DETECTION OF ALGORITHMICALLY GENERATED MALICIOUS DOMAIN

Enoch Agyepong[1], William J. Buchanan[2] and Kevin Jones[3]

[1]Cyber Operations Team, Airbus, Corsham, UK
[2]School of Computing, Edinburgh Napier University, Edinburgh, UK
[3]Cyber Operations Team, Airbus Group Innovations, Newport, Wales, UK

ABSTRACT

In recent years, many malware writers have relied on Dynamic Domain Name Services (DDNS) to maintain their Command and Control (C&C) network infrastructure to ensure a persistence presence on a compromised host. Amongst the various DDNS techniques, Domain Generation Algorithm (DGA) is often perceived as the most difficult to detect using traditional methods. This paper presents an approach for detecting DGA using frequency analysis of the character distribution and the weighted scores of the domain names. The approach's feasibility is demonstrated using a range of legitimate domains and a number of malicious algorithmically-generated domain names. Findings from this study show that domain names made up of English characters "a-z" achieving a weighted score of < 45 are often associated with DGA. When a weighted score of < 45 is applied to the Alexa one million list of domain names, only 15% of the domain names were treated as non-human generated.

KEYWORDS

Domain Generated Algorithm, malicious domain names, Domain Name Frequency Analysis & malicious DNS

1. INTRODUCTION

Domain names and DNS services are often abused by cyber-criminals to provide them with an efficient and reliable communication link for malicious activities [34], [47]. Criminal activities involving Advanced Persistent Threats (APT), malware and botnets use DNS service to locate Command and Control (C&C) servers for file transfer and updates [16], [34]. Spammers also rely on DNS service to redirect users to scams and phishing websites [35]. Zhao et al. [47] explains that these cyber-criminal activities are often successful because DNS traffic is usually unfiltered or allowed through a firewall thereby providing a stealthy and undisturbed communication channel for cyber-criminals to operate.

Cyber-criminals in recent times have been designing malware to take advantage of certain Dynamic DNS (DDNS) capabilities such as the ability to change the IP address associated to a domain [28]. Whilst DDNS provides a useful feature for organisations that need to maintain consistent services, because they rely on a dynamic IP range allocated by their Internet Service Provider (ISP) [18]. Arntz [5] explains that, cyber-criminals exploit this feature to increase the survivability of their C&C server. Zhao et al. also state that, DDNS provide the capability for cyber-criminals to maintain persistent presence on a victim's machine once it has been compromised as they can easily change their IP and domain information [47]. Stevanovic et al. [35] calls DDNS, "agile DNS" and argue that, this feature poses a serious challenge to internet

Dhinaharan Nagamalai et al. (Eds) : ACSIT, ICITE, SIPM - 2018
pp. 13–32, 2018. © CS & IT-CSCP 2018 DOI : 10.5121/csit.2018.80802

security. Agile DNS uses dynamic hosting strategies in which domain names and IP addresses associated with a particular service change over time. Well-known DDNS capabilities often exploited by cyber-criminals include: Flux services such as IP-Flux and Domain Flux and the use of Domain Generated Algorithm (DGA) [29]. Amongst the DDNS techniques, DGA is noted as the most elusive and difficult to detect using traditional detection strategies such as signature based solutions.

A number of detection approaches and filtering techniques have been proposed by various scholars and researchers for detecting DGA and flux services. However, criminals continue to adopt DGA because they see it as easy to implement yet difficult to block [5]. Cyber-criminals also employ DGA because it ensures that their C&C network can elude security researchers and law enforcement [14]. Moreover, malware that uses static domain or IP address can easily be taken down or blocked, thereby hindering the operations of the criminal [34].

This piece of work proposes an approach to analysing domain names using frequency analysis of the distribution of letters to ascertain whether the domains were algorithmically-generated or human generated. This work does not seek to compete with the existing tools for detecting DGA but rather it seeks to complement existing works [1], [2], [3], [7], [18],[27],[44]. This paper expands upon previous work related to the detection of algorithmically-generated domain names [15], and seeks to answer the following research question:

1. With what certainty can a letter frequency distribution be used to identify algorithmically-generated domain names and differentiate it from human (legitimate) domains?

2. What are the limitations of systems that have the ability to identify DGA?

2. BACKGROUND AND RELATED WORK

2.1. Abuse of Domain Names

Although there is no single definition of what constitutes domain abuse [41], domain names registered for phishing, malware, botnets and those used for spamming falls into what may be classed as abuse [45]. Often these sorts of activities are recognised in most countries as illegal. Zhao et al. [47] point out that the flexibility of domain names allows cyber-criminals to easily move the IP address of their malware C&C servers using a range of techniques to avoid detection. Zhao et al. [47] also explains that the flexibility of DNS allows cyber-criminals to hide the source of their attack behind proxy servers. To ensure that they remain in control and to maintain persistent access to a compromised device, criminals generally implement some sort of C&C architecture to send and receive information from devices they control.

2.2. Malware Command and Control

Cyber-criminals like to maintain control of devices and hosts they compromise for long term benefits such as financial gain [16]. To achieve this objective, they usually plant some form of a backdoor or create a C&C channel to allow them to re-enter the system at will [26]. C&C activity is initiated when an attacker compromises an end–user device and transforms that device into a bot that listens out for instructions issued by the botmaster [8], [26]. Norton defines a bot as “a type of malware that allows an attacker to take control over an affected computer” [23]. This kind of activity is evident in many cyber-crimes and it is well documented by various writers [16], [33], [47]. Most system hacking methodologies present the implementation of C&C or planting backdoors as a key component of a structured attack [25].

Figure 1: Cyber Kill Chain by Lockheed Martin

Figure 1 above shows an easy to understand model depicting how a cyber-criminal implements C&C in attacking their victim in what is typically known as the Lockheed Martin Cyber Kill Chain [21]. Cyber criminals typically start their attack process with the **recon** (reconnaissance). Under recon the attacker gathers information about the target. Having gathered enough information about the target, the attacker then moves to the next stage of their attack process which involves **weaponisation**. During this phase, the attacker will identify a tool or a piece of malware they intend to use against the target. The attacker will then **deliver** that piece of malware, which they will then use to **exploit** their victim to allow them long term unrestricted access. In order for the attacker to return they will often **install** some form of a backdoor program. Unrestricted access is often achieved using **C&C**. The final stage is the **actions on objectives** which involves financial gain, theft or sabotage.

Although there are several types of C&C communication channels, for example, using of protocols such as Hypertext Transfer Protocol (HTTP), Internet Relay Chat (IRC) and many more [26], traditionally, cyber-criminals tend to rely on a **centralised topology** to control their victims by hardcoding their IP addresses or domain names in their malware binaries to allow persistent access between a server they control and a compromised device [16], [30]. Often, the cyber-criminals include one or more static domains or IP addresses in their source code as observed with the WannaCry ransomware [9], [40].

However, studies have suggested that there are were a number of problems with this approach [2], [16], [47]. Firstly, the hardcoded IP address can be identified by a security researcher reverse engineering the malware, allowing mitigating action to be taken that may involve blacklisting the IP address [16]. Secondly, if the C&C server goes down or the IP address is detected, the nefarious activity can be easily identified and blocked by a security administrator which means their compromised device will be out of the attacker's control [13],[47].

To overcome some of these challenges and to maximise the availability of C&C servers, cyber-criminals have resorted to using a range of methods that avoid detection of their criminal activities yet providing them with high availability and resilience [33]. Chen et al. [13] suggest that the Peer-to-Peer (P2P) botnets infrastructure was one of the first architectural structures that criminals used for overcoming the limitations of the centralised C&C network approach. This is also known as a **decentralised topology**. The Nugache and storm bot utilised the P2P architecture and more recently Waledac and Zeus [2], [16]. The advantage of P2P is that the majority of the bots do not communicate with the centralised machine, hence the IP address is not easily detected to be taken down [13]. However, P2P has some downsides and limitations. They are usually difficult to implement and maintain due to their setup architecture [16]. This has led to the introduction of a **locomotive topology** in which the C&C entities changes over time.

In an effort to combine the simplicity of C&C and the robustness of the P2P infrastructure, cyber-criminals now employ more dynamic strategies to maintain their C&C servers [2], [28]. These new strategies involve generating a large number of domain names using a variety of jurisdictions and service providers for C&C servers [33]. It also involves rapid changes of these domain names through the implementation of Domain Generation Algorithm (DGA). Yadav et al. [44] also points out that spammers also try to avoid detection by systems that rely on regular expression by using DGA.

The objective here is to evade detection or to elude a security engineer's attempt to easily implement a mitigation strategy. Kwon et al. [18] points out that the most common agile DNS implementations associated with malicious activities include: Domain-flux, IP-flux and the use of DGA with some fundamental differences. Zhao et al. [47], however, highlights some similarities such as the "short-life" between Flux services and DGA.

2.3. Domain-flux and IP-flux

Berger et al. [17] mentions that the operation of a malicious flux service is similar to a content-delivery network (CDN) service and argues that the theory behind CDN and malicious flux is the same. Zhao et al. [47] explains that whilst CDN is used for facilitating efficient delivery of web-servers content and services, malicious flux is a DNS technique used by criminals to facilitate C&C activity. CDN comprises of a large number of legitimate servers, whereas malicious flux consists of a group of malicious networks. Both Domain-flux and IP-flux involves the continual changing and re-assignment of multiple Fully Qualified Domain Names (FQDNs) to one or more IP addresses to avoid being taken down easily [18], [30]. However, there are some fundamental differences between these two terminologies which are explained below [42]. Examples of some historical malware that employed domain-fluxing are: Cycbot and Murofet [33]. The Zbot and Kazy are also well-known malware that utilised IP-flux or Fast-Flux in their operations [20].

In order to understand domain-flux, the concept of IP-flux is first presented. Historical use of a C&C server by botnets relied only on a single IP address, this meant that once the IP is discovered it was blacklisted and taken down, thereby disabling the communication channel [16]. To get around this problem, criminals began buying a set of IP addresses to ensure that if one IP address is identified, an alternative IP will be introduced to allow the communication to continue, thus implementing IP-flux as shown in Figure 2. However, given that most botnet and malware operating under C&C use DNS to resolve IP addresses to domain names, connecting multiple IP address to a single domain name also results in a single point of failure for the criminal as the malware can be blocked at DNS level [8]. This has led to criminals finding an alternative approach resulting in the development of Domain-flux.

Domain-fluxing uses a DGA to generate many domain names and then attempts to communicate with a small section of the generated domains [37]. Bilge et al. [8] explains that the use of domain names gives the attackers the flexibility and redundancy of migrating their servers with ease. If the malware cannot find the C&C server at its previous domain, it searches the list of domains generated until it finds one that works [16]. Other flux services for example Fast-Flux involve the rapid changes of DNS "A" records. Likewise, there is Double-Flux, in which both the DNS "A" record and the "NS" records are changed rapidly to make take-down much more difficult, yet allowing the cyber-criminal control of their C&C server.

2.4. Domain Generation Algorithm (DGA)

According to [27], the first known use of DGA for malicious purposes goes back to the Sality malware observed in February 2006. Plohmann suggests that Sality was followed by Torpig in July 2007 and later by the Kraken malware [27]. However, it was not until April 2008 that DGA became publically known when Kraken, Conficker and Szribi were released [5]. DGA periodically generates algorithmically domain names in order to contact the C&C server. A compromised device will generate a large set of random domain names (Figure 5) and queries each domain until one resolves to a C&C server [47].

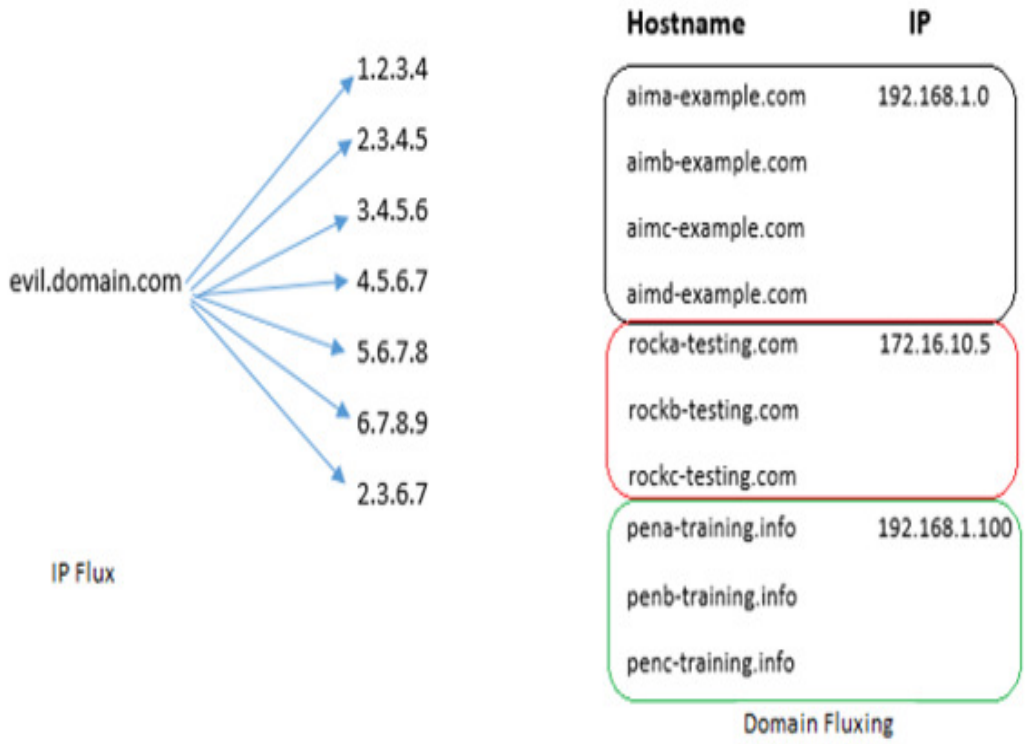

Figure 2: An example of IP-Flux and Domain-Flux

DGA is usually fed with a random seed which consist of numeric constants or a phrase, a time-based element, for example current date and time to dynamically generate often large and multiple FQDNs at run-time [28], [34]. The large number of domain names generated provides great agility and ensures that even if one or more of the domain names are eventually taken down or blacklisted, the compromised device will ultimately get the IP address of the re-allocated C&C server [33]. Antonakakis et al. [2] highlight that a compromised device running DGA will attempt to contact all the generated domains until one of the domains resolves to the IP address of the botmaster's C&C server. From the attacker's perspective, this technique is more advantageous because unless a security researcher can reverse engineer the algorithm the attacker will continue to be in control [47]. Antonakakis et al. [2] also suggest that even when one or two of the C&C servers are taken down, the criminals will simply have to register another domain and the bot will eventually get the IP address of the relocated C&C server. The difficulty is also increased by consistently changing the domain in use. Furthermore, the criminals do not have to include a hard-coded domain within the malware code [28].

Whilst there are some legitimate uses of randomly generated domain names, for example, Plohmann et al. [28] points out that early DGA were used as a backup communication mechanism. Also, Google Chrome queries three random looking domain names when started to act as a DNS check and to verify if Non-Existence Domain (thereafter NXDomain) rewriting is enabled [2], [10]. Ruan et al. [32] argues that in most cases the presence of DGA within network traffic can often signify illegal behaviour of some kind. In fact, Zhao et al. [47] points out that some malware such as Poison Ivy, and Gh0st instructs the attacker to create domains dynamically for locating C&C servers. Antonakakis et al. [2] also points out that Torpig, Srizbi, Bobax, Conficker-A/B and C are all malware that have employed DGA. For example, at its peak the Conficker-C worm, generated almost 50,000 domains per day using DGA [2], [44], [47]. Yet out of the 50,000 domain names generated, Conficker-C only queried roughly 500 of these domains per day. These domains were distributed across 110 Top Level Domain (TLD) [44]. Similarly,

both [36] and [44] highlighted how the Torpig botnet makes use of DGA and utilised the Twitter trend API (Application Programming Interface) as a seed to their algorithm to make detection and take-down difficult for security professionals.

2.5. Taxonomy of DGA

DGA differ widely in the way the algorithms are generated and seeded [28], [44]. [11] listed the main types of DGA as: static DGA, Date-based DGA, seed-based DGA and a combination of Date and Seed. Static DGA generates the same domain every time whereas the Date-based method uses the current date as an input for the algorithm to generate domains and the Seed-based DGA utilises hardcoded seed as input for their algorithm. Plohmann et al. [28] on the other hand proposes a taxonomy based on the characteristics of the seed source and identifies four different generation scheme. These include Arithmetic-based DGAs which compute a sequence of characters that have ASCII values constituting the alphabet of the DGA; a Hash-based DGA which relies on hexdigest representation of a hash to generate an algorithmically-generated domain; a wordlist-based DGA which concatenates a string of words and finally there is the permutation-based DGAs. Plohmann et al. [28] explains that permutation-based DGA generates domain names using a permutation of an initial domain name.

Barabosch et al. [6] also highlight four different DGA types shown and points out that, time and causality are the two main possible parameters for any DGA. The first class of DGA family is the deterministic and time independent DGA (**TID-DGA**) and argues that this type of DGA generates the same set of domain names every time they are executed because its use of a static seed in its algorithm. An example of malware that uses TID is Kraken [6]. The second category of DGA family is the time dependent and deterministic DGA (**TDD-DGA**). The seed used in TDD-DGA changes at a regular interval. However, precomputation of the domain names are still easy because it uses a deterministic algorithm. An example of DGA that used TDD-DGA was the Conficker worm. The next type of DGA according to [6] is the non-deterministic and time dependent DGA (**TDN-DGA**). Under TDN-DGA, the seed cannot be anticipated; hence precomputation is not possible. An example of malware that uses TDN was Torpig. Torpig uses the popular trending topics on Twitter as a seed [2]. The final category is the time independent and non-deterministic DGA (**TIN-DGA**). Barabosch et al. [6] suggest that malware that rely on TIN-DGAs have not been seen in the wild yet and argues that this class of DGA might work for small domain name, however, the chances of getting in touch with a C&C server, decreases with the increase of the domain name length.

2.6. Previous and related work

Various techniques have been proposed for the detection of Dynamic DNS domain names in DNS traffic. Yadav et al. [44] proposed a methodology for detecting “domain-fluxing and IP-fluxing” in DNS traffic by analysing patterns associated with domain names that are generated algorithmically. They observed that compromised hosts using Domain-fluxing and DGA often generate domain names that exhibit characteristics that are vastly different from legitimate domain names. Their approach relied on signal detection theory, statistical learning strategies and measures such as Kullback-Leibler divergence, Jaccaard index and Levenshtein distance to detect algorithmically-generated domain names. They also computed the distribution of alphanumeric characters as well as bi-grams in all domains that are mapped to the same set of IP addresses. Although they suggest that their methodology was successful in detecting malicious domains associated with malwares including Conficker, they mention that when their technique is applied to large data sets of test data their systems efficacy decreases significantly resulting in high false positive rate. For example, the outcome of 50 test words at 100% detection rate produced 50% false positives using a Jaccaard index.

Antonakakis et al. [1], also attempts to classify domain names (as malicious or legitimate) by dynamically assigning them a reputation score using a system known as Notos. Notos assigns a low reputation score to a domain suspected of being involved in a malicious activity such as phishing and spam campaigns and a high reputation score to those used for a legitimate purpose. Using a classification and a clustering algorithm, they were able to dynamically blacklist malicious domains, by modelling legitimate and malicious domains, thereby mitigating cyber-attack effectively. The findings of their work had true positive rate of 98%. However, Notos has the limitation of not being able to assign reputation scores to domain names with very little historic information, a key feature of most DGA.

Doyle uses frequency analysis as an approach to detect pseudo-random domain names [15]. His work reports that the pseudo-random generated domain were much more uniform and linear than real-world data sets. However, there were a number of fundamental limitations that makes generalisation difficult. Firstly, the data sample was limited to domains that use .net, .com and .org as their TLD. The problem here is that DGA are not limited to those TLDs. Further, the malicious sample was limited to the Conficker worm. Given that all algorithms are not the same, additional tests will be needed to test whether randomly generated domain names using a variety of algorithms will results in distributions of letters that are vastly different from the letter frequency distribution of human (legitimate) domains.

Bilge et al. suggests a passive DNS analysis technique and detection system known as EXPOSURE that is useful for detecting domain names that are involved in malicious activities [8]. EXPOSURE focuses on extracting certain behavioural features from DNS such as time-based features, for example life-span of the domain (short-life), TTL value and features such as the percentage of numerical characters in a DNS domain. Using these they were able to automatically identify a wide variety of malicious domains. A limitation to their system is that an attacker could adapt their attack to evade features EXPOSURE has been trained to detect. For example an attacker could attempt to change TTL values to evade EXPOSURE. Antonakakis et al. also proposes another passive DNS analysis tool known as Kopis that monitors DNS traffic at the upper levels of the DNS hierarchy [1]. Whilst Kopis is able to detect malicious domain with minimal false positives, it needs a certain mandatory time period to analyse the traffic before results are presented. Hence, DGA bots that operate within a smaller epoch will be inactive by the time Kopis provides its detection results; a limitation of their system.

Antonakakis et al. proposes *Pleiades* that utilises a clustering and classification algorithm to analyse DNS traffic and identify DGA domain names [2],[3]. *Pleiades* relies on domain name requests that result in Name Error Responses (NXDOMAIN) or unsuccessful domain name resolution responses to identify DGA-bots. *Pleiades* searches a cluster of NXDomains that have similar syntactic features and are queried by multiple potentially compromised devices during a given epoch and uses a statistical learning strategy to build a model of the DGA. However, *Pleiades* is unable to learn the exact DGA thereby generating a certain number of false positives and false negatives. Despite these limitations, when applied to real-world DNS traffic obtained from an ISP, *Pleiades* was still able to identify domain names generated using DGA.

Zhao et al. combine signature and anomaly-based solutions to design a detection system that can be placed at the edge of a network to detect DNS traffic that shows signs of malware infection based on 14 certain characteristics extracted through big data analysis of malware associated with DNS activities [14]. Some of the features extracted are similar to those identified by [2] however, their work focuses on detecting Advanced Persistent Threats (APTs), malware and C&C activities within DNS traffic. They highlight certain characteristics of a domain name that can identify potential signs of malicious domains in use. A limitation of their work is that bots that use IP addresses to directly locate the C&C server rather than domains are not detected.

Sharifnya & Abadi [33] proposes the DFBotkiller system that builds on earlier works by other researchers [1], [2], [44]. They argue that previous work does not effectively consider the history of activities on the monitored network; potentially resulting in a high proportion of false positives. To address this problem, they proposed a system that incorporates the history of domain groups linked to malicious activities and suspicious domain failures to automatically assign a reputation score. This approach combines certain characteristics associated with DGA, such as generating a large number of NXDomains and the alphanumeric distribution of Algorithmically-generated domain names, to develop a reputation system that assigns a high negative reputation score to hosts involved in suspicious domain based activities and a low negative score to legitimate domains.

More recent studies by [28] and [43] also present an alternative way of detecting DGA. Plohmann et al. [28] uses a bottom-up approach that relies on reverse engineering to analyse 43 DGA based malware. They identified 253 seeds used in previous and recent DGAs through analysis of domain names to build a ground truth about DGA with no false positive. They built a web-service known as DGArchive to ascertain whether a queried domain originated from a DGA. Wang et al. [43] proposes a DGA-Based detection system known as DBob that monitors only DNS traffic as opposed to previous works that tend to monitor all traffic. DBob, in comparison with previous systems provides a much more efficient way of detecting DGA-based activity as it does not rely on prior training of the system. A limitation of DBob is that devices are clustered based on the similarities of their query behaviours during a detection time window. This means that their system is unable to detect a compromised host if it has never been attacked before or attempted to connect to a C&C server.

2.7. Limitations of Machine Learning and Signature Based Systems

A wide variety of complementary techniques exist to detect algorithmically-generated domain names however, these techniques are overly reliant on machine learning and complex computational measures. Whilst machine learning is useful in predicting the possible outcome of an event based on previously learned data sets (training data), Wang et al. highlight that systems that rely on prior training can be evaded by DGA bots if the underlying algorithm or the learned features change [43]. A change in the learned behaviour can easily evade a system based on machine learning [4]. Jadav et al. [44] also points out the resource intensive nature of machine learning. Similarly, when it comes to rule-based systems and Intrusion Detection Systems (IDS) that use signatures, Ruan et al. explains that these systems are insufficient as a strategy as they are unable to respond to the dynamic nature of DGA [13]. All these systems suffer from false positives, making it important for further studies in strategies for detecting malicious DGA.

2.8. Frequency Analysis of Letters in a domain name

An alternative strategy to machine learning, reverse engineering of DGA bots and signature-based solution is the use of frequency analysis techniques to examine the character distribution of domain names. Frequency analysis is deeply rooted in descriptive statistics and relates to the number of times an event occurs. As a technique, it is also used in cryptanalysis, to understand how often a character or a group of characters (letters) appears in a cipher-text in order to decrypt and understand a secret text or ciphers for which the key is unknown [38]. This strategy relies on the fact that in any language, certain letters occur with varying frequencies. Whilst the focus of this work is to understand the distribution of alphanumeric characters within a domain name, an appreciation of key measures used within frequency analysis is also important.

[2], [15], [44] includes frequency analysis as part of their strategy to detect DGA albeit they differ in how they employ this technique. *Pleiades* which was developed by Antonakakis et al. uses the

character frequency distribution during the DGA discovery and clustering phase [2]. Yadav et al. for example uses a computation of the bigram of alphanumeric characters (two consecutive characters) within the domain and not the domain or the FQDN [44]. Although [15] attempts to use frequency analysis as a sole technique for the detection of DGA he points the high volume of false positive when using this strategy. However, false positive tend to be present in nearly all the currently available detection strategies. Given the small data sample used by Doyle for his baseline, it is possible that a larger data sample may achieve better results and reduce the level of false positive. This is something that can be investigated in future work.

Doyle [15] compares a small sample of legitimate domains against a well-known English frequency table and proposes a baseline for identifying DGA. His approach is based on a concept well explained by the Unicode Consortium. The Unicode Consortium explains that Domain Names were originally designed to support only the American Standard Code for Information Interchange (ASCII) [39]. They highlight that even non-ASCII Unicode characters are transposed into a special sequence of ASCII characters. This means that as far as DNS system is concerned, all domain names are just ASCII. ASCII as a character encoding standard denotes English characters as numbers with each letter being assigned a number from 0 to 127. Observations from previous studies suggest that the alphanumeric or character distribution of letters in Algorithmically-generated domain names are vastly different from pronounceable words. The fact that domain names are translated into ASCII makes frequency analysis a viable technique for detecting DGA.

Novig [24] points out that although the English language has 26 alphabet letters they do not appear equal amounts of time in an English text. Some letters are used more frequently than others and argues that for instance, the letter "Z" appears less frequently in words than the letter A or E [31]. By using frequency analysis, of letters or a group of letters, the distribution of the characters can be analysed and compared to that of English words to differentiate between legitimate and DGA domains. Figure 3 and Figure 4 below shows the frequency table used.

Letter	Frequency	Letter	Frequency
e	12.7020%	m	2.4060%
t	9.0560%	w	2.3600%
a	8.1670%	f	2.2280%
o	7.5070%	g	2.0150%
i	6.9660%	y	1.9740%
n	6.7490%	p	1.9290%
s	6.3270%	b	1.4920%
h	6.0940%	v	0.9780%
r	5.9870%	k	0.7720%
d	4.2530%	j	0.1530%
l	4.0250%	x	0.1500%
c	2.7820%	q	0.0950%
u	2.7580%	z	0.0740%

Figure 3: Relative Frequency of Letters in the English Language.
http://en.algoritmy.net/article/40379/Letter-frequency-English

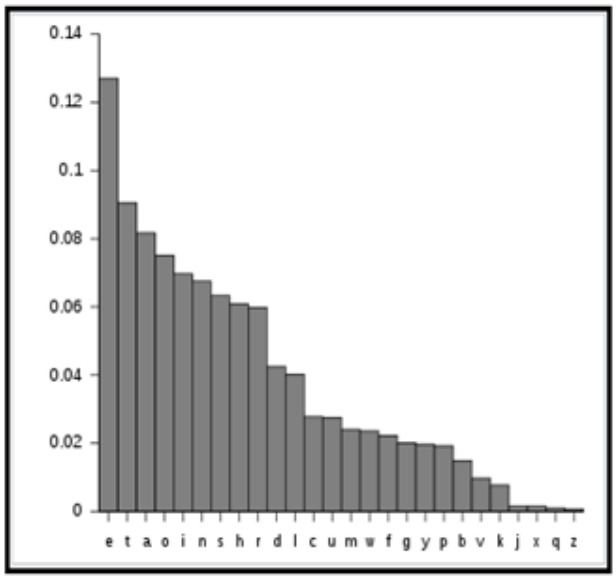

Figure 4: Relative Frequencies of Letters in English text http://en.algoritmy.net/article/40379/Letter-frequency-English

3. METHODOLOGY

3.1. Research Approach

The motivation for this work and the approach used here is based on observations by previous studies in this area such as Yadav et al., (2010) and Zhao et al., (2015) that suggests that algorithmically-generated domains do not use pronounceable or real words and hence have a distribution of characters that will be vastly different from legitimate domains. *This hypothesis will be tested using frequency analysis of the distribution of letters in the domain name and if it is true, a schema will be developed that relies on the total weighted score of a domain name to predict potentially suspicious Algorithmically-generated domain names.* Reverse engineering of DGA algorithm will not be covered in this work as it is resource and time intensive [44], moreover reverse engineering of DGA has been previously covered by Plohmann et al., (2016). Classification of different types of DGA and the identification of C&C servers fall outside the remit of this work.

3.2. Selection of Study Samples

The malicious dataset samples were selected based on a convenience samples. However, a simple random technique will be applied to both datasets to select a subset. An observation from the material reviews is that, most DGA tends to have string of characters of length greater than 8. Conficker, for example, which was rated among the top 5 DGA-based crimeware was originally designed to use random strings of between 8-11 characters [14], [29]. According to [17] detecting randomness in words or letters with six (6) characters is often a very difficult task even for humans. This argument is supported by [22] who assert that even humans will often struggle to detect randomness in a string of characters that are less than eight (8) characters in length. With this in mind, a decision was made to select a sample data from the available data set with the characteristics described above.

3.3. Data Collection

Both primary and secondary data sets was used in this work. The secondary data was taken from well-known publically available third-party databases to ensure that experimental activity and testing reflected real-life usage. In addition to the secondary data, a prototype DGA that uses only the English alphabetic characters "a-z" was also used as primary data. The reason for including the customised DGA is that, it would allow for testing of a DGA that uses only the English 26 alphabetic characters to examine how the character distribution will compare. There are different types of DGA and it is impossible to predict the kind of algorithm being utilised by the cyber-criminal hence the prototype allows for some form of testing of a previously unknown DGA [11]. The legitimate domains are obtained from Alexa. Alexa has acted as the source of data for a number of other works in this area [8], [19], [18], [33]. The malicious or algorithmically-generated domain names were sourced from third parties such as bambenekconsulting.com.

3.4. Frequency Analysis of the domain names

A customised script was applied to the domain names (excluding protocols and TLD's) to generate letter frequency distribution. The character distribution of legitimate domains was compared to a standard frequency distribution table of English text to examine their relationship. Doyle [15] observed a close similarity between legitimate domains and a well-known English frequency table, albeit his sample of legitimate domains was relatively small in comparison to the sample used in this study. Doyle [15], propose a weighted scoring technique that assigns scores to the characters and uses this information as the basis to detect a DGA. Figure 5 below shows the formula for assigning the weighted scores. The weighted score approach described above will be used to produce a cumulative frequency plot for further statistical analysis of the domains.

3.5. Weighted score-based analysis

A weighted score-based analysis was used in this work to assign scores to the domain names. However, there are differences between this work and the work conducted by [15]. Firstly, the upper limit of his domain weighted score will differed from this study because of the size of his good data which was used as the reference point. Secondly, his analysis of the domains were limited to second level domains. This work uses the entire domain excluding TLDs and protocols. Thirdly, the malicious dataset used in [15] was limited to a single malware family (Conficker). This does not give a good indication of how well his system will perform when used for the detection of other DGA based malware domains.

$$w = \frac{\sum_{i=0}^{n} x_i}{n} \times 1000$$

Where:
$w = weighting$
$n = num.letters\ in\ domain$
$x = frequency\ lookup\ of\ letter\ i$

Figure 5: Weighted Score Formula

4. EXPERIMENTAL ACTIVITIES

4.1. Analysis of the Data Sample

The non-malicious dataset (legitimate domain) used in this work was sourced from Seobook.com and contains the Alexa list of one million domains. The malicious datasets (DGA samples) were taken from bambenekconsulting.com and comprises of 5 days of DGA feeds. The DGA samples

differ in the way in which the algorithms were seeded. Differentiating these samples of malware is important because as pointed out by [44], different malware employing DGA are seeded differently. As an example, Conficker generated 250 domains every three hours using the current date and time at UTC as the seed. Torpig's seeding was based on the most popular trending topics on twitter [44]. The DGA samples obtained include: Cryptolocker, Kraken, Tinba, P2P Gameover, Zeus, Shifu, Banjori, Pykspa, Dyre, Ramnit, Cryptowall, Ranbyus, Simda, Murofet, Qakbot and Necurs. All TLDs and protocols were removed leaving just the domain names. Stripping off the TLD was crucial because there are over a thousand TLD's, any of which can be used by criminals [44]. It is difficult to predict which TLD a criminal will use. It was also felt that adding the TLD would not give an accurate representation of letters used in the generation of the domains.

The legitimate domains were used as the baseline or reference point of what a "good domain" should look like and domain names that differed vastly (based on measure of dispersion) from this baseline are treated as suspicious. In other words, the distribution table of well-known (legitimate domains) acted as the reference table for all good domains. Three groups of malicious samples; Ramnit, Tinba and Qakbot each containing 40,000 domains were selected for the initial experimental activity. The reason behind selecting 40,000 is that there is already a frequency table for 40,000 English words as observed in the literature review, hence where a comparison is required, that frequency table can be used. A customised script was applied to the domain names to create a frequency table of the character distribution.

Table 1: Character Frequency distribution of a random sample of 40,000 DGA domain names against the Alexa one million domains.

Relative Frequency of letters in English text.	English Text	Alexa 1 million	40,000 English Words	40,000 Random DGA
E	0.12702	0.095	0.1202	0.0429
T	0.09056	0.061	0.091	0.0375
A	0.08167	0.0921	0.0812	0.0294
O	0.07507	0.074	0.0768	0.0392
I	0.06966	0.0722	0.0731	0.0426
N	0.06749	0.06	0.0695	0.0396
S	0.06327	0.0645	0.0628	0.0375
H	0.06094	0.0249	0.0592	0.0428
R	0.05987	0.0639	0.0602	0.0376
D	0.04253	0.0322	0.0432	0.0425
L	0.04025	0.0472	0.0398	0.0391
C	0.02782	0.0379	0.0271	0.0428
U	0.02758	0.0333	0.0288	0.0375
M	0.02406	0.0338	0.0261	0.0392
W	0.0236	0.012	0.0209	0.0374
F	0.02228	0.0171	0.023	0.0421
G	0.02015	0.0243	0.0203	0.0421
Y	0.01974	0.0163	0.0211	0.0369

P	0.01929	0.03	0.0182	0.0379
B	0.01492	0.0238	0.0149	0.0434
V	0.00978	0.0136	0.0111	0.0374
K	0.00772	0.0188	0.0069	0.0396
J	0.00153	0.0055	0.001	0.0425
X	0.0015	0.0065	0.0017	0.0372
Q	0.00095	0.0021	0.0011	0.0375
Z	0.00074	0.0065	0.0007	0.0154

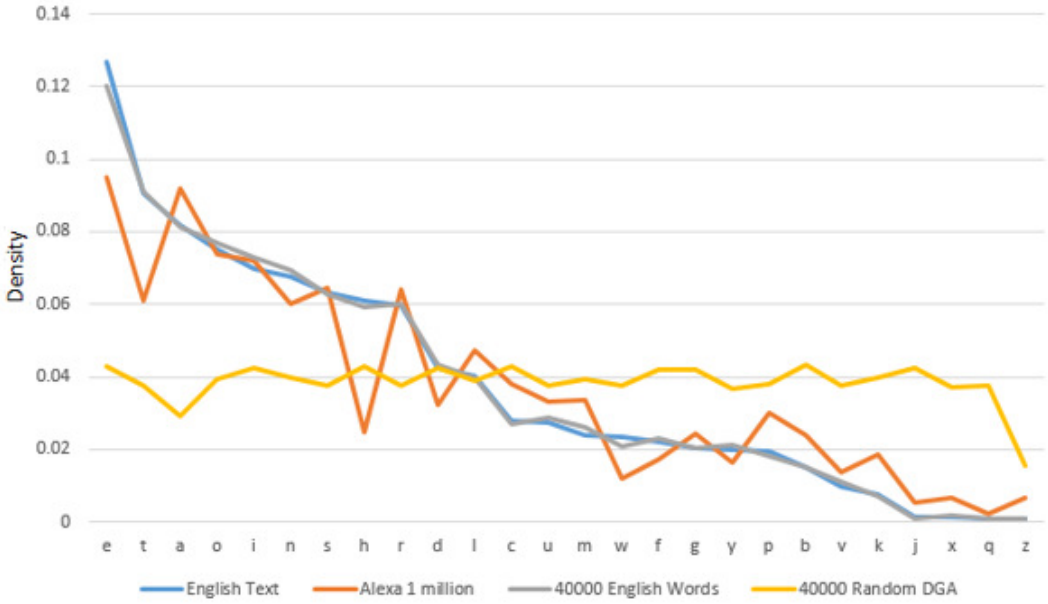

Figure 6: A line chart showing how a random sample of 40,000 DGA domain names compare to English words and text frequency table.

For the purposes of this work, the list of domains from Alexa is assumed as the "correct" frequency distribution for domain names and will therefore be used as the baseline. Domains that differ vastly from this baseline are treated as suspicious.

4.2 Findings

The character frequency distribution of the domain names from Alexa does show a close similarity with the English letter frequency table, albeit there were one or two characters such as "t" and "h" that had a wide variation as observed in Table 1 and Figure 6. In order words the frequencies are different from English text. The similarities can be linked to the fact that many "domain names follow English nouns, verbs, adjectives or a combination of these" [15].

By applying the above formula in Figure 5, the frequency of characters in the domain name were added together and divided by the number of characters in the domain. The outcome was then multipled by 1000 to facilitiate the ease of calculation. Each domain that was calculated was expected to fall within a range of:

$0 < w < 95$

The limit of 95 is obtained by multiplying the highest frequency letter "e" from the good domain (Alexa). This limit is roughly 10% less than the upper limit used by [15]. All the scores for the domain should fall within the range; when using the frequency table of the good domains taken from the Alexa top 1 million. For instance by applying this formula (using a customised script), the weighted score for the domain dailymail.co.uk and americanexpress.com can be computed.

When the TLD is excluded, "dailymail" will get a score of 56.1444 whereas "americanexpress" will get a score of 64.4267.

The calculations for "*dailymail* " is shown below:

$(0.0322_{\mathbf{d}} + 0.0921_{\mathbf{a}} + 0.0722_{\mathbf{i}} + 0.0472_{\mathbf{l}} + 0.0163_{\mathbf{y}} + 0.0338_{\mathbf{m}} + 0.0921_{\mathbf{a}} + 0.0722_{\mathbf{i}} + 0.0472_{\mathbf{l}}) / 9$ X 1000 =56.14444

Likewise the formula when applied to "*americanexpress"* achieves a score of 64.4267.

$(0.0921_{\mathbf{a}} + 0.0338_{\mathbf{m}} + 0.095_{\mathbf{e}} + 0.0639_{\mathbf{r}} + 0.0722_{\mathbf{i}} + 0.0379_{\mathbf{c}} + 0.0921_{\mathbf{a}} + 0.06_{\mathbf{n}} + 0.095_{\mathbf{e}} + 0.0065_{\mathbf{x}} + 0.03_{\mathbf{p}} + 0.0639_{\mathbf{r}} + 0.095_{\mathbf{e}} + 0.0645_{\mathbf{s}} + 0.0645_{\mathbf{s}})/15$ X 1000 =64.4267

The weighted scores of legitimate domains could be compared against the DGA domains. Doyle (2010) used the cumulative weighted scores of the domain names to analyse 3381 samples of Conficker DGA. Although, he had some interesting results including being able to block 87.35% of pseudo-random domains, his reference sample (legitimate data) was not large enough to provide a baseline of what a good domain should be like. In addition, his legitimate domains were limited to domains that use .net, .org, and .com. Furthermore, his study was limited to analysis of Conficker which makes it difficult to generalise his findings. This project uses a larger data sample and assumes the one million list of domains as a good reference table for domain names compared to the 3381 legitimate and malicious DGA used in [15]. The cumulative weighting score for a random sample of legitimate and malicious DGA comprising of Qakbot, Tinba and Ramnit is shown in Figure 7 below:

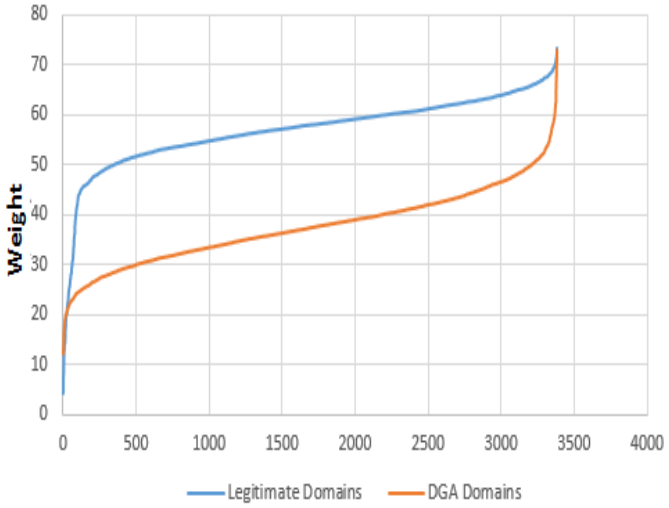

Figure 7: A plotted cumulative weights of malicious domains versus legitimate domain names.

The plotted graph (Figure 7) and the data used for the generation of the graph can be used to carry out some basic blocking of malicious DGA.

4.2.1. Test 1 - Block 80% of the sample DGA and determine the percentage of false positive.

80% blocking of the DGA domains would require a score of < 43.538 to be blocked. This score when applied to the legitimate domains would result in 3.13% blocking of these domains (over-blocking). The over-blocking in this experiment seemed to achieve a better result than [15]. This could be put down to using a large legitimate sample to create the baseline of the assumed "correct" frequency table of legitimate domains.

4.2.2. Test 2 - Block 95% of the sample DGA and determine the percentage of false positives.

This test increased the number of DGA domains that need to be blocked to 95%. A score of < 50.39 would need to be blocked in this instance. Applying this score to the domains would result in over-blocking of the legitimate samples by 11.56%.

4.2.3. Test 3 - Aim for a 20% false positives.

This test set a limit on the number of false positives generated from the legitimate domains. 20% false positive rate is too high for real-world application [15]. However, this test still gives a good indication of how the proposed schema will respond under this condition. In this situation a score of <52.95 is required. This results in 97.5% blocking of the DGA domains.

4.2.4. Test 4 - Aim for no more than 5% false positives.

This test set a limit on the number of false positive to 5%. This would require a score of < 46.22 to be blocked. This score achieves a blocking rate of 88.7% of the DGA domains. Given that the information being used is the character distribution of the letters in the domains, blocking 88.7% of the algorithmically-generated domain name with 5% over-blocking of legitimate domain would be reasonable. Although the tests achieved some impressive outcomes with the data set, there are still the issues of false positives. Plohmann et al. points out that malware that uses DGA has different characteristics and features, in terms of how they are seeded and how the algorithm works [28]. With this in mind, additional testing was conducted with a range of DGA. Overall findings shows that in almost all cases the malicious sample when plotted were linear in comparison to non-malicious samples.

5. DISCUSSION AND IMPLICATION

5.1. Efficiency of the design approach

Overall, the experimental activities were successful in terms of the ability to detect DGA domains and the results obtained supports findings from previous studies and the hypothesis that states that the distribution of alphanumeric characters in DGA domain names is vastly different from that in legitimate domain names. A top ceiling weighted score of 95 was applied to all domain names. By doing this, no domain name was expected to have a weighted score greater than 95. This is lower than the numerical value of 106 used by [15].The discrepancies in these figures can be attributed to the large sample size used in this study. As expected none of the domains had a weighted score greater than 95.

In almost all cases, the malicious sample shows a linear distribution when compared to legitimate domain names. For example, Tinba, Pyskpa, Ramnit and Qakbot were all observed to have a linear distribution. Legitimate domains tend to follow a similar pattern to the frequency

distribution of well-known English words and text. This observation can be attributed to the fact that most domain names follow nouns, verbs, adjectives and or a combination of these [15]. It can also be argued that humans will often use a memorable pattern of words whereas machine generated algorithms do not have such limitations.

In comparing this work to other approaches, it is evident that frequency analysis of the characters in the domain name combined with the weighted score approach can dynamically detect Algorithmically-generated domains in a more efficient manner than systems that rely on static signatures albeit there are some limitations. This approach does not have the same limitations as signature based solutions that lack the flexibility to respond to changes in behaviour. The statistical-based approach overcomes the non-dynamic nature of rule-based solutions. Even though this approach is solely dependent on frequency analysis of the letters in the domain names, it achieves a lower false positive when tested against some malicious DGA domain names than the KL and edit distance used by [44]. Yadav et al. reports 50% false positives in their work [44]. Other systems however do have a much lower false positive at the detection of DGA. For example, Kopis proposed by Antonakakis et al. has a much smaller false positive rate [1].

5.2. Study Limitations

Despite, some of the good outcomes generated using the proposed approach, there are a number of disadvantages to the weighted based approach. Weights and scores are susceptible to numerical variances due to human bias and other uncertainties [46]. This can be seen in the upper limits that are applied in this work. Whilst the research did not directly manipulate the ceiling of the domain names, there is uncertainty of what that ceiling will be if this test is repeated using a different number of data set.

In addition to the limitations of the weighted score approach highlighted by [46], there are problems with very short domain names that will always result in a much lower score in comparison to longer domain names. For example, bt.com and cnn.com for example will generate a much lower score than a domain such as americanexpress.co.uk or dailymail.co.uk. Also, there is nothing stopping criminals from using algorithms or seeding techniques that generates domains that fall within legitimate English text or letter frequency distribution. The approach presented in this work does not take into consideration domain names that appear to be machine-generated but are used for legitimate purposes. Also, provision was not made for misconfigured applications or domain names resulting from typos. These may be flagged as possible DGA. Future work could focus on addressing these issues.

6. CONCLUSION AND IMPLICATION OF THE FINDINGS

Based on the experimental activities, it was observed that domain names achieving a weighted score of < 45 or domains with a weighted score of approximately 20% deviation from the average weighted score of Alexa 1 million are often Algorithmically-generated (non-human generated). The average score of the domains on the Alexa list is 55.19. A legitimate domain (human generated) typically achieves a weighted score > 45 with a top limit of 95 based on earlier calculations. When a weighted score of < 45 is applied to the Alexa one million list of domains, only 15% of domains were treated as non-human generated. This is fairly impressive given that the detection is based purely on weights of the domain. Also, a group of NXDomains at a given epoch achieving a score of < 45 would be deemed as potentially malicious for further investigation. The issue of blocking legitimate traffic means that this approach can have a negative impact in real-life applications if it is used as a sole approach, rather the weighted based approach can be used by DGA detection systems to help access the overall validity of a domain. Future work could investigate how to reduce or eliminate the blocking of legitimate domain by

measuring the randomness of characters in domain names by using measures such as Kolmogorov Complexity (K-complexity) and Hidden Markov Model (HMM).

ACKNOWLEDGEMENTS

Many thanks also goes to Adam Wedgbury of Airbus Research Group Innovations for the initial idea. We will also like to thank Dr Ebenezer Paintsil, Shane Murnion and Richard Macfarlane for their feedback.

REFERENCES

[1] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon, (2011, August) "Detecting Malware Domains at the Upper DNS Hierarchy". In USENIX security symposium Vol. 11, pp. 1-16, 2011

[2] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou II, S. Abu-Nimeh, W. Lee, and D. Dagon, "From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware". In USENIX security symposium Vol. 12, 2012.

[3] M. Antonakakis, R. Perdisci, N. Vasiloglou, and W. Lee, Detecting and Tracking the Rise of DGA-Based Malware. The magazine of USENIX & SAGE, 37(6), 15-24, 2012

[4] H. Armstrong, (2015, July 05). Machine that learn in the wild. Available: https://www.nesta.org.uk/sites/default/files/machines_that_learn_in_the_wild.pdf

[5] P. Arntz, (2016, June 27). Explained: Domain Generating Algorithm. Available: https://blog.malwarebytes.com/security-world/2016/12/explained-domain-generatingalgorithm/

[6] T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla, (n.d.). Automatic Extraction of Domain Names Generation Algorithms from Current Malware. Available: https://net.cs.unibonn.de/fileadmin/user_upload/wichmann/Extraction_DNGA_Malware.pdf

[7] A. Berger, A. D'Alconzo, W.N. Gansterer, and A. Pescapé, "Mining agile DNS traffic using graph analysis for cybercrime detection". Computer Networks, 100, 28-44, 2016

[8] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, (2011, February). EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In Ndss.

[9] R. Brandom, (2017, May 17) Registering a single web address may have stopped a global malware attack -Finding the kill switch. Available: https://www.theverge.com/2017/5/13/15635050/wannacry-ransomware-kill-switchprotect-nhs-attack.

[10] CERT Polska (2015, May 26). DGA botnet domains: on false alarms in detection. Available: https://www.cert.pl/en/news/single/dga-botnet-domains-false-alarms-in-detection/

[11] A. Chailytko, and A. Trafimchuk, (2015, July 17). DGA clustering and analysis: mastering modern, evolving threats. Available: https://www.botconf.eu/wp-content/uploads/2015/12/OK-S01-Alex-Chailytko-Alex-Trafimcuk-DGA-clustering-and-analysis-mastering-modern-evolvingthreats.pdf.

[12] M. Chapple, M. (2017, July 28.). Evaluating and tuning an Intrusion Detection System. Available: http://searchsecurity.techtarget.com/tip/Evaluating-and-tuning-an-intrusion-detectionsystem.

[13] R. Chen, W. Niu, X. Zhang, Z. Zhuo, and F. Lv, "An Effective Conversation-Based Botnet Detection Method". Mathematical Problems in Engineering, 2017.

[14] Damballa (2012, July 17). DGAs in the Hands of Cyber-Criminals: Examining the state of the art in malware evasion techniques. Available: https://www.damballa.com/wpcontent/uploads/2014/02/WP_DGAs-in-the-Hands-of-Cyber-Criminals.pdf Accessed.

[15] R. Doyle, (2010, June 17). Frequency analysis of second-level domain names and detection of pseudorandom domain generation. Available: http://ryandoyle.net/assets/papers/Frequency_analysis_second_level_domains_June_2010_RDoyle.pd

[16] N. Goodman, A Survey of Advances in Botnet Technologies. arXiv preprint arXiv:1702.01132, 2017

[17] A. Kololkoltsev, (2015, July 28). Machine learning technique to detect generated domain names. Available: https://www.youtube.com/watch?v=9wB_ovM5C0M.

[18] J. Kwon, J. Lee, H. Lee,and A Perrig, "PsyBoG: A scalable botnet detection method for large-scale DNS traffic". Computer Networks, 97, 48-73, 2016

[19] J. Lee, and H. Lee, "GMAD: Graph-based Malware Activity Detection by DNS traffic analysis". Computer Communications, 49, 33-47, 2014

[20] D. Mahjoub, (2013, September). "Monitoring a fast flux botnet using recursive and passive DNS: A case study". In eCrime Researchers Summit (eCRS), 2013 (pp. 1-9). IEEE.

[21] L. Martin, (2014). "Cyber Kill Chain®". Available: http://cyber. lockheedmartin.com/hubfs/Gaining_the_Advantage_Cyber_Kill _Chain. pdf.

[22] M. Namazifar, (2015, July 17). Detecting Random strings: A language based approach. Available: https://www.youtube.com/watch?v=70q5ojxNuv4.

[23] Norton (2016, July 17). Bots and Botnets. Available: https://us.norton.com/botnet/

[24] P. Norvig, (2012). English Letter Frequency Counts:Mayzner Revisited or ETAOIN SRHLDCU. Available: http://norvig.com/mayzner.html Accessed 02 July 2017

[25] S.P. Oriyano, CEH v9: Certified Ethical Hacker Version 9 Study Guide. John Wiley & Sons 2016

[26] V. Oujezsky, T. Horvath, and V. Skorpil, "Botnet C&C Traffic and Flow Lifespans Using Survival Analysis". International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, 6(1), 38-44, 201

[27] D. Plohmann, (2015). DGAArchive – A deep dive into domain generating malware. Available:https://www.botconf.eu/wp-content/uploads/2015/12/OK-P06-Plohmann-DGArchive.pdf

[28] D. Plohmann, K. Yakdan, M. Klatt, J. Bader, and E. Gerhards-Padilla, (2016). "A Comprehensive Measurement Study of Domain Generating Malware". In 25th USENIX Security Symposium (USENIX Security 16) pp. 263-278, 2016 USENIX Association.

[29] P. Porras, H. Saidi, and V. Yegneswaran, V. "An analysis of Conficker's logic and rendezvous points". Technical report, SRI International.2009

[30] M. Poor, SANS 503: Intrusion Detection in-depth. The SANS institute, 2015

[31] D. Rodriguez-Clark, (2017). Frequency Analysis: breaking the code . Available:http://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html

[32] W. Ruan, Y. Liu, and R. Zhao, "Pattern discovery in DNS query traffic". Procedia Computer Science, 17, 80-87, 2013

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

[34] U. Sternfeld, (2016). Dissecting domain generation algorithm: eight real world DGA Variants. Available: http://go.cybereason.com/rs/996-YZT-709/images/Cybereason-Lab-Analysis-Dissecting-DGAs-Eight-Real-World-DGA-Variants.pdf

[35] M. Stevanovic, J.M. Pedersen, A. D'Alconzo, S. Ruehrup, and A. Berger, "On the ground truth problem of malicious DNS traffic analysis". Computers & Security, 55, 142-158, 2015.

[36] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, and G. Vigna,(2009, November). Your botnet is my botnet: analysis of a botnet takeover. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 635-647). ACM.

[37] A. Almomani "Fast-flux hunter: a system for filtering online fast-flux botnet". Neural Computing and Application 29(7), 483-493, 2018

[38] C. Swenson, Modern cryptanalysis: techniques for advanced code breaking. John Wiley & Sons, 2008

[39] The Unicode Consortium. Internationalized Domain Names (IDN) FAQ. Available: http://unicode.org/faq/idn.html. Accessed 07 June 2017.

[40] US-CERT (2015). Indicators Associated with WannaCry Ransomware. Avaiable: https://www.us-cert.gov/ncas/alerts/TA17-132A Accessed 30 May 2017

[41] P. Vixie, "What DNS is not". Commun. ACM, 52(12), 43-47, 2009

[42] L. Vu Hong, (2012). DNS Traffic Analysis for Network-based Malware Detection.

[43] T.S. Wang, H.T. Lin, W.T. Cheng, and Chen, C. Y. "DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis". Computers & Security, 64, 1-15, 2017

[44] S. Yadav, A.K.K Reddy, A.L. Reddy, and S. Ranjan, (2010, November). Detecting Algorithmically-generated malicious domain names. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 48-61). ACM.

[45] M. Young, M. (2014). Domain name abuse is a 4 letter word. Available: http://www.circleid.com/posts/20141112_domain_name_abuse_is_a_4_letter_word/

[46] J. Yuventi, and S. Weiss, (2013). Probabilistic Consideration Method for Weight/Score-Based Decisions in Systems Engineering-Related Applications.

[47] G. Zhao, K. Xu, L. Xu, and B. Wu, (2015). "Detecting APT Malware Infections Based on Malicious DNS and Traffic Analysis". IEEE Access, 3, 1132-1142, 2015

AUTHORS

Enoch Agyepong is a Lead Cyber Security Engineer at Airbus. He hold Bachelor's Degree in Computing and Master's Degree in Advanced Security And Digital Forensics from Edinburgh Napier University, United Kingdom

William J. Buchanan is a Professor in the School of Computing at Edinburgh Napier University, and a Fellow of the BCS. He was awarded an OBE in the Queen's Birthday awards in June 2017. Bill currently leads the Centre for Distributed Computing, Networks, and Security and The Cyber Academy. He has published over 27 academic books, and over 250 academic research papers, along with several awards for excellence in knowledge transfer, and for teaching. Bill has been included in a list of the "Top 50 Scottish Tech People Who Are Changing The World".

Kevin Jones is the Head of Cyber Security Architecture, Innovation and Scouting. He holds PhD in Computer Science and Mathematics from De Montford Univeristy, UK