A Machine Learning Approach for Detecting Fast Flux Phishing Hostnames

Thomas Nagunwa[a*], Paul Kearney[a], Shereen Fouad[b]
[a] *School of Computing and Digital Technology, Birmingham City University, Birmingham, UK*
[b] *School of Informatics and Digital Engineering, Aston University, Birmingham, UK*

Abstract

Attackers are increasingly using Fast Flux Service Networks (FFSNs), networks of compromised machines, to host phishing websites. In FFSNs, the machines rapidly change such that blacklisting them does not entirely stop the networks from operating the websites. This increases the longevity of the websites thus becoming more harmful. Existing solutions for detecting the websites are limited with relatively low or moderate prediction performances, high prediction time and use of less diversified features which increases their susceptibility to detection evasions. This paper proposes a Machine Learning (ML) based approach for detecting phishing websites hosted in FFSNs using a novel set of 56 features. Compared with previous works, the approach achieves high accuracy, a low detection time and uses highly diversified features to enhance resilience to detection evasion. The effectiveness of the features for prediction was evaluated in the context of binary and multi-class classification tasks using multiple traditional and deep learning ML algorithms. The proposed approach achieves an accuracy of 98.42% and 97.81% for binary and multi-class classification tasks respectively. Our results showed that temporal and DNS based features are the strongest predictors while network and host related features are the weakest. Our approach is a significant step towards tracking of core components of FFSNs with an aim of shutting down the entire phishing ecosystem.

Keywords; phishing hostname, fast flux service network, machine learning, deep learning, flat classification, hierarchical classification.

1. Introduction

Traditionally, phishing websites have been hosted in single machines, small networks or static botnets. Cybersecurity experts have become proficient at taking down the websites by tracking and blacklisting their hosts through their consistent IP addresses [1, 2]. In order to evade the blacklisting approach, thus increasing longevity of the websites, attackers have been increasingly using highly dynamic botnets, also known as Fast Flux Service Networks (FFSNs) to host the websites [3, 4]. In FFSNs, hostnames of the hosted websites are dynamically mapped to IP addresses of members of large, evolving pools of compromised machines, also known as flux agents. The flux agents act as proxies relaying communications between users and a small number of actual content hosts (which often are also FFSN controllers) thus hiding visibility of the hosts and controllers from the public. Since the flux agents are the ones which are visible to the public, blacklisting them will not entirely stop the FFSN operations as the motherships tend to recruit new agents to replace the old ones, thus maintaining the running of the operations. Through this way, the blacklist approach always lags the evolving networks. Consequently, the use of FFSNs makes the shutting down of phishing websites difficult, allowing them to stay alive longer and

* Corresponding author.
Email address: thomas.nagunwa@mail.bcu.ac.uk / tom.nag@gmail.com

become more impactful. An effective and efficient approach to detect the websites hosted in these networks is critical in order to effectively address the phishing problem.

Detecting these websites can be useful in complementing approaches such as that proposed by Nagunwa, et al. [5]. It can also be useful in building a blacklist of such websites. The database can be used by security stakeholders such as Internet Services Providers (ISPs) to investigate and monitor flux networks of the websites in order to identify the compromised legitimate networks hosting the flux agents. This can help the informed owners of the compromised networks to clean their machines and take precautions to prevent their machines from being infected again. Also, solutions such as those proposed by Gu, et al. [6] and Khattak, et al. [7], which monitor data traffic between flux agents in the local networks and their external motherships, can be used at the network gateways to track the motherships in order to blacklist them, thus shutting down the entire infrastructures of the phishing attacks.

This work focuses on phishing specific FFSNs because: 1) Phishing attacks are the major source of global cybersecurity attacks, causing up to 91% of global data breaches [8]. Preventing users from visiting phishing websites will significantly reduce the number of data breach incidents across the globe. 2) FFSNs hosting different types of malicious web services such as spam, malware and phishing have different DNS, host and network related characteristics which are often used to derive features for detection [4]. Thus, solutions designed to detect all types of malicious web services hosted in generic FFSNs are likely to be less effective than the ones which detect specific services hosted in their dedicated FFSNs.

The malicious web services hosted in FFSNs are often detected through their hostnames (we refer to them as Fast Flux (FF) hostnames). The most effective FF hostname detection techniques proposed to date have been based on the analysis of DNS related predictive features using a ML approach. Early techniques proposed by Passerini, et al. [9], Perdisci, et al. [10] and others monitored specific FF hostname characteristics over a period of several hours, days or weeks to identify the predictive patterns. However, during this monitoring period, the hostnames continue to operate thus causing more damage. To address the issue, some of the recent works including Hsu, et al. [11] and Jiang and Li [12] proposed faster techniques that take only a few seconds or minutes to detect the hostnames. However, they have the following limitations:

- A number of techniques achieved relatively low or moderate prediction performances. For instance, Kumar and Xu [13] obtained an accuracy of 88.03%, Stevanovic, et al. [14] achieved an F1 score of 0.85 and Almomani [1] attained misclassification rates of up to 16%.
- They rely on a restricted set of predictive features mainly DNS, network or spatial based features. With little effort, attackers can discover the features used and develop simple techniques to evade detection.
- With an exception of Chen, et al. [15], all the works have addressed the problem as a classification task by distinguishing FF hostnames from all legitimate hostnames only, ignoring non-FF malicious hostnames. However, according to our data (described in section 3.2), the majority of malicious websites are still hosted by non-FF malicious hostnames. By not considering such hostnames, their proposed solutions are deemed impractical as they do not reflect the real-world scenario in which non-FF malicious hostnames exist, thus they cannot be deployed directly to protect end users.

- IP geolocation databases do not have records of all public IP addresses. Some of the works including Stalmans, et al. [16] have based their detection features on this data. These techniques fail to detect websites whose hosts' IP geolocation records are unknown.
- The detection performances of the current techniques were not thoroughly validated using a wide range of reliable performance measures including precision, recall and false negative rates, limiting our understanding of all-round effectiveness of the techniques.

To address the above deficiencies, we propose a more robust ML-based approach to detect FF phishing hostnames. Our contributions in this paper are as follows;

1. We design the approach using a novel set of highly diversified features that allows for a reliable detection of FF phishing hostnames. The set consists of 56 features derived from DNS, host and network characteristics of the hosting networks. They are grouped into six different categories, of which 41 are newly-proposed features and the rest are adopted from existing works.
2. The problem is formulated as both binary and multi-class classification tasks in which FF phishing hostnames are distinguished from CDN hostnames, and phishing and legitimate non-flux hostnames. In binary classification, the FF phishing hostnames are distinguished from the others combined as a single hostname class. On the other hand, in the multi-class classification, which deemed as more difficult task from the ML perspective, all four hostnames are identified which allows for detecting the exact type of hostname and hence helps to take more informed decision.
3. Using flat and hierarchical classification techniques, four implementation architectures of our prediction model are proposed based on binary and multi-class classification tasks with an aim of identifying the architecture that provides the best prediction performance.
4. An approach to evaluation of the feature set was devised whereby the performance of the features was measured using a larger number of different ML algorithms than the number used in the related works. Comparing the performance results from such a large set of algorithms allows conclusions to be drawn regarding the general effectiveness of a feature set. A larger number of metrics were also used to measure and report the performances in order to inform us on the all-around effectiveness of the prediction model.

To our knowledge, this is the first work that has addressed the problem as a four-class classification task, has used the hierarchical classification approach, has applied DL algorithms to address fast detection of FF hostnames, and has compared the performance of traditional ML and DL algorithms in this context.

Our model has achieved a high prediction performance but with a prediction time that is currently higher than desirable for real-time applications. We have not yet attempted to optimise the prediction time, but this is on our work plan. Alternatively, the model, could be used to build a blacklist of FF phishing hostnames, which could be used in an effective real-time detection application and as a source of data for research purposes. We are not aware of any existing blacklist that is specific to FFSN hostnames. It is important to mention that we have not yet tried to distinguish between single-flux FFSNs (i.e. those using only hostname IP fluxing) and double-flux networks (combining hostname IP fluxing with name server IP fluxing. This will be part of the future work.

This paper is arranged in six sections. The second section provides a background to non-flux networks, CDNs, FFSNs, and flat and hierarchical classifications. Significant works related to our research are also reviewed. In section 3, we present the design of our prediction model and introduce the prediction features. In section 4 we describe various model architectures, the experiments for developing and evaluating the model and present the results of the experiments with their analysis. Section 5 compares our work with other related works, and discusses applicability and limitations of our solution. Section 6 concludes the paper by re-visiting our results and contributions, and outlines our future work.

2. Background and Related Work

2.1. Non-flux Networks

The majority of phishing and legitimate websites are still hosted in traditional (non-flux) networks. According to the data we collected, more than 71.4% of all the phishing websites are hosted in traditional networks whereas only 25 million websites of the 1.78 billion legitimate websites are hosted in CDNs, the rest being hosted in traditional networks [17, 18]. In non-flux networks, the hostname of the website is resolved to IP address(es) of one or a small number of servers consistently. If multiple servers are used, they are often located in one or a few specific locations. Except for changes due to rare events such as maintenance or upgrading of the services, subsequent queries return the same set of addresses. Since DNS records rarely change, the time to live (TTL), which is the maximum time intermediate name servers can cache the queries, is set relatively high. The default TTL value is normally between one and three days [19, 20]. We refer to such behaviour as non-fluxing.

To avoid being easily tracked, attackers increasingly host their web services in compromised legitimate servers. According to Aaron and Rasmussen [21], up to 51% of all phishing websites run on the compromised hosts. In order to optimize their limited resources, attackers usually host many of their malicious web services in one machine. Similarly, web hosting service providers co-host a large number of legitimate web services of their various customers. In both cases, a large number of different hostnames will be resolved by DNS to the same IP address.

Other attackers use proxies to hide the identities of their hosting machines [22]. Hostnames of their web services, when queried using DNS, resolve to the IP addresses of the proxies. When a proxy receives a request, it contacts the real phishing server to obtain the content and returns this to the user. Though there are legitimate uses of proxies, including protecting hosts from being probed by hackers, caching and traffic filtering [23], proxies are more likely to be deployed by attackers.

2.2. Content Delivery Networks

CDNs are networks of web server farms in dispersed locations. Their purpose is to deliver content efficiently to users scattered over a large geographical area. Copies of the content are cached in multiple farms and a user's request is served from the farm that can provide a copy most efficiently. To decide the IP addresses to return in response to a DNS query, CDNs use sophisticated techniques based on factors such as geographical distance, network topology and link health [24, 25]. Short TTLs are set to ensure users are served with freshly computed IP addresses direct from authoritative name servers.

The owners of CDNs use them to host their own services or lease the infrastructure as a service to others. For example, large organizations with complex web service ecosystems such as Google and Netflix own their specialized CDNs while others including Akamai and Limelight provide unspecialized CDNs as a service to mid-sized content providers [24]. Their server

farms are usually in locations distributed around the globe to serve local users. The number and identities of the servers in a farm changes occasionally due to, for instance, maintenance and scaling of services [26]. Content is accessed directly from the servers which are assigned with static IP addresses. Because they need to deliver large volumes of content to many users with high efficiency, high performance servers and web server software are used.

2.3. Fast Flux Service Networks

FFSNs are essentially networks of compromised machines (flux agents) that are managed by attackers for malicious purposes including hosting phishing websites, distributing spam or malware, and carrying out denial of service attacks [9]. Typically, an FFSN consists of a small number of content servers (in comparison with the number of proxies) and one or more command and control servers (motherships) that may double as content servers[1]. FFSNs recruit new flux agents by infecting vulnerable machines with malware that enables them to be controlled from the motherships [10, 19]. DNS servers map the malicious hostnames dynamically to the IP addresses of sets of flux agents, so that (as for CDNs) consecutive DNS queries return different IP addresses. To ensure updated A records (sets of IP address(es) mapped to hostnames) of active agents are returned for each DNS query, FF hostnames are set with short TTLs. By using flux agents as proxies, the motherships are invisible to users and attackers' footprints are hidden from forensic investigators. Tracking and taking down the flux agents does not destroy the FFSN as motherships can continually recruit and use new flux agents, resulting in the prolonged existence of malicious campaigns [27].

Most of the flux agents are the compromised standard computers and Internet of Things (IoT) devices in homes and small office networks [2, 11, 27], which are often less secured and have many security vulnerabilities [28]. There is an evidence showing that some FFSNs conceal identities of their flux agents by using proxy applications in the agents [29]. FFSN sizes may range up to hundreds of thousands of flux agents and their members vary continuously as new machines are infected and existing agents which are temporarily or permanently inactive are removed [1, 9, 30].

Due to a random process of malware infection, flux agents in an FFSN are typically from different IP networks [19, 31]. As many of the flux agents are owned by individual users and small businesses, their availability is likely to fluctuate as machines are turned off when not in use. Furthermore, agents may be lost from the FFSN when machines are cleansed of malware [10]. FFSNs may host the malicious services of their owners, be offered for hire, or a mixture of the two [32]. Figure 1 provides an overview of FFSN's architecture.

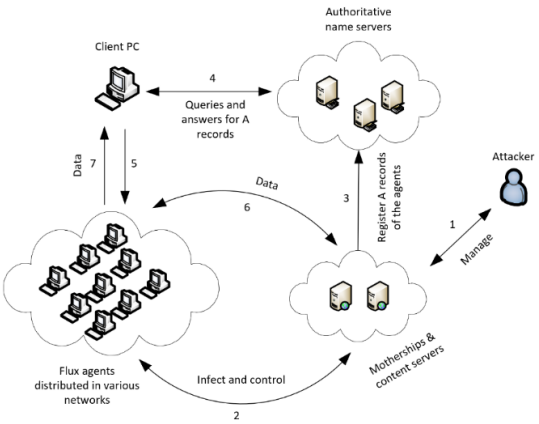

Figure 1. Architecture of FFSNs.

2.4. Related Work

Various studies have proposed techniques for detecting botnets or their members (i.e bots and command and control servers (C&C)) and domains/hostnames. For the former, signature based and anomaly behaviour-based techniques are the most common ones. Those based on signatures, including Intrusion Detection Systems (IDSs) such as Snort and Sagan, and the proposed techniques by Gu, et al. [6],

Kirubavathi and Anitha [33], Xie, et al. [34] and Khattak, et al. [7], use predefined patterns and signatures of network traffic and of malicious codes of previous known botnets to detect new botnets or the members [35, 36]. The main limitation of this approach is that it only detects known botnets, leaving unknown (zero-day) botnets undetected. The anomaly behaviour-based techniques use heuristics or statistical algorithms to analyse host and network behaviours such as load processing overheads, changing system calls, network latency, network traffic volume and ports handling the traffic to detect abnormal activities related to bots [37-41]. These techniques, however, detect only the botnets that are associated with the networks which have implemented the solutions.

The latter aim to detect botnets in the wider internet using various hostname and network characteristics. One approach in this category is to detect malicious domains from their linguistical properties. Yadav, et al. [42], Antonakakis, et al. [43], Kelley and Furey [44], Ravi, et al. [45] and Vinayakumar R. [46], for instance, aimed at detecting malicious domains generated by domain fluxing botnets using this approach. Domain fluxing botnets are the networks which generate large numbers of new domains algorithmically after every specific period or event. Consequently, their strings follow specific patterns set by their algorithms that are different to those typically chosen by people. Fu, et al. [47], argued that the detection solutions trained against specific domain-generation algorithms may not perform well against novel ones.

The other approach includes the techniques for detecting malicious FF hostnames based on DNS related features. These generally fall into two groups: those based on the monitoring of DNS related features over an extended period of time, and those based on detection features extracted from data collected at a point in time. Using the former approach, Kumar and Xu [13] proposed a SVM based classifier using seven DNS related features to detect FF hostnames. The model was trained on passive DNS data to achieve a detection accuracy of 88.03%. Some of the features, however, required long term monitoring of hosts to obtain their appropriate values. For instance, the feature 'MaxCount' counts the total number of visits to the hostname observed in a particular period, typically 24 hours. Using a LSTM deep learning technique, Chen, et al. [15] proposed a classifier to detect FF hostnames using three features queried from active domains at five different times. Though they achieved a good accuracy of 95.4%, the classifier required an input data collected at five separate times, thus increasing a detection time. Other similar works in this category include Passerini, et al. [9] and Perdisci, et al. [10]

The above works require significant time intervals to collect sufficient data to produce good prediction results. To address detection delays, some works proposed classifiers to detect the hostnames in few seconds or minutes. Huang, et al. [48] used six features based on time zones and geographical locations of hosts and name servers to develop a classifier that achieved an accuracy, false positive rate (FPR) and area under ROC curve (AUC) of 98.16%, 0.398% and 0.984 respectively However, not all public IP addresses have recorded in IP geolocation databases such as "MaxMind". Therefore, the classifier will fail to make predictions for hostnames whose IP addresses have not been recorded. Also, attackers' DNS servers may select flux agents from time zones and locations similar to the users, emulating the behaviour of CDNs, so as to increase false alarms and evade the detection. Classifiers proposed by Stalmans, et al. [16] and Wang, et al. [49], which are also based on geolocation features, face the same problem.

Hsu, et al. [11] developed an FF hostname detection classifier based on response time difference (RTD) between hosts of the same hostname. The classifier achieved FPR and false negative rate (FNR) of 0.3% and 2%

respectively. However, the feature can be neutralised by configuring the FFSN's name server to return A records of flux agents that yield similar response times. Other significant studies that proposed classifiers with short detection times are Hsu, et al. [28], Lin, et al. [25], Stevanovic, et al. [14], Jiang and Li [12] and Almomani [1].

3. Detection of Fast Flux Phishing Hostnames

3.1. Design Overview

Our proposed approach uses supervised ML techniques to train and develop a classifier that can distinguish FF phishing hostnames from CDN hostnames, and phishing and legitimate non-flux hostnames using features extracted at one point in time (instantaneous features). In order to assign class labels to the set of hostnames used to train the classifier, we monitored their fluxing behaviour over an extended period of time.

The approach has the following main steps:

1. Monitoring of the A records returned by DNS for sets of known phishing and legitimate websites in order to label their hostname classes according to their fluxing behaviour (see Figure 2). Section 3.2 provides further details.
2. Extraction of instantaneous feature data. For each URL, a range of services (shown in Figure 3) are queried and the features are extracted from the returned information to generate the training dataset. See section 3.3 for further details.
3. Training a suitable ML algorithm on the training dataset to develop a classifier.
4. The classifier accepts instantaneous features of given unknown website and predicts its hostname class.
5. Periodic incremental updating of the training dataset and re-training of the classifier to improve and update its performance. This takes into account an assessment of the prediction accuracy of the classifier over the preceding period.

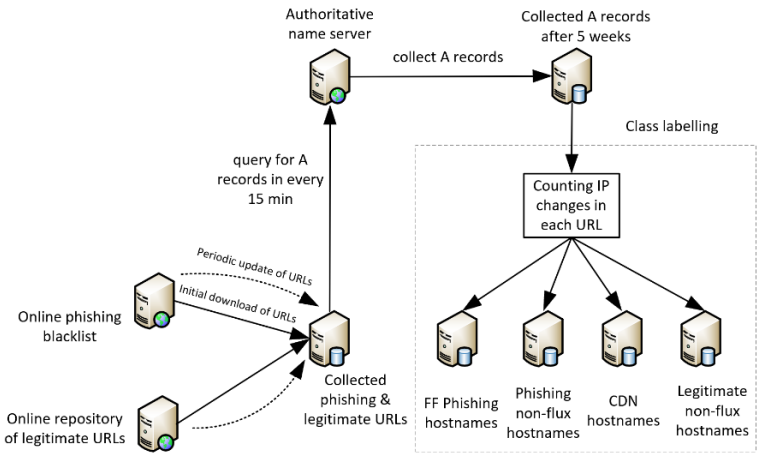

Figure 2. Monitoring of A records for 5 weeks for labelling classes of the hostnames.

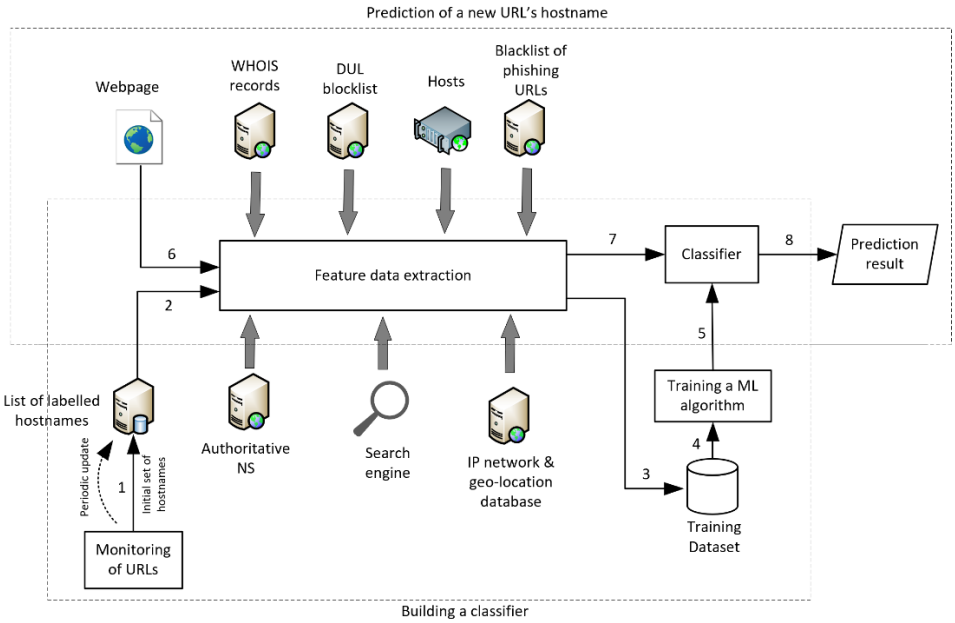

Figure 3. Components 2 - 4 of our proposed approach; building of a classifier and prediction of a new hostname.

3.2. URL Monitoring for A Records and Class Labelling

In order to be able to label a training set of websites according to their fluxing behaviour, we collected sets of known 4,271 phishing and 7,530 legitimate websites and monitored their IP addresses returned by DNS queries over an extended period. We obtained the legitimate URLs from a list of 1 million most visited website domains from Tranco[†] and the phishing URLs from two major reputable online repositories; PhishTank[‡] and OpenPhish[§]. Each URL was queried repeatedly for A records after every 15 minutes for up to 5 weeks from June 16 to July 21, 2019. The IP addresses returned in consecutive queries regarding the same hostname were compared, and the number of times a change was observed throughout the period was recorded. In a similar fashion to the 'fluxiness' approach used by Holz, et al. [31] to distinguish between FF and CDN domains, we labelled any phishing or legitimate website observed to have at least one IP change as a phishing FF or CDN hostname respectively.

Figure 4 shows the distribution of the number of IP changes observed for each monitored website in the five-week period. Only 29% of the collected phishing URLs (also indicated in Table 4) were observed to undergo at least one IP change while 17% underwent less than 10 changes. There is significant percentage of URLs with number of changes at ranges including 21 – 30, 11 – 20, 61 – 70 and 401 – 500. This suggests that most of the phishing websites are still hosted in non-flux networks. 48% of the legitimate URLs experienced at least one change (also indicated in Table 4), with 25% having less than 10 changes. Other ranges of number of IP changes with large percentages of NSs are 61 – 70, 71 – 80, 501 – 600 and 11 – 20. Generally, legitimate URLs were observed to have larger numbers of IP changes across most of the ranges compared to phishing ones. This is likely because the legitimate URLs were obtained from a list of 1000 most visited websites, which are more likely to be hosted in CDNs than lowly ranked websites [17]. IP changes in websites with a low flux rate, for instance less than 5 changes in the monitoring period, may be due to non-

[†] https://tranco-list.eu

[‡] https://www.phishtank.com

[§] https://openphish.com

fluxing behaviours such as routine maintenance of hosts and upgrading of the networks.

In order to investigate the structure of FFSNs, the cumulative records of the IP addresses that the phishing hostnames resolve to during monitoring were imported into Gephi[**], a graph visualisation software. Figure 5 shows what we suspect to be a large FFSN. The brown and green discs represent hostnames and IP addresses respectively, and an arc indicates that the relevant IP address appears in the hostname's A records at least once in the monitoring period. The size of the disc indicates how many connections it has. It is apparent that IP addresses are often shared by different hostnames. This could simply be that different FFSNs have infected the same vulnerable hosts, but given the density of interconnections, it seems more likely that this cluster is a single network. Very small networks consisting of one or two hostnames and a few IP addresses were also observed.

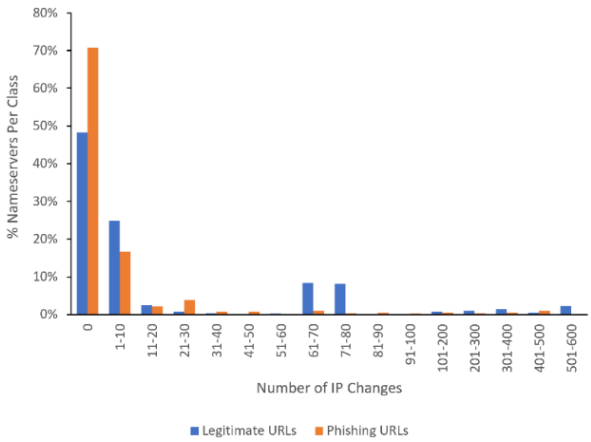

Figure 4. Distribution of number of IP changes of website hosts observed per URL type.

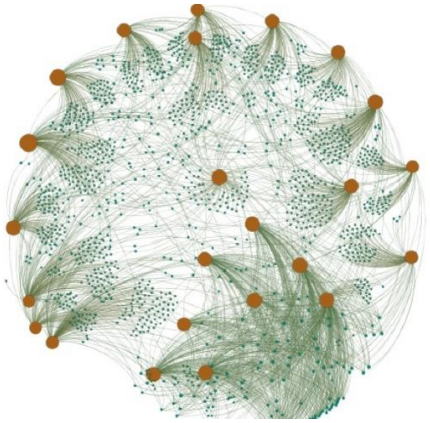

Fig 5. Some of the phishing FF hostnames and their FFSNs observed in our dataset.

3.3. Distinctive Features for Detecting Fast Flux Phishing Hostnames

We propose 83 instantaneous features grouped into 6 categories that, based on the analysis summarised in section 2, are likely to be useful in distinguishing FF phishing hostnames from CDN hostnames, and legitimate and phishing non-flux hostnames. 62 of the features are newly-introduced by this study and the remaining 21 features are adopted from the existing works. Here we describe few important features, a complete list is given in Table 1. Table 2 provides a list of sources of information we used to generate the features.

Many of the features measure the distribution of various attributes across the set of IP addresses identifying the hosts associated with a given hostname. Consequently, the first step in extracting the features is to perform a DNS query to obtain this set. The feature value is then obtained for each host and statistical measures such as average, standard deviation, entropy, minimum and maximum are calculated. Entropy features are approximated as $\Sigma_i p_i ln(p_i)$ where i runs over the set of unique values of the feature within the sample. p_i is then the number of occurrences of i within the sample divided by the number of hosts. Entropy

[**] https://gephi.org

is greatest when all the feature values are different, and least when they are all the same.

3.3.1. Temporal Features

Round Trip Time (RTT). Using the traceroute command, we measure the time interval between sending a data packet and receiving an acknowledgement from each host corresponding to the hostname in question. Five metrics are computed from these RTT values as feature 1 to 5 (see Table 1).

DNS Response Time. This is the time taken to receive an answer from the DNS server to a query for A records.

Authoritative TTL for A records. This is a maximum time set in the authoritative name server for caching A records of each hostname. It is obtained by querying A records of the hostname from its authoritative nameserver.

Uptime of Hosts. This is an estimated time, in hours, the machine has been up and running as reported by a host scanning tool (Nmap). We extract features 8–10 from the uptime recorded for each IP address associated with the hostname.

Domain Age and Domain Validity. We query a WHOIS database to extract a date of first registration and an expiry date of the hostname's domain. The former is used to compute its age with respect to the current date while the latter is used to compare with the current date to determine whether the domain has expired or is still valid.

3.3.2. Spatial Features

Geographical and Network Distances. We use the traceroute command to obtain the IP addresses of intermediate hops on the route to each host of the given hostname. The hop IP addresses are then used to generate features 16-21 and 25-28. For instance, we identify the country location of each hop and count the number of unique countries on the route to a host. By combining the numbers of all the hosts per each hostname, we derive features 19–21. Using geographical coordinates of IP addresses of a user and the hosts, various geographical distances are computed to obtain features 29–35. We obtain the coordinates from the Maxmind's Geolite2 database[††].

3.3.3. DNS Features

Characteristics of Hosts' PTR Records. PTR records hold information allowing DNS to perform an inverse look-up of a hostname given an IP address. A DNS PTR query is performed for each host of the same hostname to obtain its PTR records. Features 41– 49 are then extracted from the records.

Characteristics of Co-hosted Websites. We search for websites that are co-hosted on a machine identified by an IP address of each host of the hostname using Bing search engine with a search command 'ip:W.X.Y.Z.'. From the search results, we count the number of co-hosted websites and extract their URLs from which we generate features 50-57.

Similarity of Hostnames. Similar to the approaches used by [42] and [47], we measure similarity of all types of hostnames against the FF phishing hostnames. Three similarity measures are used namely Kullback-Leibler distance (KL) [50], Jaccard Index (JI) [51] and Edit Distance (ED) [52]. For KL and JI, the similarity is measured for unigram and bigram characters of the hostnames. The symmetric values of KL are computed instead of asymmetric ones. The metrics are computed as follows; for example, JI of unigram characters between hostnames h_1 and h_2 is defined as

$JI(h_1, h_2) = X \cap Y \,/\, X \cup Y$

where X and Y are the sets of unigram characters of h_1 and h_2. For classifier B (described in section 4) as an example, we calculate

$$\Delta JI = \frac{1}{|O|}\sum_{o \epsilon O} JI(h, o) - \frac{1}{|P|}\sum_{p \epsilon P} JI(h, p)$$

the difference between an average of JI scores of each tested hostname against all other three

[††] https://dev.maxmind.com/geoip/geoip2/geolite2/

combined hostnames (*O*) and against all FF phishing hostnames (*P*) as a feature. The small difference means *JI(h,p)* is large and therefore the hostname is closely similar to the FF phishing hostnames. Similar approach is used to compute the differences for KL and ED to obtain features 57-61.

3.3.4.Network Features

Network Characteristics. For each IP address associated with the hostname, we extract its network identity information including subnet, network and Autonomous System Number (ASN) from an IP geolocation database (we use IP2location[‡‡]), to compute features 62 to 69. For instance, for feature 62-63, subnet of each host is identified and then we count the number of unique subnets and also compute entropy of all subnets per hostname.

3.3.5.Host Features

Up State of Hosts. Using a host scanning tool (Nmap), we scan each machine hosting the given hostname to determine its availability state. We then compute a ratio of hosts in the 'up' state as feature 71.

Host's Operating System. Using Nmap, we scan each host of the given hostname and identify its operating system (OS). We count the number of unique OSs and identify the most common OS per each hostname as features 72 and 73 respectively.

Host's Webserver Software. We extract the name of the webserver software installed on each host of the given website from the response to an HTTP header request. From this we generate features 74 and 75.

Hosts with Proxy IP Addresses. We compute the proportion of the IP addresses of each host of the given website that are found in a database of known public proxy IP addresses (we use IP2Proxy[§§] database) as feature 76.

3.3.6.Reputation Features

IP Addresses shared with Other Malicious Hostnames. We identify the IP addresses associated with the given hostname that appear on a blacklist of phishing URLs collected in the past three months to generate features 77-82. For instance, in feature 77, we count the total number of times the IP addresses of all hosts for each hostname have matched in the database. Similarly, from the same blacklist, we query for NS records of each hostname to generate a list of IP addresses of name servers of the phishing websites. IP address of each host of the given hostname is also compared against the list to generate features 80–82.

Domain Registrar. We identify the registrar of the website's domain by querying against a WHOIS database to obtain feature 83.

Feature Category	Feature #	Features	Proposed or Existing Features	
Temporal	1 - 5	Round trip time: average, standard deviation, entropy, minimum and maximum	1 - 2	Existing
			3 - 5	Proposed
	6	DNS response time	6	Proposed
	7	TTL for A records	7	Existing
	8 - 10	Uptimes of hosts: average, standard deviation and entropy	8	Existing
			9 - 10	Proposed
	11 - 12	Domain age, domain validity	11	Existing
			12	Proposed
	13 - 15	Domain ages of co-hosted websites: average, standard deviation and entropy	13 - 15	Proposed
Spatial	16 - 18	# hops on route to host: average, standard deviation and entropy	16 - 18	Proposed

[‡‡] https://lite.ip2location.com

[§§] https://lite.ip2location.com/ip2proxy-lite

	19 - 21	# unique hop countries on route: average, standard deviation and entropy	19 - 21	Proposed
	22	Ratio of hosts in the same country with a user	22	Proposed
	23 - 24	# unique hosts' countries, # unique of hosts' continents	23	Existing
			24	Proposed
	25 - 27	# unique hops' continents: average, standard deviation and entropy	25 - 27	Proposed
	28 - 30	Geo-distances between the user and hosts: average, standard deviation and entropy	28 - 30	Proposed
	31 - 34	Geo-distance between the hosts: sum, average, standard deviation and entropy	31 - 34	Proposed
	35 - 38	IP range between hosts: minimum, maximum, average, standard deviation	35 - 38	Proposed
DNS	39	# unique A records	39	Proposed
	40 - 43	Ratio of hosts: with PTR records, with their PTRs containing IP addresses, with PTR's hostnames matching with the URL's hostname, with PTR's hostnames identity matching with the URL's hostname identity	40 - 43	Proposed
	44 - 47	Average: length of hosts' PTRs, # of digits in hosts' PTRs, # of hyphens in hosts' PTRs, # of dots in hosts' PTRs	44 - 47	Proposed
	48	# unique TLDs of hosts' PTRs	48	Proposed
	49 - 52	Ratio of hosts: with co-hosted websites, with co-hosted websites' hostnames matching with the URL's hostname, using private IP addresses, with dynamic IP addresses	49 - 51	Proposed
			52	Existing
	53 - 54	# co-hosted websites: average, standard deviation	53 - 54	Proposed
	55	# unique hostnames of co-hosted websites in the hosts	55	Proposed
	56	# unique TLDs of hostnames of co-hosted websites in the hosts	56	Proposed
	57 - 58	Difference of average KL divergence between hostnames of non-FF phishing and FF phishing hostnames: KL of unigram characters, KL of bigram characters	57 - 58	Existing
	59 - 60	Difference of average Jaccard Index between hostnames of non-FF phishing and FF phishing hostnames: JI of unigram characters, JI of bigram characters	59 - 60	Existing
	61	Difference of average Edit distance (ED) between hostnames of non-FF phishing and FF phishing hostnames	61	Existing
Network	62 - 63	Subnets of hosts: unique #, entropy of # of subnets	62 - 63	Existing
	64 - 65	Networks of hosts: unique #, entropy of # of networks	64 - 65	Existing
	66 - 67	ASNs of hosts: unique #, entropy of # of ASNs	66	Existing
			67	Proposed
	68 - 69	Organizations managing hosts' ASs: unique #, entropy of # of organizations	68 - 69	Existing
	70	Ratio of hosts' networks with generic gateways	70	Proposed
Host	71	Ratio of available (up) hosts	71	Proposed
	72 - 73	OS of hosts: unique #, common OS	72 - 73	Proposed
	74 -75	Webserver software of hosts: unique #, common webserver software	74 -75	Proposed
	76	Ratio of hosts with known proxy IP addresses	76	Proposed
Reputation	77 - 79	Hosts' IP addresses in a blacklist of phishing IP addresses: total # of occurrences, average #, ratio of hosts with their IP addresses matched	77 - 79	Proposed

	80 - 82	Hosts' IP addresses in a blacklist of IP addresses of name servers of phishing websites: total # of occurrences, average #, ratio of hosts with their IP addresses matched	80 - 82	Proposed
	83	Domain registrar	83	Existing

Table 1. A list of the proposed features for predicting phishing FF hostnames.

Feature #	Source/Tool
1 – 5, 8 – 10, 16 – 18, 70 - 73	Network queries with traceroute and Nmap commands
6 – 7, 36 – 39, 40 – 49	Authoritative name server
11 – 15, 83	WHOIS database
13 – 15, 50 – 51, 54 - 56	Bing Search engine
19 – 35, 62 – 69, 76	IP geolocation database
51	A list of private IP addresses provided by Internet Assigned Numbers Authority (IANA)
52	Dynamic User List (DUL) block list
57 - 61	Phishing blacklist (PhishTank, OpenPhish) and Tranco's list of top ranked websites
74 - 75	HTTP response header
77 - 82	Phishing blacklist (PhishTank)

Table 2. Sources of data for each feature.

4. Experiments and Results

Our experiments aim at evaluating the effectiveness of the proposed features (described in section 3.3) in predicting FF phishing hostnames using supervised ML techniques. In particular, we propose multiple architectures designed to solve this problem in the context of binary and multi-class classification tasks using flat and hierarchical classification techniques. We designed two sets of experiments, the first one evaluates the performance of the features using traditional ML algorithms and the second one uses DL algorithms. All experiments were run using Python and Jupyter hosted on Google's Colab platform.

4.1. Flat and Hierarchical Techniques for Binary and Multi Classification Tasks

Some of the four hostname classes share common characteristics and therefore have parent-child relationships. In this case, both flat and hierarchical classification techniques can be applied. In the former, in which relationships between classes are ignored, one binary or multi-class classifier assigns instances to their respective classes in a single step (see Figure 6a). In the latter, a hierarchical class structure based on the parent-child relationships is used to break down the overall multi-class classification task into layers of simpler binary or multi-class classification tasks [53]. Local Classifier per Parent Node (LCPN), illustrated in Figures 6b, in which for each parent node, a classifier is trained to classify its child nodes, is the most preferred technique to implement hierarchical classification [54, 55].

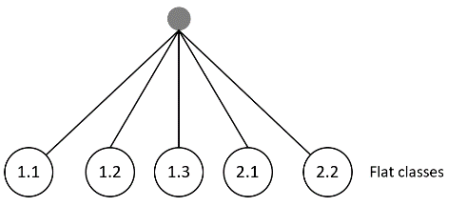

Figure 6a. No parent-child relationships in flat classification.

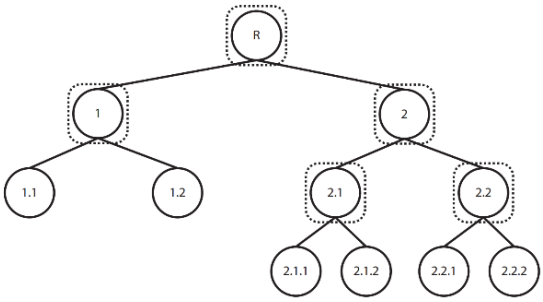

Figure 6b. LCPN technique for hierarchical classification (dashed squares represent binary or multi-class classifiers) [53].

Which of the two classification techniques achieves better prediction results has been shown to depend on the specific problem [53]. Based on the two classification techniques, we therefore designed four model architectures (see Figures 7), two architectures for each technique, to evaluate and compare their performances. Architectures A and B take the flat classification techniques (Figures 7a and 7b respectively). In the multi-class classification-based Architecture A, the FF phishing hostnames are classified against the three other classes, whereas in the binary classification-based Architecture B, the FF phishing hostnames are classified against the other three classes combined.

Architectures C and D apply the hierarchical classification techniques (Figures 7c and 7d, respectively). The multi-class classification is performed through layers of binary classification tasks. In each architecture, we applied the LCPN technique to build the classifiers. We use selective classifier approach, proposed by Silla and Freitas [56], to evaluate various ML algorithms to identify the best performing classifier to use at each node. A feature selection method (for traditional ML algorithms) is also performed at each node to determine the most relevant features for each classifier. The classifiers used within the architectures employ different prediction classes and dataset sizes (see Tables 3 and 4), and therefore are expected to produce different prediction performances.

We build each classifier using a different set of the best features selected from the same original set of features proposed in section 3.3. Each classifier is evaluated using the ML algorithms (named in section 4.3) to identify the best performing algorithm for the classifier. For each hierarchical architecture, we combine performances of the classifiers, from the parent node of the hierarchy to the child node in which FF phishing hostnames belong, to obtain the overall performance. We then compare performances of all architectures to determine the best performing architecture as a recommendation for the implementation of the model.

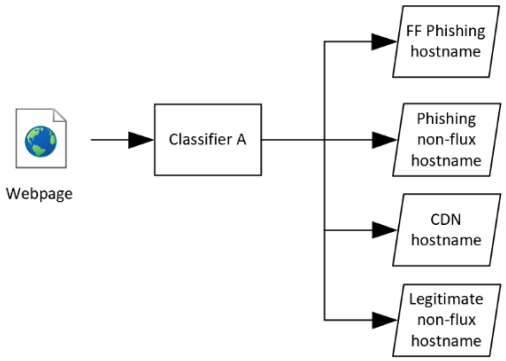

Figure 7a. Architecture A - flat classification-based architecture

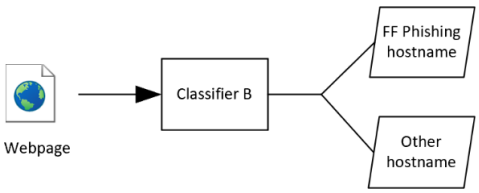

Figure 7b. Architecture B - flat classification-based architecture.

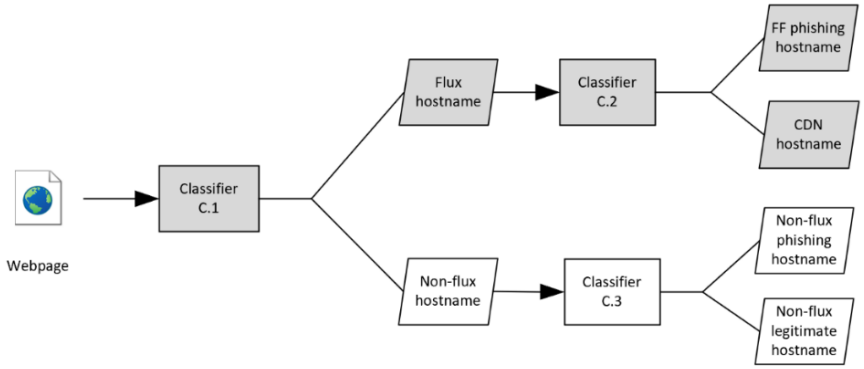

Figure 7c. Architecture C – hierarchical classification-based architecture.

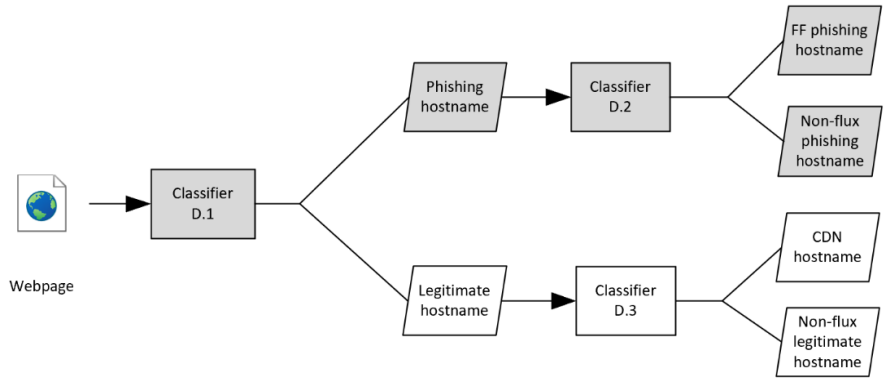

Figure 7d. Architecture D - hierarchical classification-based architecture.

Classifier	Classifier Description	Architecture	Classification Type
Classifier A	Classifies a webpage into four classes of hostnames; FF phishing hostname, phishing non-flux hostname, CDN hostname and legitimate non-flux hostname	A	Multi-class classification
Classifier B	Classifies a webpage as FF phishing hostname or other hostnames (phishing non-flux hostname, CDN hostname and legitimate non-flux hostname combined)	B	Binary classification
Classifier C.1	Classifies a webpage as flux or non-flux hostname	C	Multi-class classification
Classifier C.2	Classifies a webpage as FF phishing or CDN hostname		
Classifier D.1	Classifies a webpage as phishing or legitimate webpage	D	Multi-class classification
Classifier D.2	Classifies a webpage as FF phishing or phishing non-flux hostname		

Table 3. FF phishing hostname detection classifiers forming various architectures of the model.

4.2. Training Datasets

Features described in section 3.3 were extracted from all the monitored and labelled URLs (described in section 3.2) to create a training dataset from which specific training datasets for

each classifier were derived. As indicated in Table 4, the training datasets of classifiers A, B, C.1 and D.1 were formed by labelling the entire dataset with their respective classes whereas in classifier C.2, all URLs with no IP changes were removed from the dataset. In classifier D.2, we removed all legitimate URLs from the dataset.

Classifier	Class Labels	Class Size	Dataset Size
Classifier A	FF phishing hostname	1257	11801
	Phishing non-flux hostname	3014	
	CDN hostname	3867	
	Legitimate non-flux hostname	3663	
Classifier B	FF phishing hostname	1257	11801
	Other hostnames	10544	
Classifier C.1	Flux hostname	5124	11801
	Non-flux hostname	6677	
Classifier C.2	FF phishing hostname	1257	5124
	CDN hostname	3867	
Classifier D.1	Phishing webpage	4271	11801
	Legitimate webpage	7530	
Classifier D.2	FF phishing hostname	1257	4271
	Phishing non-flux hostname	3014	

Table 4. Classes and dataset sizes of training datasets used for each classifier.

4.3. Performance Results

We used various evaluation measures to report the predictive performance of the proposed features. Individual classifier performances were assessed in terms of accuracy, FPR, FNR, precision, recall, F1-score, ROC curve and AUC metrics [57-59] , and they are defined as follows:

- $Accuracy = TP + TN \,/\, TP + TN + FP + FN$
- $FPR = FP \,/\, FP + TN$
- $FNR = FN \,/\, FN + TP$
- $Precision = TP \,/\, TP + FP$
- $Recall = TP \,/\, TP + FN$
- $F1 - score = 2 * Precision * Recall \,/\, Precision + Recall$
- *ROC curve - A graph of Recall against FPR values for thresholds ranging from 0 to 1.*
- *AUC - Area under the ROC curve.*

The above performance measures are derived from the counts of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). Note that, if an instance is positive and it is classified as positive, it is defined as TP. If the instance is negative and it is classified as positive, it is FP. While, a negative instance classified as negative is TN and if it is classified as positive, it is called FN. A positive instance in this problem is the FF phishing hostname. We present and compare results of individual classifiers for both ML and DL experiments in the following subsections.

4.3.1. Results of Individual Classifiers Using Machine Learning Algorithms

The predictive ability of the proposed features was evaluated using eight traditional ML algorithms namely Logistic Regression (LR), k-Nearest Neighbour (k-NN), Decision Tree (DT), Naive Bayes (NB), Support Vector Machine (SVM), Artificial Neural Network (ANN), Random Forest (RF) and Gradient Boosting (GB) [38-41]. These algorithms have been successfully used by other researchers to address various classification problems in cybersecurity [12, 60-63]. Firstly, we performed feature selection using wrapper subset feature evaluation method with RF algorithm [64] to find the best subset of features for each classifier. For instance, for classifier B, we ranked the results (features) of the selection method according to their importance weight. Starting with the full feature set, we evaluated various subsets of the ranked list of features using RF by eliminating the least important feature in each evaluation round. Optimal accuracy was obtained with a subset of 56 features and therefore it was selected as the best feature set for the classifier.

In each classifier, we applied ML algorithms on the best features subset to determine the best performing algorithm and its best performance results. The stratified cross validation technique (k-fold where k is 10) [65] was applied on the algorithms to obtain average scores. We then tuned the best classifier using a random search method [66] to obtain its optimal performance. Figures 8 show the performances of ML algorithms in each classifier across all threshold values in the ROC curve.

Tables 5a-b summarize the results of the four best performing algorithms for each of the individual classifiers (indicated in Table 3). The results indicate that RF yields the best performance across most metrics in each classifier. Table 5c indicates the tuned hyperparameters of RF for classifier B (the best overall classifier) and their values which yielded optimal performance.

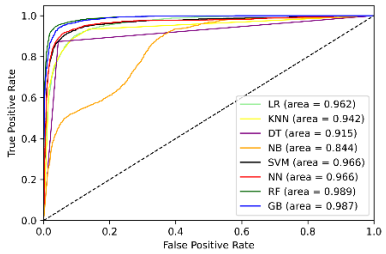

Figure 8a. ROC curves of the traditional ML algorithms for classifier A.

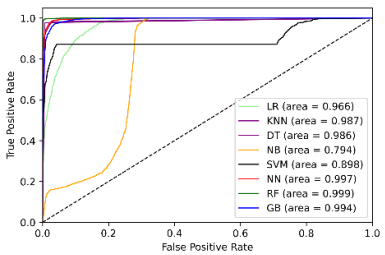

Figure 8b. ROC curves of the traditional ML algorithms for classifier B.

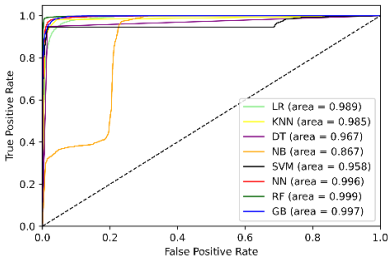

Figure 8c. ROC curves of the traditional ML algorithms for classifier C.1.

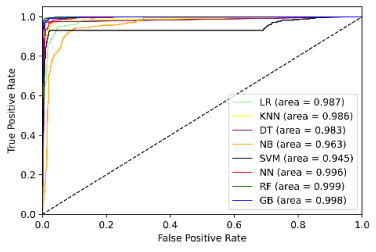

Figure 8d. ROC curves of the traditional ML algorithms for classifier C.2.

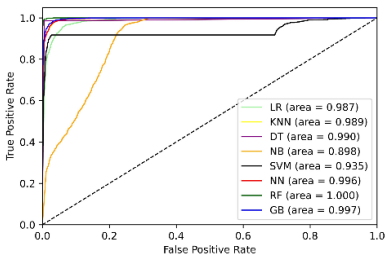

Figure 8e. ROC curves of the traditional ML algorithms for classifier D.1.

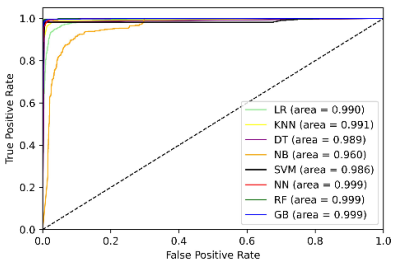

Figure 8f. ROC curves of the traditional ML algorithms for classifier D.2.

Algorithm	Classifier A	Classifier B	Classifier C.1

	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
DT	88.54	5.20	12.27	0.89	97.23	1.37	13.88	0.98	95.44	4.62	6.10	0.95
ANN	89.13	4.23	11.15	0.89	98.07	1.25	14.02	0.98	97.60	1.67	4.97	0.97
RF	**93.58**	**1.82**	**8.62**	**0.94**	**98.42**	**0.57**	**5.88**	**0.99**	**98.36**	**0.69**	**5.05**	**0.99**
GB	91.94	3.73	7.64	0.92	95.05	4.26	7.79	0.95	96.57	2.04	10.33	0.96

Table 5a. Performance results of top four best performing ML algorithms for classifiers A, B and C.1.

Algorithm	Classifier C.2				Classifier D.1				Classifier D.2			
	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score	Acc. (%)	FPR (%)	FNR (%)	F1 Score
DT	97.14	1.65	4.68	0.98	97.92	1.53	3.08	0.98	98.14	1.63	9.83	0.98
ANN	97.44	1.64	5.72	0.98	97.19	2.48	3.35	0.97	98.70	0.75	8.36	0.99
RF	**98.49**	**0.56**	**3.82**	**0.99**	**98.64**	**0.62**	**2.15**	**0.99**	**99.16**	**0.19**	**7.07**	**0.99**
GB	97.63	1.28	5.77	0.98	95.67	3.71	4.88	0.96	98.48	0.91	9.62	0.98

Table 5b. Performance results of top four best performing ML algorithms for classifiers C.2, D.1 and D.2.

Hyperparameter	Description	Value
n_estimators	Number of trees	1000
max_features	Max number of features considered for splitting a node	log2
max_depth	Max number of levels in each decision tree	38
min_samples_split	Min number of data points placed in a node before the node is split	2
min_samples_leaf	Min number of data points allowed in a leaf node	3
bootstrap	Method for sampling data points	false

Table 5c. The optimal values of the tuned RF hyperparameters for classifier B.

4.3.2. Results of Individual Classifiers Using Deep Learning Algorithms

Here, we assess the performance of the introduced features using three DL algorithms which are Fully Connected feedforward Deep Neural Networks (FC-DNN), Long Short-Term Memory (LSTM) and one-dimension Convolutional Neural Network (1D CNN) [41-43]. The DL algorithms were tuned by a random search method. We first identified key hyperparameters for tuning and their considerable range of values for evaluation. In each hyperparameter, we first identified a set of considerable values for performance tuning (indicated in Table 6). We also attempted to tune with multiple hidden layers. We found that only one hidden layer was sufficient to produce optimal performance in each algorithm. Additional layers did not improve the performances. The identified optimal values of all the hyperparameters were then used to build the classifiers. The final result of each classifier was obtained by taking an average of the performances of five runs of the tuned classifier. Figures 9 show the three network architectures of the tuned classifier B (the best classifier), as an example, along with the tuned hyperparameters and their optimal values. As the final results in Tables ~~6~~ 7 indicate, FC-DNN has performed best across most of the

measures in most of the classifiers (A, C.2 and D.1), followed by 1D CNN (in C.1 and D.2). LSTM has outperformed others in classifier B only.

Hyperparameter	Range of Evaluated Values
Number of neurons in dense layers / memory units in a hidden layer of LSTM / filters in a convolution layer of CNN)	10, 30, 50, 80, 100, 150, 200, 300, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2800, 3000
Activation functions	Relu, tanh, sigmoid, hard_sigmoid, linear, softmax, softplus, softsign
Optimization algorithms	SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
Learning rates	0.001, 0.01, 0.1, 0.2, 0.3
Kernel initializers	Uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform
Dropout rates	0.1, 0.2, 0.3, 0.4, 0.5
Batches	15, 30, 50, 70, 90, 110, 130, 150
Epochs	10, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300

Table 6. Hyperparameters and their ranges of values evaluated for tuning the three DL algorithms.

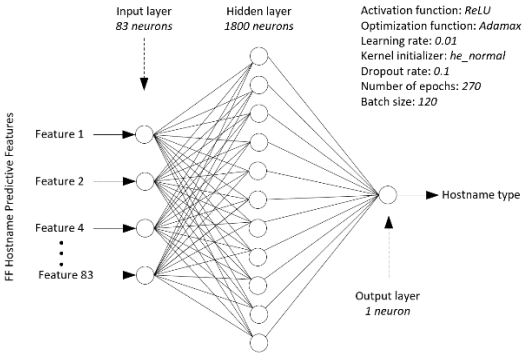

Figure 9a. The tuned FC-DNN architecture of classifier B.

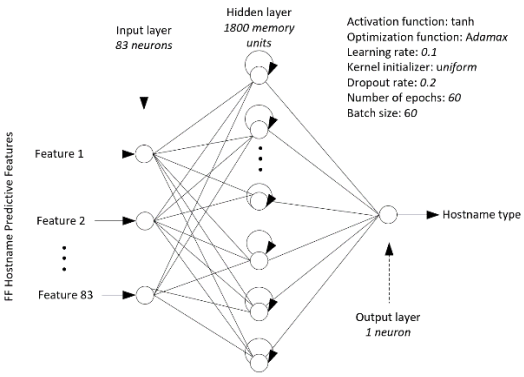

Figure 9b. The tuned LSTM architecture of classifier B.

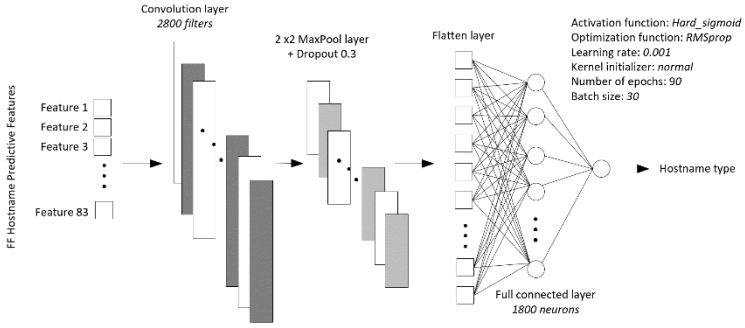

Figure. 9c. The tuned 1D CNN architecture of classifier B.

Algorithm	Classifier A				Classifier B				Classifier C.1			
	Acc. (%)	**FPR (%)**	**FNR (%)**	**F1 Score**	**Acc. (%)**	**FPR (%)**	**FNR (%)**	**F1 Score**	**Acc. (%)**	**FPR (%)**	**FNR (%)**	**F1 Score**
FC-DNN	**83.56**	**6.96**	**20.12**	**0.72**	94.86	0.79	36.52	0.94	92.60	4.68	19.24	0.92
LSTM	82.60	7.91	28.23	0.73	**96.29**	**0.63**	**35.07**	**0.95**	93.39	1.28	24.44	0.93
CNN	81.89	7.45	31.05	0.71	94.50	1.18	32.46	0.93	**93.78**	**1.10**	**23.22**	**0.94**

Table 7a. Performance results of the evaluated DL algorithms for classifiers A, B and C.1.

Algorithm	Classifier C.2				Classifier D.1				Classifier D.2			
	Acc. (%)	**FPR (%)**	**FNR (%)**	**F1 Score**	**Acc. (%)**	**FPR (%)**	**FNR (%)**	**F1 Score**	**Acc. (%)**	**FPR (%)**	**FNR (%)**	**F1 Score**
FC-DNN	**93.18**	**3.66**	**16.63**	**0.93**	**94.55**	**6.60**	**3.13**	**0.95**	93.13	3.15	34.27	0.94
LSTM	91.54	5.44	21.35	0.92	90.24	10.14	10.26	0.90	94.69	1.47	41.03	0.94
CNN	90.58	2.50	37.67	0.90	91.89	6.37	12.04	0.90	**94.62**	**1.35**	**42.68**	**0.94**

Table 7b. Performance results of the evaluated DL algorithms for classifiers C.2, D.1 and D.2.

4.3.3. Overall Results of the Proposed Architectures

By comparing results of ML and DL algorithms for each classifier, we found that the tuned RF has also outperformed the DL algorithms in all measures. Recalling the proposed architectures in section 4.1, our aim is to determine the architecture that offers the FF phishing hostname prediction performance. We first determined overall accuracy rates of the architectures. For hierarchical architectures, we applied the approach used by Kowsari, et al. [55] in which the overall accuracy at the leaf class is obtained by taking accuracy of the child classifier as a fraction of the accuracy of its parent classifier. Since LCPN hierarchical approach propagates misclassification errors from top to bottom [53], we compute the errors at the leaf class by summing up errors of the parent and child classifiers.

$$Overall\ (leaf)\ accuracy = Accuracy\ of\ a\ parent\ classifier * Accuracy\ of\ a\ child\ classifier$$

$$Overall\ (leaf)\ error\ rate = Error\ rate\ of\ a\ parent\ classifier + Error\ rate\ of\ a\ child\ classifier$$

Table 8 below summarizes the best performance of each architecture. As explained above, the performance of architecture C was obtained by combining performances of classifiers C.1 and C.2 while for architecture D, the performances of classifiers D.1 and D.2 were combined. Of the multi-class classification-based architectures, architecture D produced the overall best performance by yielding the highest accuracy and relatively low FPR. However, architecture B, which is based on binary classification, produced the best performance overall in all three metrics.

Architecture	Acc. (%)	FPR (%)	FNR (%)	Classification Type
A	93.58	1.82	8.62	Multi-class
B	**98.42**	**0.57**	**5.88**	**Binary**
C	96.87	1.25	8.87	Multi-class
D	**97.81**	**0.81**	**9.22**	**Multi-class**

Table 8. Prediction performances of the architectures.

4.3.4. Feature Performance Analysis

This section studies the importance of individual features and feature categories in the performance of the best performing classifier (classifier B). The classifier achieved the best performance with 56 features, of which 41 are novel and 15 were adopted from other works. 51 of these features were derived from third party services while only 5 of them were derived from the hostname string (local features). The importance weights of the best features of the classifier (shown in Figure 10) were computed using the tuned RF algorithm (described in section 4.3.1). To help explain this ranking, we will examine the data distributions of some of the best features with respect to the four hostname classes.

The experiment reveals that counts of hosts of a hostname matching with the blacklisted phishing IP addresses (features 77 in Table 1) is the strongest predictor. In observing the data, the boxen plot in Figure 11a shows that hosts of FF phishing hostnames have the largest median counts (indicated by the horizontal black line in) while a large number of them have the large counts of up to nearly 250 compared to phishing non-flux and CDN hostnames. Legitimate non-flux hostnames have the lowest count by far. Similar pattern is observed in a binary classification of classifier B in Figure 11b. This suggests that phishing FFSNs re-use a larger number of machines which were found to host known phishing websites in hosting their phishing websites. This could be through an FFSN hosting multiple phishing websites with the same pool of flux agents or multiple FFSNs hosting their unique websites using the same pool of flux agents.

The high ranking of other IP reputation features (for instance features # 82 and 81 at positions 4 and 5 respectively) and the features related to co-hosted websites (for instance features #53 and 55 at positions 14 and 17 respectively) affirms this trend. For instance, we observed that although hosts of phishing non-flux hostnames have the highest median number of co-hosted websites (indicated in Figure 12a), their resulting median number when they are combined with both CDN and legitimate non-flux hostnames is lowered to 5.5 against 7 of the FF phishing hostnames (Figure 12b). In this feature, we expected phishing FFSNs and CDNs would co-host a large number of malicious and legitimate websites respectively compared to non-flux networks. A possible explanation for the observed data distribution is that phishing non-flux networks require the fewest resources and therefore they are cheap to build and maintain compared to the other networks. Since phishers are driven by profits, the networks make an ideal platform to host multiple phishing websites in order to maximize returns of investment. While hosts in CDNs will be high performance machines, the services they support will a have a high demand for resources, so that the number of services running on a given host is likely to be modest. Furthermore, very large internet companies such as Google and Microsoft are likely to own CDNs dedicated to their own services.

The hostname similarity features (KL, JI and ED) are ranked at positions 3, 6, 7, 13 and 16. Although the features were originally used by Yadav, et al. [42] and Fu, et al. [47] to detect algorithmically generated domains (AGDs) hosted in domain flux botnets, their high ranking demonstrates that they are also effective in detecting hostnames hosted in phishing FFSNs. Figure 13a shows the distribution of differences in average edit distances (feature #61) between the sum of averages of ED of hostnames of the three categories (both legitimate and phishing non-flux hostnames) and the average ED of FF phishing hostnames. The distribution illustrates that FF phishing hostnames have the largest median difference followed closely by phishing non-flux hostnames while both legitimate hostnames have the smallest median. This indicates that FF phishing hostnames have the smallest average distance when compared to each other, followed by phishing non-flux and legitimate hostnames. The pattern, also illustrated in Figure 13b in which the

differences were obtained by subtracting the average ED of FF phishing hostnames from the average ED of the non-FF phishing hostnames, shows that FF phishing hostnames require the fewest number of operations to transform one hostname string to another, thus they have the closest resemblance to each in terms of character composition compared to the rest. Similar observations were made in the other four hostname similarity features. The close similarity in median values and patterns between the two categories of phishing hostnames when compared to those of legitimate hostnames suggests that FFSN and phishing non-flux network owners use nearly similar strategies in composing their phishing hostnames, different from the strategies that are used to generate legitimate hostnames.

TTL, domain age and average domain age of co-hosted hostnames lead the ranking of temporal based features at positions 2, 8 and 10. In observing the TTL data, Figure 14a shows that the majority of CDN and FF phishing hostnames have the lowest TTLs compared to non-flux hostnames, as we expected. Most of the CDN hostnames have the lowest TTLs of 20s and 60s compared to the rest, while a significant number of them have a TTL of 300s. Most of the FF phishing hostnames, on the other hand, have TTLs of 60s followed by 300s and 3600s. The majority of the non-flux hostnames have a TTL value of 300s while a significant percentage have TTLs of 60s, 600s and 3600s. Phishing non-flux hostnames are observed to have the largest range of TTLs of up to 144000s. However, when combining CDN and the two non-flux hostnames as a single category, a large percentage of their hostnames have high TTLs values at 300s, 600s and 3600s when compared to those of FF phishing hostnames, which most of them have TTL of 60s and 90s. The prediction significance of domain age in classifier B is also reflected in our data analysis in Figure 15. Although the median domain age of phishing non-flux hostnames is the lowest (Figure 15a), the value is increased considerably to 1219 days when they are combined with both categories of legitimate hostnames (Figure 15b). The median age of FF phishing hostnames is 3571 days. Similar pattern is observed in the other domain age related features. With both phishing hostnames having the lowest medians of domain age, this confirms the trend that phishers use significant share of newly registered domains to operate their phishing websites

Average number of hops and average geo-distance between user and hosts of a hostname are the highest ranked spatial based features at positions 11 and 12. The data in Figure 16a shows the distribution of average number of hops between user and hosts of a hostname. Unexpectedly, CDN hostnames have the highest median number while non-flux hostnames have the lowest median number. We expected FF phishing hostnames to have the largest median number due to our understanding that flux agents are often distributed across many networks whereas CDN hostnames would have the smallest median number since CDNs deliver contents to the local users. However, it is important to note that the distribution of data of any feature relating user and the hosts depends on the current geographical location of the user relative to the hosts since hosts of non-FF phishing contents are located in few and specific locations. Different distributions are expected to be generated when users are at different locations. In this case, the feature is among the strong predictors because when combining CDN and non-flux hostnames, the median number is lowered to 9 while that of FF phishing hostname is at 14.5 (see Figure 16b). The low-ranking positions of standard deviation and entropy of countries of hops between user and hosts suggest that the majority of hops in all hostname categories are located in a small number of unique countries, indicating that most of the hostnames are hosted in regions closer to the users.

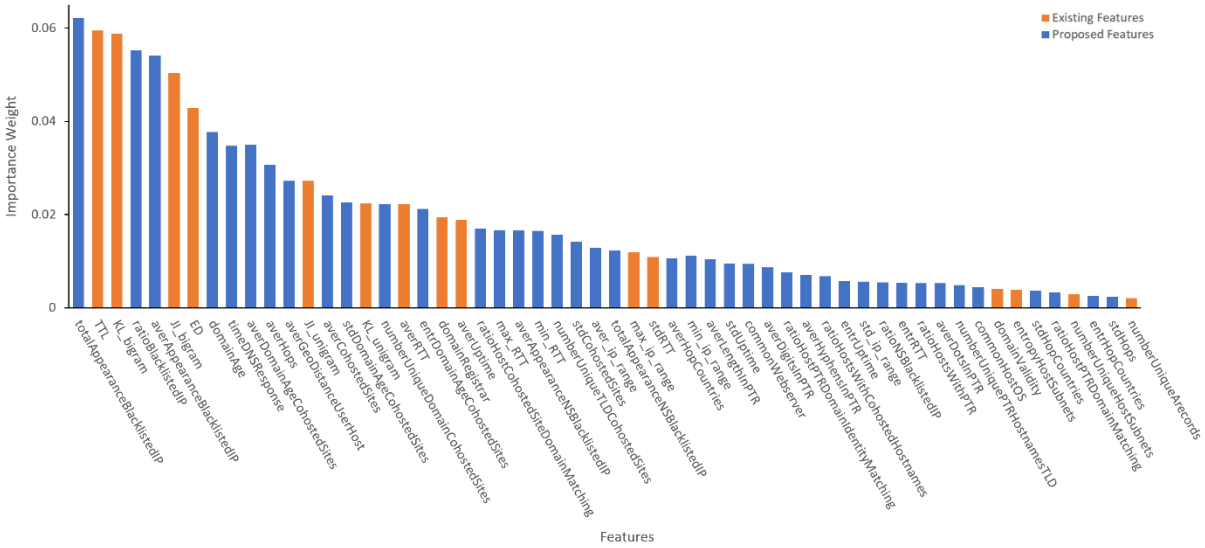

Fig 10. Importance weights of the best features of classifier B.

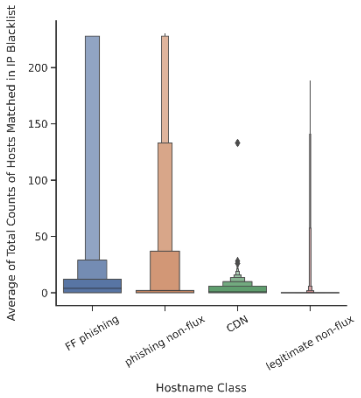

Fig 11a. Distribution of average of total number of occurrences of hosts' IP addresses of the four categories of hostnames in a phishing blacklist.

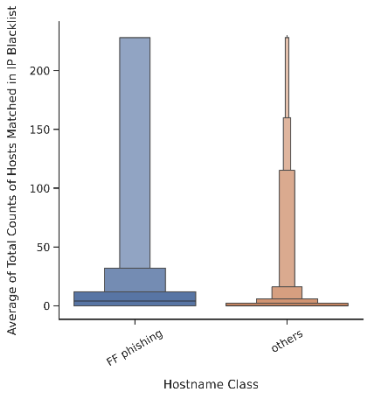

Fig 11b. Distribution of average of total number of occurrences of hosts' IP addresses of two categories of hostnames in classifiers B in a phishing blacklist.

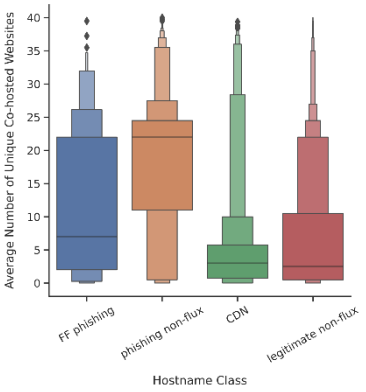

Fig 12a. Distribution of average number of unique co-hosted websites in the hostname's hosts for all four categories of hostnames.

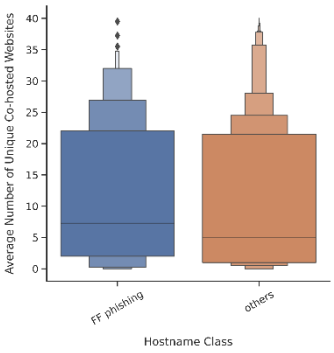

Fig 12b. Distribution of average number of unique co-hosted websites in the hostname's hosts for the two categories of hostnames in classifier B.

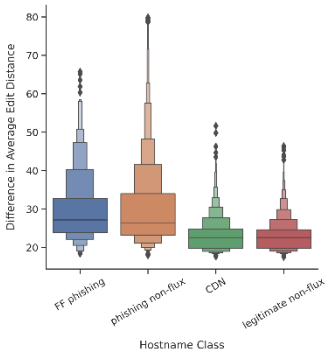

Fig 13a. Distribution of differences of edit distances (Levenshtein) between FF phishing hostnames and each of the four categories of the hostnames.

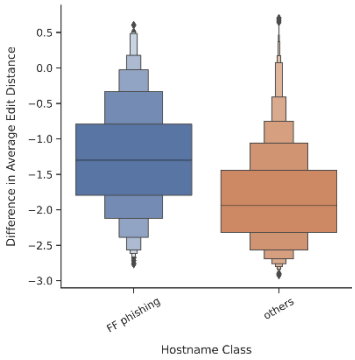

Fig 13b. Distribution of differences of edit distances between FF phishing hostnames

and each of the two categories of the hostnames in classifier B.

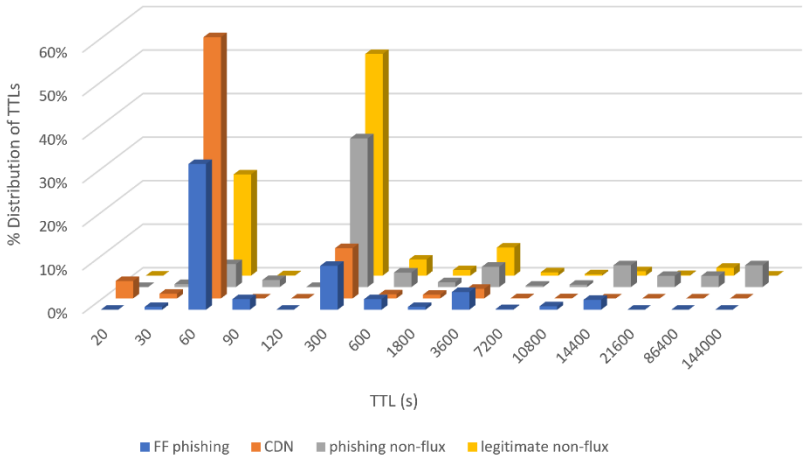

Fig 14a. Distribution of TTLs of the four categories of hostnames.

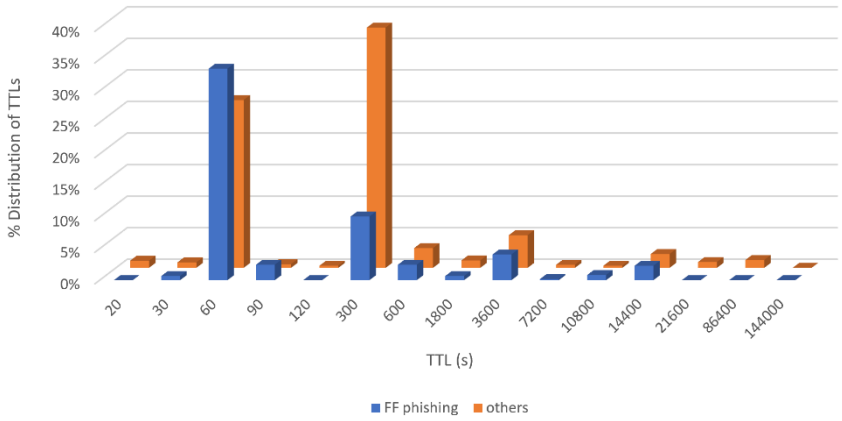

Fig 14b. Distribution of TTLs of the two categories of hostnames in classifier B.

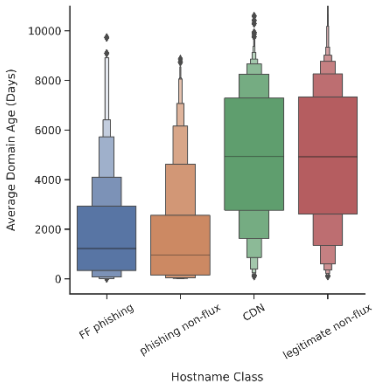

Fig 15a. Distribution of average domain age of the four categories of hostnames.

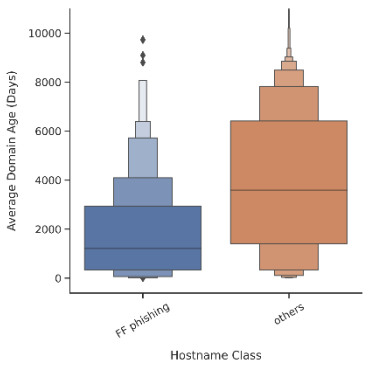

Fig 15b. Distribution of average domain age of hostnames of the two categories of hostnames in classifier B.

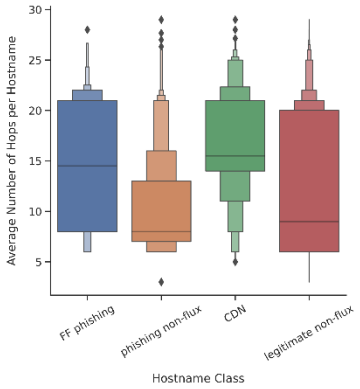

Fig 16a. Distribution of average number of hops between user and hosts of a hostname in each of the four categories of hostnames.

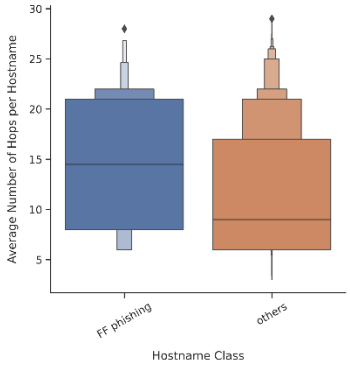

Fig 16b. Distribution of average number of hops between user and hosts of a hostname in each of the two categories of hostnames.

We also analysed the composition and the performance contribution of each feature category in the best feature subset of classifier B. Table 9 indicates the breakdown of the best features of classifier B by category. Compared with the full set of features proposed in section 3.3, there is a representation of features from each category, indicating that all the categories are important in predicting the FF phishing hostnames. Existence of all the proposed temporal, DNS and reputation features in the best feature list indicates that the features categories are the most important in this classification task. As illustrated in Figures 17a-b, temporal and DNS categories produced the highest accuracy rates and among the lowest error rates, along with spatial features, and therefore determined as strong predictors. The network and the host features, on the other hand, are the weakest predictors because they produced the lowest accuracy rates and the highest error rates. This result coincides with the results in Table 9 which shows that temporal and DNS categories have among the highest percentages of features in the best feature subset and the results in Figure 10 which indicates that majority of them are highly ranked. This is opposite to the results of the network and host categories. The poor performances of network features compared to temporal and spatial features suggests that flux agents in FFSNs are located in small number of unique networks but are largely dispersed than the hosts of non-FFSNs. Compared with the overall accuracy, FPR and FNR, each category has performed lesser than the overall performance in all three metrics. This shows that the combination of all feature categories has improved the overall performance of the model.

Similarly, we evaluated and compared performance contributions of the proposed and the existing features against the overall performance of the model (results illustrated in Figures 17c-d). While our proposed features have achieved better results compared to the existing features, the combination of the two has improved the overall accuracy, and more significantly to FPR and FNR, suggesting that their combination is important as for the case of feature categories, not only to achieve optimal prediction performances, but also to increase the diversity of features for hardening the model against detection evasion techniques.

Feature Category	**# of Full Features**	**# of Best Features**	**Best Features # (# from Table 1)**
Temporal	15	15	1-15
Spatial	23	10	16 - 20, 28, 35 - 38

DNS	23	20	39, 40, 42 - 50, 53 - 61
Network	9	2	62, 63
Host	6	2	73, 75
Reputation	7	7	77 - 83

Table 9. Composition of categories in the best feature set of classifier B.

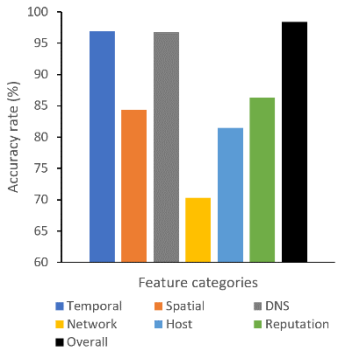

Fig 17a. Comparison of accuracy rates of feature categories of classifier B.

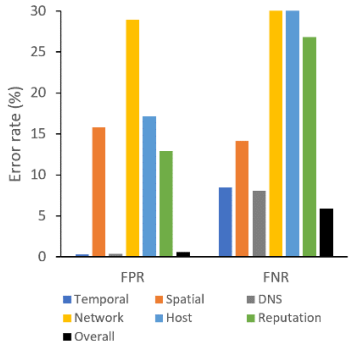

Fig 17b. Comparison of error rates of feature categories of classifier B.

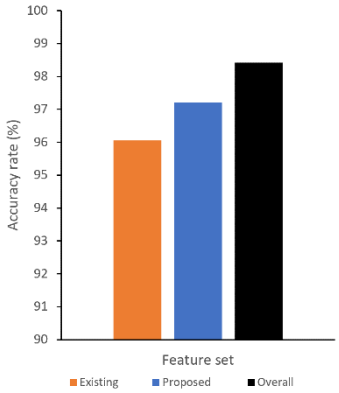

Fig 17c. Comparison of accuracy rates between existing, proposed and the overall features of classifier B.

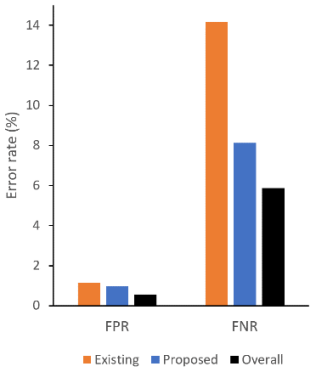

Fig 17d. Comparison of error rates between existing, proposed and the overall features of classifier B.

4.4. Detection Time

We measured the runtimes of the model's three phases, feature extraction, dataset training and prediction, to understand its efficiency in detecting FF hostnames. Table 10 summarizes the times. To determine the actual detection time per webpage, we sum up an average time to extract the 56 features per a webpage and an average time to predict a new webpage, which gives a total of 162.64 seconds. Almost all of this time is due to the retrieval of features' data from their sources. Most features are the results of queries of data from online third-party services, which entail data retrieval and network overheads. All the features were extracted and generated sequentially. Figure 18 below indicates a distribution of the feature extraction time by activities showing a considerable variation of times, with host scanning (using Nmap) and traceroute querying consuming the most time. It is important to note that the runtime of each activity was affected by the Python libraries we selected to use as well as the function(s) and the coding style we used to implement the activity. We expect a considerable speed-up to be achievable through more efficient coding and choice of libraries,

and by querying services concurrently where possible. Nevertheless, it is likely that the latency will not be acceptable for real-time use.

Phase	Time (s)
Training the dataset	8.50
Feature extraction per webpage	162.64
Prediction per webpage	0.0003
Detection Time	**162.64**

Table 10. Runtimes of the classifier's three stages for predicting FF phishing hostnames.

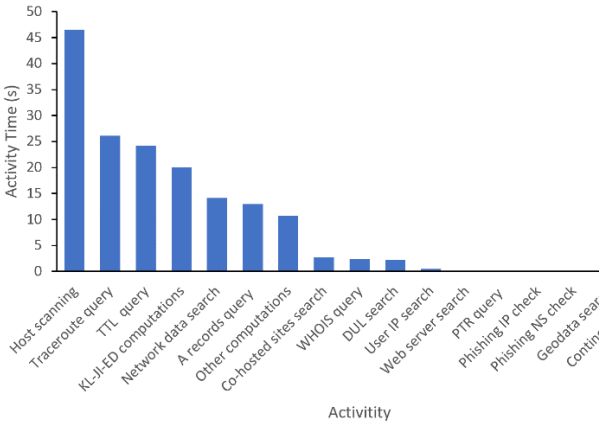

Fig 18. Distribution of feature extraction times by activities.

5. Discussions

5.1. Comparison with Existing Works

We compare previously proposed fast FF hostname detection methods with our work in terms of prediction performance and execution time. For fair comparison, we focus on methods which utilized supervised ML techniques and used datasets of a similar size to ours. Table 11 summarizes the comparisons. The below points summarize the main benefits of our methods compared to existing ones.

While all other works have proposed classifiers to detect FF hostnames operating general malicious services including spam, malware and phishing, our work has refined the scope by focusing on those hosting phishing websites only. Having a detection solution specific for phishing-based FF hostnames is important because FFSNs hosting different types of attacks have significant variations in network behaviours which are used to determine most of the common detection features for FF hostnames. Generic FF hostname detection solutions, thus, are likely to be less effective in detecting specific FF hostnames than the solutions for specific FF hostnames. The difference in network behaviours of various FFSNs was empirically investigated by Caglayan, et al. [4]. The study showed that different types of FFSNs have different characteristics due to their unique economic models. For instance, they observed that spam FFSNs have longer lifespans, between 30 and 90 days while most of the phishing FFSNs live less than a week. Since the longer the FFSN exists, the more agents and domains are being recruited and hosted respectively, spam FFSNs tend to have large network sizes as well as large numbers of hosted domains compared to phishing FFSNs. The study illustrated this difference by observing that the average number of domains hosted in spam FFSNs and phishing FFSNs were 71 and 19 respectively. Such differences in their structures result in differences in a number of characteristics that are often used in detecting FF hostnames. The characteristics include network and geographical dispersion of hosts, and distribution of co-hosted websites. Phishing attacks are, by far, the leading cause of all global security breaches, contributing up to 91% of global data breaches and 93% of data breaches occurring specifically in organizations [8, 67, 68]. Given the significance of phishing to the global security, it is vital to have specific methods for detecting FF phishing hostnames to help prevent internet users falling victim to phishing attacks, thus reducing the number of data breaches.

In contrast to the existing works, our dataset includes all four types of networks (described in sections 2 and 4.2) with the majority of phishing websites being hosted in non-flux networks. This setting is more realistic and relevant to real world applications. It has also allowed us to evaluate the effectiveness of

different classification architectures described in section 4.1.

We have used features belonging to six different feature categories, the most compared to the other studies. In comparison, Yadav, et al. [42] used features from a single category while Hsu, et al. [28], Lin, et al. [25] and Almomani [1] used features from two categories and Jiang and Li [12] from four. Our widely diverse set of features is likely to increase resilience of the solution to detection evasion techniques. This is because the attacker has to develop different evasion methods to compromise the features from each or most of the feature categories to increase significantly their chance of evading detection. In addition, most of our best features were extracted from third-party services. Third party features are difficult to emulate or forge since they are generated from highly secure and verified information. It would take extensive skills and time for an attacker to forge the features. Attackers will be unwilling to incur such costs as they look to maximize their financial gains using minimum effort.

In contrast to current methods, our method has been evaluated using a large number of measures to prove its reliability and all-round robustness. Existing approaches have evaluated their methods using only a small number of performance measures (as indicated in Table 11), omitting other significant measures such as FNR, which we consider to be one of the most important measures for this problem. This limits our understanding on the all-round effectiveness of these solutions.

Most existing methods work by distinguishing FF hostnames from both CDN and legitimate non-flux hostnames. Malicious non-flux hostnames, which host the majority of malicious websites, were not considered in their datasets. However, our approach distinguishes FF hostnames from all possible types of hostnames (including phishing non-flux hostnames), thus it is more related to the real-world use cases. Notably, Lin, et al. [25] reported a performance similar to ours but with a smaller number of features and lower detection time. Their work, however, did not include malicious non-flux hostnames in their dataset. To be useful in real-world applications, their classifier needs to be merged with other methods, for instance, be preceded in the processing pipeline by a filter to exclude phishing non-flux hostnames from the data stream. We evaluated three of their four features, which we also used, using our dataset with classifier B. We evaluated them first using a three-class dataset, from which we had eliminated phishing non-flux hostnames, and then a four-class dataset. In the former, we obtained an accuracy rate, FPR and FNR of 73.96%, 23.49% and 72.36% respectively whereas in the latter, an accuracy of 60.74%, FPR of 56.42% and FNR of 37.86% were obtained. Note that the accuracy and FPR decreased as a result of adding the fourth hostname class. We expect a similar reduction in the performance of their classifier would be observed if their dataset were expanded to include the fourth hostname class. Also, the features performed less well against our phishing-based dataset, suggesting that the performance is sensitive to the type of FFSNs in the dataset. In addition, it is important to note that our work has achieved better accuracy rate and FPR than their work.

We also compare our work against Yadav, et al. [42]'s work from which we adopted four of the hostname similarity features (features # 57-58 and 60-61). Their work used the features to distinguish AGDs from the legitimate hostnames. We evaluated all the four features on the dataset of our classifier D.1 which distinguishes phishing (both AGDs and non-AGDs) from legitimate hostnames. We obtained an accuracy of 92.37% and FPR of 4.82%. Comparing with performances reported by Yadav, et al. [42] (see Table 11), the results show that the features are more effective in detecting AGDs than the combination of AGDs and non-AGDs. This is because AGDs are more

likely to resemble in terms of composition of their characters compared to non-AGDs since the malware of each domain flux network generating them follows a specific and uniform pattern. Nevertheless, the features have shown to increase significantly the performances of our classifiers, suggesting that they are also useful in classifying other categories of hostnames. For instance, when excluding the features, classifier D.1 produced an accuracy of 96.37% and FPR of 3.85% but the performances were improved to 98.64% and 0.62 respectively after including the features. The improvements were also observed in the other classifiers. Yadav, et al. [42], observed that JI performed the best followed by ED and lastly KL. Our results showed that ED performed the best followed by JI and KL.

Good performances of our proposed features compared to the existing features on the same dataset and that of our model against the models developed by other works suggest that there are other undiscovered features that are as effective as or more than the existing ones. Given that phishers continue to learn about the existing features with an aim of subverting the solutions, it is important that researchers also continue to explore and add new potential features so as to be ahead of the race against the attackers, thus, making the solutions relevant and effective in different periods and seasons.

It is important to note that the results obtained in our work are based on the size and nature of distribution of our dataset. Attackers are likely to vary configurations of their FFSNs over the time which may result in variations in some of the detection features. This is likely to impact the distribution of the training dataset, thus the performances of the classifiers and the architectures. We insist that continuous observations of behaviours of FFSNs and evaluation of the resulting collected dataset are important to ensure that the detection results are up to date and relevant on each monitoring period.

Work	Features	Data size (URLs)	Classification Type	Evaluation Algorithms	Performance	Detection Time (s)
Hsu, et al. [28]	2 temporal features, 4 host features	17, 214	Binary classification (FF bots versus benign server)	SVM	Acc. = 95% AUC = 0.993	Unknown
Lin, et al. [25]	3 DNS features, 1 temporal feature	12,952	Binary classification (FF hostnames versus benign hostnames)	Genetic algorithm	Acc. = 98.2% FPR = 1.78%	18.54
Jiang and Li [12]	2 DNS features, 1 network feature, 1 temporal feature, 1 spatial feature	26,873	Binary classification (FF hostnames versus benign hostnames)	SVM, Naïve Bayes, K-NN	Acc. = 96.7% Prec. = 0.965 Recall = 0.981 F1 = 0.973 AUC = 0.989	400
Almomani [1]	11 DNS features, 3 temporal features	7,615	Binary classification (FF hostnames versus benign hostnames)	Adaptive evolving fuzzy neural network, SVM, Naïve Bayes	Acc. = 98% Error rate = 1 – 16 (%)	Unknown

Yadav, et al. [42]	4 features DNS features	79,930	Binary classification (AGDs versus benign hostnames)	L1-Regularized Linear Regression	Acc. = 100% FPR = 2.0%	Unknown
Our work	15 temporal, 23 spatial, 20 DNS, 4 network, 3 host, 7 reputation	11,801	Binary classification (FF phishing hostnames versus other hostnames)	Linear Regression, K-NN, Decision Tree, Naive Bayes, SVM, Neural Network, Random Forest, Gradient Boosting, Fully-connected Deep Neural Network, LSTM, CNN	Acc. =98.42% FPR = 0.57% FNR = 5.88% Prec. = 0.99 Recall = 0.99 F1 = 0.99 AUC = 0.99	162.64
			Multi-class classification (FF phishing hostnames versus phishing non-flux, CDN hostnames and legitimate non-flux hostnames)		Acc. =97.81% FPR = 0.81% FNR = 9.22%	

Table 11. A summarized comparison chart between our work and the related works.

5.2. Applications, Limitations and Future Work

Though the subset of 56 best features have produced the optimal performance, smaller subsets have also produced good accuracy and moderate error rates. For instance, the top 7 best features produced an accuracy, FPR and FNR of 96.24%,15.87% and 1.56% respectively. If accuracy and FNR are the most desirable performances, then a trade-off can be made between accuracy and number of features in order to reduce computing overheads, thus, the detection time. However, from the cybersecurity point of view, reducing the number of features increases the risk of detection evasion which may led to serious damage to users. We argue that the use of a large number of features, not only improves the performance across all measures, but also enhances resilience to detection evasion for the long-term reliability value of the solution. The impact of longer detection time due the use of large number of features can be countered by using the classifier to build the solution (described below) that will provide real time detection to end users. When applied in this way, our classifier achieves three important goals; high detection performance, enhanced resilience to detection evasion and real time detection.

Our detection time is higher than the typical delay a user can expect when accessing a webpage, measured to be in the region of 8-9s in [69]. Consequently, our model is not suitable for providing direct real-time protection to end users. Instead, we suggest that it would make effective use in building and maintaining a blacklist of FF phishing hostnames. Used in this way, the classifier would be fed with a stream of hostnames obtained from various sources such as user emails, network traffic, and databases of legitimate and phishing websites. Those websites classified as FF phishing hostnames would be added to the blacklist. The blacklist then can be used in various ways to provide real time protection to end users.

Examples of such applications are a web browser plug-in and a cloud anti-malware suite with its clients installed at the end users. A blacklist approach has proved to be efficient for real-time applications in other cybersecurity related areas [70, 71]. The blacklist can also be a useful resource for security researchers, vendors and authorities for further investigations on the operations of phishing FFSNs. Currently there are no active blacklists consisting of known FF phishing hostnames only that we are aware of. Also, our classifier D.2, which distinguishes FF phishing and phishing non-flux hostnames with a high performance (indicated in Table 5b), can be integrated (as a second layer) with a real time solution which distinguishes phishing and legitimate hostnames, such as the one proposed in Nagunwa, et al. [5], to offer a more comprehensive solution for the mitigation of phishing attacks. In this approach, the former detects FF phishing hostnames from the phishing hostnames detected by the latter.

The proposed multi-class model using architecture D has produced relatively good results. The advantage of this model is that it predicts each hostname type as an independent outcome. This can be useful in cases where security experts want to distinguish FF phishing hostnames from phishing non-flux hostnames. By doing so, specific measures to address the attacks at the network level can be applied. For instance, for the former, this would require an approach described in section 1. For the latter, hosts of the hostnames can be directly identified through querying A records of the hostnames and then blacklist the returned IP addresses.

The current detection time of our models can be reduced in an attempt to achieve a latency suitable for a potential real-time application. For instance, most of the features can be extracted in parallel instead of sequentially. Query of A records would be performed first to retrieve hosts' IP addresses before extracting the rest of the features in parallel. Furthermore, features obtained from some of the time-consuming feature extraction activities (as described in section 4.4) can be dropped to further reduce the time at the expense of detection performance. For instance, by dropping all best features of classifier B obtained through host scanning, the detection time obtained was 44.6s while the accuracy attained was 96.85%. We will further explore this analysis as part of our future work.

One of the key limitations of our solution is that data from online third-party services, from which the majority of our features are derived, may be missing from time to time, for reasons including poor network connection, temporary unavailability of servers or unfound records. A high percentage of missing data is likely to reduce detection performance. However, our experience during data collection suggests that scenarios that could give rise to missing data are relatively rare in normal daily circumstances. In some features missing data is quite common, for instance 56.4% and 44.6% of data related to PTR and uptime of hosts respectively was missing. Though we attempted to drop features with missing data, better results were obtained with the full feature set. This suggests that our solution is resilient against missing data in several features.

Recently, much attention has been paid to adversarial attacks and how to make ML/DL algorithms resistant to them. See e.g. the MITRE Adversarial Threat Landscape for Artificial-Intelligence Systems (ATLAS) knowledge [72]. A comprehensive study of the robustness of our proposed feature set combined with each of the ML/DL algorithms against such attacks is beyond the scope of the current paper. However, it is under consideration as a topic for a follow-on project.

6. Conclusion

In this paper, we have proposed a comprehensive set of predictive features for the model for detecting FF phishing hostnames using a supervised ML approach. In particular,

we have introduced 41 new predictive features alongside 15 features, categorized in six groups, that were used in existing works to build the model. Four model implementation architectures based on binary and multi-class classification approaches were proposed and evaluated using eight ML and three DL algorithms. The model performance was evaluated using seven reliable performance measures. The binary classification-based architecture achieved the highest accuracy of 98.42%, in which the FF phishing hostnames are distinguished from the other three hostnames combined as a single hostname class. The multi-class classification-based architecture yielded a highest accuracy of 97.81%, where all four hostnames are identified, which allows for detecting the exact type of hostname and hence help to take more informed decision. We also investigated the importance of the proposed features with respect to the best performing architecture and revealed that temporal and DNS related features are strongest predictors while network and host related features are the weakest. The proposed approach has delivered a comparable detection performance when compared against other similar works in the literature. However, unlike exiting approaches, we propose a more robust solution as we have used many new features, have addressed the problem in a four-class classification context and have reported our results using a wider range of performance measures.

As part of our future work, we intend to analyse changes in the performance of the model in response to the removal of features that contribute most to extraction times, with an aim of finding the optimal combination of detection time and detection performance. As observed by Salusky and Danford [2] and Yadav, et al. [42], it is likely that attackers can combine IP flux with domain flux or name server flux or both of them in the same botnets. We aim at extending our work by exploring and identifying hostnames that are hosted in botnets with multiple fluxing behaviours.

Acknowledgment

This research was funded by Commonwealth Scholarship Commission (UK) and Birmingham City University (UK). The results and the views of this research are from the authors and were not influenced by the funding bodies.

References

[1] A. Almomani, "Fast-flux hunter: a system for filtering online fast-flux botnet," *Neural Computing and Applications*, vol. 29, pp. 483-493, 2018.

[2] W. Salusky and R. Danford. (2008, April 2017). Know your Enemy: fast-flux service networks. *The Honeynet Project*. Available: https://www.researchgate.net/publication/328781281_Know_Your_Enemy_Fast-Flux_Service_Networks

[3] O. Katz, Perets,R., Matzliach, G., "Digging Deeper – An In-Depth Analysis of a Fast Flux Network " 2017.

[4] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Behavioral analysis of botnets for threat intelligence," *Inf. Syst. E-bus. Manag.*, vol. 10, pp. 491-519, 2012.

[5] T. Nagunwa, S. Naqvi, S. Fouad, and H. Shah, "A Framework of New Hybrid Features for Intelligent Detection of Zero Hour Phishing Websites," in *12th International Conference on Computational Intelligence in Security for Information Systems (CISIS)*, Seville, Spain, 2019, pp. 36–46.

[6] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: detecting malware infection through IDS-driven dialog correlation," presented at the Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, Boston, MA, 2007.

[7] S. Khattak, Z. Ahmed, A. A. Syed, and S. A. Khayam, "BotFlex: A community-driven tool for botnet

detection," *Journal of Network and Computer Applications,* vol. 58, pp. 144-154, 2015.

[8] Sophos. (2017, June 2020). *Don't take the bait* Available: https://www.cygnussystems.com/wp-content/uploads/2017/08/dont-take-the-bait.pdf

[9] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "FluXOR: Detecting and Monitoring Fast-Flux Service Networks," in *Proc. 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Paris, France, 2008, pp. 186-206.

[10] R. Perdisci, I. Corona, D. Dagon, and W. Lee, "Detecting malicious flux service networks through passive analysis of recursive dns traces," presented at the Proc. 2009. ACSAC'09. Annual Computer Security Applications Conference Honolulu, HI, USA, 2009.

[11] F. Hsu, C. Wang, C. Hsu, C. Tso, L. Chen, and S. Lin, "Detect Fast-Flux Domains Through Response Time Differences," *IEEE Journal on Selected Areas in Communications,* vol. 32, pp. 1947-1956, 2014.

[12] C. Jiang and J. Li, "Exploring Global IP-Usage Patterns in Fast-Flux Service Networks," *Journal of Computers,* vol. 12, pp. 371-379, 2017.

[13] S. Kumar and B. Xu, "A Machine Learning Based Approach to Detect Malicious Fast Flux Networks," presented at the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018.

[14] M. Stevanovic, J. M. Pedersen, A. D'alconzo, and S. Ruehrup, "A method for identifying compromised clients based on DNS traffic analysis," *International Journal of Information Security,* vol. 16, pp. 115-132, 2017.

[15] X. Chen, G. Li, Y. Zhang, X. Wu, and C. Tian, "A Deep Learning Based Fast-Flux and CDN Domain Names Recognition Method," presented at the Proceedings of the 2019 2nd International Conference on Information Science and Systems, Tokyo, Japan, 2019.

[16] E. Stalmans, S. O. Hunter, and B. Irwin, "Geo-spatial autocorrelation as a metric for the detection of Fast-Flux botnet domains," presented at the 2012 Information Security for South Africa, 2012.

[17] BuiltWith. (2020, March, 2020). *Content Delivery Network Usage Statistics*. Available: https://trends.builtwith.com/CDN/Content-Delivery-Network

[18] Hosting Facts. (2020, June, 2020). *Internet Stats & Facts (2020)*. Available: https://hostingfacts.com/internet-facts-stats/

[19] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains," *ACM Trans. Inf. Syst. Secur.*, vol. 16, pp. 1-28, 2014.

[20] D. Barr. (1996, August, 2019). *Common DNS Operational and Configuration Errors*. Available: https://www.ietf.org/rfc/rfc1912.txt

[21] G. Aaron and R. Rasmussen. (2017, December, 2017). *Global Phishing Survey: Trends and Domain Name Use in 2016*. Available: http://docs.apwg.org/reports/APWG_Global_Phishing_Report_2015-2016.pdf

[22] Y. Chang, K. Yoon, and D. Park, "A Study on the IP Spoofing Attack through Proxy Server and Defense Thereof," in *2013 International Conference on Information Science and Applications (ICISA)*, 2013, pp. 1-3.

[23] R. Prego. (2016, August, 2019). *5 Reasons Your Company Should Use Proxy Servers*. Available: https://www.cmswire.com/information-management/5-reasons-your-company-should-use-proxy-servers/

[24] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem," *Telecommunications Policy,* vol. 41, pp. 1003-1016, 2017/11/01/ 2017.

[25] H.-T. Lin, Y.-Y. Lin, and J.-W. Chiang, "Genetic-based real-time fast-

flux service networks detection," *Computer Networks*, vol. 57, pp. 501-513, 2013/02/04/ 2013.

[26] Vivek. (2017, March, 2020). *Does ever CDN server goes down?* Available: https://www.quora.com/Does-ever-CDN-server-goes-down

[27] S. Campbell, S. Chan, and J. R. Lee, "Detection of fast flux service networks," presented at the Proceedings of the Ninth Australasian Information Security Conference - Volume 116, Perth, Australia, 2011.

[28] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen, "Fast-Flux Bot Detection in Real Time." vol. 6307, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 464-483.

[29] ThreatX Labs. (2017, August, 2019). *Malicious Bot Detection through A Complex Proxy Network*. Available: https://blog.threatxlabs.com/malicious-bot-detection-through-complex-proxy-network

[30] M. Konte, N. Feamster, and J. Jung, "Dynamics of online scam hosting infrastructure," presented at the Proc. International conference on passive and active network measurement, 2009.

[31] T. Holz, C. Gorecki, K. Rieck, and F. Freiling, "Measuring and Detecting Fast-Flux Service Networks," presented at the Proc. 16th Annual Network & Distributed System Security Symposium (NDSS), San Diego, CA, 2008.

[32] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," presented at the 3rd International Conference on Malicious and Unwanted Software (MALWARE), Fairfax, VI, USA 2008.

[33] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," *Computers & Electrical Engineering*, vol. 50, pp. 91-101, 2016.

[34] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: signatures and characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 171–182, 2008.

[35] C. Lai, A. Chavez, C. Jones, N. Jacobs, S. Hossain-McKenzie, J. Johnson, *et al.*, *Review of Intrusion Detection Methods and Tools for Distributed Energy Resources*, 2021.

[36] Y. Xing, H. Shu, H. Zhao, D. Li, and L. Guo, "Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation," *Mathematical Problems in Engineering*, vol. 2021, 2021.

[37] A. Karim, R. B. Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: review, future trends, and issues," *Journal of Zhejiang University SCIENCE C*, vol. 15, pp. 943-983, 2014/11/01 2014.

[38] Y. j. Ou, Y. Lin, Y. Zhang, and Y. j. Ou, "The Design and Implementation of Host-Based Intrusion Detection System," in *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, 2010, pp. 595-598.

[39] W. Lu, G. Rammidi, and A. A. Ghorbani, "Clustering botnet communication traffic based on n-gram feature selection," *Comput. Commun.*, vol. 34, pp. 502–514, 2011.

[40] H. Choi and H. Lee, "Identifying botnets by capturing group activities in DNS traffic," *Computer Networks*, vol. 56, pp. 20-33, 2012/01/12/ 2012.

[41] G. Creech and J. Hu, "A Semantic Approach to Host-Based Intrusion Detection Systems Using Contiguousand Discontiguous System Call Patterns," *IEEE Transactions on Computers*, vol. 63, pp. 807-819, 2014.

[42] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with DNS traffic analysis," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1663-1677, 2012.

[43] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, *et al.*, "From throw-away traffic to bots: detecting the rise of DGA-based malware," in *Proc. 21st USENIX conference on Security symposium*, Bellevue, WA, 2012, pp. 24-24.

[44] T. Kelley and E. Furey, "Getting prepared for the next botnet attack: Detecting algorithmically generated domains in botnet command and control," in *2018 29th Irish Signals and Systems Conference (ISSC)*, 2018, pp. 1-6.

[45] V. Ravi, M. Alazab, S. Srinivasan, A. Arunachalam, and K. P. Soman, "Adversarial Defense: DGA-Based Botnets and DNS Homographs Detection Through Integrated Deep Learning," *IEEE Transactions on Engineering Management*, pp. 1-18, 2021.

[46] S. K. P. Vinayakumar R., Poornachandran P., Alazab M., Jolfaei A., "DBD: Deep Learning DGA-Based Botnet Detection," in *Deep Learning Applications for Cyber Security*, T. M. Alazab M., Ed., ed: Springer, 2019.

[47] Y. Fu, L. Yu, O. Hambolu, I. Ozcelik, B. Husain, J. Sun, *et al.*, "Stealthy domain generation algorithms," *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 1430-1443, 2017.

[48] S.-Y. Huang, C.-H. Mao, and H.-M. Lee, "Fast-flux service network detection based on spatial snapshot mechanism for delay-free detection," presented at the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010.

[49] H. Wang, C. Mao, K. Wu, and H. Lee, "Real-Time Fast-Flux Identification via Localized Spatial Geolocation Detection," presented at the 2012 IEEE 36th Annual Computer Software and Applications Conference, 2012.

[50] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, pp. 79-86, 1951.

[51] H. Small, "Co-citation in the scientific literature: A new measure of the relationship between two documents," *Journal of the American Society for information Science*, vol. 24, pp. 265-269, 1973.

[52] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, 1966, pp. 707-710.

[53] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31-72, 2011.

[54] N. Weiss. (2020). *Hierarchical Classification with Local Classifiers: Down the Rabbit Hole*. Available: https://towardsdatascience.com/hierarchical-classification-with-local-classifiers-down-the-rabbit-hole-21cdf3bd2382

[55] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," presented at the 2017 16th IEEE international conference on machine learning and applications (ICMLA), 2017.

[56] C. N. Silla and A. A. Freitas, "Novel top-down approaches for hierarchical classification and their application to automatic music genre classification," presented at the 2009 IEEE International Conference on Systems, Man and Cybernetics, 2009.

[57] J. Brownlee. (2014, August, 2018). *Classification Accuracy is Not Enough: More Performance Measures You Can Use*. Available: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/

[58] A. Müller and S. Guido, *Introduction to Machine Learning with Python*, 1 ed. U.S: O'Reilly Media, 2017.

[59] J. Brownlee. (2018, January, 2019). *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. Available: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

[60] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, *et al.*, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35365-35381, 2018.

[61] M. Al-Garadi, M. Amr, A. Al-Ali, X. Du, and M. Guizani, "A survey of machine and deep learning methods

for internet of things (IoT) security," *arXiv preprint arXiv:1807.11023*, 2018.

[62] G. Apruzzese, Colajanni, M., Ferretti, L., Guido, A., Marchetti, M., "On the effectiveness of machine and deep learning for cyber security," in *2018 10th International Conference on Cyber Conflict (CyCon)*, Estonia, 2018, pp. 371-390.

[63] J.-h. Li, "Cyber security meets artificial intelligence: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 1462-1474, 2018/12/01 2018.

[64] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1200-1205.

[65] J. Brownlee, *Machine Learning Mastery With Python*, 1.4 ed., 2016.

[66] P. Worcester. (2019, May, 2021). *Comparison of Grid Search and Randomized Search Using Scikit Learn*. Available: https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85

[67] Sophos. (2019, June, 2020). *Don't Take the Bait*. Available: https://secure2.sophos.com/en-us/medialibrary/Gated-Assets/white-papers/Dont-Take-The-Bait.pdf

[68] Verizon. (2018, May, 2020). *2018 Data Breach Investigations Report*. Available: https://www.phishingbox.com/assets/files/images/Verizon-Data-Breach-Investigations-Report-2018.pdf

[69] MachMetrics. (2018, February, 2018). *Average Page Load Times for 2018 – How does yours compare?* Available: https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/

[70] H. Kordestani and M. Shajari, "An entice resistant automatic phishing detection," in *Proc. 5th Conference on Information and Knowledge Technology (IKT)*, Shiraz, Iran, 2013, pp. 134-139.

[71] Y.-S. Chen, Y.-H. Yu, H.-S. Liu, and P.-C. Wang, "Detect phishing by checking content consistency," in *Proc. 15th IEEE International Conference on Information Reuse and Integration (IRI)*, Redwood City, CA, USA, 2014, pp. 109-119.

[72] MITRE. (n.d, December, 2021). *MITRE Adversarial Threat Landscape for Artificial-Intelligence Systems*. Available: https://atlas.mitre.org/