Malicious URL Detection via Machine Learning

Daniel Ventura, Isabella Oren, Dana Engel and Anna Aharonov.
Based on the research of: Chen Hajaj, Nitay Hason, Nissim Harel and Amit Dvir

Abstract—In our everyday life we manage personal risk and infer which situations may be dangerous and avoid them accordingly. However, it translates poorly to the context of the malicious URLs, there are few effective heuristics to differentiate safe URLs from dangerous ones. Internet criminals depend on the absence of such indicators to prey on their marks. A simple URL can cause a lot of damage. The potential harm is so great that malicious links are considered one of the biggest threats to the digital world. Some of the most popular cyber attacks use URLs for the attacks, such as C&C and phishing. Even though there are a lot of studies conducted about the detection of malicious URLs, there's still a lot to discover especially about the weak spots of the defense mechanisms. Since machine learning has become one of the most prominent methods of malware detection, A robust feature selection mechanism is proposed that results in malicious domain detection models that are resistant to evasion attacks. This mechanism exhibits high performance based on empirical data. This paper is meant to help identify whether a link is malicious or benign. We rely on the thesis written by Nitay Hason "Robust Malicious URL Detection". We have tried to find additional features to improve the accuracy of the URL detection, and improved the scores of the classifier. Furthermore, it introduces novel features that are robust to the adversary's manipulation. Based on extensive evaluation of the different feature sets and commonly used classification models this paper show that models which are based on robust features are resistant to malicious perturbations, and at the same time useful for classifying non-manipulated data.

Index Terms—Malware detection, Robust features, Domain

1 Introduction

In the past two decades, cybersecurity attacks have become a major issue for governments and civilians [2]. Many of these attacks are based on malicious web domains or URLs (See Figure 1 for the structure of a URL). These domains are used for phishing [3], [4], [5], [6], [7] (e.g. spear phishing), Command and Control (C&C) [8] and a vast set of virus and malware [9] attacks.

The Domain Name System (DNS) is a hierarchical and decentralized naming system for computers, services, or other resources connected to the internet or a private network. It maps human-readable domain names to their associated IP addresses (e.g., google.com to 172.217.16.174). DNS services can be abused in different ways, in which the adversary manipulates them to conduct various attacks [10], [11]. Therefore, the ability to identify a malicious domain in advance is a massive game-changer [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26].

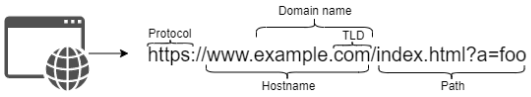


Fig. 1: The URL structure

A common way of identifying malicious/compromised domains is to collect information about the domain names (alphanumeric characters) and network information (such as DNS and passive DNS data[1]). This information is used for extracting a set of features using machine learning algorithms that are trained on massive amount of data. Measuring the distance between a known malicious domain name and the analyzed domain (benign or malicious) is one of many ways to obtain a mathematical approach of detecting the malicious domains [11], [13], [14], [15], [16], [18], [19], [20], [21], [22], [23], [24], [26], [27], [28].

Many ML-based that are used are not robust. An adversary can bypass these models with minimal feature perturbations. Therefore, the main challenge is to identify malicious domains manipulated by an intelligent adversary. The goal is to achieve A feature selection mechanism that is robust to adversarial manipulations for identifying malicious domains. This will prevent even an adversary with black-box access to the model from tampering with the domain properties. A large set of malicious and benign URLs were collected and surveyed for commonly used features, that were then manipulated to show that some of them are slightly robust or not robust at all. Thus, novel features were engineered to enhance the robustness of the models. These novel features support the detection of malicious domains from new angles, but the accuracy of the models that were solely based on these features is not necessarily higher than the original features model accuracy. Therefore, A hybrid set of features, combining A subset of the well-known features

- *C. Hajaj was with Ariel Cyber Innovation Center, Data Science and Artificial Intelligence Research Center, IEM Department, Ariel University, Israel.*
- *N. Hason was with Ariel Cyber Innovation Center, CS Department, Ariel University, Israel.*
- *N. Harel was with CS Department, Holon Institute of Technology, Israel*
- *A. Dvir was with Ariel Cyber Innovation Center, CS Department, Ariel University, Israel.*
  E-mail: amitdv@g.ariel.ac.il

A preliminary version of some of these results appeared in the Proceedings of the Fourth International Symposium on Cyber Security Cryptology and Machine Learning (CSCML-2020) [1].

1. Most works dealing with malicious domain detection are based on DNS features, and only some take the passive DNS features into account as well.

and the novel features was generated. Finally, the different sets of features were evaluated using well-known machine learning and deep learning algorithms, which led to A robust and efficient malicious domain detection system.

The rest of the paper is organized as follows: Section 2 presents the evaluation metric used, and Section 3 summarizes related works. Section 4 describes the methodology and the novel features. Section 5 presents the empirical analysis and evaluation. Finally, Section 6 concludes and summarizes this work.

2 Evaluation Metrics

Machine Learning (ML) is a subfield of Computer Science aimed at getting computers to act and improve over time autonomously by feeding them data in the form of observations and real-world interactions. In contrast to traditional programming, when one provides input and algorithm and receives an output when using ML, one provides a list of inputs and their associated outputs, in order to extract the algorithm that maps the two.

ML algorithms are often categorized as either supervised or unsupervised. In supervised learning, each example is a pair consisting of an input vector (also called data point) and the desired output value (class/label). Unsupervised learning learns from data that has not been labeled, classified, or categorized. Instead of responding to feedback, unsupervised learning identifies commonalities in the data and reacts based on the presence or absence of such commonalities in each new piece of data.

In order to evaluate how a supervised model is adapted to a problem, the dataset needs to be split into two, the training set and the testing set. The training set is used to train the model, and the testing set is used to evaluate how well the model "learned" (i.e. by comparing the model predictions with the known labels). Usually, the train/test distribution is around 75%/25% (depending on the problem and the amount of data). Standard evaluation criteria are as follows: Recall, Precision, Accuracy, F1-score, and Loss. All of these criteria can easily be extracted from the evaluation's confusion matrix.

Confusion matrix (Table 1) is commonly used to describe the performance of a classification model. WLOG, we define positive instances as malicious and negative as benign. Recall (Eq. 1) is defined as the number of correctly classified malicious examples out of all the malicious ones. Similarly, Precision (Eq. 2) is the number of correctly classified malicious examples out of all examples classified as malicious (both correctly and wrongly classified). Accuracy (Eq. 3) is used as a statistical measure of how well a classification test correctly identifies or excludes a condition. That is, the accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. Finally, F1-score (Eq. 4) is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The F1-score is the harmonic average of the precision and recall, where an F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0. These criteria are used as the main evaluation metric. Since a classification model is being examined here, the Logarithmic loss (Eq. 5) was chosen as the loss function.

		Prediction Outcome		
		Positive	Negative	**Total**
Actual Value	Positive	True Positive	False Negative	TP+FN
	Negative	False Positive	True Negative	FP+TN
	Total	P	N	

TABLE 1: Confusion Matrix

In this research, the problem of identifying malicious web domains can be classified as supervised learning, as the correct label (i.e. malicious or benign) can be extracted using a blacklist-based method, as we describe in the next chapter.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{P} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{T}{P + N} \tag{3}$$

$$F_1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4}$$

$$Loss = -\frac{1}{N} \sum_{i=1}^{N} (y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)) \tag{5}$$

3 Related Work

The issue of identifying malicious domains is a fundamental problem in cybersecurity. This section discusses recent results in identifying malicious domains, focusing on three significant methodologies: Mathematical Theory approaches, ML-based techniques, and Big Data approaches.

The use of graph theory to identify malicious domains was more pervasive in the past [17], [26], [29], [30], [31]. Yadav et al. [26] presented a method for recognizing malicious domain names based on fast flux. Fast flux is a DNS technique used by botnets to hide phishing and malware delivery sites behind an ever-changing network of compromised hosts acting as proxies. Their methodology analyzed the DNS queries and responses to detect if and when domain names are being generated by a Domain Generation Algorithm (DGA). Their solution was based on computing the distribution of alphanumeric characters for groups of domains and by statistical metrics with the KL (Kullback Leibler) distance, Edit distance and Jaccard measure to identify these domains. Their results for a fast-flux attack using the Jaccard Index achieved impressive results, with 100% detection and 0% false positives. However, for smaller numbers of generated domains for each TLD, their false positive results were much higher, at 15% when 50 domains were generated for the TLD using the KL-divergence over unigrams, and 8% when 200 domains were generated for each TLD using Edit distance.

Dolberg et al. [17] described a system called *Multi-dimensional Aggregation Monitoring (MAM)* that detects

anomalies in DNS data by measuring and comparing a "steadiness" metric over time for domain names and IP addresses using a tree-based mechanism. The steadiness metric is based on a similar domain to IP resolution patterns when comparing DNS data over a sequence of consecutive time frames. The domain name to IP mappings were based on an aggregation scheme and measured steadiness. In terms of detecting malicious domains, the results showed that an average steadiness value of 0.45 could be used as a reasonable threshold value, with a 73% true positive rate and only 0.3% false positives. The steadiness values might not be considered a good indicator when fewer malicious activities were present (e.g. $<$10%).

However, the most common approach to identifying malicious domains is using machine learning (ML) [10], [11], [15], [21], [24], [27], [28], [32], [33]. Using a set of extracted features, researchers can train ML algorithms to label URLs as malicious or benign. Shi et al. [10] proposed a machine learning methodology to detect malicious domain names using the Extreme Learning Machine (ELM) [20] which is closest to the one employed here. ELM is a new neural network with high accuracy and fast learning speed. The authors divided their features into four categories: construction-based, IP-based, TTL-based, and WHOIS-based. Their evaluation resulted in a high detection rate, an accuracy exceeding 95%, and a fast learning speed. However, as shown below, a significant fraction of the features used in this work emerged as ineffective in the presence of an intelligent adversary.

Sun et al. [24] presented a system called *HinDom*, that generate a heterogeneous graph (in contrast to homogeneous graphs created by [23], [26]) in order to robustly identify malicious attacks (e.g. spams, phishing, malware and botnets). Even though HinDom collected DNS and pDNS data, it also has the ability to collect information from various clients inside networks (e.g. CERNET2 and TUNET) and by that the perspective of it is different from the perspective of this study (i.e. client perspective). Nevertheless, HinDom has achieved remarkable results using transductive classifier it managed to achieve high accuracy and F1-score with 99% and 97.5% respectively.

Bilge et al. [14] created a system called *Exposure*, designed to detect malicious domain names. Their system uses passive DNS data collected over some period of time to extract features related to known malicious and benign domains. Passive DNS Replication [11], [14], [21], [23], [25], [27], [28] refers to the reconstruction of DNS zone data by recording and aggregating live DNS queries and responses. Passive DNS data could be collected without requiring the cooperation of zone administrators. The Exposure is designed to detect malware- and spam-related domains. It can also detect malicious fast-flux and DGA-related domains based on their unique features. The system computes the following four sets of features from anonymized DNS records: (a) Time-based features related to the periods and frequencies that a specific domain name was queried in; (b) DNS-answer-based features calculated based on the number of distinctive resolved IP addresses and domain names, the countries that the IP addresses reside in, and the ratio of the resolved IP addresses that can be matched with valid domain names and other services; (c) TTL-based features that are calculated based on statistical analysis of the TTL over a given time series; (d) Domain name-based features are extracted by computing the ratio of the numerical characters to the domain name string, and the ratio of the size of the longest meaningful substring in the domain name. Using a Decision Tree model, Exposure reported a total of 100,261 distinct domains as being malicious, which resolved to 19,742 unique IP addresses. The combination of features that were used to identify malicious domains led to the successful identification of several domains that were related to botnets, flux networks, and DGAs, with low false positive and high detection rates. It may not be possible to generalize the detection rate results reported by the authors (98%) since they were highly dependent on comparisons with biased datasets. Despite the positive results, once an identification scheme is published, it is always possible for an attacker to evade detection by mimicking the behaviors of benign domains.

Another way to detect malicious URLs was introduced in the article "Malicious URL detection via spherical classification" [34]. This article introduces and test a binary classification method aimed at detecting malicious URL on the basis of some information on both the URL syntax and its domain properties.

In the article "Malicious URL Detection By Dynamically Mining Patterns Without Predefined Elements" [35]- instead of using any predefined features or fixed delimiters for feature selection, proposed to dynamically extract lexical patterns from URLs. The novel model of URL patterns provides new flexibility and capability on capturing malicious URLs which are generated by malicious programs. Also, they have developed a new method to mine the novel URL patterns, which are not assembled using any predefined items and thus cannot be mined using any existing frequent pattern mining methods.

In another paper "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods" [36] another solution was proposed: a combination of linear and non-linear space transformation methods. For linear transformation, a two-stage distance metric learning approach was developed: first, singular value decomposition was performed to get an orthogonal space, and then linear programming was used to solve an optimal distance metric. For nonlinear transformation, a Nyström method was introduced for kernel approximation and used the revised distance metric for its radial basis function such that the merits of both linear and non-linear transformations can be utilized.

Some researches used Machine learning approach like "Using Lexical Features for Malicious URL Detection - A Machine Learning Approach" [37]- which recognized the extant need and shortcomings of blacklists-based methods, a machine learning based ensemble classification approach is proposed herein to combat the above threat. The approach uses static lexical features extracted from the URL string, with the underlying assumption that the distribution of these features is different for malicious and benign URLs. And such as "Detecting Malicious Web Links and Identifying Their Attack Types" [38], in this paper, proposed A method using machine learning to detect malicious URLs of all the popular attack types and identify the nature of attack a malicious URL attempts to launch. This method uses a va-

riety of discriminating features including textual properties, link structures, web page contents, DNS information, and network traffic. Many of these features are novel and highly effective.

Rahbarinia et al. [23] presented a system called *Segugio*, an anomaly detection system based on passive DNS traffic to identify malware-controlled domain names based on their relationship to known malicious domains. The system detects malware-controlled domains by creating a machine domain bipartite graph that represents the underlying relations between new domains and known benign/malicious domains. The system operates by calculating the following features: (a) Machine Behavior, based on the ratio of "known malicious" and "unknown" domains that query a given domain d over the total number of machines that query d. The larger the total number of queries and the fraction of malicious related queries, the higher the probability that d is a malware controlled domain; (b) Domain Activity, where given a time period, domain activity is computed by counting the total number of days in which a domain was actively queried; (c) IP Abuse, where given a set of IP addresses that the domain resolves to, this feature represents the fraction of those IP addresses that were previously targeted by known malware controlled domains. Using a Random Forest model, Segugio was shown to produce high true positive and very low false positive rates (94% and 0.1% respectively). It was also able to detect malicious domains earlier than commercial blacklisting websites. However, Segugio is a system that can only detect malware related domains based on their relationship to previously known domains and therefore cannot detect new (unrelated to previous malicious domains) malicious domains. More information about malicious domain filtering and malicious URL detection can be found in [32], [33].

Adversarial Machine Learning is a subfield of Machine Learning in which the training and testing set do not share the same distribution, for example, given perturbations on a malicious instance so that it will be falsely classified. These manipulated instances are commonly called *adversarial examples (AE)* [39]. AE are samples an attacker changes, based on some knowledge of the model classification function. These examples are slightly different from correctly classified examples. Therefore, the model fails to classify them correctly. AE are widely used in the fields of spam filtering [40], network intrusion detection systems (IDS) [41], Anti-Virus signature tests [42] and bio-metric recognition [43].

Attackers commonly follow one of two models to generate adversarial examples: 1) white-box attacker [44], [45], [46], which has full knowledge of the classifier and the train/test data; 2) black-box attacker [44], [47], [48], which has access to the model's output for each given input.

Various methods emerged to tackle AE-based attacks and make ML models robust. The most promising are those based on game-theoretic approaches [49], [50], [51], robust optimization [44], [45], [52], and adversarial retraining [53], [54], [55]. These approaches mainly concern *feature-space models* of attacks where feature space models assume that the attacker changes values of features directly. Note that these attacks may be an abstraction of reality as random modifications to feature values may not be realizable or avoid the manipulated instance functionality.

Big Data is an evolving term that describes any voluminous amount of structured, semi-structured and unstructured data that can be mined for information. Big data is often characterized by 3Vs: the extreme Volume of data, the wide Variety of data types and the Velocity at which the data must be processed. To implement Big Data, high volumes of low-density, unstructured data need to be processed. This can be data of unknown value, such as Twitter data feeds, click streams on a web page or a mobile app, or sensor enabled equipment. For some organizations, this might be tens of terabytes of data. For others, it may be hundreds of petabytes. Velocity is the fast rate at which data are received and (perhaps) acted on. Normally, the highest velocity of data streams directly into memory rather than being written to disk.

Torabi et al. [25] surveyed state of the art systems that utilize passive DNS traffic for the purpose of detecting malicious behaviors on the Internet. They highlighted the main strengths and weaknesses of these systems in an in depth analysis of the detection approach, collected data, and detection outcomes. They showed that almost all systems have implemented supervised machine learning. In addition, while all these systems require several hours or even days before detecting threats, they can achieve enhanced performance by implementing a system prototype that utilizes big data analytic frameworks to detect threats in near real-time. This overview contributed in four ways to the literature. (1) They surveyed implemented systems that used passive DNS analysis to detect DNS abuse/misuse; (2) they performed an in-depth analysis of the systems and highlighted their strengths and limitations; (3) they implemented a system prototype for near real-time threat detection using a big data analytic framework and passive DNS traffic; (4) they presented real-life cases of DNS misuse/abuse to demonstrate the feasibility of a near real time threat detection system prototype. However, the cases that were presented were too specific. In order to understand the real abilities of their system, the system must be analyzed with a much larger test dataset.

As part of this approach in order to have a better analyze of the dataset we decided to look for new features. To understand it better we've read more articles such as: "Feature-based Malicious URL and Attack Type Detection Using Multi-class Classification" [56]. This paper proposes a methodology to detect malicious URLs and the type of attacks based on multi-class classification. In this work, there proposed 42 new features of spam, phishing and malware URLs. These features are not considered in the earlier studies for malicious URLs detection and attack types identification. In the article "What's in a URL: Fast Feature Extraction and Malicious URL Detection" [57] the writers were building a system for URL analysis and classification to primarily detect phishing attacks. URL analysis is attractive to maintain distance between the attacker and the victim, rather than visiting the website and getting features from it. It is also faster than Internet search, retrieving content from the destination web site and network-level features used in previous research. Investigate several facets of URL analysis, e.g., performance analysis on both balanced and unbalanced datasets in a static as well as live experimental setup and online versus batch learning. This paper inspired

us to implement the Shannon Entropy, Special characters ratio and Special character count features. Another study that contributed to our knowledge is "Detecting Malicious Websites by Learning IP Address Features" [58]. The aim of this study is not to provide a single solution that effectively detects web-based malware but to develop a technique that compensates the drawbacks of existing approaches. Validate the effectiveness of this paper's approach by using real IP address data from existing blacklists and real traffic data on a campus network. This paper inspired us to implement the Ip in the domain feature. Another paper - "A Lexical Approach for Classifying Malicious URLs" [59] describes a lightweight approach for classifying malicious web pages using URL lexical analysis alone. Exploring the upper-bound of the classification accuracy of a purely lexical approach. Another aim is to develop an approach which could be used in a real-time system. This paper inspired us to implement the Tokens, Special characters ratio and Special character count features.

4 METHODOLOGY

The following criteria are used: Section 4.1 outlines the characteristics and methods of collection of the dataset. Section 4.2, defines each of the well known features evaluated. Section 4.3 covers the evaluation of their robustness, and Section **??** presents novel features and evaluates their robustness.

4.1 Data Collection

Machine Learning models are trained on data, that in our case includes both malicious and benign URLs. Our benign URLs are taken from the Alexa top 1 million [60], and the malicious URLs were crawled from multiple sources [61], [62], [63] due to the fact they are quite rare.

For each URL, we crawled the URL and domain information properties from Urlscan, Whois, VirusTotal and DNS records. Urlscan is a service that scans and analyzes websites and records the activity that this page navigation creates. This includes the related domains, IPs contacted and the resources (JavaScript, CSS, etc) requested from those domains. For each IP, Urlscan also returns the related ASN and the country. We denote these IPs as "Communication IPs", the IPs associated ASNs as "Communication ASNs" and the countries as "Communication countries". Whois is a widely used Internet record listing that identifies who owns a domain, how to get in contact with them, the domain's creation, update and expiration date of the domain. Whois records have proven to be extremely useful and have developed into an essential resource for maintaining the integrity of the domain name registration and website ownership process. Note that according to a study by ICANN [2] [64], many malicious attackers abuse the Whois system - hence we only used the information that could not be manipulated.

Based on these four resources (Urlscan, Whois, VirusTotal and DNS records), we generated the following features: the length of domain, the number of consecutive characters and the entropy of the domain from the URLs datasets. Next, we calculated the lifetime of A domain and the active time of A domain from the Whois data. Based on the DNS response dataset (total of 334,295 DNS records) we extracted the number of IP addresses, distinct geolocations of the IP addresses, average Time to Live (TTL) value and the Standard deviation of the TTL. We collected passive DNS records and SSL certificate information.

The above data collection framework is presented in Figure 4.1. These resources yielded 1,710 malicious active unique URLs and 5,533 benign active unique URLs (76% benign, 24% malicious).

2. Internet Corporation for Assigned Names and Numbers

4.2 Feature Engineering

Based on previous works surveyed, a set of features which are commonly used for malicious domain classification [10], [11], [14], [23], [27], [28], [33], [65], [66] were extracted. Specifically,the following nine features were used as the baseline:

- **Length of domain**:

$$\text{Length of domain} \quad = \quad length(Domain_{(i)}) \tag{6}$$

Length of domain is the number of characters in the domain name followed by the TLD (Including the dot). For example, for the URL 'https://www.google.com' the length of the domain is 10.

- **Number of consecutive characters**:

$$\text{Number of consecutive characters} = \max\{\text{consecutive repeated characters in } Domain_{(i)}\} \tag{7}$$

The maximum number of consecutive repeated characters in the domain name and the TLD. For example, for the URL 'xxxyyzzzz.com' the number of consecutive characters is 4.

- **Entropy of the domain**:
The calculation of the entropy of a given domain Domain(i) consisting of n_i distinct characters $\{c_1^i, c_2^i, \ldots, c_{n_i}^i\}$. Calculation is according to the formula:

$$\text{Entropy of the domain} = -\sum_{j=1}^{n_i} \frac{count(c_j^i)}{length(Domain_{(i)})} \cdot \log \frac{count(c_j^i)}{length(Domain_{(i)})} \tag{8}$$

For example for the domain "google.com" the entropy is:
$-(5 \cdot (\frac{1}{10} \cdot \log \frac{1}{10}) + 2 \cdot (\frac{2}{10} \cdot \log \frac{2}{10}) + 3(\cdot \frac{3}{10} \cdot \log \frac{3}{10})) = 1.25$
The domain has 5 characters that appear once ('l', 'e', '.', 'c', 'm'), one character that appears twice ('g') and one character that appears three times ('o').

- **Number of IP addresses**:

$$\text{Number of IP addresses} = \|\text{distinct IP addresses}\| \tag{9}$$

The number of distinct IP addresses in the domain's DNS record. For example, for the list ["1.2.3.4", "1.2.3.4","4.4.4.4"] the number of distinct IP addresses is 2.

- **Distinct Geo-locations of the IP addresses**:
The country of each IP address in the domain's DNS record. For example, for the list of IP addresses ["1.2.3.4", "1.2.3.4","4.4.4.4"] the list of countries is ["USA", "USA", "ISRAEL"] and the number of the different countries is 2.

- **Mean TTL value**:
Average the TTL values of all the DNS records of the domain in the DNS dataset. For example, if we conducted 30 checks of some domain's DNS records, and if in 20 of these checks the TTL value was "60" and in 10 checks the TTL value was "1200", the average is $\frac{20 \cdot 60 + 10 \cdot 1200}{30} = 440$.

- **Standard deviation of the TTL**:
he standard deviation of the TTL values of all the DNS records of the domain in the DNS dataset. For example, For the "Mean TTL value" example, the standard deviation of the TTL values is 537.401.

- **Lifetime of domain**:

$$\text{Lifetime of domain} = Date_{Expiration} - Date_{Created} \tag{10}$$

The interval between a domain's expiration date and creation date in years. For example, according to Whois information, for the domain "ariel-cyber.co.il", the creation date is 2015-05-14, and the expiration date is 2020-05-14, and it was updated on 2015- 05-14. Therefore, the lifetime of the domain is 5.

- **Active time of domain**:

$$\text{Active time of domain} = Date_{Updated} - Date_{Created} \tag{11}$$

The interval between a domain's update date and creation date in years. For the same example as in " Lifetime of domain", the active time of the domain "ariel-cyber.co.il" is 0.

- **Shannon Entropy of URL**:

$$\text{H} = -\sum_i p_i \log_b p_i \tag{12}$$

Shannons Entropy is the probability of a character to appear in the URL, and it's calculated by the formula above. Where p_i is the probability of a character to appear in the URL.

- **Ip in the domain**:
This feature checks if the ip is a substring of the domain. If it returns true, it's most likely that the URL is malicious. The name of the domain is supposed to contain a name, if it contains the ip it means that it is not a benign website.

- **Numbers of Distinct characters**:
Counts how many special characters there are in the URL and how many times they appear. The higher the count will be, the most likely that the URL is malicious.

- **Ratio of Distinct characters**:
Calculates the ratio between the amount of special characters in the URL, and the total amount of characters in the URL. The higher the ratio is, it's more likely that the URL is malicious.

- **Secure Connection**:
Checks weather the connection is secure. In other words it checks weather the 'http' contains 's', and creates 'https'. If the connection is secure its probably a benign URL since the malicious ones are not secured and they can't contain 'https'.

- **Tokens**:
It removes the non-alphanumeric characters, and splits them into token words. It counts how many token words the URL contains and checks if those words are suspicious or not.

- **Communication Countries Rank (CCR)**:

$$CCR = \text{CRA}(\text{the communication countries list of } URL_{(i)}) \tag{13}$$

This feature looks at the communication countries with respect to the communication IPs, and uses the countries ratio table to rank a specific URL. The countries ratio table is calculated in the article we've based our research on, and it calculates the ratio of a URL being benign or malicious by using the URL dataset and the Urlscan service (Figures 2, 3) and an algorithm that is detailed later (CRA).

- **Communication ASNs Rank (CAR)**:

$$CAR = \text{CRA}(\text{the communication ASNs list of } URL_{(i)}) \tag{14}$$

Similarly, this feature analyzes the communication ASNs with respect to the communication IPs, and uses the ASNs ratio table to rank a specific URL. the communication ASNs is a list of ASNs, that was extracted using Urlscan, each IP address, and appended the ASNs list. While there is some correlation between the ASNs and the countries, the second feature examines each AS (usually ISPs or large companies) within each country to gain a wider perspective.

- **Number of passive DNS changes**:

$$PDNS = \text{count}(\text{list of passive DNS records of } Domain_{(i)}) \tag{15}$$

When inspecting the passive DNS records, benign domains emerged as having much larger DNS changes that the sensors (of the company that collects the DNS records) could identify, unlike malicious domains (i.e. 26.4 vs. 8.01, as reported in Table **??**). For the "Number of passive DNS changes" the number of DNS records changes were counted, which is somewhat similar to other features described in [11], [25]. Still, these features require much more elaborated information which is not publicly available. On the other hand, this feature can be extracted from passive DNS records obtained from VirusTotal, which are scarce (in terms of record types).

- **Remaining time of SSL certificate**:

$$SSL = Certificate_{Valid} \cdot \cdot (Certificate_{Expiration} - Certificate_{Updated}) \tag{16}$$

When installing an SSL certificate, there is a validation process conducted by a Certificate Authority (CA). Depending on the type of certificate, the CA verifies the organization's identity before issuing the certificate. When analyzing our data it was noted that most of the malicious domains do not use valid SSL certificates and those that do only use one for a short period. Therefore, this feature was engineered which represents the time the SSL certificate remains valid. For the "Remaining time of SSL certificate", in contrast to a binary feature version used by [65], this feature extends the scope and represents both the existence of an SSL certificate and the remaining time until the SSL certificate expires.

The CRA (Algorithm 1) gets a URL as an input and returns its country communication rate or the ASN communication rate (based on the type in the input of the algorithm).

For each item (i.e., country or ASN), first the algorithm initialized the value of the ratio variable to 0.75 (according to [67], 25% of all URLs in 2017 were malicious, suspicious or moderately risky) and the normalized total occurrences (Total_norm) of an item to be 1. Next, in Step 9, if the total

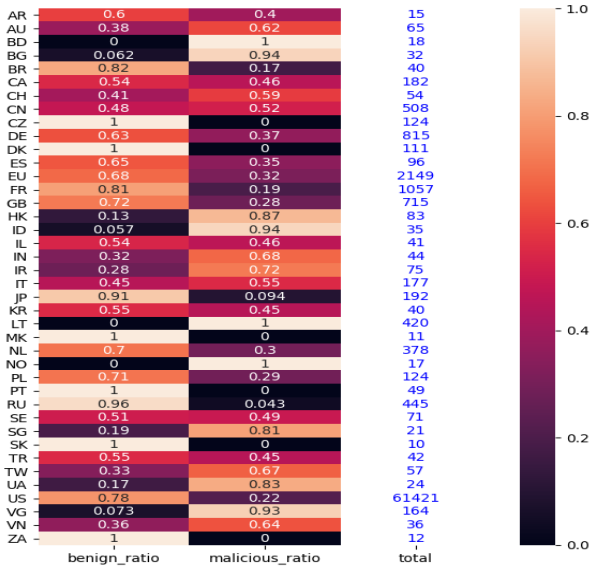


Fig. 2: Communication Countries Ratio

number of occurrences of an item was $\geq$ to the threshold, the algorithm replaced the ratio and normalized occurrences to the correct values according to the ratio tables given in Figures 2 and 3. Finally, the algorithm sums the rank with a log base 0.5 of the ratio (+ some epsilon) and divide this value by the normalized total occurrences.

Algorithm 1 Communication Rank

Input: URL, Threshold, Type
Output: Rank (CCR or CAR)

1: **if** Type = Countries **then**
2: ItemsList = communication countries list of the URL
3: **else**
ItemsList = ASNs list of the URL
4: **end if**
5: $Rank = 0$
6: **for** $Item$ in $ItemsList$ **do**
7: $Ratio = 0.75$ {Init value}
8: $Total\_norm = 1$ {Init value}
9: **if** $TotalOccurrences(Item) >= Threshold$ **then**
10: $Total\_norm = Normalize(Item)$
11: $Ratio = BenignRatio(Item)$
12: **end if**
13: $Rank+ = (\log_{0.5}(Ratio + \epsilon)/Total\_norm)$
14: **end for**=0

4.3 Comparison

As a part of this article we to compare between the results of the article we based our research on and between our results. In the article we relied on, they started with the main idea which was to figure out how robust are the features. In the following analysis, the robustness of these features' is analysed between their results and our results.

The evaluation metric, the prediction percentage, was defined as the average detection rate after modification.

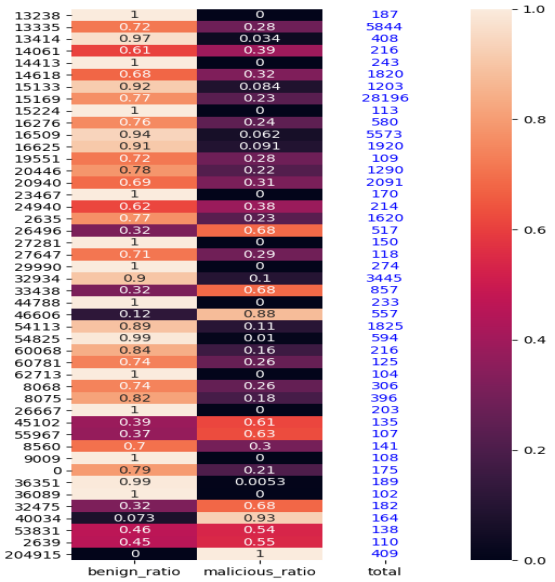


Fig. 3: Communication ASNs Ratio

1) **"Length of domain":** This feature is a number that represents the length of the domain which is followed by the TDL. An adversary can easily purchase a short or long domain to result in a benign classification for a malicious domain; hence this feature has low impact over classification. In the previous study this feature is claimed to be weak and does not help much with detecting malicious URLs from benign URLs. From the results and graphs we've received, we can conclude that the feature is indeed easily manipulated and has low impact over classification as expected.
2) **"Number of consecutive characters":** surprisingly, as depicted in Figure 4, manipulating the "Number of consecutive characters" feature can significantly lower the prediction percentage (e.g. move from three consecutive characters to one or two). On average, there were 1.46 consecutive characters in malicious domains. Therefore, the previous study expects this feature to have a strong impact over malicious and benign classification. But, according to multiple results and tests we have done, this feature had a very low impact on classifying URls.
3) **"Entropy of the domain":** In order to manipulate the "Entropy of the domain" feature as benign domain entropy, the adversary can create a domain name with entropy < 4. For example, the domain "ddcd.cc" which is available for purchase, the entropy for this domain is 3.54. This value falls precisely in the entropy area of the benign domains defined by the trained model. This example breaks the model and causes a malicious domain to look like a benign URL. Hence, this feature is not expected to have a high impact on classification. During the tests we have done, this feature showed consistency in results. But, the results showed low feature values which means exactly what the previous study claimed. This feature is not solid enough for classifying and can be manipulated by attackers.
4) **"Number of IP addresses":** Note that an adversary can add many A records to the DNS zone file of its domain to imitate a benign domain. Thus, to manipulate the number of IP addresses, an intelligent adversary only needs to have several different IP addresses and add them to the zone file. In our study, this feature showed some accuracy spikes and some good results. But, overall this feature was mainly on poor performance. Most likely because of the easy bypass an adversary can do to avoid being classified as malicious by this feature.
5) **"Distinct Geolocations of the IP addresses":** In order to be able to break the model with the "Distinct Geolocations of the IP addresses" feature, the adversary needs to use several IP addresses from different geolocations. If the adversary can determine how many different countries are sufficient to mimic the number of distinct countries of benign domains, he will be able to append this number of IP addresses (a different IP address from each geo-location) to the DNS zone file. Because the manipulation of this feature is not straight forward, not all malicious links were able to "fool" this feature. Thus, this feature has some impact over the classification but not much.
6) **"Mean TTL value"** and **"Standard deviation of the TTL":** There is a clear correlation between the "Mean TTL value" and the "Standard deviation of the TTL" features since the value manipulated by the adversary is the TTL itself. Thus, it makes no difference if the adversary cannot manipulate the "Mean TTL value" feature if the model uses both. In order to robustify the more, it is better to use the "Mean TTL value" feature without the "Standard deviation of the TTL" one. Solely in terms of the "Mean TTL value" feature, Figure 4 shows that manipulation will not result in a false classification since the prediction percentage does not drop dramatically, even when this feature is drastically manipulated. An adversary can set the DNS TTL values to [0,120000] (according to the RFC 2181 [62] the TTL value range is from 0 to 2 311). Figure 4 shows that even manipulating the value of this feature to 60000 will break the model and cause a malicious domain to be wrongly classified as a benign URL. As expected the "Mean TTL value" showed good results, solid and precision percentage. In addition the "Standard deviation of the TTL" had low results and was clearly manipulated.
7) **"Lifetime of domain"**: As for the lifetime of domains, based on [10] we know that a benign domain's lifetime is typically much longer than a malicious domain's lifetime. In order to break the model by manipulating the "Lifetime of domain" feature, the adversary must buy an old domain that is available on the market. Even though it is possible to buy an appropriate domain, it will take time to find one, and it will be expensive. Hence this feature is expected to shows good results. But, it actually

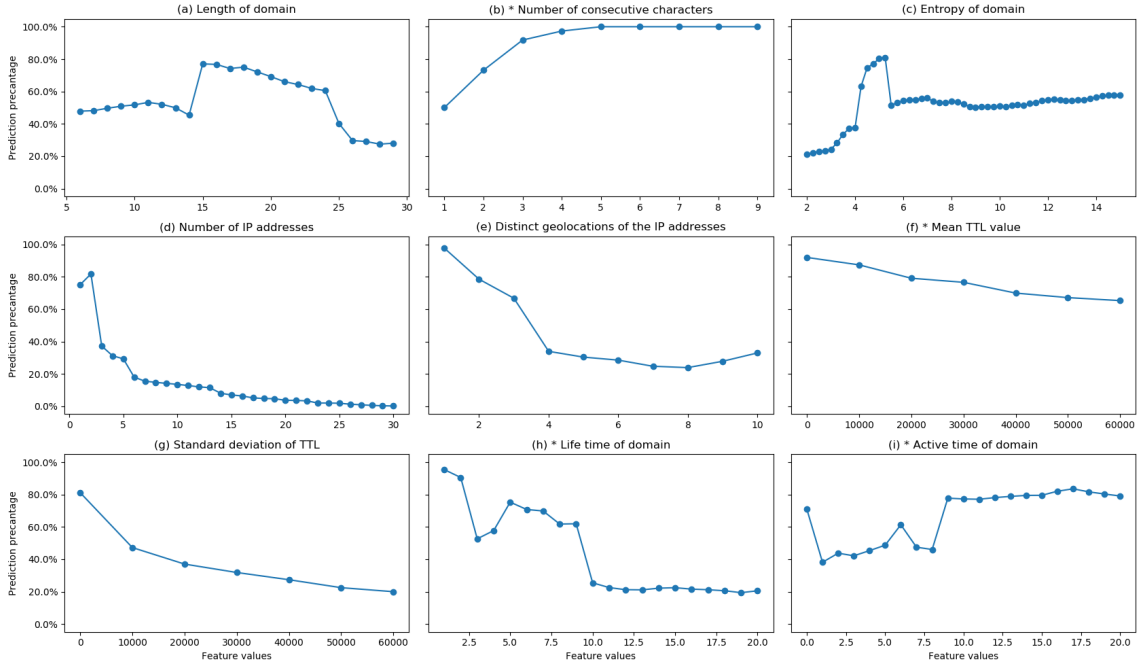


Fig. 4: Base feature manipulation graphs (* robust or semi-robust features)

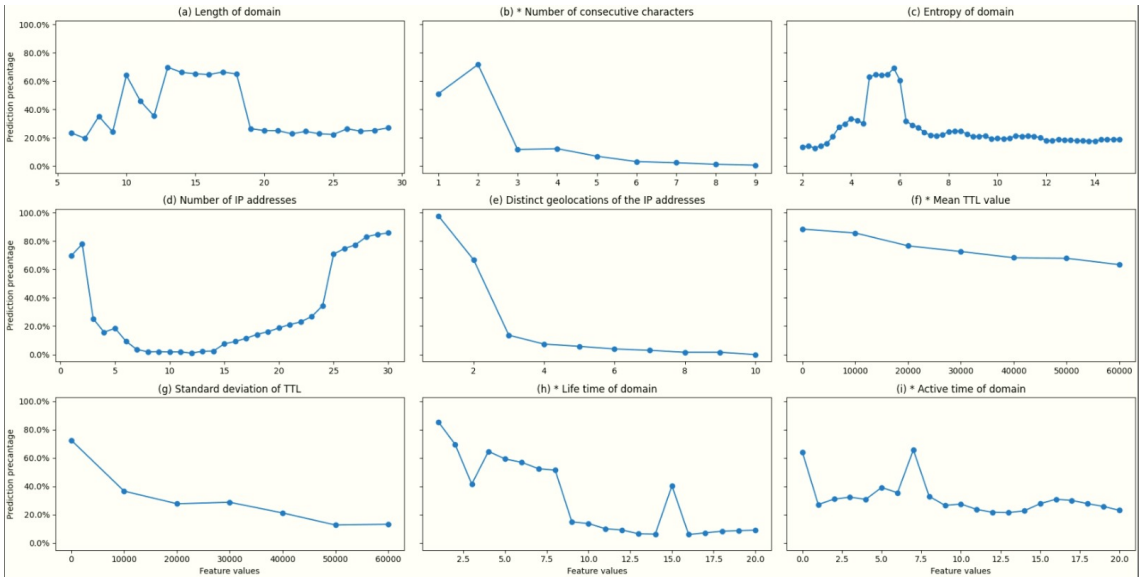


Fig. 5: Base feature manipulation graphs - Our results

didn't. while the average precision percentage was mostly the same throughout the tests, it was relatively low.

8) **"Active time of domain"**: Similar to the previous feature, in order to break "Active time of domain", an adversary must find a domain with a particular active time (Figure 4), which is much more tricky. It is hard, expensive, and possibly unfeasible. Therefore we still expect good performance from this feature, and this time we got both. The average precision was relatively high and it was mostly the same throughout the testings.
9) **"Secure Connection"**: This feature checks for the "Https" protocol inside links. This can be easily manipulated by adversaries by purchasing a Certificate and gaining the secure connection status, though many malicious sites, either do not bother purchasing a certificate or don't use https if they use some sort of eavesdroppers or tampering. Thus, we expected this feature to only have a small impact on identifying malicious links and the results show that out of the 6 features we have added had the least score.
10) **"IP in the Domain"**: This feature checks for any IP addresses that were not converted into names in the domain of a link. Avoiding this feature as an attacker is relatively easy. An adversary would purchase and use a domain that does not use it's IP

as a part of its "name". This feature got the second worst score right after "Secure connection". These two features are easy to manipulate but do help classifying lazy and simple malicious URLs handled by a very simple adversary.

11) **"Tokens"**: Tokens are keywords that malicious URLs (mainly phishing) would probably use. This feature is meant to check for the most common words malicious URLs might contain. As not every malicious URLs would contain such framing evidence of maliciousness, phishing links often do. Phishing URLs most likely want to obtain personal information. Thus, they would use words like "login" "purchase" and so on. This feature can be manipulated by not using such words in a malicious URL but could be hard for Phishing sites to avoid these tokens and make their site look reliable for clients.
12) **"Ratio of Distinct characters' and 'Numbers of Distinct characters"**: These features calculate the percentage and number of distinct characters in a URL. An adversary could manipulate this feature by choosing a short URL with not many distinct characters. Despite that, many malicious sites either use long names or names that were available for purchase (domain) - which means that in both cases the URL is likely to have multiple different characters, more than the usual benign URL. Those two features got the first and third best score out of the features we have added.
13) **"Shannon Entropy of URL"**: Similarly to the Shannon entropy of a domain we also chose to check the entropy of the hole URL. Of course, a smart adversary would be able to manipulate the entropy in his URL by choosing the right domain name, keeping the overall URL short and minimizing the probability of his malicious URL to be detected by Shannon entropy. This feature received the second highest score.

In the article we based our article on, the robust features from were selected, and the non-robust ones were dropped. Using that subset the model was trained and an accuracy of 95.71% with an F1-score of 88.78% was achieved can be improved. Therefore, we extended our analysis and searched for new features like the ones above that would meet the requirements to build a good model with a higher F1-score, ending up with a model that results in an accuracy of 99.73% and F1-score of 99.39%.

5 Empirical Analysis and Evaluation

This section describes the testbed used for the evaluation of models based on the types of features (both robust and not). General settings are provided for each of the models (e.g. the division of the data into training and test set), as are the parameters used to configure each of the models, followed by the efficiency of each model. [3]

3. Our code is publicly available at https://github.com/orenIsabella/malicious-URL

5.1 Experimental Design

Apart from intelligently choosing the model parameters, one should verify that the data used for the learning phase accurately represents the real-world distribution of domain malware. Hence, the dataset was constructed such that 75% were benign domains, and the remaining 25% were malicious domains (~5,000 benign URLs and ~1,700 malicious domains respectively) [67].

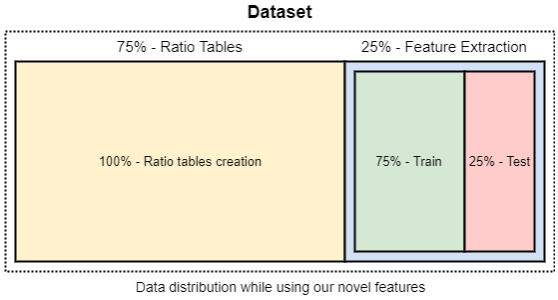
There are many ways to define the efficiency of a model. To account for most of them, a broad set of metrics was extracted including accuracy, recall, F1-score, and training time. Note that for each model, the dataset was split into train and test sets where 75% of the data (both benign and malicious) was randomly assigned to the train test, and the remaining domains are assigned to the test set.

The evaluation step measured the efficiency of the different models while varying the robustness of the features included in the model. Specifically, four different models (i.e. Logistic Regression, SVM, ELM, ANN, Random Forest, K-neighbors and Naive-Bayes) were trained using the following feature sets:

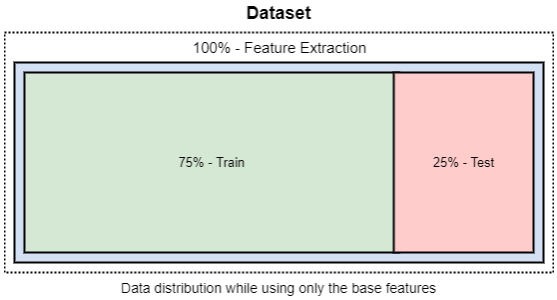
- Base (*B*) - The set of commonly used features in previous works (see Table **??** for more details) plus the 6 features we have added in our research.
- Base Robust (*BR*) - the subset of robust base features (marked with a * in Figure 4).
- "TCP" (*TCP*) - The four novel features: Time of SSL certificate, Communication ranks (CCR and CAR) and PassiveDNS changes (see Table **??**).
- Base Robust + "TCP"(*BRTCP*) - the union of *BR* and *TCP*, the robust subset of all features.
- Base + "TCP" (*BTCP*) - the union of *B* and *TCP*.

Recall that feature sets (i.e. TCP, BRTCP, and BTCP) which are based on the novel features (i.e. CCR and CAR) , require communication ratio tables. Hence, to keep the models unbiased (i.e. omit the data that was used for generating the ratio tables from the learning process), it was necessary allocate some of the data to creation of the ratio tables. Due to the variety of countries and ASNs, it was decided to dedicate the majority of the data to this process. For the evaluation, the dataset was split into two parts: when using the novel features 75% of the data was used to create the ratio tables and the remaining 25% of the data was used to extract the features, 100% of the data was used for the feature extraction as can be seen in Figure 6b. Evaluations for the opposite split ratio (i.e. 25/75) were also conducted, this time allocating most of the data to the learning phase. The findings showed that the trend between these two ratios was similar (as presented in Section 5.2). Both ratios returned high results but, for most of the models, the 75/25 ratio returned a higher F1-Score and higher Recall measures (e.g. LR, SVM, Random Forest, K-neighbors, Naive-Bayes, and ANN for each feature set). For the 25/75 distribution, the F1-Score and the Recall measures were only higher for the ELM (around 1%-2% higher). Therefore, while both results are reported in the end of this section, WLOG, the main focus of the discussion is on the 75/25 distribution. Therefore, while both results are reported in the end of this section, WLOG, the main focus of the discussion is on the 75/25 distribution.

In the process of researching this topic and mostly coming to the same results, we have encountered technical issues, problems running the code (both on Windows, Linux and on multiple IDEs I.e. Pycharm and VS-Code). We had compatibility issues and defective classes, but eventually, we solved all of those problems, we were able to confirm the distribution above, create a DB and test it with the 6 new features we've added.



(a) Data distribution while using our novel features



(b) Data distribution while using only the base features

Fig. 6: Data Distribution

5.2 Models and Parameters

In the thesis there were four classification models used for analysis: Logistic Regression (LR), Support Vector Machines (SVM), Extreme Learning Machine (ELM), and Artificial Neural Networks (ANN), on which we will elaborate further in this article. All the models were trained and evaluated on a Linux Mint 20 Cinnamon OS version 4.6.6 with 3.00GHz Intel Core i7-9700 CPU, 16GB of RAM, and Intel Corporation UHD Graphics 630. In the following paragraphs, for the first four models, the hyper-parameters used for the evaluation are first described, followed by the empirical, experimental results (which sums up several test results using different random train-test sets), and a short discussion of the findings and their implications.In addition we tested the K-Neighbors, Naive-Bayes and Random Forest models, to see if there is even more powerful models than the ones showed in the thesis.

5.2.1 Logistic Regression

The LR classification model was used as a baseline before using the nonlinear models for the evaluation process. The model was trained on the five features using the strength of hyperparameters to maximize the models performance.

Feature set	Accuracy	Recall	F1-Score
Base	89.99%	38.82%	53.21%
Robust Base	88.33%	38.87%	49.42%
TCP	86.20%	8.30%	14.99%
BRTCP	88.82%	52.46%	65.57%
BTCP	92.86%	64.14%	72.48%

TABLE 2: Model performance - Logistic Regression (in %)

Feature set	Accuracy	Recall	F1- Score
Base	89.53%	41.18%	54.86%
Robust Base	88.37%	37.59%	49.98%
TCP	87.93%	35.59%	50.39%
BRTCP	89.17%	51.14%	63.61%
BTCP	88.41%	41.76%	57.15%

TABLE 3: Our model performance - Logistic Regression (in %)

Hyperparameters: Polynomial degree: 1;K-Fold CrossValidation k=10.

Table 2 shows the results of the thesis writers result, and Table 4 shows our result. In general,the different feature sets resulted in similar accuracy rates. However, the accuracy rate measures how well the model predicts (i.e. TP+TN) with respect to all the predictions (i.e. TP+TN+FP+FN). Thus given the unbalanced dataset (75% of the dataset are benign and 25% are malicious domains), 90% accuracy is not necessarily a sufficient result in terms of malware detection. In comparison with the thesis results, it is showing that they are very close to match. On the one hand the accuracy is practically the same, which is a good sign, but on the other hand there are large differences in the recall and F1-score. However, the difference in the results only reinforces that the high accuracy is not high as we would like it to be. The reason for that is because the Recall (which represents the ratio of malicious instances detected) is not high at all. Next, it was decided to use the SVM model with an RBF kernel as a nonlinear model.

5.2.2 Support Vector Machine (SVM)

Hyperparameters: Polynomial degree: 3; K-Fold CrossValidation k=10; =0.00035; Kernel: RBF [64] [68].

Compared to the results of the LR model (Table 2), the results of the SVM model (Table 4) show a significant improvement in the recall and F1-score measures; e.g. for *Base*, the recall and the F1-score measures were both above 90% in the thesis runs and the ours. An advantage of including the novel features is the fact that models converge much faster. The results are based on analyzing a non-manipulated dataset.The Base feature set includes some non-robust features. Hence, an intelligent adversary can manipulate the values of these features, resulting in a wrong classification of malicious instances. However, an intelligent adversary should invest much more effort for a model that was trained using the Robust Base or TCP features, since each of them was specifically chosen to avoid such manipulations. In our results it is shown that the recall collapsed for almost all of the features and showed worse results than in the thesis. All the other results are in the 20% range to those in the thesis. While it is not completely clear to us why the results have a massive gap, we tried modifying the hyper parameters, changing datasets and averaging 10 runs.

Feature set	Accuracy	Recall	F1-Score
Base	96.49%	91.20%	91.36%
Robust Base	90.14%	56.51%	69.93%
TCP	83.10%	60.21%	54.21%
BRTCP	96.78%	91.37%	92.02%
BTCP	97.95%	90.73%	92.83%

TABLE 4: Model performance - SVM (in %)

Feature set	Accuracy	Recall	F1- Score
Base	92.60%	57.80%	70.72%
Robust Base	89.70%	35.96%	51.91%
TCP	87.76%	36.99%	50.99%
BRTCP	94.99%	79.14%	85.40%
BTCP	90.69%	54.31%	57.15%

TABLE 5: Our model performance - SVM (in %)

yet, the score remained low. To find efficient models on non-manipulated dataset, we also examined in the analysis the ELM model as now will be presented, and the ANN model after that. Our results can be found in Table 3

5.2.3 ELM

Hyperparameters: One input layer, 100 hidden layers, and one output layer. Activation function: first layer - ReLU [69]; hidden layer - Sigmoid. K-Fold Cross-Validation k=10 [10]. Overall, the ELM model resulted in high accuracy and higher Recall rates compared to Table 2, for any feature set, and on both test runs, the one from the thesis and the one we tested. The only feature with significant difference is Robust Base which collapsed on our test run. When compared to the SVM models, the Base model resulted in lower recall (though a higher F1-score was achieved with the ELM model). On the other hand, the Robust Base resulted in a higher recall in the ELM model compared to the SVM one. Even though the Robust Base feature set had a low dimensional space, the three rates (i.e. Accuracy, Recall, and F1-score) were higher than those of the Base feature set. Moving to the sets that include the novel features increased these metrics, while improving the robustness of the model at the same time. When compared to the SVM models, the *Base* model resulted in lower recall (though a higher F1-score was achieved with the ELM model). On the other hand, the *Robust Base* resulted in a higher recall in the ELM model compared to the SVM one. Even though the *Robust Base* feature set had a low dimensional space, the three rates (i.e. Accuracy, Recall, and F1-score) were higher than those of the *Base* feature set. Moving to the sets that include the novel features increased these metrics, while improving the robustness of the model at the same time. Our results can be found in Table 6

Feature set	Accuracy	Recall	F1- Score
Base	99.70%	98.22%	99.00%
Robust Base	91.38%	47.98%	55.09%
TCP	99.50%	97.60%	98.52%
BRTCP	98.78%	94.89%	97.02%
BTCP	99.34%	97.78%	98.66%

TABLE 6: Our model performance - ELM (in %)

Feature set	Accuracy	Recall	F1-Score
Base	98.17%	88.81%	92.92%
Robust Base	98.83%	92.24%	95.81%
TCP	98.88%	94.64%	96.84%
BRTCP	98.86%	95.82%	97.07%
BTCP	98.19%	93.09%	95.34%

TABLE 7: Model performance - ELM (in %)

Feature set	Accuracy	Recall	F1-Score
Base	97.20%	88.03%	90.23%
Robust Base	95.71%	83.63%	88.78%
TCP	98.03%	96.83%	95.24%
BRTCP	99.36%	98.77%	98.42%
BTCP	99.82%	99.47%	99.56%

TABLE 8: Model performance - ANN (in %)

5.2.4 ANN

Hyperparameters: One input layer, three hidden layers, and one output layer. Activation function: first layer - ReLU; first hidden layer - RELU; Second hidden layer - LeakyReLU; Third hidden layer - Sigmoid. Batch size -150, learning rate of 0.01; Solver: Adam [70] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. K-Fold Cross-Validation k=10. Similarly to the ELM results, the ANN results show high performance on all feature sets, and on both test runs. For the "basic" feature sets (i.e. Base and Robust Base) the ELM models resulted in higher recall and F1-score. Still, the main focus was in the BTCP feature set and more specifically in the BRTCP variant, and for those feature sets, the ANN models resulted in higher recall and F1-score.Comparing the results on the thesis and the ones we tested, it shows that there is a 5% difference between the results' which is in the normal and expected range for averaging results. Overall this model appeared to be the most effective out of all the other models. Our results can be found in Table 9

5.2.5 K-Neighbors

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors As shown in the table' the results are not great. They are somewhat similar to the results we got on the LR model. And like argued before, because of the low recall outputs, the accuracy is not reliable at all. Our results can be found in Table 10

5.2.6 Naive- Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The classifier is based on the Bayes theorem.The assumption made here is that the predictors/features are independent. That is, the

Feature set	Accuracy	Recall	F1- Score
Base	97.85%	85.69%	89.47%
Robust Base	94.32%	80.56%	87.46%
TCP	99.27%	94.23%	96.17%
BRTCP	97.99%	97.55%	94.67%
BTCP	99.73%	99.01%	99.39%

TABLE 9: Our model performance - ANN (in %)

Feature set	Accuracy	Recall	F1- Score
Base	61.65%	57.39%	51.72%
Robust Base	61.12%	69.87%	62.71%
TCP	82.71%	66.24%	59.45%
BRTCP	83.61%	63.00%	56.88%
BTCP	83.90%	65.71%	58.89%

TABLE 10: Model performance - K-Neighbors (in %)

Feature set	Accuracy	Recall	F1- Score
Base	69.29%	55.72%	54.21%
Robust Base	65.94%	45.58%	43.67%
TCP	90.20%	60.72%	58.75%
BRTCP	91.62%	61.93%	60.32%
BTCP	91.15%	62.93%	61.00%

TABLE 11: Model performance - Naive- Bayes (in %)

presence of one particular feature does not affect the other. Hence it is called naive. The results argue the same as before. The only somewhat good results are on the BRTCP and BTCP features, as shown in Table 11. This can be helpful to the statement that the model tests the independence of a feature.

5.2.7 Random Forest

This model consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. This model's results are almost the same as the Naive-Bayes model's results and all arguments are similar.The results are in Table 12.

5.2.8 Discussion

We conclude the analysis with tables (13, 14 and 15 that summarize the evaluation of the previous research in comparison to tables 16 and 17 that summarize the evaluation of the research we conducted) which depicts all the evaluation matrices (i.e. accuracy, precision, recall, F1-score and loss). The results of both the 25-75 and the 75-25 distribution were slightly better after adding our non robust features, which means that even though most malicious links would be able to manipulate these features, they still help detecting the part that can't manipulate them. The features we added had more effect when we gave 25% of the data to the ratio tables, as expected, because it had more links to train and test. When 75% of the data was passed to the ratio tables, the base features had slim to no impact on the results. Compared with the results provided from the thesis, one can say that the more features there are to examine the URL, the better the chances for better results. Furthermore, even if we provide the ratio tables with 75% of the data, we will still get slightly better results if we have more features.

Feature set	Accuracy	Recall	F1- Score
Base	69.31%	55.36%	53.81%
Robust Base	65.94%	45.58%	43.61%
TCP	90.20%	60.57%	58.61%
BRTCP	91.60%	61.98%	60.22%
BTCP	91.57%	63.76%	61.61%

TABLE 12: Model performance - Random Forest (in %)

Model \ Feature set	Base	Base Robust
Logistic Regression	Accuracy: 0.90 Precision: 0.84 Recall: 0.39 F1-score: 0.53 Loss: 3.45 AUC: 0.85	Accuracy: 0.88 Precision: 0.68 Recall: 0.39 F1-score: 0.49 Loss: 4.03 AUC: 0.81
SVM	Accuracy: 0.96 Precision: 0.91 Recall: 0.91 F1-score: 0.91 Loss: 1.20 AUC: 0.96	Accuracy: 0.90 Precision: 0.91 Recall: 0.56 F1-score: 0.69 Loss: 3.40 AUC: 0.92
ELM	Accuracy: 0.98 Precision: 0.98 Recall: 0.88 F1-score: 0.92 Loss: 0.63 AUC: 0.99	Accuracy: 0.98 Precision: 0.99 Recall: 0.92 F1-score: 0.95 Loss: 0.40 AUC: 0.99
ANN	Accuracy: 0.97 Precision: 0.92 Recall: 0.88 F1-score: 0.90 Loss: 0.44 AUC: 0.98	Accuracy: 0.95 Precision: 0.94 Recall: 0.83 F1-score: 0.88 Loss: 0.68 AUC: 0.97

TABLE 13: Raw data results for the models that trained with the base features (100% of the dataset records)

Model/Feature set	TCP	Base Robust + TCP	Base + TCP
Logistic Regression	Accuracy: 0.86 Precision: 0.77 Recall: 0.08 F1-score: 0.15 Loss: 4.76 AUC: 0.79	Accuracy: 0.88 Precision: 0.87 Recall: 0.52 F1-score: 0.65 Loss: 3.86 AUC: 0.93	Accuracy: 0.92 Precision: 0.83 Recall: 0.64 F1-score: 0.72 Loss: 2.46 AUC: 0.94
SVM	Accuracy: 0.83 Precision: 0.60 Recall: 0.49 F1-score: 0.54 Loss: 5.83 AUC: 0.80	Accuracy: 0.96 Precision: 0.92 Recall: 0.91 F1-score: 0.92 Loss: 1.11 AUC: 0.98	Accuracy: 0.97 Precision: 0.95 Recall: 0.90 F1-score: 0.92 Loss: 0.70 AUC: 0.98
ELM	Accuracy: 0.98 Precision: 0.99 Recall: 0.94 F1-score: 0.96 Loss: 0.38 AUC: 0.99	Accuracy: 0.98 Precision: 0.98 Recall: 0.95 F1-score: 0.97 Loss: 0.39 AUC: 0.99	Accuracy: 0.98 Precision: 0.97 Recall: 0.93 F1-score: 0.95 Loss: 0.62 AUC: 0.98
ANN	Accuracy: 0.98 Precision: 0.93 Recall: 0.96 F1-score: 0.95 Loss: 0.31 AUC: 0.99	Accuracy: 0.99 Precision: 0.98 Recall: 0.98 F1-score: 0.98 Loss: 0.10 AUC: 0.99	Accuracy: 0.99 Precision: 0.99 Recall: 0.99 F1-score: 0.99 Loss: 0.02 AUC: 0.99

TABLE 14: Raw data results for the models that trained on the 75/25 dataset distribution, 42,578/10,979 URLs (ratio tables/feature extraction)

Model/Feature set	TCP	Base Robust + TCP	Base + TCP
Logistic Regression	Accuracy: 0.82 Precision: 0.80 Recall: 0.07 F1-score: 0.13 Loss: 5.94 AUC: 0.70	Accuracy: 0.87 Precision: 0.85 Recall: 0.39 F1-score: 0.53 Loss: 4.24 AUC: 0.89	Accuracy: 0.90 Precision: 0.82 Recall: 0.57 F1-score: 0.68 Loss: 3.42 AUC: 0.93
SVM	Accuracy: 0.82 Precision: 0.89 Recall: 0.07 F1-score: 0.13 Loss: 5.88 AUC: 0.76	Accuracy: 0.91 Precision: 0.81 Recall: 0.65 F1-score: 0.72 Loss: 3.09 AUC: 0.94	Accuracy: 0.96 Precision: 0.91 Recall: 0.87 F1-score: 0.89 Loss: 1.33 AUC: 0.97
ELM	Accuracy: 0.99 Precision: 0.99 Recall: 0.96 F1-score: 0.97 Loss: 0.25 AUC: 0.99	Accuracy: 0.99 Precision: 0.99 Recall: 0.98 F1-score: 0.98 Loss: 0.16 AUC: 0.99	Accuracy: 0.98 Precision: 0.97 Recall: 0.93 F1-score: 0.95 Loss: 0.54 AUC: 0.99
ANN	Accuracy: 0.94 Precision: 0.99 Recall: 0.72 F1-score: 0.83 Loss: 0.81 AUC: 0.97	Accuracy: 0.98 Precision: 0.99 Recall: 0.90 F1-score: 0.94 Loss: 0.31 AUC: 0.99	Accuracy: 0.98 Precision: 0.97 Recall: 0.95 F1-score: 0.96 Loss: 0.18 AUC: 0.99

TABLE 15: Raw data results for the models that trained on the 25/75 dataset distribution 10,979/42,578 URLs (ratio tables/feature extraction)

Model \ Feature set	TCP	Base Robust + TCP	Base + TCP
Logistic regression	Accuracy: 0.87 Precision: 0.86 Recall: 0.35 F1 score: 0.50 Loss: 4.16	Accuracy: 0.89 Precision: 0.84 Recall: 0.51 F1 score: 0.63 Loss: 3.73	Accuracy: 0.88 Precision: 0.90 Recall: 0.41 F1 score: 0.57 Loss: 4.00
SVM	Accuracy: 0.11 f1 Score: 0.41 precision: 0.34 recall: 0.63 Loss: 2.5	Accuracy: 0.41 f1 Score: 0.61 precision: 0.54 recall: 0.80 Loss: 1.66	Accuracy: 0.53 f1 Score: 0.69 precision: 0.63 recall: 0.8 Loss: 1.7
ELM	Accuracy: 0.91 Precision: 0.83 Recall: 0.47 F1 score: 0.55 Loss: 2.97	Accuracy: 0.99 Precision: 0.99 Recall: 0.97 F1 score: 0.98 Loss: 0.22	Accuracy: 0.98 Precision: 0.99 Recall: 0.94 F1 score: 0.97 Loss: 0.41
ANN	Accuracy: 0.92 Precision: 0.94 Recall: 0.68 F1-score: 0.74 Loss: 0.92	Accuracy: 0.95 Precision: 0.98 Recall: 0.88 F1-score: 0.95 Loss: 0.42	Accuracy: 0.99 Precision: 0.99 Recall: 0.93 F1-score: 0.97 Loss: 0.16
K-NEIGHBORS	Accuracy: 0.82 Precision: 0.56 Recall: 0.66 F1-score: 0.59 Loss: 1.37	Accuracy: 0.83 Precision: 0.54 Recall: 0.63 F1-score: 0.57 Loss: 1.43	Accuracy: 0.84 Precision: 0.56 Recall: 0.65 F1-score: 0.59 Loss: 1.34
NAIVE-BAYES	Accuracy: 0.9 Precision: 0.58 Recall: 0.61 F1-score: 0.59 Loss: 1.32	Accuracy:0.91 Precision: 0.59 Recall: 0.61 F1-score: 0.6 Loss: 1.29	Accuracy: 0.91 Precision: 0.6 Recall: 0.63 F1-score: 0.61 Loss: 1.25
Random Forest	Accuracy: 0.9 Precision: 0.57 Recall: 0.6 F1-score: 0.58 Loss: 1.35	Accuracy: 0.91 Precision: 0.59 Recall: 0.61 F1-score: 0.60 Loss: 1.29	Accuracy:0.91 Precision: 0.6 Recall: 0.63 F1-score: 0.61 Loss: 1.25

TABLE 16: Our raw data results for the models that trained on the 75/25 dataset distribution (ratiotables / feature extraction)

6 Conclusion

Despite existing defenses, malicious URLs remain a severe threat for the internet users. To protect end users from visiting these sites, we can try to identify suspicious URLs by analyzing their features, but many of those attempts fail to successfully classify malware in realistic environments. To prevail in this contest, we need algorithms that continually adapt to new examples and features. In this article, we experimented with different approaches for detecting malicious URLs with an eye toward ultimately deploying a real-time system. We looked for features that would help us find the best way to make sure whether a URL is malicious or begging. The features effectiveness was evaluated based on well known machine and deep learning models (i. e. LR,SVM,Random Forest,K-neighbors,Naive-Bayes,and ANN) over a sizable realistic dataset that we collected from different sources. A particular challenge in this domain is that URL classifiers must operate in a dynamic landscape; one in which criminals are constantly evolving new strategies to counter our defenses. To prevail in this contest, we need algorithms that continually adapt to new examples and features. In this article, we experimented with different approaches for detecting malicious URLs with an eye toward ultimately deploying a real-time system. Experiments on a live feed of labeled examples revealed the limitations of batch algorithms in this domain. Most fundamentally, their accuracy appears to be limited by the number of training examples that can fit into memory. After observing this limitation in practice, we investigated the problem of URL classification in an online setting. We found that the best-performing features were Ratio of distinct characters, Numbers of distinct character and the feature Shannon Entropy on the URL, while the most inefficient features were Ip in the domain and Security. Furthermore we noticed that the best results were conducted when we used ANN (F1 Score: 99.39% and Accurancy: 99.73%). Further research is needed to find new features and to create models that can not only decide wether a URL is malicious but also classify malicious URLs into malicious attack types. [58]

References

[1] N. Hason, A. Dvir, and C. Hajaj, "Robust malicious url detection," in *the Fourth International Symposium on Cyber Security Cryptology and Machine Learning*, 2020.

[2] X. Shu, K. Tian, A. Ciambrone, and D. Yao, "Breaking the target: An analysis of target data breach and lessons learned," *arXiv preprint:1701.04940*, 2017.

[3] A. Blum, B. Wardman, T. Solorio, and G. Warner, "Lexical feature based phishing url detection using online learning," in *Workshop on Artificial Intelligence and Security*, 2010, pp. 54–60.

[4] M. Khonji, Y. Iraqi, and A. Jones, "Phishing detection: a literature survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2091–2121, 2013.

[5] A. Le, A. Markopoulou, and M. Faloutsos, "Phishdef: Url names say it all," in *INFOCOM*, 2011, pp. 191–195.

Model \ Feature set	TCP	Base Robust + TCP	Base + TCP
Logistic regression	Accuracy: 0.87 Precision: 0.86 Recall: 0.35 F1 score: 0.50 Loss: 4.16	Accuracy: 0.89 Precision: 0.84 Recall: 0.5 F1 score: 0.63 Loss: 3.73	Accuracy: 0.88 Precision: 0.90 Recall: 0.41 F1 score: 0.56 Loss: 4.01
SVM	Accuracy: 0.87 Precision: 0.82 Recall: 0.36 F1 score: 0.5 Loss: 4.22	Accuracy: 0.9 Precision: 0.91 Recall: 0.52 F1 score: 0.66 Loss: 3.32	Accuracy: 0.95 Precision: 0.93 Recall: 0.8 F1 score: 0.86 Loss: 1.63
ELM	Accuracy: 0.99 Precision: 0.98 Recall: 0.96 F1 score: 0.97 Loss: 0.28	Accuracy: 0.97 Precision: 0.96 Recall: 0.87 F1 score: 0.91 Loss: 0.97	Accuracy: 0.96 Precision: 0.92 Recall: 0.95 F1 score: 0.93 Loss: 1.17
ANN	Accuracy: 0.96 Precision: 0.91 Recall: 0.9 F1-score: 0.91 Loss: 0.15	Accuracy: 0.97 Precision: 0.95 Recall: 0.96 F1-score: 0.99 Loss: 0.14	Accuracy: 0.99 Precision: 0.98 Recall: 0.99 F1-score: 0.97 Loss: 0.07
K-NEIGHBORS	Accuracy: 0.83 Precision: 0.57 Recall: 0.66 F1-score: 0.59 Loss: 1.35	Accuracy: 0.84 Precision: 0.54 Recall: 0.63 F1-score: 0.57 Loss: 1.42	Accuracy: 0.84 Precision: 0.56 Recall: 0.66 F1-score: 0.58 Loss: 2.76
NAIVE-BAYES	Accuracy: 0.12 Precision: 0.07 Recall: 0.08 F1-score: 0.07 Loss: 3.61	Accuracy: 0.34 Precision: 0.17 Recall: 0.2 F1-score: 0.18 Loss: 3.22	Accuracy: 0.28 Precision: 0.13 Recall: 0.16 F1-score: 0.14 Loss: 3.3
Random Forest	Accuracy: 0.9 Precision: 0.57 Recall: 0.61 F1-score: 0.59 Loss: 1.48	Accuracy: 0.91 Precision: 0.59 Recall: 0.62 F1-score: 0.6 Loss: 1.31	Accuracy: 0.91 Precision: 0.6 Recall: 0.63 F1-score: 0.61 Loss: 1.41

TABLE 17: Our raw data results for the models that trained on the 25/75 dataset distribution (ratiotables / feature extraction)

[6] P. Prakash, M. Kumar, R. R. Kompella, and M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks," in *INFOCOM*, 2010, pp. 1–5.

[7] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, "An empirical analysis of phishing blacklists," in *Conference on Email and Anti-Spam*, 2009.

[8] N. Sandell, P. Varaiya, M. Athans, and M. Safonov, "Survey of decentralized control methods for large scale systems," *IEEE Transactions on Automatic Control*, vol. 23, no. 2, pp. 108–128, 1978.

[9] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *International Conference on World Wide Web*, 2011, pp. 197–206.

[10] Y. Shi, G. Chen, and J. Li, "Malicious domain name detection based on extreme machine learning," *Neural Processing Letters*, pp. 1–11, 2017.

[11] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for dns." in *USENIX*, 2010, pp. 273–290.

[12] M. Ahmed, A. Khan, O. Saleem, and M. Haris, "A fault tolerant approach for malicious url filtering," in *International Symposium on Networks, Computers and Communications*, 2018, pp. 1–6.

[13] H. Berger, A. Z. Dvir, and M. Geva, "A wrinkle in time: A case study in DNS poisoning," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1906.10928

[14] L. Bilge, S. Sen, D. Balzarotti, E. Kirda, and C. Kruegel, "Exposure: A passive dns analysis service to detect and report malicious domains," *Trans. Inf. Syst. Secur.*, vol. 16, no. 4, pp. 1–28, 2014.

[15] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton, "Real-time detection of fast flux service networks," in *Conference For Homeland Security, Cybersecurity Applications & Technology*, 2009, pp. 285–292.

[16] H. Choi, B. B. Zhu, and H. Lee, "Detecting malicious web links and identifying their attack types." *WebApps*, vol. 11, no. 11, p. 218, 2011.

[17] L. Dolberg, J. François, and T. Engel, "Efficient multidimensional aggregation for large scale monitoring." in *LISA*, 2012, pp. 163–180.

[18] N. Harel, A. Dvir, R. Dubin, R. Barkan, R. Shalala, and O. Hadar, "Misal-a minimal quality representation switch logic for adaptive streaming," *Multimedia Tools and Applications*, 2019.

[19] Z. Hu, R. Chiong, I. Pranata, W. Susilo, and Y. Bao, "Identifying malicious web domains using machine learning techniques with online credibility and performance data," in *Congress on Evolutionary Computation (CEC)*, 2016, pp. 5186–5194.

[20] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[21] T. Nelms, R. Perdisci, and M. Ahamad, "Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates." in *USENIX*, 2013, pp. 589–604.

[22] T. Peng, I. Harris, and Y. Sawa, "Detecting phishing attacks using natural language processing and machine learning," in *International Conference on Semantic Computing*. IEEE, 2018, pp. 300–301.

[23] B. Rahbarinia, R. Perdisci, and M. Antonakakis, "Efficient and accurate behavior-based tracking of malware-control domains in large isp networks," *ACM Transactions on Privacy and Security*, vol. 19, no. 2, p. 4, 2016.

[24] X. Sun, M. Tong, J. Yang, L. Xinran, and L. Heng, "Hindom: A robust malicious domain detection system based on heterogeneous information network with transductive classification," in *International Symposium on Research in Attacks, Intrusions and Defenses*, 2019, pp. 399–412.

[25] S. Torabi, A. Boukhtouta, C. Assi, and M. Debbabi, "Detecting Internet Abuse by Analyzing Passive DNS Traffic: A Survey of Implemented Systems," *Communications Surveys & Tutorials*, 2018.

[26] S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *Transactions on Networking*, vol. 20, pp. 1663–1677, 2012.

[27] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon, "Detecting malware domains at the upper dns hierarchy." in *USENIX*, vol. 11, 2011, pp. 1–16.

[28] R. Perdisci, I. Corona, and G. Giacinto, "Early detection of malicious flux networks via large-scale passive dns traffic analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 5, pp. 714–726, 2012.

[29] J. Jung and E. Sit, "An empirical study of spam traffic and the use of dns black lists," in *SIGCOMM Conference on Internet Measurement*, 2004, pp. 370–375.

[30] I. Mishsky, N. Gal-Oz, and E. Gudes, "A topology based flow model for computing domain reputation," in *IFIP Annual Conference on Data and Applications Security and Privacy*, 2015, pp. 277–292.

[31] H. Othman, E. Gudes, and N. Gal-Oz, "Advanced flow models for computing the reputation of internet domains," in *IFIP International Conference on Trust Management*, 2017, pp. 119–134.

[32] A. Das, G. Data, A. Platform, E. Jain, and S. Dey, "Machine learning features for malicious url filtering–the survey," 2019.

[33] D. Sahoo, C. Liu, and S. C. Hoi, "Malicious url detection using machine learning: A survey," *arXiv preprint:1701.07179*, 2017.

[34] M. G. A. P. A. Astorino, A. Chiarello1, "Malicious url detection via spherical classification," *The Natural Computing Applications Forum 2016*, 2016.

[35] D. Huang, "Malicious url detection by dynamically mining patterns without pre-defined elements," *SIMON FRASER UNIVERSITY Spring 2012*, 2012.

[36] Y. P. Tie Li, Gang Kou, "Improving malicious urls detection via feature engineering: Linear and nonlinear space transformation methods," *Information Systems 91 (2020) 101494*, 2020.

[37] P. W. Apoorva Joshi, Levi Lloyd and S. Seethapathy, "Using lexical features for malicious url detection - a machine learning approach," *arXiv:1910.06277 [cs.CR]*, 2019.

[38] H. L. Hyunsang Choi, Bin B. Zhu, "Detecting malicious web links and identifying their attack types," *WebApps'11: Proceedings of the 2nd USENIX conference on Web application development*, 2011.

[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint:1412.6572*, 2014.

[40] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. A. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, pp. 1–9, 2008.

[41] P. Fogla, M. I. Sharif, R. Perdisci, O. M. Kolesnikov, and W. Lee, "Polymorphic blending attacks." in *USENIX*, 2006, pp. 241–256.

[42] J. Newsome, B. Karp, and D. Song, "Paragraph: Thwarting signature learning by training maliciously," in *International Workshop on Recent Advances in Intrusion Detection*, 2006, pp. 81–105.

[43] R. N. Rodrigues, L. L. Ling, and V. Govindaraju, "Robustness of multimodal biometric fusion methods against spoof attacks," *Journal of Visual Languages & Computing*, vol. 20, no. 3, pp. 169–179, 2009.

[44] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint:1706.06083*, 2017.

[45] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," *arXiv preprint:1801.09344*, 2018.

[46] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," *arXiv preprint:1710.10766*, 2017.

[47] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Asia Conference on Computer and Communications Security*, 2017, pp. 506–519.

[48] M. Shahpasand, L. Hamey, D. Vatsalan, and M. Xue, "Adversarial attacks on mobile malware detection," in *International Workshop on Artificial Intelligence for Mobile*, 2019, pp. 17–20.

[49] M. Brückner and T. Scheffer, "Stackelberg games for adversarial prediction problems," in *International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 547–555.

[50] A. Singh and A. Lakhotia, "Game-theoretic design of an information exchange model for detecting packed malware," in *International Conference on Malicious and Unwanted Software*, 2011, pp. 1–7.

[51] M. Zolotukhin and T. Hämäläinen, "Support vector machine integrated with game-theoretic approach and genetic algorithm for the detection and classification of malware," in *Globecom Workshops*, 2013, pp. 211–216.

[52] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *Journal of Machine Learning Research*, vol. 10, pp. 1485–1510, 2009.

[53] B. Li and Y. Vorobeychik, "Evasion-robust classification on binary domains," *Transactions on Knowledge Discovery from Data*, vol. 12, no. 4, p. 50, 2018.

[54] N. Nissim, R. Moskovitch, O. BarAd, L. Rokach, and Y. Elovici, "Aldroid: efficient update of android anti-virus software using designated active learning methods," *Knowledge and Information Systems*, vol. 49, no. 3, pp. 795–833, 2016.

[55] "improving robustness of ml classifiers against realizable evasion attacks using conserved features."

[56] D. R. Patil and J. B. Patil, "Feature-based malicious url and attack type detection using multi-class classification," *July 2018, Volume 10, Number 2 (pp. 141162) http://www.isecure-journal.org, year=2018*.

[57] A. D. Rakesh Verma, "What's in a url: Fast feature extraction and malicious url detection," *http://dx.doi.org/10.1145/3041008.3041016*, 2017.

[58] T. M. Daiki Chiba, Kazuhiro Tobe and S. Goto, "Detecting malicious websites by learning ip address features," *arXiv preprint arXiv:1412.6980*, 2012.

[59] M. Darling, "A lexical approach for classifying malicious urls," *https://digitalrepository.unm.edu/ece$_e$tds*/63, 2015.

[60] "Alexa." [Online]. Available: https://www.alexa.com

[61] "Phishtank." [Online]. Available: https://www.phishtank.com

[62] "Url abuse." [Online]. Available: https://urlhaus.abuse.ch

[63] "Scumware." [Online]. Available: https://www.scumware.org

[64] "A study of whois privacy and proxy service abuse." [Online]. Available: https://gnso.icann.org/sites/default/files/filefield_41831/pp-abuse-study-20sep13-en.pdf

[65] D. Ranganayakulu and C. Chellappan, "Detecting malicious urls in e-mail–an implementation," *AASRI*, vol. 4, pp. 125–131, 2013.

[66] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *Transactions on Information and System Security*, vol. 14, no. 2, p. 21, 2011.

[67] "Webroot." [Online]. Available: https://www-cdn.webroot.com/9315/2354/6488/2018-Webroot-Threat-Report_US-ONLINE.pdf

[68] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[69] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.