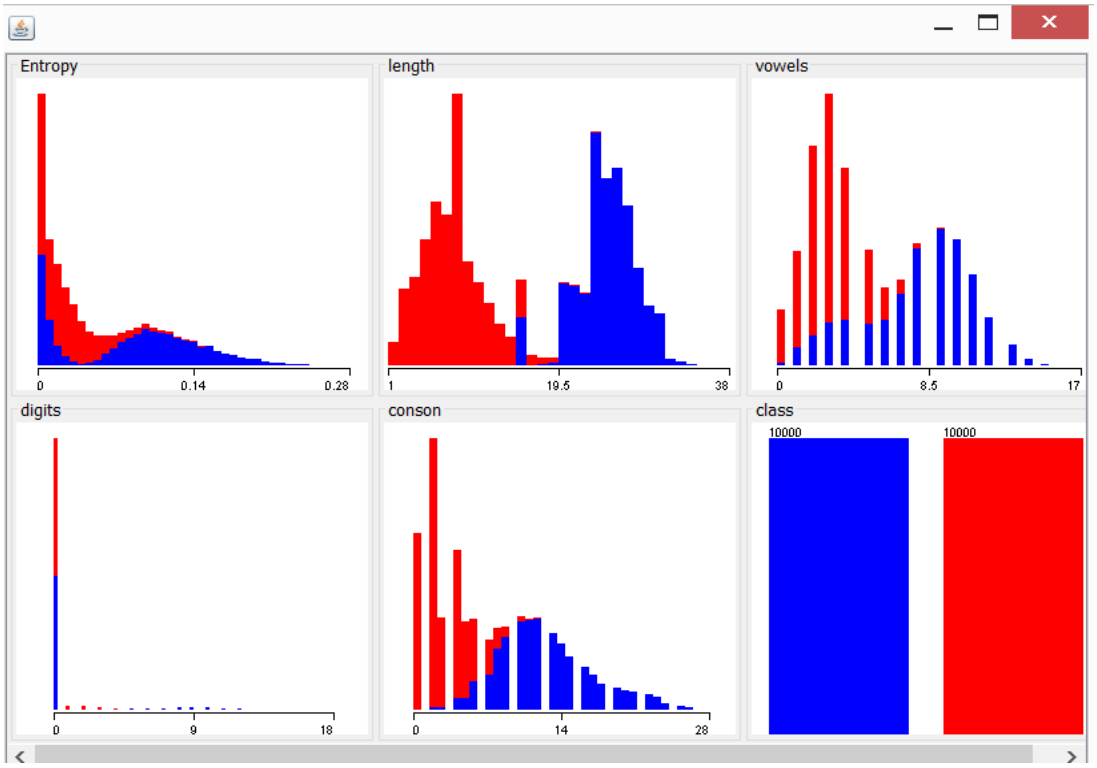

Figure 5. Basic features distributions based on 50:50 mix of legitimate and DGA data; the blue represents the legitimate domains subset while the red is the DGA subset.

Table 1. Sample DGA domain names with corresponding malware

Short DGA Domains (length 5 -7)	Long DGA Domains (Length 12 -30)
jaorw.com - Cryptolocker	lhxmfkhppoww.com - Tinba
conxyb.cc – Zeus	uocwkscmkwwmkomc.org - Ramdo
klvupgfm.biz - Conficker	gmacsmsgecquasam.org - Ramdo
azrvtgf.net - Cryptolocker	mcgakswwegmeuwma.org -Ramdo
jltkyg.com - Cryptolocker	stabletiredflowerwardisappointed.com - Matsnu
pbfxmi.biz - Cryptolocker	herequiresassembledcircumstances.com - Rovnix

bfuqnb.info - Zeus	thatunanimoushiswarcorrespondence.com- Rovnix
dvlnrl.ws - Zeus	nzzhyhemzswcyrwpzjztdmbyptktk.ru - GameOver Zeus
vkdjisc.com - Conficker	uvgxmjeiucrkeqcmaeuokzganjjbvo.ru – GameOver Zeus
uhbqolxf.org - Conficker	eminpvkskrdygqphyhhypfahm.ru – GameOver Zeus

Table 2. Sample legitimate domain names from Alexa.com

Legitimate Domain Names
google.com
baidu.com
wikipedia.org
stackoverflow.com
taobao.com
sina.com.cn
bing.com
yandex.ru
hao123.com
instagram.com
9gag.com
americanexpress.com
adnetworkperformance.com

Likewise, the primary primitive feature in our model is the Length for a given domain. This is calculated by counting all the characters in the domain name excluding the TLD. Additional primitive features that capture linguistic and structural characteristics of the domain name, considered in our model include the following:

- **Vowels:** This feature is calculated by counting the number of vowels in the second level domain name (SLD) only.
- **Consonants:** Similarly, as above, this is the count of the consonants present in the second level domain name.
- **Digits:** The count of digits in the second level domain name.

Figure 5 illustrates the distribution of the basic features from a dataset consisting of 20,000 domain names, with 10,000 DGA and 10,000 normal domain names. The distributions indicate strong capability of these basic features to discriminate legitimate domains from malicious DGA domains. Therefore, using the aforementioned basic features model, and simple statistical techniques, we can achieve encouraging detection capability.

3.3 N-gram Model

N-grams are one of the most powerful models for speech and language processing. N-gram distribution can be viewed as sequence of words or as continuous character strings. For most machine learning applications in this field, we cannot rely completely on n-gram based models. There is a need for understanding prior distributions and their likelihood result. Modern statistical models are typically made up of two parts, a prior distribution describing the inherent likelihood of a possible result and a likelihood function used to assess the compatibility of a possible result with observed data [19].

Among other uses, N-gram model can be used to determine the next element in sequence based on probability distribution and calculation. The intuition of the N-gram model is that instead of

computing the probability of a word given its entire history, we can approximate the history by just the last few words [20].

As part of our experiments, this concept has been modified as splitting our words into *n* character substrings .A simple example can be calculating sequence of the 2-grams and 3-grams, referred to as bigrams and trigrams, respectively, for the domain name , “google.com”. The following bi-grams and trigrams can be generated from top level domain, "google":

Bigrams = {“go”, “oo”, “og”, “gl”, “le”}

Trigrams = {“goo”, “oog”, “ogl”, “gle”}

Important observation here is that we do not consider all possible permutations of bigrams/trigrams from the input string, but only the ones that are naturally present as sequences in the given string.

Considering a bigram model, the probability of a bigram occurring next, given all the previous bigrams occurring in the domain name string, can be calculated by using only the conditional probability of the preceding bigram. This approach can be generalised for occurrences of n-grams with n = {2,3,4….N}.

Because of its flexibility and simplicity, N-gram model has many advantages and future scopes in language modelling applications.

3.4 N-gram Model and Conditional Probability

Structural and syntactic considerations of domain names can contribute to the likelihood function as mentioned above. The features could be ranging from length related observations to position of characters or ratios of substrings present. Therefore, by introducing N-gram model we

can assign probabilities to sequence of strings or words. The probabilities of an N-gram model come from the corpus it is trained on.

Another way to estimate probabilities is the maximum likelihood estimation (MLE). We get maximum likelihood estimates for the parameters of an N-gram model by getting counts from a corpus, and normalizing the counts so that they lie between 0 and 1 [21].

Next step would be to calculate a particular bigram or trigram probability of a domain name using probabilities. Probability multiplication can lead to numerical overflow, and can be solved by using log probabilities. Another advantage of using log probabilities is easy multiplication and faster computation and efficient storage.

3.5 Domain Name N-gram Entropy

The aforementioned basic features capture the idiosyncrasies of DGA domains names. But as mentioned earlier, an important characteristic of DGA domains is the lack (by design) of meaningful information involved. The amount of information can be calculated by the entropy.

By analyzing sample domain names, we found out that the entropy can be sensitive to the domain length. It is effective mainly for longer domains, and tend to struggle for shorter ones. Such sensitivity can be alleviated, by analyzing domain n-gram frequency subject to the length, in other words, the conditional probability with respect to the domain length.

Therefore, our feature model involves 6 features: the aforementioned 4 primitive features, and two advanced features consisting of the domain n-gram entropy and conditional probability. Analysis of different kinds of n-grams on sample data showed that n=3 yields better results. So, we analyze and use trigrams in our feature model.

We compute the domain trigram entropy using word segmentation. First, we extract the SLD of the domain, and for each SLD we compute the entropy of the domain based on the trigram frequency. We derive trigram frequencies from the Google n-corpus. Google n-corpus is a large database created to help machine learning initiatives across the world [5]. It has English language words and their respective n-gram counts, such as two, three, four or five-letter word sequences.

Let *trigram(d)* denote the set of all trigrams involved in SLD of domain *d*, and *Trigram(n-corpus)* be the set of all trigrams involved in the n-corpus.

Given trigram $t_r \, \epsilon \, trigram(d)$, let $count(t_r, n-corpus)$ denote its occurrence count based on the (Google) n-corpus.

The occurrence probability for trigram t_r for domain *d* is calculated as follows:

$$P(t_r) = \frac{count(t_r, n-corpus)}{\sum_{x_r \epsilon Trigram(n-corpus)} count(x_r, n-corpus)} \qquad \text{(i)}$$

The entropy of domain *d*, denoted *H(d)* is computed as follows:

$$H(d) = -\sum_{t_r \epsilon trigram(d)} P(t_r)\log(\mathrm{P}(t_r)) \qquad \text{(ii)}$$

A trigram may occur more than once, say *n* times in a domain name. Then probability of occurrence of the trigram in the domain is considered $n \times f$, where *f* is the frequency of occurrence of the trigram.

3.6 Domain Name N-gram Conditional Probability

As mentioned earlier, we observed that the entropy values for domain length around some threshold value of *l* are very close for DGAs and legitimate domain names. This makes it difficult to distinguish between them. Hence, conditional probability is a parameter that takes into consideration the domain length.

The conditional probability *P(Y|X)* quantifies the outcome of a random variable *Y* given that the value of another random variable *X* is known.

Let *l* be the domain length threshold. Let *L* denote the list of domains from the training datasets having length lesser of equal to *l*:

$$L = \{d: D | d.length \leq l\} \quad \text{(iii)}$$

We consider separate domain subsets for positive and negative samples (i.e. DGA and normal). But as the process is the same in both cases; we will describe only one case.

For a given domain *d* (*d∈ L)*, let *trigram(d)* be the set of corresponding trigrams. Let *Trigram(L)* denote the set of all trigrams in L: $Trigram(L) = \bigcup_{d \epsilon L} trigram(d)$.

Given a trigram t_r in *Trigram(L)*, let $count(t_r, L)$ denote the total occurrence number of t_r in *Trigram(L)*. The conditional probability of trigram t_r with respect to length *l* is calculated as follows:

$$P(t_r|l) = \frac{count(t_r, L)}{|Trigram(L)|} \quad \text{(iv)}$$

Where $|Trigram(L)|$ denote the size (or cardinality) of *Trigram(L)*.

The conditional probability of domain *d* with respect to length *l* is calculated as follows:

$$P(d|l) = \sum_{t_r \epsilon trigram(d)} P(t_r|l) \qquad \text{(v)}$$

3.7 DGA Domains Detection Model

Our detection model consists of extracting the aforementioned features and classifying them using machine learning. We investigated two different classification techniques in our detection approach. Specifically, the following classification techniques were considered:

- J48 decision tree
- Random Forests algorithm

We used the Weka machine learning tools to simulate the aforementioned classifiers. Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

We studied the impact of using each of the features separately and by grouping them in different subsets. The combination of all the features yield the best performance. However, a naïve combination, where all the features are grouped and used equally in a straightforward way is sensitive to the length of the domains, and yield reduced performance for shorter domains. We refer to this feature model as the "naïve model". We propose a simple alternate model, which uses different feature model based on whether the domain is greater or smaller than some length threshold value *l*. We refer to this model as the "split model", and it can be summarized as follows:

If (domain.length < l) then use (vowels, consonants, digits, conditional probability)

Else use (vowels, consonants, digits, conditional probability, entropy)

Our experiments show that the “split model” is less sensitive to the domain length, and yields much improved performance compared to the “naïve model”.

Summary

This chapter discusses the layout and characteristics of domain names and introduces our proposed feature model. It is observed that majority of legitimate domain names and AGDs exhibit major differences in terms of degree of familiarity and human expectations. Bots do not need to convey any meaningful information other than hosts referencing so it is common for them to have domain names that are not human-understandable. In the next chapter, we conduct different experiments to study the effectiveness of our proposed model.

Chapter 4

Domain Length Impact Study

4.1 Datasets

Our study includes experimentation with various non-malicious subsets of domains from Alexa's top million website's [16]. Websites belonging to Alexa database are ones that receive highest traffic through users on the Internet across the globe. The ranks reflect user behaviour based on unique page views and visitors. We identify these domain names as non malicious.

It is not an easy task to obtain data from botnet malicious activities. The malicious datasets of domains are collected from various open source projects such as DGA classifier and other DGA related projects on GitHub [17] [18].

We used in this work a global dataset of 100,000 legitimate domains consisting of legitimate domain names from alexa.com, and a dataset of 100,000 DGA domains, giving a complete dataset of 200,000 domain names.

4.2 Machine Learning Classifiers

Machine learning is the process of learning from patterns in data and making improved decisions in the future. Machine learning approaches are similar to data mining and predictive modeling as they involve parsing of data for patterns and adjusting future actions accordingly. Common machine learning uses are fraud detection, spam filtering, network security threat detection, predictive maintenance and building news feeds.

In our work, we investigated two popular machine learning techniques, Decision Tree classifier and Random forest algorithm. Both belong to class of supervised classification algorithms.

4.3 Domain Length Impact Study

In order to assess the impact of the domain length on detection performance, we run several experiments using different length threshold values. We run these experiments over different feature subsets.

Given a feature subset, we vary the length threshold denoted *l*, and measure the detection performance for decision tree and random forests classifiers, respectively.

The experiments were conducted using a subset Δ of our global dataset consisting of 40,000, with a mix of 50% DGAs and 50% legitimate domains. In each run of the experiment for length threshold *l*, the subset $\Delta(l)=\{d \epsilon \Delta | d.length \leq l\}$ (of the domains of length smaller than or equal to *l*) is used for training and testing.

We run each of the aforementioned experiments using 10-fold cross validation. We split the input dataset Δ(l) into 10 subsets of data. In each validation run, we train the machine learning classifier on all but one of the subsets, and then evaluate the model on the subset that was not used for training.

4.3.1 Experimental Results from Domain Trigram Entropy only Model

The first experiment was run by varying length threshold *l* on dataset Δ(l), using as feature the domain trigram entropy only. Tables 3 and 4 show the performance obtained for J48 decision tree and random forests, respectively. It can be noted in both cases that the performance improves as the domain length increases

Table 3. Performance of J48 Decision tree on domain trigram entropy only-model, when varying length threshold (experiment 1)

Length	**Detection rate (%)**	**FP rate (%)**
6	82.7	17.3
7	87	13
8	89.55	10.5
9	91.55	8.5
10	93.05	7

Table 4. Performance of Random Forests on domain trigram entropy only-model, when varying length threshold (experiment 1)

Length	**Detection rate (%)**	**FP rate (%)**
6	78.1	21.9
7	81.65	18.4
8	84.35	15.7
9	88.95	11.1
10	90.55	9.5

4.3.2 Experimental Results from Domain Trigram Conditional Probability only Model

The second experiment was run under the same condition as previously, but this time using the domain trigram conditional probability as only feature. Tables 5 and 6 show the performance obtained for J48 decision tree and random forests, respectively.

Table 5. Performance of J48 Decision tree on domain trigram conditional probability only-model, when varying length threshold (experiment 2)

Length	**Detection rate (%)**	**FP rate (%)**
6	85.15	14.9
7	90.05	10
8	91.8	8.2
9	92.45	7.6
10	94	6

Table 6. Performance of Random Forests on domain trigram conditional probability only-model, when varying length threshold (experiment 2)

Length	**Detection rate (%)**	**FP rate (%)**
6	85.15	14.9
7	90	10
8	91.35	8.7
9	92	8
10	94.25	5.8

We can notice that like in the previous experiment as the length threshold increases the accuracy of the detector improves. However, it can also be noted that the use of conditional probability alone achieves improved performance compared to using entropy alone.

4.3.3 Experimental Results from Model with all Features excluding Domain Trigram Conditional Probability

The third experiment was run by varying length threshold l on dataset $\Delta(l)$, using our basic feature model (i.e. number of digits, number of consonants, and number of vowels) and the domain trigram entropy, excluding the domain trigram conditional probability. Tables 7 and 8 show the performance obtained for J48 decision tree and random forests, respectively.

Table 7. Performance of J48 Decision tree on basic features and domain trigram entropy, excluding domain trigram conditional probability, when varying length threshold (experiment 3)

Length	**Detection rate (%)**	**F Prate (%)**
6	87.42	13.4
7	89.19	10.9
8	92.381	7.8
9	93.5	6.5
10	94.043	5.9

Table 8. Performance of Random Forests on basic features and domain trigram entropy, excluding domain trigram conditional probability, when varying length threshold (experiment 3)

Length	Detection rate (%)	FP rate (%)
6	82.98	17.7
7	87.52	12.6
8	91.09	9
9	92.75	7.3
10	93.88	6.2

4.3.4 Experimental Results from Model with all Features excluding Domain Trigram Entropy

The fourth experiment was run under the same condition as above, but this time using our basic feature model (i.e. number of digits, number of consonants, and number of vowels) and the domain trigram conditional probability, excluding the domain trigram entropy. Tables 9 and 10 show the performance obtained for J48 decision tree and random forests, respectively.

The combined model achieves some improvement for the decision tree, but the opposite occurs for random forests. So it can be said no clear cut improvement is achieved by excluding the conditional probability.

Table 9. Performance of J48 Decision tree on basic features and domain trigram conditional probability, excluding domain trigram entropy, when varying length threshold (experiment 4)

Length	Detection rate (%)	FP rate (%)
6	89	11
7	90.9	9.1
8	93.2	6.8
9	93.1	6.9
10	95.6	4.5

Table 10. Performance of Random Forests on basic features and domain trigram conditional probability, excluding domain trigram entropy, when varying length threshold (experiment 4)

Length	Detection rate (%)	FP rate (%)
6	88.35	11.7
7	90.2	9.8
8	92.8	7.2
9	93.3	6.7
10	94.45	5.6

We can notice a notable improvement when using conditional probability and the basic features over when using entropy and the basic features model. There is also slight improvement when using conditional probability and the basic features over when using conditional probability only.

4.3.5 Experimental Results from Model with ALL Features

The fifth experiment was run using all features (i.e. basic features + entropy + conditional probability). We refer to this model as the "naive" model, as it combines the features without any particular transformation or restrictions. Tables 11 and 12 show the performance obtained for J48 decision tree and random forests, respectively. There is a slight improvement in performance over the previous experiments.

This is also highlighted by Figure 6, which depicts the ROC curves outlining the performances for the different experiments. The combined "naive" model achieves better performance over the other models involving a subset of the features, with Random Forests coming as best performing algorithm.

Table 11. Performance of J48 Decision tree on all features combined (basic features, domain trigram conditional probability, and domain trigram entropy), when varying length threshold (experiment 5)

Length	**Detection rate (%)**	**FP rate (%)**
6	89.1	10.9
7	91.95	8.1
8	93.7	6.3
9	93.9	6.1
10	94.9	5.1

Table 12. Performance of Random forests on all features combined (basic features, domain trigram conditional probability, and domain trigram entropy), when varying length threshold (experiment 5)

Length	**Detection rate (%)**	**FP rate (%)**
6	88.05	12
7	90.55	9.5
8	93.9	6.1
9	94.25	5.8
10	95.55	4.5

4.4 Discussion

Based on the experimental evaluation, we plotted the results obtained from the above five sets of experiments in Figure 6. All features were tested for different combinations and feature pairing to achieve maximum performance, i.e. Highest Detection Rate. The plot below shows the results from experiment 5, with highest Detection Rate and minimum False Positive Rate. This is the experiment result of applying random Forest technique to all features (basic, conditional probability and entropy). Experiment 1 can be seen as applying Random Forest on Domain Trigram Entropy only model. The experiment results have lowest detection rate around 78%. Hence, it can be said that domain entropy is not a highly contributing feature for our model.

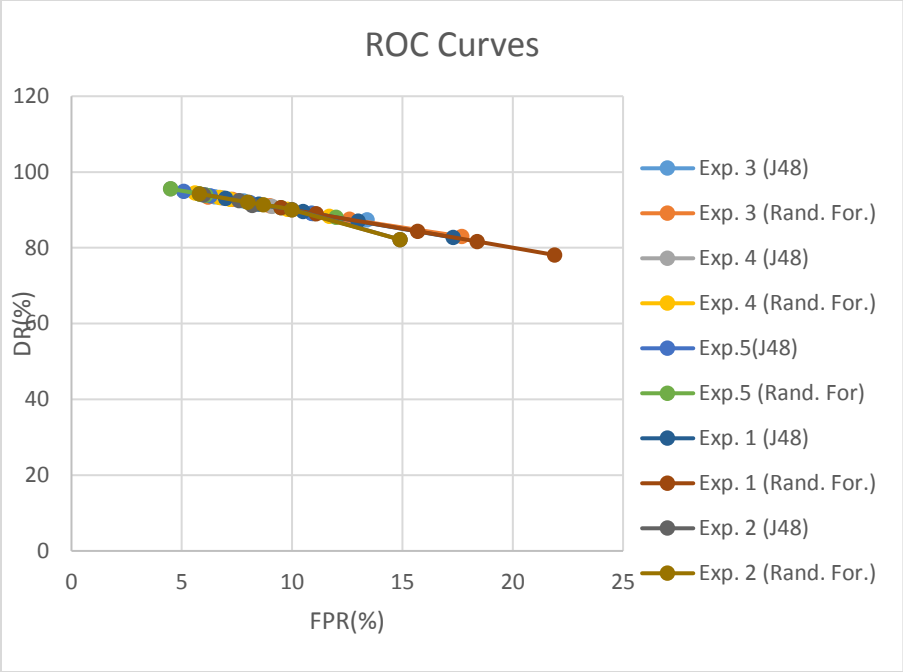

Figure 6. ROC curves illustrating the performances obtained by varying the length thresholds with different features models.

In the field of network security, if a malicious activity is not detected or alerted it is called as False Alarm or False Positive Error of the detection algorithm. In our experiment, the false positives are the normal domains that end up being misclassified as DGA values. The reason behind the higher number of false positives for the shorter domains can be the similarity in trigrams and distribution of characters in both normal and DGA domains. This is in accordance with our previous results that detection rate improves with increase in length. Also in our case, false negatives are algorithmically generated domains that get misclassified as Normal. Both these parameters are essential for verifying the performance of detection model. The sum of Detection Rate (DR) and False Negative Rate (FNR) is always one.

4.5 Split Model Study

Our split model relies on the consideration that the domain entropy does not contribute much for short domain length. We studied different values by varying the length threshold using the same dataset of 40,000 domains as in the previous subsection. Tables 13-16 show the performance obtained when varying the threshold l from 7 to 10. We can notice a notable improvement in performance when using the split model.

Important Aspects of Split Model:

- **Deciding Length threshold**: The split is controlled by a length threshold.
- **Short domains**: The basic features and the domain trigram probability is used as feature model
- **Longer domain names**: All the features are used

4.5.1 Performance of Split Model with varying length thresholds:

Table 13. Performance of Split Model when length threshold l=7

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	95.59	4.45
Random Forest	96.68	3.4

Table 14. Performance of Split Model when length threshold l=8

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	95.59	4.2
Random Forest	96.81	3.2

Table 15. Performance of Split Model when length threshold l=9

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	96.82	3.3
Random Forest	97.21	2.9

Table 16. Performance of Split Model when length threshold l=10

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	97.10	3.2
Random Forest	97.53	2.6

From the above tables, it is clear that as the length threshold is increased there is an increase in the detection accuracy and performance of the Split Model. Additionally, Random Forest algorithm performs better than decision classifier. Thus, this model is applied to our large dataset (200,000 domain names) experiment for final results.

4.5.2 Comparison of Split and Straight Models Performances

Figure 7 depict ROC curves comparing the performance of the Split model against the Straight (i.e. naïve) model. The split model achieves far better performance. The best performance is achieved when using threshold value of length l = 10.

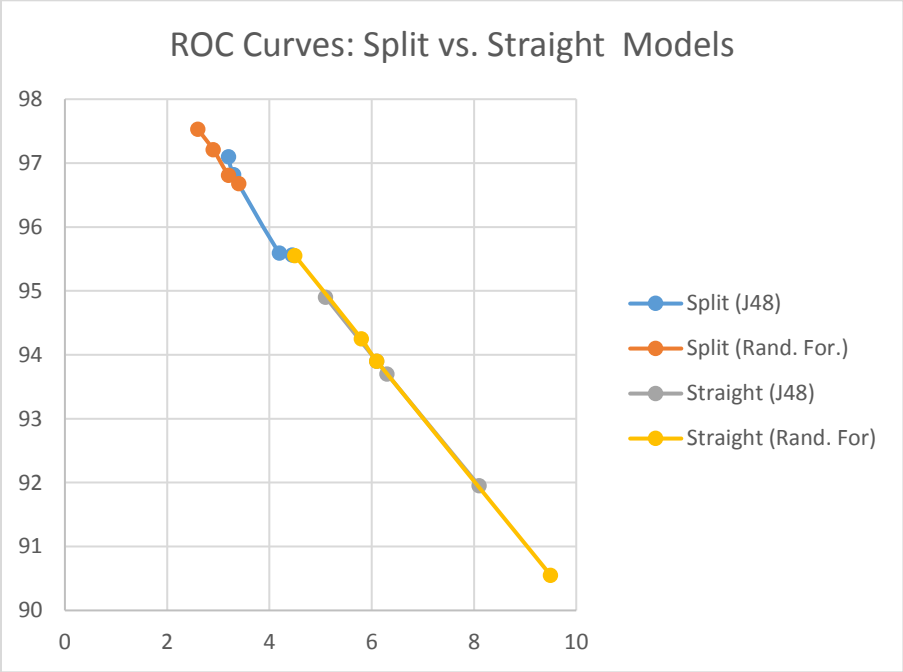

Figure 7. ROC curves comparing the performances of the split model against the combined or straight model obtained when varying the length threshold values.

4.6 Final Experimental Evaluation

As seen above, the Split model has greater performance and better results than the remaining models tested in our study. It is observed that highest performance is achieved when length threshold is 10. Hence for the final experiment , we run the split model using threshold l=10 on a larger dataset consisting of 200,000 domain names, with 50% legitimate domains and 50% DGA

domains. The experiment is conducted using 5-fold cross validation. Table 17 outlines the average performance obtained, which confirms and even slightly improves the performance shown in Table 16, which was obtained on a smaller dataset.

Tables 18-19 show the performance of the algorithms on the two upper and lower partitions of the dataset (i.e. domains of length smaller than 10, and domains of length greater or equal 10).

Table 17. Performance of Split Model on large dataset (200,000 domains) and threshold l=10

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	97.33	2.95
Random Forest	98.96	2.10

Table 18. Performance of Split Model on large dataset – subset of domains of length < 10

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	95.72	4.4
Random Forest	99.24	0.7

Table 19. Performance of Split Model on large dataset – subset of domains of length ≥ 10

Algorithm	**Detection rate (%)**	**FPR (%)**
J48	98.94	1.5
Random Forest	98.69	1.4

The Random forests algorithm achieves the best performance, and it performs almost similarly on shorter and longer domains. So the split model allows smooth transition, in quasi transparent way.

Summary

This chapter briefly describes our study on the impact of domain length on detection performance. The chapter discusses Split Model Study to overcome shortcomings of Straight Model. We study the key properties of Split Study model and compare the performance of the two models, Split versus Straight for varying length thresholds from length l = 7 to 10. The experiment was conducted by varying length thresholds and modelling results using machine learning techniques- Decision Classifier and Random Forest Algorithm.

Chapter 5

Conclusion

5.1 Summary

Over the years, researchers have proposed various approaches to tackle the challenging botnet problem. Unfortunately, Bot-masters and exploit kits developers have always responded with alternative methodologies or minor upgrades to their techniques and managed to evade detection approaches. Therefore, there is a need for an active monitoring and botnet detection mechanism. Continuous monitoring schemes can be developed that involve learning new patterns and adapt to the changes in the botnet evolution.

Modern botnets, such as HTTP botnets, heavily use AGDs as one of the preferred methods to evade detection. DGA-based malware try to contact C&C servers more than thousands of times in a day. The DGA domains are short-lived and difficult to blacklist. Hence it is very important to recognise the unique characteristics of DGA domains as part of any comprehensive strategy to detect botnets. Reverse engineering the Domain Generation Algorithm is a very tedious and difficult task, hence it is important to detect the generated domains themselves in a timely and efficient manner.

As a result of experimentation, we can say that *Length* is an extremely important factor in determining if the domain is normal or DGA-based. In this work, we can see that for domains with length below some threshold value, the characteristics of normal and DGA domains are very similar and as such, are difficult to distinguish. We developed in this work a simple model that uses information theoretic metrics and length threshold value as pivot to transparently and dynamically detect DGA domains of various lengths, whether short or long. Two different machine

learning classification techniques, namely, decision tree and random forests, yielded very encouraging detection performances.

While detecting automatically generated domains is our main focus in this work, identifying the malware family to which the DGAs belong would be beneficial.

5.2 Future Work

Future work will consist of expanding our feature space and algorithms to identify and categorize the specific algorithms used in generating the AGDs. Future work will also include testing the ability of the model for detecting novel, unseen form of AGD. This will be conducted by training the model on one dataset and testing it on another, totally different dataset. Novelty detection is important as one of the characteristics of malware such as botnets is evolution to escape detection.

References

1. Antonakakis M., Perdisci R., Nadji Y., Vasiloglou N., Abu-Nimeh S., Lee W., and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware", 21st Usenix Security Symposium, August 8-10, 2012.
2. Ma J., Saul L.K., Savage S., and G. M. Voelker, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs", Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 1245-1254, Paris, France — June 28 - July 01, 2009.
3. McGrath D. K. and M. Gupta, "Behind Phishing: An Examination of Phisher Modi Operandi', Proceedings OF 1ST USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 15, 2008, San Francisco, CA, USA.
4. Mowbray M, and J. Hagen, "Finding Domain-generation algorithms by looking at length distributions", 2014 IEEE International Symposium Software Reliability Engineering Workshops (ISSREW), Naples, Italy, 3-6 Nov. 2014.
5. Norvig P., "Natural Language Corpus Data*", Beautiful Data*, chapter 14, Pages 219-242, June 2009.
6. Sharifnya R, Abadi M. A novel reputation system to detect DGA-based botnets, in 2013 3th International eConference on Computer and Knowledge Engineering (ICCKE), 2013, pp. 417- 423.
7. Schiavoni S., Maggi F., Cavallaro L., Zanero S, "Phoenix: DGA-Based Botnet Tracking and Intelligence", In: Dietrich S. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2014. Lecture Notes in Computer Science, vol. 8550. Springer.

8. Yadav S., Reddy A. K. K., Reddy A.L.N., and S. Ranjan, “Detecting algorithmically generated malicious domain names”, In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC ’10). ACM, New York, NY, USA,2010, pages 48-61.
9. Wang W., and K. Shirley, “Breaking Bad: Detecting malicious domains using word segmentation”, In Proceedings of the 9th Workshop on Web 2.0 Security and Privacy (W2SP), 2015.
10. Weymes B., "DNS anomaly detection: Defend against sophisticated malware", May 28, 2013; Web. June 28, 2017.https://www.helpnetsecurity.com/2013/05/28/dns-anomaly-detection-defend-against-sophisticated-malware/
11. Zhao D., Traore I., Sayed B., Lu W., Saad S., Ghorbani A., and D. Garant,“Botnet detection based on traffic behavior analysis and flow intervals”, Elsevier Journal of Computers and Security, Volume 39, November, 2013, pages 2-16.
12. Zhao D. and I. Traore, “P2P Botnet Detection through Malicious Fast Flux Network Identification”, 7th International Conference on P2P, Parallel, Grid, Cloud, and Internet Computing -3PGCIC 2012, November 12-14, 2012, Victoria, BC, Canada
13. Feily M., Shahrestani A.,Ramadass S,“A Survey of Botnet and Botnet Detection.”, Third International Conference on Emerging Security Information, Systems and Technologies,2009, pages 268-272
14. Elaheh Biglar Beigi et al., "Towards effective feature selection in machine learning-based botnet detection approaches", Communications and Network Security (CNS) 2014 IEEE Conference on. IEEE, 2014.

15. J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," preprint arXiv:1611.00791, 2016
16. Alexa one million top websites, 2015. https://support.alexa.com/hc/en-us/articles/200449834-Does-Alexa-have-a-list-of-its-top-ranked-websites
17. Abakumov A, DGA repository, 2015, https://github.com/andrewaeva/DGA
18. Jacob J, "Building a DGA Classifier", September 30, 2014, http://datadrivensecurity.info/blog/posts/2014/Sep/dga-part1/ Accessed 15/10/2016
19. Fodor P ,"Grammars and introduction to machine learning", Stony Brook University, 2011, http://www3.cs.stonybrook.edu/~pfodor/courses/HON111/L5-NLP.pdf
20. Daniel Jurafsky , James H. Martin,"Language Modeling with Ngrams", Speech and Language Processing, August 7, 2017, https://web.stanford.edu/~jurafsky/slp3/4.pdf
21. Daniel Jurafsky, James H. Martin, , "N-Grams", Speech and Language Processing, September 2014, https://lagunita.stanford.edu/c4x/Engineering/CS-224N/asset/slp4.pdf
22. T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla, "Automatic extraction of domain name generation algorithms from current malware", in Proceedings of the NATO Symposium IST-111 on Information Assurance and Cyber Defense, Koblenz, Germany, September 2012.
23. P. Royal, "On the Kraken and Bobax botnets," 2008. [Online]. Available:www.damballa.com/downloads/press/Kraken Response.pdf, Accessed 20 September, 2016
24. B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel and G.Vigna, "Your Botnet is My Botnet: Analysis of a Botnet Takeover," Security, p. 635–647, 2009.

25. F. Leder and T. Werner, “Know your enemy: Containing conficker,” The Honeynet Project, 2009.