Phishing URL Detection Through Top-Level Domain Analysis: A Descriptive Approach

Orestis Christou, Nikolaos Pitropakis, Pavlos Papadopoulos, Sean McKeown and William J. Buchanan

School of Computing, Edinburgh Napier University, Edinburgh, United Kingdom
Christouorestis@gmail.com, {N.Pitropakis, pavlos.papadopoulos, S.McKeown, B.Buchanan}@napier.ac.uk

Keywords: Phishing Detection, Machine Learning, Domain Names, URL

Abstract: Phishing is considered to be one of the most prevalent cyber-attacks because of its immense flexibility and alarmingly high success rate. Even with adequate training and high situational awareness, it can still be hard for users to continually be aware of the URL of the website they are visiting. Traditional detection methods rely on blacklists and content analysis, both of which require time-consuming human verification. Thus, there have been attempts focusing on the predictive filtering of such URLs. This study aims to develop a machine-learning model to detect fraudulent URLs and be used within the Splunk platform. Inspired from similar approaches in the literature, we trained the SVM and Random Forests algorithms using malicious and benign datasets found in the literature and one dataset that we created. We evaluated the algorithms' performance with precision and recall reaching up to 85% precision and 87% recall in the case of Random Forests while SVM achieved up to 90% precision and 88% recall using only descriptive features.

1 Introduction

The past few years have seen an outburst of high-impact breaches and issues, showing that sole reliance on traditional mitigation and prevention approaches is not ideal for providing adequate protection against such fluctuant environments. Domain Name System (DNS), being one of the principal elements of the web, is not only a prime target for attacks involving system downtime but also used as a means for the execution of further and more complex social engineering and botnet attacks. In the report by IDC (Fouchereau and Rychkov, 2019) 82% of the companies undertaking their survey reported suffering from at least one DNS-related attack. The average number of attacks experienced per company was 9.45, placing the average cost of damages at $1,000,000. From the "dangerously diverse" and ever-growing threat landscape, the most ubiquitous DNS-related threat was phishing, with Malware, DDoS and Tunnelling coming not far behind it. FireEye's recent report (Hirani et al., 2019) on a grand-scale DNS hijacking attack by alleged Iran-based actors for record manipulation purposes reinforces this notion.

Adversaries do not have to be networking experts, nor possess knowledge of the underlying operation of the DNS to misuse it. To execute a successful phishing attack, the only thing an adversary needs to do is to select the right domain name to host their malicious website. Instead of merely choosing a generic and innocent-appearing name, the process of selecting the domain name may include techniques such as homograph spoofing or squatting (Kintis et al., 2017; Moubayed et al., 2018; Nikiforakis et al., 2014).

One of the most popular approaches of dealing with such websites is using blacklists (OpenDNS, 2016). It is simple and accurate as each entry in the blacklist is usually manually verified as malicious. The problem with the latter approach is that it requires frequent updating of the blacklist through constant scanning for new entries. Moreover, the systems creating these blacklists tend to have high operational costs, which lead to the companies requiring payment to access them. Usually, adversaries that utilise these malicious URLs do not keep them active for very long as they risk being detected and blocked.

Machine Learning techniques use features extracted from the URLs and their DNS data to analyse and detect whether they are malicious or benign. Usually, methods which rely on the analysis of the content of such URLs come at a high computational cost. (Blum et al., 2010), compute MD5 hashes of the main index of their webpages and compare them with the hashes of known phishing sites. In their work, they mention that this technique is easily bypassed

just by obfuscating the malicious contents. This limitation constrains approaches to exclusively analysing the URL strings to classify the URLs. More recently, López et al. (López Sánchez, 2019) attempted to detect phishing by using "Splunk" and taking into consideration only the use of typosquatting (Nikiforakis et al., 2014) and homograph squatting.

To the best of our knowledge, our work is the first attempt that takes into consideration all the forms of domain squatting and produces an automated mechanism to detect malicious phishing URLs, thus increasing the situational awareness of the user against them. The contributions of our work can be summarised as follows:

- The Machine Learning system is trained using descriptive features extracted from the URL strings without utilising host-based or lexical (bag-of-words) features. The domain names come from real-world, known phishing domains (blacklist) and benign domain names (whitelist).
- The popular classification algorithms SVM and Random Forests are used and compared empirically based on their performance.
- The process is heavily automated as it relies on Splunk software, thus being fit against new datasets and generating alerts when new malicious entries are detected.

The rest of the paper is organised as follows: Section 2 briefly describes the related literature with regards to phishing; Section 3 introduces our methodology while Section 4 describes the results of our experimentations along with their evaluation. Finally, Section 5 draws the conclusions giving some pointers for future work.

2 Background and related work

2.1 Phishing

"Phishing" as a term did not exist until 1996 when it was first mentioned by "2600" a popular hacker newsletter after an attack on "AOL" (Ollmann, 2004). Since then, there has been an exponential increase in phishing attacks, with it becoming one of the most prevalent methods of cybercrime. According to Verizon (Verizon, 2019), phishing was part of 78% of all Cyber-Espionage incidents and 87% of all installations of C2 malware in the first quarter of 2019. In the earlier report by Verizon (Verizon, 2018), it is reported that "78% of people didn't click a single phish all year", that means that 22% clicked. Therefore, that could be rephrased to be: "One in five people clicks on a phishing e-mail at least once a year". Moreover, users only reported an alarming 17% of the campaigns ran. It is also emphasised that even though training can reduce the number of incidents, "phish happens". Kaspersky has recorded over 11 million blocked redirect attempts to phishing sites just in the first quarter of 2019 (Vergelis and Shcherbakova, 2019), a 31.5% increase from the last quarter of 2018. Since only a single e-mail is needed to compromise an entire organisation, protection against it should be taken seriously.

Cyber-criminals use phishing attacks to either harvest information or steal money from their victims through deceiving them with a reflection of what would seem like a regular e-mail or website. By redirecting the victim to their disguised website, they can see everything they insert in any forms, login pages or payment sites.

Cyber-criminals copy the techniques used by digital marketing experts to guarantee a high click rate. They also tend to take advantage of the fuss created by viral events or stories to increase their potential victims. Vergelis and Shcherbakova (Vergelis and Shcherbakova, 2019), reported a spike in phishing redirects to apple sites before each new product announcement.

Regular phishing attacks do not care about their target; they are usually deployed widely and are very generic so that they can be deployed to target as many people as possible. A **Spear-Phishing attack** is similar; it targets a specific individual instead. Information gathering against the victim needs to be performed beforehand to craft a successful spear-phishing e-mail. A more advanced version of this attack is a **Whaling attack**; A spear phishing attack that specifically targets a company's senior executives to obtain higher-level access in the organisation's system. Targeted phishing attacks are increasingly gaining popularity because of their high success rates (Krebs, 2018).

Pharming is a different approach in which the attacker will attempt to direct their victims to a malicious website. There are various methods to execute this without even needing the user to make a mistake. For example, if the attacker manages to poison the cache of the local DNS server fake records, then they can redirect the user to their malicious website.

2.1.1 Attacks

If the end-goal of a phishing attack is to ensure that the victim is ultimately redirected to the phishing website without being aware of it, then the adversary needs to use several techniques to guarantee that. Some of those techniques include: **URL hiding** a most commonly used technique, where the attacker obfuscates a malicious URL in a way that does not raise any suspicions and ultimately gets clicked on by

the victim. One way to execute this would be to replace a valid URL link with a malicious one.

Shortened links from services such as Bitly can be used to obfuscate malicious links easily. There is no way to know the actual destination of an obfuscated link without visiting it.

Homograph spoofing is a method which depends on the replacement of characters in a domain name with other visually similar characters. An example of that would be to replace 0 with o, or I with 1 or an exclamation mark (Rouse et al., 2019). So, for a URL "bingo.com" the spoofed URL would be "b1ng0.com". Characters from other alphabets such as Greek have also been used in the past for such attacks. The Greek 'o' character is visually indistinguishable from the English "o" even though their ASCII codes are different and would redirect to different websites. **Squatting** is the term used to describe the use of a variation of a popular domain name for spoofing purposes. **Polymorphism** in phishing was initially a synonym for squatting as it was only applied to URLs. Now polymorphism is also applied in the contents of phishing websites and e-mails. By making minor alterations to the e-mail contents it is much easier to bypass conventional anti-phishing mechanisms (Jain and Gupta, 2017). Content polymorphism is addressed using visual similarity analysis of the contents; an early example of such an application is illustrated by Lam et al. (Lam et al., 2009).

Typosquatting is a similar method to homograph spoofing, but it targets common typographic errors in domain names. For example, an attacker could use the domain "www.goooogle.com" to target users who incorrectly type "google.com" or to trick them into clicking on a regular link. (Moubayed et al., 2018) combat this issue using a Machine Learning approach. They use the K-Means Clustering Algorithm to observe the lexical differences between benign and malicious domains and extract the features needed to detect them successfully. They propose a majority voting system that takes into consideration the outputs of five different classification algorithms. In the report by Proofpoint (Proofpoint, 2018), the most popular typosquatting approach is to swap an individual character, followed by inserting an additional one.

Combosquatting is different from typosquatting as it depends on altering the target domain by adding familiar terms inside the urls. An example of this technique would be "bankofscotland-live.com" or "facebook-support.com". Research performed by Kintis et al. (Kintis et al., 2017) show a steady increase in the use of combosquatting domains for phishing as well as other malicious activities over time. It is also reported that combosquatting domains are more resilient to detection than typosquatting. Moreover, they report that the majority of the combosquatting domains they were monitoring remained active for extended periods, sometimes exceeding three years. This suggests that the measures set in place to counter these are inadequate and that if that remains as the status quo, then combosquatting could grow into a genuine and dangerous threat.

Soundsquatting targets voice-operated software with the use of words that sound alike (homophones). In their research, Nikiforakis et al. (Nikiforakis et al., 2014) show that for a domain "www.test.com", an adversary may use dot-omission typos ("wwwtest.com"), missing-character typos ("www.tst.com"), character-permutation typos ("www.tset.com"), character-replacement typos ("www.rest.com") and character-insertion typos ("www.testt.com"). The same author and his team (Nikiforakis et al., 2014) illustrate how they used Alexa's Top one million domain list to create and register their soundsquatting domains, measuring the traffic from users accidentally visiting them. Through their research, they have proven the significance of taking into account homophone confusion through abuse of text-to-speech software when tackling the issue of squatting.

2.1.2 Suggested Defences

The term Passive DNS refers to the indirect collection and archiving of DNS data locally for further analysis. In the early days of passive DNS (pDNS) URL analysis (Spring and Huth, 2012), where privacy was still not considered an issue, the pioneer system for malicious domain detection through pDNS was Notos (Antonakakis et al., 2010) with its reputation-based classification of domains. Notos extracts a variety of features from the DNS queries and creates a score for each entry to represent the likelihood of it being malicious. The system gathered DNS traffic collected from two ISP locations in the USA and extracted information such as geographical locations, the number of IP addresses historically related to a domain and the number of malware samples related to IP addresses that a domain points to. A similar approach is taken for EXPOSURE (Bilge et al., 2011), expanding upon the work of Notos. EXPOSURE is a large-scale pDNS analysis system developed using a gathered dataset of 100 billion entries. Bilge et al. (Bilge et al., 2011), differ in their approach by operating with fewer data compared to Notos. Khali et al. (Khalil et al., 2016), expand upon the work of Notos and Exposure by focusing on the global associations between domains and IPs instead of looking at their local features. This way, they also address any privacy issues as they only extract information relevant

to their research from the gathered dataset. Because of their alternative approach, they view their work as complementary to Notos and Exposure. Okayasu and Sasaki (Okayasu and Sasaki, 2015), compare the performance of SVM and quantification theory in a similar setting. SVM was proved to be superior in their comparison.

Since the lexical contents of malicious URLs play a significant role in their victim's susceptibility, squatting detection should play a vital role in their detection. Kintis et al. (Kintis et al., 2017), study a subcategory of domain squatting, as mentioned in section 2.1.1 named "combosquatting". For their analysis, they use a joint DNS dataset comprised of 6 years of collected DNS records from both passive and active datasets that amount to a total of over 450 billion records. They find that the majority of combosquatting domains involve the addition of just a single token to the original domain. While it succeeds in establishing that combosquatting is a real threat, there is no mention of any future directions for research on this topic.

A novel approach is taken by Blum et al. (Blum et al., 2010), where URLs are classified without the need for host-based features. They found that lexical classification of malicious URLs can rival other conventional methods in accuracy levels. Their dataset was created utilising a technique called Deep MD5 Hashing (Wardman et al., 2010). The technique is used to compare the contents of known malicious websites to those being tested by comparing their Kulczynski 2 coefficients to check for their similarity (Kulczyński, 1928). Lin et al. (Lin et al., 2013), propose a similar ML approach, which can detect malicious URLs by only looking at the URL strings. They use two sets of features to train their online learning algorithm: lexical and descriptive. The lexical features are extracted by taking the name of the domain, path and argument of each entry and using a dictionary remove less useful words from them. The descriptive features are static characteristics derived from the URLs such as total length or symbol count. The main focus in their approach is to reduce the resources required for the analysis, which they achieve with a 91% hit rate. Both Notos (Antonakakis et al., 2011) and EXPOSURE (Bilge et al., 2011) scraped the surface of using descriptive features in URL analysis by measuring the domain length and character frequency. However, they did not dive into any depth because utilising other features such as TTL, geographical locations and historical IP address relationships were much more effective. The separate analysis of features based on their respective categories is a common trend within the literature. Darling et al. (Darling et al., 2015) show that classification speed and accuracy increases significantly compared to other more complete approaches when the classification system is created based on lexical features.

A Machine Learning system that uses the K-Means algorithm to label the data before applying the ensemble learning classifier mentioned earlier is proposed in the paper by Moubayed et al. (Moubayed et al., 2018). Shorter length domain names with fewer unique characters were found to be more likely to be benign than malicious. The ensemble classifier outperformed the individual algorithms in their tests. While promising, the algorithms considered should not be selected solely based on their popularity but rather their efficiency when implemented together. The authors do not mention if any other combinations were considered. The number of features taken into consideration is scarce, and the features are mostly length and character count related. Mamun et al. (Mamun et al., 2016) achieve a 97% average classification performance using a similar approach. They use the random forests algorithm in their lexical analysis of the URLs. Through their approach, they find that the Random Forests algorithm yields significantly better results than multi-class classification. K-Nearest Neighbours placed 2nd with an average performance of 94%.

Malicious URLs vary in their nature depending on their purpose, for example, a phishing URL might take advantage of squatting methods to deceive their victims while a botnet C&C will probably use a random generator as its looks are not essential and its lifespan might be limited. Da Luz and Marques (da Luz, 2014) expand upon this, building upon the work of Notos (Antonakakis et al., 2011) and Exposure (Bilge et al., 2011) to detect botnet activity using both host-based and lexical features. They give a comparison of the performance of the K-Nearest Neighbours, Decision Tree and Random Forests algorithms, showing that Random Forests performed significantly better. Moreover, in their feature significance comparison, the number of digits divided by the domain name length is shown to be the most influential feature. Feroz et al. (Feroz and Mengel, 2015) use a very similar methodology to target phishing domains specifically. Their work uses the K-Means algorithm to perform clustering using the lexical and host-based features, creating a new set of URL ranking features which is in turn used in the classification. Their results demonstrate a significant increase in accuracy when the clustering features are included.

Nikiforakis et al. (Nikiforakis et al., 2014) mention the issue of evaluating domain names comprised of foreign words since they use an English dictionary

to detect and replace accidental words. The issue becomes even more complicated as they propose the detection of which language the domain name is written in. Multilingual domain names would also be flagged as false positives in their system as it is tough to distinguish what language each word token is written in. Another issue that is raised is the splitting of the domain into different words, as there is no space separator in URLs. They extracted soundsquatting domains from the Alexa top 10k dataset and managed to classify them with an 18.9% false-positive rate. This paper does not present a solution to the problem but rather an evaluation of a new squatting method.

Lin et al. (Lin et al., 2013), makes the distinction between descriptive and lexical features. The purpose of that division is to separate the features derived directly from the domain names strings and the features derived using their bag-of-words model. They use the Passive-Aggressive algorithm to classify their dataset and then use the Confidence Weighted algorithm to alter the characteristic's weight based on their "confidence". Their model operates much more efficiently than other content-based models and is compatible with the volatile lifetime of malicious URLs.

Our approach differentiates from these as it only emphasises the descriptive characteristics of URLs in order to observe and attempt to improve the performance of a model without taking into consideration host-based or lexical (bag-of-words) features. Moreover, none of the approaches in the literature explored the creation and use of a model in a widely used platform such as Splunk to provide alerting capabilities to automate the detection of malicious URLs.

3 Methodology

As malicious parties continue to abuse DNS to achieve their goals, the means of stopping them should also be constantly developed. When observing the literature historically, it can be deduced that modern approaches are becoming more and more focused on the detection of specific problems. Therefore, following the same trend, our work will be assisted by the Splunk platform to train and use a classifier to detect phishing domains through their extracted descriptive features. This section will describe the methodology undertaken to select and prepare the training datasets, choose and extract the features, train the classifier and use it within the regular Splunk environment. Figure 1, illustrates the architecture of the proposed system and Table 1 shows our test environment's technical specifications.

Table 1: System Specifications

Model	Inspiron 7537
CPU	Intel® Core™ i5-4210U @1.70GHz
OS	Windows 10 (64-bit)
RAM	6 GiB

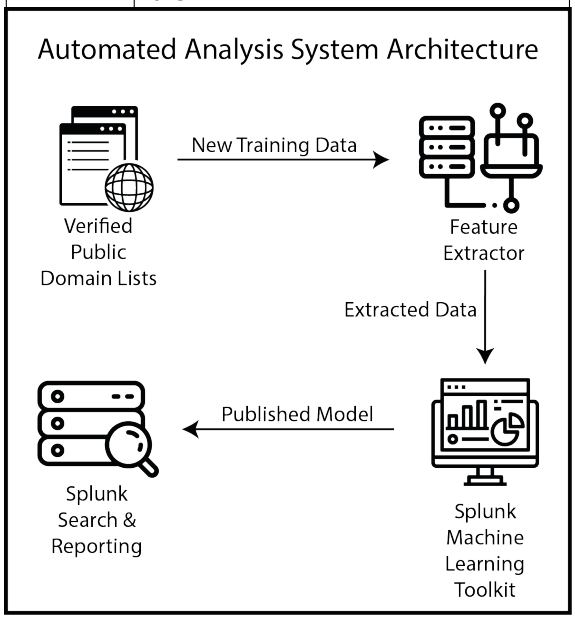

Figure 1: System Architecture Diagram

3.1 Dataset Selection

The quality of the prediction of a ML algorithm is strongly related to the quality of its training set. The Machine Learning approach will require a supervised learning algorithm, and therefore, the samples will need to be labelled as either 'benign' or 'malicious'. To reduce bias in our results, three tests were conducted using a total of six lists, three whitelists and three blacklists.

The first benign list was derived from TLDs in the Alexa top 1M domain database as of September 2019. This database contains 1 million entries of the most popular websites worldwide. As this list contains domains ranked by popularity, we manually verified their authenticity and content the first 5,000 domains from this list. We therefore populated our whitelist using the 5,000 most popular Alexa domains.

The first malicious list was created using Phishtank's active blacklist (OpenDNS, 2016). The blacklist consists of more than 400,000 phishing domain entries and is continuously updated with active domains. 5,000 of those domains were selected to populate the blacklist. The two lists are joined, and overlapping domain names are removed to avoid creating any noise. More entries from the blacklists are included in the following tests.

As the Alexa database is not validated for potentially malicious entries like combosquatting domains

as proved in the related literature (Kintis et al., 2017), we turned our attention to established datasets used by the cyber security community. Therefore, the legitimate and malicious lists provided by Sahingoz et al. (Sahingoz et al., 2019) were used together in the second test. The legitimate list is reported to have been cross-validated and thus can be used at full scale.

The phishing/legitimate URL set published in "Phishtorm" (Marchal et al., 2014) is used in the third test. The comparison of the performance of our feature selection using each of the different sets will be crucial in determining their relevance through the elimination of dataset bias from the algorithm's perspective.

3.2 Analysis

The previous sections have illustrated how miscreants can misuse DNS in their attempts to perform phishing attacks. From the knowledge extracted from the literature, a set of features can be selected and extracted from the gathered pDNS data. The set of features will allow for the classifier to divide the domain names into either benign or malicious.

3.2.1 Feature Extraction

The Splunk ML Toolkit has no functionality for extracting features from strings, and therefore, the features were extracted manually using python libraries. The Pandas Python library was used to import the two datasets into python for the extraction of the features. A Type column was created to mark entries as either benign or malicious as the two datasets were combined in a single dataset. IP addresses that were listed as domain names in blacklists were removed automatically using regex. Using a simple while loop, it was possible to iterate through the newly formatted joint dataset and create the features one by one.

We extracted a total of 18 features (See Tables 2 and 3) from each domain name in the DNS dataset. Since the benign dataset did not include other details such as the TTL or the path, certain types of features could not be derived. Therefore, more weight is given to analysing the lexical characteristics of our domains. The features are split into two groups: descriptive features and statistical features. The full list can be seen below in Table 2. The rationale behind the division of the features into these two categories is that descriptive features are simple variables derived directly from the domains while statistical features are derived from applying mathematical statistic operations on either the strings themselves or the descriptive features. A joint CSV file was created containing both the entries from the benign and the malicious datasets. An additional column was created to flag each entry as either malicious or benign.

Table 2: Descriptive features

ID	*Feature Description*
1	Count of URL unique characters
2	Count of Domain unique characters
3	Count of Suffix unique characters
4	Domain length
5	Suffix length
6	Total length
7	Count of Domain numbers
8	Count of URL numbers
9	Count of Suffix numbers
10	Count of symbol characters in domain
11	Count of symbol characters in Suffix
12	Total count of symbol characters

Table 3: Statistical features

ID	*Feature Description*
13	Domain Character Continuity Rate
14	Suffix Character Continuity Rate
15	Shannon entropy of domain string
16	Shannon entropy of suffix string
17	Standard deviation of the two domain levels' entropy.
18	Mean of the entropy of the two domain levels.

3.2.2 Descriptive features

A total of 12 descriptive features were extracted from each domain name string. These features were extracted based on the reasoning of the previous approaches mentioned in the research. Malicious domain names tend to have a higher number of symbols or numbers than benign ones, either because of squatting or because they are randomly generated. Therefore, we extracted features 10-12 to represent the number of symbols and numbers found in the different parts of the domain.

Malicious domains also tend to be longer than benign ones (Moubayed et al., 2018). However, after observing the entries in the phishing dataset, it was noticed that many entries would have longer subdomains but short domains. This would mean that even though they would look disproportionate, they would still be flagged as being of a length similar to benign entries. To counter this issue features 4-6 were set to contain the length of each domain part.

The number of unique characters also differs in malicious URLs because legitimate website owners tend to choose simpler and easier to remember words for their URLs. Using this reasoning, the number of

unique characters in the domain and suffix of each URL was used to populate the features 1-3. Moreover, since those unique characters are often numbers, features 7-9 were selected to represent the number of numeric characters present in each URL.

3.2.3 Statistical Features

Features 12 & 13 constitute the character continuity rate of the domain and suffix. In general, as mentioned earlier, website owners tend to go for simpler names for memorisation purposes. Because simpler domain names are usually more expensive to buy, it is unlikely for attackers to pay large sums for a domain that will most likely serve them for a short period. (Lin et al., 2013), use this idea to design the character continuity rate feature. To create this feature, the domain name is split into tokens of sequential characters based on their type (letter, number or symbol). Once the domain is split, the length of each token is measured and compared to the other tokens in its respective category. Then, the longest token for each character type is selected, and their total length is added together and divided by the total length of the token.

Take for example a domain string of "abcdef-12345ab1-ab12". It will be split into the following tokens: "abcdef" , "-" , "12345" ,"ab" , "1" , "-" , "ab" , "12". The longest letter token is "abcdef" which has a length of 6. The longest number and symbol tokens are "12345" and "-" with lengths of 5 and 1 respectively. Therefore, 6+5+1=12 and 12 divided by the total length of 20 will equal 0.6, which is the character continuity rate. Features 15-16 contain the Shannon entropy of the domain and suffix strings. This feature is used to detect randomised domain strings or at least detect randomisations within them (Lin et al., 2013). Shannon entropy H is calculated using the formula seen in Equation 4, where p_i is the chance for a character i to appear in a given string (Marchal et al., 2012).

$$H = \sum_i p_i log_b p_i \tag{1}$$

Equation 1: Shannon Entropy

In our scenario, p_i is replaced with the count of different characters divided by the length of the string. Features 17 and 18 are the mean and standard deviation of the features 15-16. These features were extracted to test if using a more median number would produce better results than using the initially extracted entropies.

3.3 Training, Application and Alerting

To train the algorithms, the training set containing all the features and the malicious and benign labels was exported to a CSV file. Splunk was set to monitor that CSV file so that if any changes needed to be made to it, they could be updated in Splunk instantly. This experiment was split into Tests 1, 2 & 3. Test 1 was performed using a set of 10,000 data entries from the Alexa and Phishtank datasets (Alexa, 2019; OpenDNS, 2016). A 50/50 split was performed on the datasets for the training phase using Splunk to provide a better outlook of the algorithms' efficiency by using one half for training and the other for testing.

The chosen algorithms for this experiment are SVM and Random Forests because of their reported performance in the literature as mentioned earlier. Test 1.1 will examine the performance of the Random Forests algorithm with minimal changes to its default configuration of the parameters: infinite maximum depth, features and maximum leaf nodes, ten N estimators and two minimum samples per split. Test 1.2 will take a similar approach against the SVM algorithm, with slight alterations to the C and Gamma parameters which are set by default to 1 and 1/500 respectively.

Test 2 was separated into Test 2.1 and Test 2.2 to evaluate the SVM and Random Forests algorithms using a larger dataset of 70,000 data entries (Sahingoz et al., 2019). Likewise, Test 3 was divided into Test 3.1 and Test 3.2 to compare SVM and Random Forests with the Phishstorm dataset (Marchal et al., 2014). Test 4 serves as a "what if" scenario that allows the comparison of all the available algorithms in the Splunk ML toolkit to see if there could have been alternatives not mentioned in the literature.

After training the algorithm, Splunk will be configured to periodically fit the algorithm on a continuously monitored file so that any new entries are checked immediately for their maliciousness. If the algorithm is accurate enough, it will be configured for scheduled re-training to ensure that it is up to date with recently found phishing domain entries.

4 Results & Evaluation

This section will include the outcome of the experiment using the methodology described previously in the form of tables. The results for each test will be displayed, explained and evaluated. The tables will show the performance of each of the algorithms against a small, medium and large-sized dataset. Finally, the selected features will be evaluated for their importance.

4.1 Results

4.1.1 Training

The first algorithm to be tested was Random Forests in Test 1.1. In Table 4, the performance of the algorithm is evaluated. Moreover, the fine-tuning of the algorithm's parameters to achieve its full potential can also be seen. The final run used ten N estimators, ten minimum samples per split, two maximum features, two minimum samples per split and infinite maximum leaf nodes.

Table 4: Test 1.1 Random Forests Alexa Evaluation

N Estimators	Max Depth	Max Features	Precision	Recall
10	∞	∞	0.87	0.86
10	10	∞	0.89	0.86
10	10	2	0.89	0.87

Test 1.2 implemented the SVM algorithm against the same dataset as Test 1.1, with an initial precision & recall of 0.89 and 0.86 respectively as shown in Table 5. The algorithm's performance increased immensely by simply using a larger C value to increase the hyperplane's flexibility. However, reducing the influence of the points placed far from the hyperplane resulted in a drop in accuracy. As shown in Table 5, a precision of 0.90 and a recall of 0.88 was achieved after the tweaking.

Table 5: Test 1.2 SVM Alexa Evaluation

C	Gamma	Precision	Recall
1	1/18	0.89	0.87
1	1/50	0.83	0.83
10	1/18	0.90	0.88

Test 2.1 evaluated the performance of the Random Forests algorithm using a larger dataset containing 70,000 entries. Once again, the features were tweaked until the perfect combination was found. As shown in Table 6, the performance of Random Forests peaked with a 0.84 precision and recall when 10 Estimators (Decision Trees) and an infinite max depth of nested statements were set. The alteration of the minimum samples per split had no impact on the overall performance, and therefore, it was kept to its default value. The decrease of the maximum number of features to consider per split negatively impacted the overall accuracy and thus was kept as the default value.

The same procedure as earlier is repeated using the SVM algorithm in Test 2.2. Using the default settings with a C of 1 and a Gamma of 1/18, the algorithm achieved a 0.79 precision and a 0.77 recall. In Table 7, it can be observed that this time, the peak performance was achieved using a C value of 100 and a Gamma of 1/500.

Table 6: Test 2.1 Random Forests

N Estimators	Max Depth	Max Features	Precision	Recall
10	∞	∞	0.84	0.84
1	∞	∞	0.81	0.81
10	10	∞	0.80	0.80
10	∞	2	0.84	0.84

Table 7: Test 2.2 SVM

C	Gamma	Precision	Recall
1	1/18	0.76	0.76
10	1/18	0.77	0.77
100	1/18	0.78	0.77
100	1/500	0.79	0.77

Table 8 shows the performance of the Random Forests algorithm against the Phishstorm dataset (Marchal et al., 2014) with 96,000 data entries in Test 3.1. Tweaking the algorithm parameters did not yield better results and thus the default configuration of the algorithm was kept.

Table 8: Test 3.1 Random Forests

N Estimators	Max Depth	Max Features	Precision	Recall
10	∞	∞	0.85	0.85
1	∞	∞	0.83	0.83
10	10	∞	0.83	0.83
10	∞	2	0.85	0.85

Once more, the process is repeated using SVM in Test 3.2 (See Table 9). When a C value of 100 was used the algorithm reached its peak precision & recall rates of 0.81 and 0.81. Altering the Gamma value only reduced our rates and thus was kept to its default of 1/18.

Table 9: Test 3.2 SVM

C	Gamma	Precision	Recall
1	1/18	0.79	0.79
100	1/18	0.81	0.81
100	1/100	0.81	0.80
100	1/500	0.80	0.79

4.1.2 Alerting

After completing the experiments, the experimental setup of Test 3.1 was chosen to create a model in Splunk. The model allows for the fitting of the now

trained algorithm into other datasets which are imported in Splunk. The method to fit it is straightforward and is shown in Figure 2 where "randomforestsfull" is the name of the model and "inputtest.csv" is the new input file.

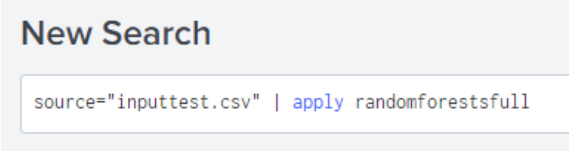

Figure 2: Model fit method in Splunk Search

Figure 3 illustrates the top 3 results from the previous query. All the features from the entries in the input file are now separate fields of events in Splunk. Each entry in the results now has a new field named "predicted(Type)" which is the algorithm's prediction for if it will be malicious or benign.

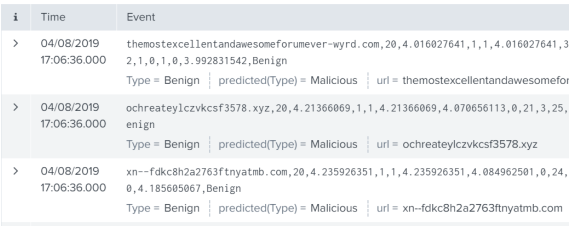

Figure 3: Sample results from Splunk Search model fit

Splunk was configured to continuously monitor the input file to update any further additions or removals. An alert was created as seen in Figure 4, which runs the query against the input file every hour and notifies the user if a new entry is flagged as malicious. The alert is then added to the triggered alerts. With this, the automated detection system is complete. Finally, the model is set to re-train itself using any new data added to the initial dataset.

Figure 4: Splunk Machine Learning alert

4.2 Evaluation

The outcomes of Tests 1.1 & 1.2 established that SVM performed slightly better than Random Forests. With Random Forests, the precision & recall rates did not vary significantly when changing the algorithm's parameters. Decreasing the Gamma value in SVM however proved to have great significance on its results, indicating that even the furthest points from the hyperplane were of great importance. The evaluation of Tests 1.1 & 1.2 should yield better results than the other tests due to the reduced number of entries.

The full dataset containing 70,000 entries was used in Tests 2.1 & 2.2 as the benign entries were already validated (Sahingoz et al., 2019). Test 2.1 hit its peak performance without any tweaking, while Test 2.2 required a less straight hyperplane to do so. The evaluation results of SVM and Random Forests were not ideal for an automated filtration system but are still usable in our experiment. This time Random Forests was the predominant algorithm with a significant difference in precision & recall. Overall both algorithms achieved lower rates, the sudden drop in accuracy of SVM indicates that Random Forests could be more ideal for a large-scale application.

Using the Phishstorm dataset (Marchal et al., 2014) of 90,000 entries in Tests 3.1 & 3.2 achieved very similar but slightly better results than Tests 2.1 & 2.2. This means that the selected feature set is not dataset-biased and is robust when handling new data. Even though the achieved rates are not perfect, the model can still be used in the passive detection of malicious URLs.

While SVM performed better than Random Forests in Test 1, Tests 2 & 3 showed that Random Forests does not perform much differently when using a larger dataset, contrary to SVM's performance drop. Finally, Random Forests was selected for the final model simply because of the stability in the outcome it provides by utilising results from multiple decision trees.

After the model was created, it could be used as a standard search parameter in the original search and reporting app by Splunk and not just in the ML toolkit. This allowed for the easy creation of customised periodic checks for new malicious entries in the original dataset. In combination with the pDNS collector, this finalises the automated phishing URL detector. The feature for scheduled training could also be very easily implemented. However, in this scenario, it was not feasible to schedule the script to extract the features periodically because of the script's long execution time duration on the system used (3 days). In a realistic scenario where the system is more powerful than an old laptop, the system would certainly have worked.

4.2.1 Feature Comparison

After the "publishing" of the model, the summary command in Splunk was used to evaluate the individual importance of the features used using the Random Forests algorithm. The chart in Figure 5 illustrates a comparison of the performance in all three tests, listing them using the IDs defined previously in Table 2.

From the chart, it is clear that F1, F6, F16 and F18 held the most weight for Test 1, all of which indicate that longer URLs (with more unique characters) tend to be malicious. These however may also be the differences between popular-expensive domains and cheaper ones. Test 2's results were much more close, with F10, F17 and F18 being the most important ones. For Test 3, F6, F10 and F13 were the top 3. An interesting observation can be made about the similarity in feature importance in Tests 2 & 3 and their extreme contrast in performance with some features with Test 1 such as F10 or F13. Another interesting observation is that the F13, Character Continuity Rate feature presented by (Lin et al., 2013) had the third least importance of all the other features in Test 1, while in their experiment it ranked first. F9, the count of numbers in the domain suffix was of no importance as there were no numbers in any of our URL suffixes.

5 Discussion

The precision and recall rates achieved here are good enough to indicate that the approach may be useful in real-life scenarios. They may not be ideal for an automated filtration system but can still provide a list of possibly malicious URLs, narrowing the list down and reducing the human input required to spot them. The sole use of descriptive features in a single classification may not be the correct approach if the goal is to achieve the efficiency levels required for an automated detection system to work. If the system had reached higher efficiency levels, then that would mean that the multi-lingual domain classification difficulties mentioned by (Nikiforakis et al., 2014) would be avoided. The purpose of selecting this specific approach was to further the trend of separate classification of features seen in the literature by not considering lexical features. It is an essential step towards understanding which groups of features work best together so that future developed multi-classifier systems are built knowing those relationships.

The detection of phishing URLs is more challenging than the detection of botnet C&C URLs as by their nature, phishing domain names attempt to mirror the appearance of benign domains. In retrospect, this model could have been a better fit for detecting randomly generated C&C domains as their randomisation would lead to higher entropy values, longer URLs and would have used a multitude of unique characters and symbols. This would yield higher precision and recall rates which would, in turn, produce more accurate alerts once the model was "published" in the Splunk search and reporting app. For the detection of phishing domains, more focus should have been given towards features explicitly targeting the detection of squatting domains (Moubayed et al., 2018).

This difference in feature importance between the three tests as well as from other approaches in the literature suggests that the quality of the datasets used can entirely change which features will be more critical for the classification. It is concerning how a feature such as F13, Character Continuity Rate, had so little importance in the first experiment while in Tests 2 & 3 and in the literature (Mamun et al., 2016), it was one of the most important. In the second approach taken by (da Luz, 2014), their Shannon entropy features are of similar importance to F15 and F16 in Test 1: high in 3LD and low in 2LD. However, in their first approach, which used a different dataset, they differed completely, with both features holding no importance. The positive evaluation of the Unique Characters features by (Moubayed et al., 2018), mirrored their performance in the first test. The domains' length and symbol count features proved to be the most important in Tests 2 & 3 but not the count of unique characters; homograph spoofed and typosquatted domains are the likely culprits responsible for this.

As demonstrated, it is possible to use the trained model to make predictions against new data within the Splunk Search and Reporting app. Though with the current model, it would be ill-advised to do so as the surge of false positive and false negative alerts generated would cause more harm than good. It would be best, however, if a classifier with higher accuracy was used.

5.1 Attacks against Machine Learning

The best method of evading detection from a ML algorithm is to just use an expensive domain name, meaning that a wealthy assailant can operate relatively unhindered. For this reason, a lexical analysis system should never be relied upon exclusively, but should be incorporated into a larger system.

The ML system is at its most vulnerable during training (Pitropakis et al., 2019), as that is where human error can thrive. If a malicious entry is included in the benign training dataset then not only will that particular entry not be detected later, but other malicious URLs with similar characteristics may also escape detection. A malicious entry in a whitelist (poisoning) has much more potential to cause damage than a benign entry in a blacklist.

A creative adversarial approach to invalidate a ML system and avoid detection would be to buy a swarm of malicious names with similar features and associate them with malicious activity. In time, after they are included in blacklists because their similarities are known to the adversary, it would be easier to select domain names which would bypass detection as they

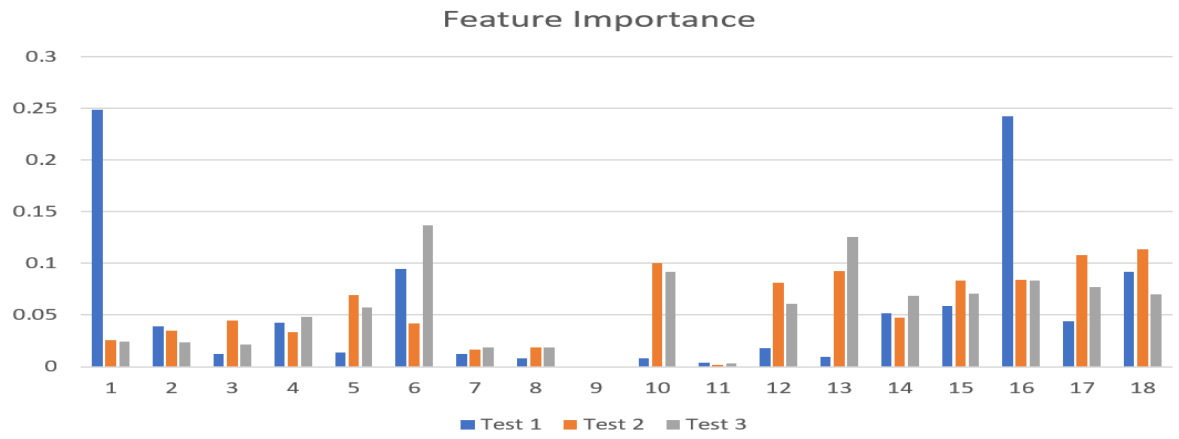

Figure 5: Feature importance comparison graph

would have a hand in the training. Although in a smaller scale, this method would not have much accuracy, it is always a possibility in a grandiose cyberwarfare scenario. However, with so many resources, it would be much easier to simply buy an expensive domain.

6 Conclusions and future work

As adversaries keep inventing different means of abusing the DNS, the only certainty is that Machine Learning will continue to play a vital role in the future of malicious URL filtering. In this work, the descriptive features derived from a benign and malicious domain name datasets were used to make predictions on their nature using the Random Forests and SVM algorithms. The final precision and recall rates produced when only using descriptive features and not considering host-based features were up to 85% and 87% for Random Forests and up to 90% and 88% for SVM respectively. Those results play a vital role in the understanding of the operational relationship between features and thus contribute knowledge into the correct grouping of features and the creation of multi-model classifiers. The features were found to have significantly different impact factors than some other cases in the literature, proving the importance of placing great care in the selection of training datasets. After the model was finalised and fine-tuned, it was "published" in the Splunk Search and Reporting app where it was used against new data to generate alerts. This was the final step toward the automation of the detection process. Scheduled training was also configured using Splunk, furthering the system's autonomy.

There are several research pathways which can be undertaken to improve the performance of this system, some being parallel to and some being stemming from the existing literature. A great addition to our methodology would be the use of a dataset composed of real-world passive DNS data for the training phase that would allow for the generation of more features, thus leading towards the elimination of noise. As we have a passive DNS infrastructure under development, we plan in the near future to make use of a higher volume of real-world data as training datasets, which would lead to the further improvement of our model.

REFERENCES

Alexa (2019). The top 1.000.000 sites on the web.

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

Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N., and Dagon, D. (2011). Detecting malware domains at the upper dns hierarchy. In *USENIX security symposium*, volume 11, pages 1–16.

Bilge, L., Kirda, E., Kruegel, C., and Balduzzi, M. (2011). Exposure: Finding malicious domains using passive dns analysis. In *Ndss*, pages 1–17.

Blum, A., Wardman, B., Solorio, T., and Warner, G. (2010). Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, pages 54–60. ACM.

da Luz, P. M. (2014). Botnet detection using passive dns. *Radboud University: Nijmegen, The Netherlands*.

Darling, M., Heileman, G., Gressel, G., Ashok, A., and Poornachandran, P. (2015). A lexical approach for classifying malicious urls. In *2015 international conference on high performance computing & simulation (HPCS)*, pages 195–202. IEEE.

Feroz, M. N. and Mengel, S. (2015). Phishing url detection using url ranking. In *2015 ieee international congress on big data*, pages 635–638. IEEE.

Fouchereau, R. and Rychkov, K. (2019). Global DNS Threat Report Understanding the Critical Role of DNS in Network Security.

Hirani, M., Jones, S., and Read, B. (2019). Global dns hijacking campaign: Dns record manipulation at scale. *blog, Jan*.

Jain, A. K. and Gupta, B. B. (2017). Phishing detection: analysis of visual similarity based approaches. *Security and Communication Networks*, 2017.

Khalil, I., Yu, T., and Guan, B. (2016). Discovering malicious domains through passive dns data graph analysis. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 663–674. ACM.

Kintis, P., Miramirkhani, N., Lever, C., Chen, Y., Romero-Gómez, R., Pitropakis, N., Nikiforakis, N., and Antonakakis, M. (2017). Hiding in plain sight: A longitudinal study of combosquatting abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 569–586. ACM.

Krebs, B. (2018). The Year Targeted Phishing Went Mainstream.

Kulczyński, S. (1928). *Die pflanzenassoziationen der pieninen*. Imprimerie de l'Université.

Lam, I.-F., Xiao, W.-C., Wang, S.-C., and Chen, K.-T. (2009). Counteracting phishing page polymorphism: An image layout analysis approach. In *International Conference on Information Security and Assurance*, pages 270–279. Springer.

Lin, M.-S., Chiu, C.-Y., Lee, Y.-J., and Pao, H.-K. (2013). Malicious url filteringa big data application. In *2013 IEEE international conference on big data*, pages 589–596. IEEE.

López Sánchez, J. (2019). Métodos y técnicas de detección temprana de casos de phishing.

Mamun, M. S. I., Rathore, M. A., Lashkari, A. H., Stakhanova, N., and Ghorbani, A. A. (2016). Detecting malicious urls using lexical analysis. In *International Conference on Network and System Security*, pages 467–482. Springer.

Marchal, S., François, J., State, R., and Engel, T. (2014). Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471.

Marchal, S., François, J., Wagner, C., State, R., Dulaunoy, A., Engel, T., and Festor, O. (2012). Dnssm: A large scale passive dns security monitoring framework. In *2012 IEEE Network Operations and Management Symposium*, pages 988–993. IEEE.

Moubayed, A., Injadat, M., Shami, A., and Lutfiyya, H. (2018). Dns typo-squatting domain detection: A data analytics & machine learning based approach. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE.

Nikiforakis, N., Balduzzi, M., Desmet, L., Piessens, F., and Joosen, W. (2014). Soundsquatting: Uncovering the use of homophones in domain squatting. In *International Conference on Information Security*, pages 291–308. Springer.

Okayasu, S. and Sasaki, R. (2015). Proposal and evaluation of methods using the quantification theory and machine learning for detecting c&c server used in a botnet. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 3, pages 24–29. IEEE.

Ollmann, G. (2004). The phishing guide–understanding & preventing phishing attacks. *NGS Software Insight Security Research*.

OpenDNS, L. (2016). Phishtank: An anti-phishing site. *Online: https://www. phishtank. com*.

Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E., and Loukas, G. (2019). A taxonomy and survey of attacks against machine learning. *Computer Science Review*.

Proofpoint (2018). THE HUMAN FACTOR: PEOPLE CENTERED THREATS DEFINE THE LANDSCAPE.

Rouse, M., Bedell, C., and Cobb, M. (2019). DEFINITION phishing.

Sahingoz, O. K., Buber, E., Demir, O., and Diri, B. (2019). Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357.

Spring, J. M. and Huth, C. L. (2012). The impact of passive dns collection on end-user privacy. *Securing and Trusting Internet Names*.

Vergelis, M. and Shcherbakova, T. (2019). Spam and phishing in Q1 2019.

Verizon (2018). 2018 Data Breach Investigations Report.

Verizon (2019). 2019 Data Breach Investigations Report.

Wardman, B., Warner, G., McCalley, H., Turner, S., and Skjellum, A. (2010). Reeling in big phish with a deep md5 net. *Journal of Digital Forensics, Security and Law*, 5(3):2.