Malicious URL Detection using Machine Learning: A Survey

Doyen Sahoo, Chenghao Liu, and Steven C.H. Hoi

***Abstract*—Malicious URL, a.k.a. malicious website, is a common and serious threat to cybersecurity. Malicious URLs host unsolicited content (spam, phishing, drive-by exploits, etc.) and lure unsuspecting users to become victims of scams (monetary loss, theft of private information, and malware installation), and cause losses of billions of dollars every year. It is imperative to detect and act on such threats in a timely manner. Traditionally, this detection is done mostly through the usage of blacklists. However, blacklists cannot be exhaustive, and lack the ability to detect newly generated malicious URLs. To improve the generality of malicious URL detectors, machine learning techniques have been explored with increasing attention in recent years. This article aims to provide a comprehensive survey and a structural understanding of Malicious URL Detection techniques using machine learning. We present the formal formulation of Malicious URL Detection as a machine learning task, and categorize and review the contributions of literature studies that addresses different dimensions of this problem (feature representation, algorithm design, etc.). Further, this article provides a timely and comprehensive survey for a range of different audiences, not only for machine learning researchers and engineers in academia, but also for professionals and practitioners in cybersecurity industry, to help them understand the state of the art and facilitate their own research and practical applications. We also discuss practical issues in system design, open research challenges, and point out some important directions for future research.**

***Index Terms*—Malicious URL Detection, Machine Learning, Online Learning, Internet security, Cybersecurity**

I. Introduction

The advent of new communication technologies has had tremendous impact in the growth and promotion of businesses spanning across many applications including online-banking, e-commerce, and social networking. In fact, in today's age it is almost mandatory to have an online presence to run a successful venture. As a result, the importance of the World Wide Web has continuously been increasing. Unfortunately, the technological advancements come coupled with new sophisticated techniques to attack and scam users. Such attacks include rogue websites that sell counterfeit goods, financial fraud by tricking users into revealing sensitive information which eventually lead to theft of money or identity, or even installing malware in the user's system. There are a wide variety of techniques to implement such attacks, such as explicit hacking attempts, drive-by exploits, social engineering, phishing, watering hole, man-in-the middle, SQL injections, loss/theft of devices, denial of service, distributed denial of service, and many others. Considering the variety of attacks, potentially new attack types, and the innumerable contexts in which such attacks can appear, it is hard to design robust systems to detect cyber-security breaches. The limitations of traditional security management technologies are becoming more and more serious given this exponential growth of new security threats, rapid changes of new IT technologies, and significant shortage of security professionals. Most of these attacking techniques are realized through spreading compromised URLs (or the spreading of such URLs forms a critical part of the attacking operation) [1].

URL is the abbreviation of Uniform Resource Locator, which is the global address of documents and other resources on the World Wide Web. A URL has two main components : (i) protocol identifier, it indicates what protocol to use, (ii) resource name, it specifies the IP address or the domain name where the resource is located. The protocol identifier and the resource name are separated by a colon and two forward slashes. An example is shown in Figure 1.

Compromised URLs that are used for cyber attacks are termed as *malicious URLs*. In fact, it was noted that close to one-third of all websites are potentially malicious in nature [2], demonstrating rampant use of malicious URLs to perpetrate cyber-crimes. A Malicious URL or a malicious web site hosts a variety of unsolicited content in the form of spam, phishing, or drive-by-exploits in order to launch attacks. Unsuspecting users visit such web sites and become victims of various types of scams, including monetary loss, theft of private information (identity, credit-cards, etc.), and malware installation. Popular types of attacks using malicious URLs include: Drive-by Download, Phishing and Social Engineering, and Spam [3]. Drive-by-download [4] refers to the (unintentional) download of malware upon just visiting a URL. Such attacks are usually carried out by exploiting vulnerabilities in plugins or inserting malicious code through JavaScript. Phishing and Social Engineering attacks [5] trick the users into revealing private or sensitive information by pretending to be genuine web pages. Spam is the usage of unsolicited messages for the purpose of advertising or phishing. These types of attacks occur in large numbers and have caused billions of dollars worth of damage every year. Effective systems to detect such malicious URLs in a timely manner can greatly help to counter large number of and a variety of cyber-security threats. Consequently, researchers and practitioners have worked to design effective solutions for Malicious URL Detection.

The most common method to detect malicious URLs de-

Doyen Sahoo is with the School of Information Systems, Singapore Management University, Singapore, email: doyen.2014@phdis.smu.edu.sg

Chenghao Liu was with the School of Information Systems, Singapore Management University, Singapore, email: twinsken@gmail.com

Corresponding author: Steven C.H. Hoi is with the School of Information Systems, Singapore Management University, Singapore, email: chhoi@smu.edu.sg

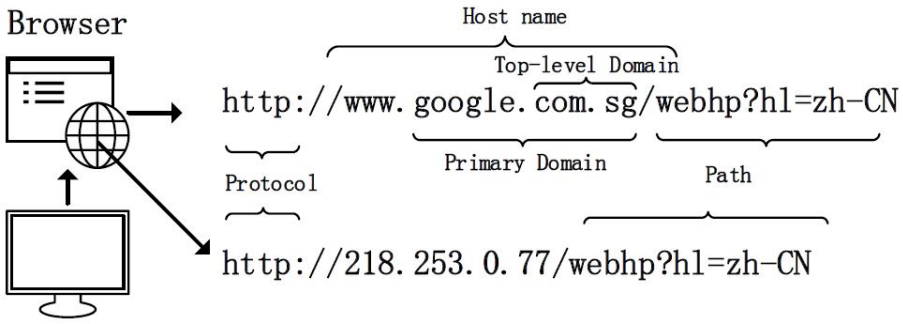

Fig. 1. Example of a URL - "Uniform Resource Locator"

ployed by many antivirus groups is the black-list method. Black-lists are essentially a database of URLs that have been confirmed to be malicious in the past. This database is compiled over time (often through crowd-sourcing solutions, e.g. PhishTank [6]), as and when it becomes known that a URL is malicious. Such a technique is extremely fast due to a simple query overhead, and hence is very easy to implement. Additionally, such a technique would (intuitively) have a very low false-positive rate (although, it was reported that often blacklisting suffered from non-trivial false-positive rates [7]). However, it is almost impossible to maintain an exhaustive list of malicious URLs, especially since new URLs are generated everyday. Attackers use creative techniques to evade blacklists and fool users by modifying the URL to "appear" legitimate via obfuscation. Garera et. al. [8] identified four types of obfuscation: Obfuscating the Host with an IP, Obfuscating the Host with another domain, Obfuscating the host with large host names, and misspelling. All of these try to hide the malicious intentions of the website by masking the malicious URL. Recently, with the increasing popularity of URL shortening services, it has become a new and widespread obfuscation technique (hiding the malicious URL behind a short URL) [9], [10]. Once the URLs appear legitimate, and user's visit them, an attack can be launched. This is often done by malicious code embedded into the JavaScript. Often the attackers will also try to obfuscate the code so as to prevent signature based tools from detecting them. Attackers use many other simple techniques to evade blacklists including: fast-flux, in which proxies are automatically generated to host the web-page; algorithmic generation of new URLs; etc. Additionally, attackers can often simultaneously launch more than one attack, which alters the attack-signature, making it undetectable by tools that focus on specific signatures. Blacklisting methods, thus have severe limitations, and it appears almost trivial to bypass them, especially due to the fact that blacklists are useless for making predictions on new URLs.

To overcome these issues, in the last decade, researchers have applied machine learning techniques for Malicious URL Detection [3], [8], [11]–[17]. Machine Learning approaches, use a set of URLs as training data, and based on the statistical properties, learn a prediction function to classify a URL as malicious or benign. This gives them the ability to generalize to new URLs unlike blacklisting methods. The primary requirement for training a machine learning model is the presence of training data. In the context of malicious URL detection, this would correspond to a set of large number of URLs. Machine learning can broadly be classified into supervised, unsupervised, and semi-supervised, which correspond to having the labels for the training data, not having the labels, and having labels for limited fraction of training data. Labels correspond to the knowledge that a URL is malicious or benign.

After the training data is collected, the next step is to extract informative features such that they sufficiently describe the URL and at the same time, they can be interpreted mathematically by machine learning models. For example, simply using the URL string directly may not allow us to learn a good prediction model (which in some extreme cases may reduce the prediction model to a blacklist method). Thus, one would need to extract suitable features based on some principles or heuristics to obtain a good feature representation of the URL. This may include lexical features (statistical properties of the URL string, bag of words, n-gram, etc.), host-based features (WHOIS info, geo-location properties of the host, etc.), etc. These features after being extracted have to be processed into a suitable format (e.g. a numerical vector), such that they can be plugged into an off-the-shelf machine learning method for model training. The ability of these features to provide relevant information is critical to subsequent machine learning, as the underlying assumption of machine learning (classification) models is that the feature representations of the malicious and benign URLs have different distributions. Therefore, the quality of feature representation of the URLs is critical to the quality of the resulting malicious URL predictive model learned by machine learning.

Finally, using the training data with the appropriate feature representation, the next step in building the prediction model is the actual training of the model. There are plenty of classification algorithms can be directly used over the training data (Naive Bayes, Support Vector Machine, Logistic Regression, etc.). However, there are certain properties of the URL data that may make the training difficult (both in terms of scalability and learning the appropriate concept). For example, the number of URLs available for training can be in the order of millions (or even billions). As a result, the training time for traditional models may be too high to be practical. Consequently, Online Learning [18], a family of scalable learning techniques have been heavily applied for this task. Similarly, for this task, URLs are represented using the bag-of-words (BoW) features. These features basically

indicate whether a particular word (or string) appears in a URL or not - as a result every possible type of word that may appear in any URL becomes a feature. This representation may result in millions of features which would be very sparse (most features are absent most of the time, as a URL will usually have very few of the millions of possible words present in it). Accordingly, a learning method should exploit this sparsity property to improve learning efficiency and efficacy. Despite the promising generalizing ability of machine learning approaches, one potential shortcoming of these approaches for malicious URL detection may be their resource intensive nature (especially while extracting features that are non-trivial and expensive to compute), reducing their practical value when requiring real-time security assurance compared to blacklisting methods.

In this survey, we review the state-of-the-art machine learning techniques for malicious URL detection in literature. We specifically focus on the contributions made for feature representation and learning algorithm development in this domain. We systematically categorize the various types of feature representation used for creating the training data for this task, and also categorize various learning algorithms used to learn a good prediction model. We also discuss the open research problems and identify directions for future research. In the rest of the survey, we first discuss the broad categories of strategies used for detecting malicious URLs - Blacklists, Heuristic and Machine Learning. We formalize the setting as a machine learning problem, where the primary requirement is good feature representation and the learning algorithm used. We then comprehensively present various types of feature representation used for this problem. This is followed by presenting various algorithms that have been used to solve this task, and have been developed based on the properties of URL data. Finally we discuss the newly emerging concept of Malicious URL Detection as a service and the principles to be used while designing such a system. We end the survey by discussing the practical issues and open problems in this domain.

II. Malicious URL Detection

In this section, we first present the key principles used by researchers and practitioner to solve the problem of Malicious URL detection, followed by formalizing it as a machine learning task.

A. Principles of Detecting Malicious URLs: An Overview

A variety of approaches have been attempted to tackle the problem of Malicious URL Detection. According to the fundamental principles, these approaches can be broadly grouped into two major categories: (i) Blacklisting or Heuristics, and (ii) Machine Learning approaches [19], [20]. Below we briefly describe the key principles of each category.

1) Blacklisting or Heuristic Approaches: Blacklisting approaches are a common and classical technique for detecting malicious URLs, which often maintains a list of URLs that are known to be malicious. Whenever a new URL is visited, a database lookup is performed. If the URL is present in the blacklist, it is considered to be malicious and then a warning will be generated; else it is assumed to be benign. Blacklisting suffers from the inability to maintain an exhaustive list of all possible malicious URLs, as new URLs can be easily generated daily, thus making it impossible for them to detect new threats [21]. This is particularly of critical concern when the attackers generate new URLs algorithmically, and can thus bypass all blacklists. Despite several problems faced by blacklisting [7], due to their simplicity and efficiency, they continue to be one of the most commonly used techniques by many anti-virus systems today.

Heuristic approaches [22] are some kind of extensions of Blacklist based methods, wherein the idea is to create a "blacklist of signatures". Common attacks are identified, and based on their behaviors, a signature is assigned to this attack type. Intrusion Detection Systems can scan the web pages for such signatures, and raise a flag if some suspicious behavior is found. These methods have better generalization capabilities than blacklisting, as they have the ability to detect threats in new URLs as well. However, such methods can be designed for only a limited number of common threats, and can not generalize to all types of (novel) attacks. Moreover, using obfuscation techniques, it is not difficult to bypass them. A more specific version of heuristic approaches is through analysis of execution dynamics of the webpage (e.g. [23]–[27] etc.). Here also, the idea is to look for a signature of malicious activity such as unusual process creation, repeated redirection, etc. These methods necessarily require visiting the webpage and thus the URLs actually can make an attack. As a result, such techniques are often implemented in controlled environment like a disposable virtual machine. Such techniques are very resource intensive, and require all execution of the code (including the rich client sided code). Another drawback is that websites may not launch an attack immediately after being visited, and thus may go undetected.

2) Machine Learning: These approaches try to analyze the information of a URL and its corresponding websites or webpages, by extracting good feature representations of URLs, and training a prediction model on training data of both malicious and benign URLs. There are two-types of features that can be used - *static* features, and *dynamic* features. In static analysis, we perform the analysis of a webpage based on information available without executing the URL (i.e., executing JavaScript, or other code) [12], [13], [20], [28]. The features extracted include lexical features from the URL string, information about the host, and sometimes even HTML and JavaScript content. Since no execution is required, these methods are safer than the Dynamic approaches. The underlying assumption is that the distribution of these features is different for malicious and benign URLs. Using this distribution information, a prediction model can be built, which can make predictions on new URLs. Due to the relatively safer environment to extracting important information, and the ability to generalize to all types of threats (not just common ones which have to be defined by a signature), static analysis techniques have been extensively explored by applying machine learning techniques. In this survey, we focus primarily on the static analysis techniques where machine

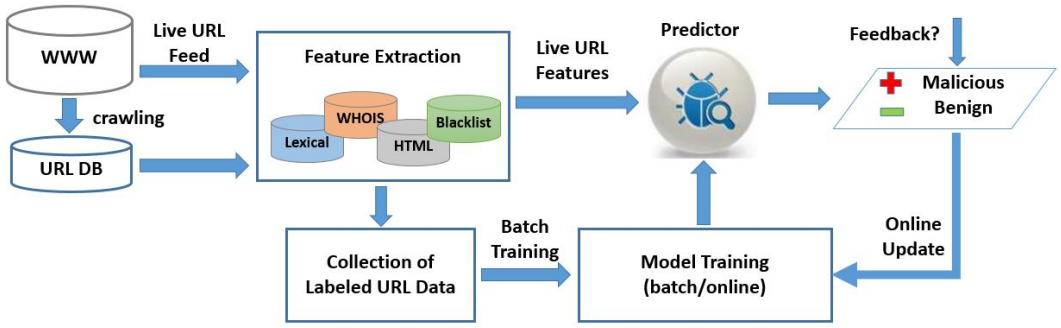

Fig. 2. A general processing framework for Malicious URL Detection using Machine Learning

learning has found tremendous success. Dynamic analysis techniques include monitoring the behavior of the systems which are potential victims, to look for any anomaly. These include [29] which monitor the system call sequences for abnormal behavior, and [30] which mine internet access log data for suspicious activity. Dynamic analysis techniques have inherent risks, and are difficult to implement and generalize.

In the following, we formalize the problem of malicious URL detection as a machine learning task which allows us to generalize most of the existing work in literature. Alternate problem settings will also be discussed in Section IV.

B. Problem Formulation

We formulate the problem of malicious URL detection as a binary classification task for two-class prediction: "malicious" versus "benign". Specifically, given a data set with T URLs $\{(\mathbf{u}_1, y_1), \ldots, (\mathbf{u}_T, y_T)\}$, where $\mathbf{u}_t$ for $t = 1, \ldots, T$ represents a URL from the training data, and $y_t \in \{1, -1\}$ is the corresponding label where $y_t = 1$ represents a malicious URL and $y_t = -1$ represents a benign URL. The crux to automated malicious URL detection is two-fold:

1) *Feature Representation*: Extracting the appropriate feature representation: $\mathbf{u}_t \rightarrow \mathbf{x}_t$ where $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional feature vector representing the URL; and
2) *Machine Learning*: Learning a prediction function $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}$ which predicts the class assignment for any URL instance $\mathbf{x}$ using proper feature presentations.

Consider a binary classification task, the goal of machine learning for malicious URL detection is to maximize the predictive accuracy. Both of the folds above are important to achieve this goal. While the first part of feature representation is often based on domain knowledge and heuristics, the second part focuses on training the classification model via a data driven optimization approach. Fig. 2 illustrates a general architecture of solving Malicious URL Detection using machine learning.

The first key step is to convert a URL $\mathbf{u}$ into a feature vector $\mathbf{x}$, where several types of information can be considered and different techniques can be used. Unlike learning the prediction model, this part cannot be directly computed by a mathematical function (not for most of it). Using domain knowledge and related expertise, a feature representation is constructed by crawling all relevant information about the URL. These range from lexical information (length of URL, the words used in the URL, etc.) to host-based information (WHOIS info, IP address, location, etc.). Once the information is gathered, it is processed to be stored in a feature vector $\mathbf{x}$. Numerical features can be stored in $\mathbf{x}$ as is, and identity related information or lexical features are usually stored through a binarization or bag-of-words (BoW) approach. Based on the type of information used, $\mathbf{x} \in \mathbb{R}^d$ generated from a URL is a d-dimensional vector where d can be less than 100 or can be in the order of millions. A unique challenge that affects this problem setting is that the number of features may not be fixed or known in advance. For example, using a BoW approach one can track the occurrence for every type of word that may have occurred in a URL in our training data. A model can be trained on this data, but while predicting, new URLs may have words that did not occur in the training data. It is thus a challenging task to design a good feature representation that is robust to unseen data.

After obtaining the feature vector $\mathbf{x}$ for the training data, to learn the prediction function $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}$, it is usually formulated as an optimization problem such that the detection accuracy is maximized (or alternately, a loss function is minimized). The function $\mathbf{f}$ is (usually) parameterized by a $d-$dimensional weight vector $\mathbf{w}$, such that $\mathbf{f}(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x})$. Let $\hat{y}_t = \text{sign}(\mathbf{f}(\mathbf{x}_t))$ denote the class label prediction made by the function $\mathbf{f}$. The number of mistakes made by the prediction model on the entire training data is given by: $\sum_{t=1}^{T} \mathbb{I}_{\hat{y}_t = y_t}$ where $\mathbb{I}$ is an indicator which evaluates to 1 if the condition is true, and 0 otherwise. Since the indicator function is not convex, the optimization can be difficult to solve. As a result, a convex loss function is often defined, and is denoted by $\ell(\mathbf{f}(\mathbf{x}), y)$ and the entire optimization can be formulated as:

$$\min_{\mathbf{w}} \sum_{t=1}^{T} \ell(\mathbf{f}(\mathbf{x}_t), y_t) \tag{1}$$

Several types of loss functions can be used, including the popular hinge-loss $\ell(\mathbf{f}(\mathbf{x}), y) = \frac{1}{2}\max(1 - y\mathbf{f}(\mathbf{x}), 0)$, or the squared-loss $\ell(\mathbf{f}(\mathbf{x}), y) = \frac{1}{2}(\mathbf{f}(\mathbf{x}) - y)^2$. Sometimes, a regularization term is often added to prevent over-fitting or to learn sparse models, or the loss function can be modified based on cost-sensitive nature of the data (e.g., class imbalanced distribution, different costs for diverse threats).

In the following, we will discuss the existing studies on feature representation for malicious URL detection and appropriate machine learning algorithms design in detail.

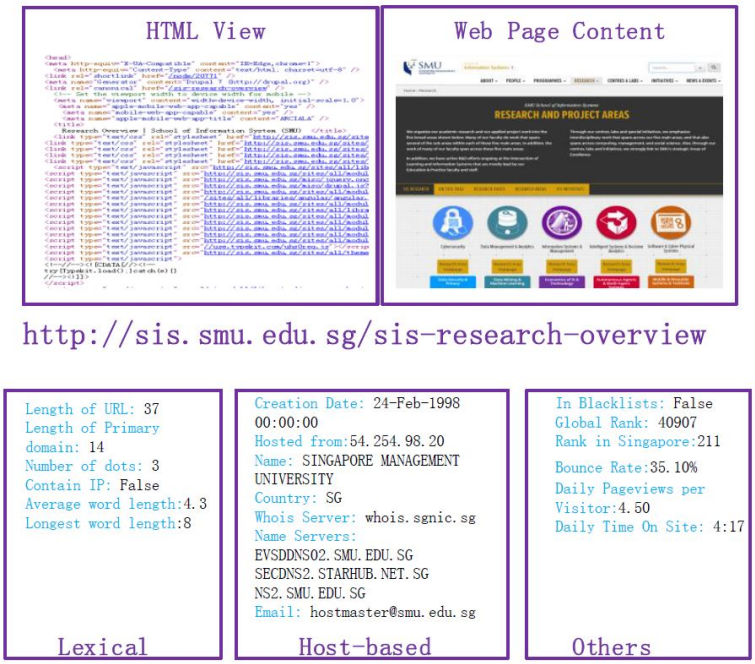

Fig. 3. Example of information about a URL that can be obtained in the Feature Collection stage

III. Feature Representation

As stated earlier, the success of a machine learning model critically depends on the quality of the training data, which hinges on the quality of feature representation. Given a URL $\mathbf{u} \in \mathbb{U}$, where $\mathbb{U}$ denotes a domain of any valid URL strings, the goal of feature representation is to find a mapping $g : \mathbb{U} \rightarrow \mathbb{R}^d$, such that $g(\mathbf{u}) \rightarrow \mathbf{x}$ where $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional feature vector, that can be fed into machine learning models. The process of feature representation can be further broken down into two steps:

1) *Feature Collection*: This phase is engineering oriented, which aims to collect most if not all relevant information about the URL. This includes information such as presence of the URLs in a blacklist, the direct features of the URL such as the URL String and information about the host, the content of the web-site such as HTML and JavaScript, popularity information, etc. Figure 3 gives an example to demonstrate various types various types of information that can be collected from a URL to obtain the feature representation.
2) *Feature Preprocessing*: In this phase, the unstructured information about the URL (e.g. textual description) is appropriately formatted, and converted to a numerical vector so that it can be fed into machine learning algorithms. For example, the numerical information can be used as is, and the BoW is used for representing textual or lexical content. Besides, some data normalization (e.g., Z-score normalization) may often be used to handle the scaling issue.

For malicious URL detection, researchers have proposed several types of features that can be used to provide useful information. We categorize these features into: Blacklist Features, URL-based Lexical Features, Host-based features, Content-based Features, and Others (Context, Popularity, etc.). All have their benefits and short-comings - while some are very informative, obtaining these features can be very expensive. Similarly, different features have different preprocessing challenges and security concerns. Next, we will discuss each of these feature categories in detail, followed by comparing their pros and cons.

A. BlackList Features

As mentioned before, a trivial technique to identify malicious URLs is to use blacklists. An existing URL as having been identified as malicious (either through extensive analysis or crowd sourcing) makes its way into the list. However, it has been noted that blacklisting, despite its simplicity and ease of implementation, suffers from nontrivial high false negatives [7] due to the difficulty in maintaining exhaustive up-to-date lists. Consequently, instead of using blacklist presence alone as a decision maker, it can be used as a powerful feature. In particular, [12] used the presence in a blacklist as a feature, from 6 different blacklist service providers. They also analyzed the effectiveness of these features compared to other features, and observed that blacklist features alone did not have as good a performance as other features, but when used in conjunction with other features, the overall performance of the prediction model improved.

[31] observed that to evade detection via blacklisting, many attackers made minor modifications to the original URL. They proposed to extend the blacklist by deriving new URLs based on five heuristics including: Replacing Top-Level Domain (TLDs), IP Address Equivalence, Directory Structure Similarity, Query String substitution, and brand name equivalence. Since, even a minor mismatch from the blacklist database can cause a malicious URL to go undetected, they also devised an

approximate matching solution. Similar heuristics potentially could be used when deriving blacklist features for machine learning approaches. A similar methodology was adopted for automated URL blacklist generation by [32], [33]. [34] developed a method to proactively perform domain blacklisting.

B. Lexical Features

Lexical features are features obtained based on the properties of the URL name (or the URL string). The motivation is that based on how the URL "looks" it should be possible to identify malicious nature of a URL. For example, many obfuscation methods try to "look" like benign URLs by mimicking their names and adding a minor variation to it. In practice, these lexical features are used in conjunction with several other features (e.g. host-based features) to improve model performance. However, using the original URL name directly is not feasible from a machine learning perspective. Instead, the URL string has to be processed to extract useful features. Next we review some of the lexical features used for malicious URL detection.

Traditional Lexical Features: The most commonly used lexical features include statistical properties of the URL string, like the length of the URL, length of each of the components of the URL (Hostname, Top Level Domain, Primary domain, etc.), the number of special characters, etc. [35] were one of the first to suggest extracting words from the URL string. The string was processed such that each segment delimited by a special character (e.g. "/", ".", "?", "=", etc.) comprised a word. Based on all the different types of words in all the URLs, a dictionary was constructed, i.e., each word became a feature. If the word was present in the URL, the value of the feature would be 1, and 0 otherwise. This is also known as the bag-of-words model.

Directly using the bag-of-words model, causes a loss of information about the order in which the words occurred in the URL. [12], [28] also used similar lexical features, but they made the distinction between the tokens belonging to the hostname, the path, the top-level domain and the primary domain name. This was done by having a separate dictionary for each of these segments. The distinction would allow for preserving some of the order in which the words occurred. For example, it allows us to distinguish between the presence of "com" in the top-level domain vs other parts of the URL. [36] try to enhance the lexical features by considering the usage of bi-gram features, i.e., they construct a dictionary, where in addition to single-words the presence of a set of 2-words in the same URL is considered a feature. In addition, they record the position of sensitive tokens and bigrams to exploit the token context sensitivity.

The entire bag-of-word features approach can be viewed as a form of machine learning compatible fuzzy blacklist approach. Instead of focussing on the entire URL string, it assigns scores to the URL based on smaller components of the URL string. While this approach offers us an extensive number of features, it can become problematic while running sophisticated algorithms on them. For example, [28] collected a dataset of 2 million URLs, having almost as many lexical features. This number may grow even larger if bi-gram features were considered. [35] considered n-gram features (same as bi-gram, but n can be > 2), and devised a feature selection scheme based on relative entropy to reduce the dimensionality. A similar feature extraction method was used by [37], where the feature weights were computed based on the ratio of their presence in one class of URLs against their presence in both classes of URLs.

In order to avoid being caught by blacklists, hackers can generate malicious URLs algorithmically. Using bag-of-words feature for such URLs is likely to give a poor performance, as algorithmically generated URLs may produce never before seen words (hence never before seen features). To detect such algorithmically generated malicious URLs, [38] analyzed character level strings to obtain the features. They argued that algorithmically generated domain names and those generated by humans would have a substantially different alpha-numeric distribution. Further, since the number of characters is small, the number of features obtained would also be small. They performed their analysis based on KL-divergence, Jaccard Coefficient, and Edit-distance using unigram and bigram distributions of characters.

Advanced Lexical Features: Traditional lexical features were directly obtained from the URL string without significant domain knowledge or computation. Researchers have proposed several advanced lexical features that exploit properties of URL strings, to get more informative features.

[39] derive new lexical features using heuristics with the objective of being obfuscation resistant. Based on the obfuscation types identified by [8], five categories of features are proposed: URL-related features (keywords, length, etc.), Domain features (length of domain name, whether IP address is used as domain name, etc.), Directory related features (length of directory, number of subdirectory tokens, etc.), File name features (length of filename, number of delimiters, etc.), and Argument Features(length of the argument, number of variables, etc.).

Another feature is based on the *Kolmogorov Complexity* [40]. Kolmogorov Complexity is a measure of complexity of a string s. Conditional Kolmogorov Complexity is the measure of the complexity of a string s given another string for free. This means that the presence of the free string does not add to the complexity of the original input string s. Based on this, for a given URL, we compute the URL's Conditional Kolmogorov Complexity with respect to the set of Benign URLs and the set of Malicious URLs. Combining these measures we get a sense of whether the given URL is more similar to the Malicious URL database or the Benign URL database. This feature, though useful, may not be easy to scale up to very large number of URLs. [41], [42] define a new concept of intra-URL relatedness which is a measure to quantify the relations between different words that comprise the URL with specific focus on relationship between the registered domain and the rest of the URL. [43] propose new distance based metrics, called *domain brand name distance* and *path brand name distance*. These are essentially types of edit distance between strings aimed at detecting those malicious URLs which try to mimic popular brands or websites.

C. Host-based Features

Host-based features are obtained from the host-name properties of the URL [28]. They allow us to know the location of malicious hosts, the identity of the malicious hosts, and the management style and properties of these hosts.

[11] studied the impact of a few host-based features on the maliciousness of URLs. Some of the key observations were that phishers exploited Short URL services; the time-to-live from registration of the domain was almost immediate for the malicious URLs; and many used botnets to host themselves on multiple machines across several countries. Consequently, host-based features became an important element in detecting malicious URLs.

[12], [28] borrowed ideas from [11] and proposed the usage of several host-based features including: *IP Address properties*, *WHOIS information*, *Location*, *Domain Name Properties*, and *Connection Speed*. The IP Address properties comprise features obtained from IP address prefix and autonomous system (AS) number. This included whether the IPs of A, MX or NS records are in the same ASes or prefixes as one another. The WHOIS information comprises domain name registration dates, registrars and registrants. The Location information comprises the physical Geographic Location - e.g. country/city to which the IP address belongs. The Domain Name properties comprise time-to-live values, presence of certain keywords like "client" and "server", if the IP address is in the host name or not and does the PTR record resolve one of the host's IP addresses. Since many of the features are identity related information, a bag-of-words like approach is required to store them in a numerical vector, where each word corresponds to a specific identity. Like the lexical features, adopting such an approach leads to a large number of features. For the 2 million URLs, [28] obtained over a million host-based features. Exclusive usage of IP Address Features has also been considered [44]. IP Address Features are arguably more stable, as it is difficult to obtain new IP Addresses for malicious URLs continuously. Due to this stability, they serve as important features in malicious URL detection. However, it is cumbersome to use IP Address directly. Instead, it is proposed to extract IP Address features based on a binarization or categorization approach through which octet-based, extended-octet based and bit-string based features are generated.

DNS Fluxiness features were proposed to look for malicious URLs that would hide their identity by using proxy networks and quickly changing their host [45], [46]. [43] define *domain age* and *domain confidence* (dependent on similarity with a white-list) level which help determine the fluxiness nature of the URL (e.g. malicious URLs using fast flux will have a small domain age). [47] propose new features to detect malicious URLs that are hidden within trusted sites. They extract *header* features from HTTP response headers. They also use the *age* obtained from the time stamp value of the last modified header. [48] propose *Application Layer* features and *Network Layer* features to devise a cross-layer mechanism to detect malicious URLs. [49] suggest the usage of *temporal variation patterns* based on active analysis of DNS logs, to help discover domain names that could be abused in the future.

D. Content-based Features

Content-based features are those obtained upon downloading the entire web-page. As compared to URL-based features, these are "heavy-weight", as a lot of information needs to be extracted, and at the same time, safety concerns may arise. However, with more information available about a particular web-page, it is natural to assume that it would lead to a better prediction model. Further, if the URL-based features fail to detect a malicious URL, a more thorough analysis of the content-based features may help in early detection of threats [19]. The content-based features of a web-page can be drawn primarily from its HTML content, and the usage of JavaScript. [50] categorize the content based features of a web-page into 5 broad segments: Lexical features, HTML Document Level Features, JavaScript features, ActiveX Objects and feature relationships. [51], [52] proposed CANTINA and its variants for detecting phishing websites using a comprehensive feature-based machine learning approach, by exploiting various features from the HTML Document Object Model (DOM), search engines and third party services. In the following we discuss some of these categories, primarily focusing on the HTML Document Level Features and JavaScript Features.

1) HTML Features: [50] proposed the usage of lexical features from the HTML of the web-page. These are relatively easy to extract and preprocess. At the next level of complexity, the HTML document level features can be used. The document level features correspond to the statistical properties of the HTML document, and the usage of specific types of functionality. [50] propose the usage of features like: length of the document, average length of the words, word count, distinct word count, word count in a line, the number of NULL characters, usage of string concatenation, unsymmetrical HTML tags, the link to remote source of scripts, and invisible objects. Often malicious code is encrypted in the HTML, which is linked to a large word length, or heavy usage of string concatenation, and thus these features can help in detecting malicious activity.

Similar features with minor variations were used by many of the subsequent researchers including [46] (number of iframes, number of zero size iframes, number of lines, number of hyperlinks, etc.). [19] also used similar features, and additionally proposed to use several more descriptive features which were aimed at minor statistical properties of the page. These include features such as number of elements with a small area, number of elements with suspicious content (suspiciousness was determined by the length of the content between the start and end tag), number of out of place elements, presence of double documents, etc. [53] developed a *delta* method, where *delta* represented the change in different versions of the website. They analyzed whether the change was malicious or benign.

2) JavaScript Features: [50] argue that several JavaScript functions are commonly used by hackers to encrypt malicious code, or to execute unwanted routines without the client's permission. For example extensive usage of function *eval()* and *unescape()* may indicate execution of encrypted code within the HTML. They aim to use the count of 154 native JavaScript functions as features to identify malicious URLs.

[46] identify a subset (seven) of these native JavaScript functions that are often in Cross-site scripting and Web-based malware distribution. These include: escape(), eval(), link(), unescape(), exec(), and search() functions. [19] propose additional heuristic JavaScript features including: keywords-to-words ratio, number of long strings, presence of decoding routines, shell code presence probability, number of direct string assignments, number of DOM-modifying functions, number of event attachments, number of suspicious object names, number of suspicious strings, number of "iframe" strings and number of suspicious string tags. In [54], the authors try to detect JavaScript Obfuscation by analyzing the JavaScript codes using n-gram, Entropy and Word Size. n-gram and word size are commonly used to look for character/word distribution and presence for long strings. For Entropy of the strings, they observe that obfuscated strings tend to have a lower entropy. More recently, [55] applied deep learning techniques to learn feature representations from JavaScript code.

3) Visual Features: There have also been attempts made at using images of the webpages to identify the malicious nature of the URL. Most of these focus on computing visual similarity with *protected pages*, where the protected pages refer to genuine websites. Finding a high level of visual similarity of a suspected malicious URL could be indicative of an attempt at phishing. One of the earliest attempts at using visual features for this task was by computing the Earth Mover's Distance between 2 images [56]. [57], [58] addressed the same problem and developed a system to extract visual features of web pages based on text-block features and image-block features (using information such as block size, color, etc.). More advanced computer vision technologies were adapted for this task. Contrast Context Histogram (CCH) features were suggested [59], and so were Scale Invariant Feature Transform (SIFT) features [60]. Another approach using visual feature was developed by [61], where an OCR was used to read the text in the image of the webpage. [62] combine both textual and visual features for measuring similarity. With recent advances in Deep Learning for Image Recognition [63], [64], it may be possible to extract more powerful and effective visual features.

4) Other Content-based Features: [50] argued that due to the powerful functionality of ActiveX objects, they can be used to create malicious DHTML pages. Thus, they tried to compute frequency for each of eight ActiveX objects. Examples include: "Scripting.FileSystemObject" which can be used for file system I/O operations, "WScript.Shell" which can execute shell scripts on the client's computer, and "Adodb.Stream" which can download files from the Internet. [65] try to find the identity and keywords in the DOM text and evaluate the consistency between the identity observed and the identity it is potentially trying to mimic which is found by searching. [66] used the directory structure of the websites to obtain insights.

E. Other Features

Recent years have seen the growth of Short URL service providers, which provide the original URL to be represented by a shorter string. This enables sharing of the URLs in on social media platforms like twitter, where the originally long URLs would not fit within the 140 character limit of a tweet. Unfortunately, this has also become a popular obfuscation technique for the malicious URLs. While the Short URL service providers try their best to not generate short URLs for the malicious ones, they struggle to do an effective job as they also rely primarily on blacklists [67], [68]. As a result, a recently emerging research direction has become active where *context-features* of the URL are obtained, i.e., the features of the background information where the URL has been shared. [69] use context information derived from the tweets where the URL was shared. [70] used click traffic data to classify short URLs as malicious or not. [71] propose forwarding based features to combat forwarding-based malicious URLs. [72] propose another direction of features to identify malicious URLs - they also focus on URLs shared on social media, and aim to identify the malicious nature of a URL by performing behavioral analysis of the users who shared them, and the users who clicked on them. These features are formally called "Posting-based" features and "Click-based" features. [10] approach this problem with a systematic categorization of context features which include content-related features (lexical and statistical properties of the tweet), context of the tweet features (time, relevance, and user mentions) and social features (following, followers, location, tweets, retweets and favorite count).

Some other features used were designed heuristics to measure the *popularity* of the URL. One of the earliest approaches to applying statistical techniques to detect malicious URLs [8] aimed at probabilistically identifying the importance of specific hand-designed features. These include Page-based features (Page rank, quality, etc.), Domain-based features (presence in *white domain table*), Type-based features (obfuscation types) and Word-based features(presence of keywords such as "confirm", "banking", etc.). [73] use both the URL-based and content based features, and additionally record the initial URL, the landing URL and the redirect chain. Further they record the number of popups and the behavior of plugins, which have been commonly used by spammers. [46] proposed the usage of new categories of features: Link Popularity and Network Features. Link Popularity is scored on the basis of incoming links from other webpages. This information was obtained from different search engines. In order to make the usage of these features robust to manipulation, they also propose the usage of certain metrics that validate the quality of the links. They also use a metric to detect spam-to-spam URL links. For their work, they use these features in conjunction with lexical features content-based feature, and host-based features. [20] used social reputation features of URLs by tracking their public share count on Facebook and Twitter. [74] incorporated information on redirection chains into redirection graphs, which provided insight into detecting malicious URLs. [42] use search engine query data to mine for word relatedness measurement.

F. Summary of Feature Representations

There is a wide variety of information that can be obtained for a URL. Crawling the information and transforming the

unstructured information to a machine learning compatible feature vector can be very resource intensive. While extra information can improve predictive models (subject to appropriate regularization), it is often not practical to obtain a lot of features. For example, several host-based features may take a few seconds to be obtained, and that itself makes using them in real world setting impractical. Another example is the Kolmogorov Complexity - which requires comparing a URL to several malicious and benign URLs in a database, which is infeasible for comparing with billions of URLs. Accordingly, care must be taken while designing a Malicious URL Detection System to tradeoff the usefulness of a feature and the difficulty in retrieving it. We present a subjective evaluation of different features used in literature. Specifically, we evaluate them on the basis of Collection Difficulty, Associated Security Risks, need for an external dependency to acquire information, the associated time cost with regard to feature collection and feature preprocessing, and the dimensionality of the features obtained.

Collection difficulty is refers to the engineering effort required to obtain specific information about the features. Blacklist, context and popularity features require additional dependencies and thus have a higher collection overhead, whereas the other features are directly obtained from the URL itself. This also implies, that for a live-system (i.e. real-time Malicious URL Detection), obtaining features with a high collection time may be infeasible. In terms of associated security risks, the content-features have the highest risk, as potential malware may be explicitly downloaded while trying to obtain these features, while other features do not suffer from these issues. The collection time of the blacklist features can be high if the external dependency has to be queried during runtime, however, if the the entire blacklist can be stored locally, the collection overhead is very small. Collection of the lexical features is very efficient, as they are basically direct derivatives of the URL string. Host-based features are relatively time-consuming to obtain. Content-features usually require downloading the web-page which would affect the feature collection time. For preprocessing, once the data has been collected, deriving the features in most cases is computationally very fast. For dimensionality size, the lexical features have a very high-dimensionality (and so do unstructured Host-features and content features). This is largely because they are all stored as Bag-of-Words features. This feature size consequently affects the training and test-time. These properties are summarized in Table I. We also categorize the representative references according to the feature representation used, in Table II.

IV. Machine Learning Algorithms for Malicious URL Detection

There is a rich family of machine learning algorithms in literature, which can be applied for solving malicious URL detection. After converting URLs into feature vectors, many of these learning algorithms can be generally applied to train a predictive model in a fairly straightforward manner. However, to effectively solve the problem, some efforts have also been explored in devising specific learning algorithms that either exploit the properties exhibited by the training data of Malicious URLs, or address some specific challenges which the application faces. In this section, we categorize and review the learning algorithms that have been applied for this task, and also suggest suitable machine learning technologies that can be used to solve specific challenges encountered. We categorize the learning algorithms into: Batch Learning Algorithms, Online Algorithms, Representation Learning, and Others. Batch Learning algorithms work under the assumption that the entire training data is available prior to the training task. Online Learning algorithms treat the data as a stream of instances, and learn a prediction model by sequentially making predictions and updates. This makes them extremely scalable compared to batch algorithms. Next, we discuss representation learning methods, which in the context of Malicious URL Detection are largely concentrated towards feature selection techniques. Lastly, we discuss other learning algorithms, in challenges specific to Malicious URL Detection are addressed, including cost-sensitive learning, active learning, similarity learning, unsupervised learning and string pattern matching.

A. Batch Learning

Following the previous problem setting, consider a URL data set with T URLs $\{(\mathbf{u}_1, y_1), \ldots, (\mathbf{u}_T, y_T)\}$, where $\mathbf{u}_t \in \mathbb{U}$ for $t \in 1, \ldots, T$ represents a URL from the training data, and $y_t \in \{1, -1\}$ is its class label where $y = 1$ indicates a malicious URL and $y = -1$ indicates a benign URL. Using an appropriate feature representation scheme ($g : \mathbb{U} \mapsto \mathbb{R}^d$) as discussed in the previous section, one can map a URL instance into a d-dimensional feature vector, i.e., $g(\mathbf{u}_i) \rightarrow \mathbf{x}_i$. As a result, one can apply any existing learning algorithm that can work with vector space data to train a predictive model for malicious URL detection tasks. In this section we review the most common popular batch learning algorithms that have been applied for Malicious URL Detection.

A popular family of batch learning algorithms can be categorized under a discriminative learning framework using regularized loss minimization as:

$$\min_{\mathbf{f}} \sum_{t=1}^{T} \ell(\mathbf{f}(\mathbf{x}_t), y_t) + \lambda \mathcal{R}(\mathbf{w}) \tag{2}$$

where $\mathbf{f}(\mathbf{x}_t)$ can be either a linear model, e.g., $\mathbf{f}(\mathbf{x}_t) = \mathbf{w} \cdot \mathbf{x}_t + b$, or some nonlinear models (kernel-based or neural networks), $\ell(\mathbf{f}(\mathbf{x}_t), y_t)$ is some loss function to measure the difference between the model's prediction $\mathbf{f}(\mathbf{x}_t)$ and the true class label y, $\mathcal{R}(\mathbf{w})$ is a regularization term to prevent overfitting, and λ is a regularization parameter to trade-off model complexity and simplicity. In the following, we discuss two popular learning algorithms under this framework: Support Vector Machines and Logistic Regression.

1) Support Vector Machine: (SVM) is one of most popular supervised learning methods. It exploits the structural risk minimization principle using a maximum margin learning approach, which essentially can be viewed as a special instance of the regularized loss minimization framework. Specifically, by choosing the hinge loss as the loss function and maximiz-

TABLE I
PROPERTIES OF DIFFERENT FEATURE REPRESENTATIONS FOR MALICIOUS URL DETECTION

Features	Category	Criteria					
		Collection Difficulty	Risk	External Dependency	Collection Time	Processing Time	Feature Size
Blacklist	**Blacklist**	Moderate	Low	Yes	Moderate	Low	Low
Lexical	**Traditional**	Easy	Low	No	Low	Low	Very High
	Advanced	Easy	Low	No	Low	High	Low
Host	**Unstructured**	Easy	Low	No	High	Low	Very High
	Structured	Easy	Low	No	High	Low	Low
Content	**HTML**	Easy	High	No	Depends	Low	High
	JavaScript	Easy	High	No	Depends	Low	Moderate
	Visual	Easy	High	No	Depends	High	High
	Other	Easy	High	No	Depends	Low	Low
Others	**Context**	Difficult	Low	Yes	High	Low	Low
	Popularity	Difficult	Low	Yes	High	Low	Low

TABLE II
REPRESENTATIVE REFERENCES OF DIFFERENT TYPES OF FEATURES USED BY RESEARCHERS IN LITERATURE

Feature	Sub Category	Representative References
Blacklist	Blacklist	[8], [12], [31]–[34]
Lexical	Lexical	[8], [12], [13], [19], [20], [28], [35]–[43], [46], [48], [62], [73], [75]–[91]
Host	Host-based	[11]–[13], [17], [19], [28], [44]–[46], [46]–[48], [73], [76], [77], [80], [85]
Content	HTML	[2], [19], [20], [22], [46], [48], [50], [73], [78], [79], [83], [84], [90], [92]–[95]
	JavaScript	[2], [4], [19], [20], [26], [48], [50], [54], [55], [73], [83], [84], [94]
	Visual	[56]–[62], [79], [96]
	Others	[50], [65], [66], [94]
Others	Context-based	[10], [16], [69]–[72], [97]
	Popularity-based	[8], [20], [42], [46], [73], [77], [83], [84], [98]–[102]

ing the margin, SVM can be formulated into the following optimization:

$$(\mathbf{w}, b) \leftarrow \arg\min_{\mathbf{w},b} \frac{1}{T} \sum_{t=1}^{T} \max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t + b)) + \lambda \|\mathbf{w}\|_2^2$$

In addition, SVM can learn nonlinear classifiers using kernels [103]. SVMs are probably one of the most commonly used classifiers for Malicious URL Detection in literature [10], [12], [35], [40]–[43], [47], [48], [50], [69], [70], [79], [81], [92], [93].

2) Logistic Regression: is another well-known discriminative model which computes the conditional probability for a feature vector $\mathbf{x}$ to be classified as a class $y = 1$ by

$$P(y = 1|\mathbf{x}; \mathbf{w}, b) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}} \quad (3)$$

Based on the maximum-likelihood estimation (equivalently defining the loss function as the negative log likelihood), the optimization of logistic regression can be formulated as

$$(\mathbf{w}, b) \leftarrow \arg\min_{\mathbf{w},b} \frac{1}{T} \sum_{t=1}^{T} -\log P(y_t|\mathbf{x}_t; \mathbf{w}, b) + \lambda \mathcal{R}(\mathbf{w}) \quad (4)$$

where the regularization term can be either L2-norm $\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||_2$ or L1-norm $\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||_1$ for achieving a sparse model for high-dimensional data. Logistic Regression has been a popular learning method for Malicious URL Detection [8], [12], [19], [48], [50], [70], [79].

Other commonly used supervised learning algorithms focus on feature-wise analysis to obtain the prediction model. These include the Naive Bayes Classifier which computes the posterior probability of the class label assuming feature independence, and Decision Tree which adopts a greedy approach to constructing if-else rules based on the features offering the best splitting criteria.

3) Naive Bayes: is a generative model for classification, which is "naive" in the sense that this model assumes all features of $\mathbf{x}$ are independent of each other. Specifically, let $P(\mathbf{x}|y)$ denote the conditional probability of the feature vector given a class, the independence assumption implies that $P(\mathbf{x}|y) = \Pi_{i=1}^{d} P(x_i|y)$, where d is the number of features. By applying the Bayes Theorem, one can compute the posterior probability that a feature vector $\mathbf{x}$ is a malicious URL by

$$P(y = 1|\mathbf{x}) = \frac{P(\mathbf{x}|y = 1)}{P(\mathbf{x}|y = 1) + P(\mathbf{x}|y = -1)} \quad (5)$$

Naive Bayes has been used for Malicious URL Detection by [19], [48], [50], [62], [71], [97].

4) Decision Trees: is one of most popular methods for inductive inference and has a major advantage of its highly interpretable decision tree classification models which can also been converted into a rule set for human readability. Decision Trees have been used for malicious URL/web classification by [10], [12], [19], [22], [41], [42], [44], [48], [70], [71], [80], [97]. A closely related approach which gives us rules in the form of If-then was applied in using Associative Classification mining by [104].

5) Others and Ensembles: In addition to the above, other recently proposed approaches include applying Extreme Learning Machines (ELM) for classifying the phishing web sites using ELM by combining hybrid features in [105], and the spherical classification approach that allows batch learning models to be suitable for a large number of instances [106].

Beyond the binary classification approaches, [46] formulated the problem of malicious URL detection as a *multi-label classification* task. The argument for the need of multi-label classification is that different attacks have varying degrees of threats. For example, a spam URL is not as deadly as a malware infection. They proposed a two-step method: first using SVM for classifying a URL as malicious or benign; and second, performing multi-label classification on the malicious URLs using some popular multi-label learning methods (e.g., RAkEL and ML-kNN).

In addition, there are quite a few malicious ULR detection approaches using ensemble learning methods. For example, [107] applied Adaboost for detecting phishing websites using a content-based approaching together with Latent Dirichlet Allocation (LDA) for topic modeling. [20] employed an ensemble of multiple classifiers to make a weighted prediction. They independently train Decision Trees, Random Forests, Bayesian classifiers, Support Vector Machines and Logistic Regression, and design a confidence weighted majority voting scheme to make the final prediction. [87] adopt a multi-view analysis where a logistic regression model is trained on different portions of the URL lexical features, and their optimal combination is learnt. [83], [84] adopt an evolutionary optimization method to search for the best combination of features and models to obtain the final ensemble. In practice, ensemble learning is a common and very successful learning strategy when there is a need for boosting the predictive performance.

Although batch learning algorithms are popular and easy to use, they can suffer from several major limitations when dealing with real-world malicious URL detection tasks. For example, batch learning methods often suffer from expensive retraining cost when the new training data may arrive frequently. Moreover, due to expensive retraining cost, batch learning algorithms often do not update the model frequently, making them difficult to capture some emerging threats in a timely way. Last but not least, batch learning methods may poorly adapt to the concept drift due to their nature of batch training. To address these limitations, online learning algorithms have been emerging as a promising direction for resolving the Malicious URL Detection tasks.

B. Online Learning

Online Learning represents a family of efficient and scalable learning algorithms that learn from data sequentially [18], [108]. Consider malicious URL detection, given a sequence of T labeled instances, denoted by $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$, where $\mathbf{x}_t \in \mathbb{R}^d$ denotes the URL's feature representation, and $y_t \in \{-1, +1\}$ is the class label. $y = +1$ denotes a malicious URL, and $y_t = -1$ denotes a benign URL. At each iteration t, the algorithm makes a prediction $\mathbf{f}(\mathbf{x}_t) = sgn(\mathbf{w} \cdot \mathbf{x}_t)$ where $\mathbf{w}$ is a d-dimensional weight vector initialized to $\mathbf{0}$ at $t = 0$. After the prediction, the true class label y_t is revealed to the learner, and based on the loss suffered, the learner makes an update of the model in order to improve predictions in the future. The general framework of an online learning algorithm is outlined in Algorithm 1.

Algorithm 1: The Online Learning Procedure

```
Initialize the prediction function as w_1 = 0;
for t = 1, 2, ..., T do
    Receive instance: x_t ∈ R^d;
    Predict ŷ_t = f_t(x_t)(= sign(w_t^⊤ x_t) for binary
    classification);
    Receive correct label: y_t ∈ {−1, +1};
    Suffer loss: ℓ_t(w_t), which depends on the difference
    between w_t^⊤ x_t and y_t;
    Update the prediction function w_t to w_{t+1};
end for
```

Online learning algorithms are often much more scalable than traditional batch learning algorithms. Both the learning (model updates) and forecasting are computationally very efficient, making it especially suitable for malicious URL detection tasks with increasingly large amounts of training data (often with millions of instances and millions of features), where batch learning algorithms may suffer due to their expensive retraining and the high memory and computational constraints. Online learning algorithms are often developed with strong theoretical guarantees such that they are able to asymptotically learn the prediction models as good as the batch algorithms under mild assumptions.

Online learning has been actively explored and applied to resolve the malicious URL Detection tasks [28]. In the following, we categorize the existing online learning algorithms roughly into two major categories: (i) First-order online algorithms, and (ii) Second-order online algorithms, and highlight some important concerns for their applications to malicious URL detection.

1) First Order Online Learning: First-order algorithms learn by updating the weight vector $\mathbf{w}$ for classification sequentially by utilizing only the first-order information with training data. We briefly describe some popular first-order online algorithms applied to Malicious URL Detection.

Perceptron [109] is the earliest online learning algorithm. In each iteration, whenever a mistake is made by the prediction model, Perceptron makes an update as follows:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t \tag{6}$$

Online Gradient Descent (OGD) [110] updates the weight vector $\mathbf{w}$ by applying the (Stochastic) Gradient Descent principle only to a single training instance arriving sequentially. Specifically, OGD makes an online update iteratively as:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla \ell(\mathbf{w}_t, \mathbf{x}_t; y_t) \tag{7}$$

where η is a step size parameter, and $\ell(\mathbf{w}_t, \mathbf{x}_t; y_t)$ is some predefined loss function, e.g., Hinge-Loss, Negative Log-Likelihood, Squared Loss, etc.

Passive-Aggressive learning (PA) [111] is an online learning method that trades off two concerns: (i) passiveness: to avoid the new model deviating too much from the existing one, and (ii) aggressiveness: to update the model by correcting the

prediction mistake as much as possible. The optimization of PA learning can be cast as follows:

$$\mathbf{w}_{t+1} \leftarrow \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}_t - \mathbf{w}||^2 \quad \text{subject to } y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq 1 \quad (8)$$

The closed-form solution to the above can be derived as the following update rule:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$$
$$\text{where} \qquad \tau_t = \max\left(\frac{1-y_t(\mathbf{w}_t \cdot \mathbf{x}_t)}{||\mathbf{x}_t||^2}, 0\right) \quad (9)$$

The above model assumes a hard margin exists, that is, data can be linearly separable, which may not be always true, especially when data is noisy. To overcome this limitation, soft-margin PA variants, such as PA-I and PA-II, are often commonly used, which also have closed-form solutions [111].

The above first-order online learning algorithms have been widely applied for malicious URL detection tasks in literature [13], [28], [37], [73], [112], which are efficient, scalable, simple to understand, and easy to implement.

2) Second Order Online Learning: Unlike the first order online learning, second order online learning aims to boost the learning efficacy by exploiting second-order information, e.g., the second order statistics of underlying distributions. For example, they usually assume the weight vector $\mathbf{w}$ follows a Gaussian distribution $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean vector $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. This is particularly useful for malicious URL Detection where data is sparse and high dimensional (due to the bag-of-words or alike representations of lexical features). Below we briefly describe some popular second-order algorithms applied to Malicious URL Detection.

Confidence-Weighted learning (CW) [113] is similar to the PA learning algorithms in terms of passiveness and aggressiveness tradeoff, except that CW exploits the second-order information. In particular, CW learning maintains a different confidence measure for each individual feature, such that weights of lower confidence will be updated more aggressively than those of higher confidence. Specifically, by modeling the weight vector as a Gaussian distribution, CW trades off between (i) passiveness: to avoid the new distribution of the model from deviating too much from the existing one; and (ii) aggressiveness: to update the model by not only correcting the prediction mistake if any, but also improving the classification confidence. More specifically, the CW learning can be cast into the following optimization:

$$(\boldsymbol{\mu}_{t+1}, \Sigma_{t+1}) \leftarrow \underset{\boldsymbol{\mu}, \Sigma}{\operatorname{argmin}} \; D_{KL}(\mathcal{N}(\boldsymbol{\mu}, \Sigma)||\mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)) \quad (10)$$

$$\text{subject to } y_t(\boldsymbol{\mu}, \mathbf{x}_t) \geq \phi^{-1}(\eta)\sqrt{\mathbf{x}_t^\top \Sigma \mathbf{x}_t} \quad (11)$$

Like the PA algorithms, the closed-form solutions for the CW optimization can be derived. CW algorithms have been applied for detecting malicious URLs by [28], [36].

CW online learning and its variants have been explored for malicious URL detection in literature. For example, [76] applied the CW learning for malicious URL detection by improving the efficiency when exploiting the full covariance matrix for high-dimensional features, which uses an approximation technique to accelerate the covariance computation (although it may be still quite slow for very high-dimensional data). Further, Adaptive Regularization of Weights (AROW) [114], an improved CW learning algorithm, for learning with non-separable data, was also used for Malicious URL Detection in [39]. [85] adopted a hybrid online learning technique by combining both CW and PA algorithms. Specifically, CW is used for learning from purely lexical features (e.g., bag of words), and PA is used for learning from descriptive features (e.g., statistical properties of the lexical features). They assume lexical features are more effective at detecting maliciousness, while they could change frequently (short-lived), whereas descriptive properties are more stable and static.

Besides, there are many other kinds of online learning algorithms (both first-order and second-order) in literature [18], which may also be applicable to Malicious URL Detection, but yet to be extensively studied.

3) Cost-Sensitive Online Learning: Unlike a regular binary classification task, Malicious URL Detection often faces the challenges of imbalanced label distribution (i.e., different number of malicious and benign URLs), and also a differential misclassification cost (malware installation is much more sever than a simple spam). Accordingly, the designed learning algorithms have to account for this differential misclassification rate in the optimization problem. There have been several algorithms in literature (for both batch and online settings) which address this issue. An example is Cost-Sensitive Online Learning [115]–[120]. While traditional online learning algorithms simply optimize the classification accuracy or mistake rate (which could be misleading for an extremely imbalanced data set since a trivial algorithm that declares every URL as benign may achieve a very high accuracy), cost-sensitive online learning aims to optimize either one of two cost-sensitive measures: *sum* and *cost*, where *sum* is a weighted combination of specificity and sensitivity, and *cost* is the weighted summation of misclassification costs on positive and negative instances. By defining cost-sensitive loss functions, cost-sensitive online learning algorithms can be derived by applying the similar techniques (e.g., online gradient descent).

4) Online Active Learning: Traditional supervised learning (either batch or online) methods often assume labels of training data can always be obtained by the learners at no cost. This is not realistic in real systems since labeling data can be quite expensive and time-consuming. Online Active Learning aims to develop an online learning algorithm for training a model that queries the label of an incoming unlabeled URL instance only if there is a need (e.g., according to some uncertainty measure) [121]–[123]. Typically, an active learner works in an online learning manner for a real system. For example, [116], [121] proposed a cost-sensitive online active learning (CSOAL) approach for Malicious URL detection, where the online learner decides to query the label on the fly for an incoming unlabeled URL instance, such that the label will be queried only when the classification confidence of the URL instance is low or equivalently there is a high uncertainty for making a correct prediction.

C. Representation Learning

There are a large number and variety of features used for Malicious URL Detection. In particular, the usage of Bag of Words features for many of the feature categories results in millions of features. Moreover, as the number of URLs to be processed increases (which is the case in the real world setting), the number of features keeps growing as well. Learning prediction models using so many features suffers from two main drawbacks:

- Computationally Expensive: Training and test time become significantly high, not only because of the many mathematical operations to be performed, but also because of collecting and preprocessing so many features. In many cases (e.g. using bi-gram and tri-gram features), we obtain so many features that it is practically infeasible to perform any optimization.
- Noisy Models: Malicious URL Detection often exhibits number of features being larger than the number of instances available for training. Optimizing such models may result in overfitting.

To overcome these problems, researchers in both machine learning and cyber-security have proposed representation learning techniques, which in the context of Malicious URL Detection are mostly concentrated in the domain of feature selection techniques, i.e., the optimal subset of the given representation needs to be learnt. Here we discuss two categories of representation learning: feature selection, where the features are evaluated and selected based on their performance, and sparsity regularization, where the feature selection is implicitly done by incorporating it into the objective function.

1) Feature Selection: There are two categories of feature selection where the features are scored on the basis of their performance and accordingly selection. These are *Filter Methods and Wrapper Methods* [124], [125].

Filter methods usually use a statistical measure to evaluate the suitability of a particular feature. Based on this score, a set of features can be selected (and the poor features are filtered out). In most cases, this evaluation is done independently (i.e., independent of other features). Some popular approaches include the Chi squared test (χ^2) [92], [126] and information gain scores [35], [127], [128].

Wrapper methods try to select the best subset of features, by modeling the feature selection as a search problem. Different subsets of features are used to learn a prediction model, and are evaluated based on their performance [129], [130]. [131] use Genetic Algorithms to perform feature selection and divides the features into critical and non-critical. The critical features are used as is, while projection of the non-critical features are used to provide supplementary information, instead of being discarded. [125] adopt a maximum relevance and minimum redundancy criterion to select a robust subset of features.

2) Sparsity Regularization: Due to a large number of features that are collected, in particular, lexical features, feature selection needs to be performed over millions of features. Applying filter and wrapper methods may not be very practical. Feature selection can be induced by appropriately modifying the objective function, i.e., (also called *Embedded methods*) to embed the feature selection into the optimization. Consider the generic optimization problem discussed before:

$$\min_{\mathbf{f}} \sum_{t=1}^{T} \ell(\mathbf{f}(\mathbf{x}_t), y_t) + \lambda \mathcal{R}(\mathbf{w})$$

where the first term aims to optimize the model performance, and the second term is the regularizer. λ is the tradeoff parameter to regulate the effect of regularization. A common technique to induce feature selection (or alternately learn sparse models), is to use the L1-norm regularizer:

$$\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||_1 \tag{12}$$

Unlike other popular regularizers (like L2-norm), which try to reduce model complexity, L1-regularization encourages zero values in the weight vector $\mathbf{w}$, which results in the corresponding features to not be selected for the final prediction model. This approach has been commonly used in conjunction with SVM and Logistic Regression models [73] in the batch setting. From the perspective of Online Setting, Sparse Online Learning [132], [133] has been developed to learn sparse models for high-dimensional sparse data (as exhibited in URL features that are very sparse and in the order of millions). [134], [135] proposed *Online Feature Selection* as a principled way for feature selection in an online learning manner, which aims to bound the number of selected features based on some given budget based on the first order online learning approaches. There is also a second-order Online Feature Selection approach [136].

D. Other Learning Methods

While most of the algorithms used for Malicious URL Detection take the form of binary classification, other problem settings have also been studied. Some of these settings are designed for problems specifically arising in Malicious URL Detection. These include application of unsupervised learning to improve detection, learning similarity functions for detecting URLs, and learning interpretable models by string pattern mining for matching URLs. In the following we briefly discuss these problem settings for application to Malicious URL Detection.

1) Unsupervised Learning: Unsupervised learning is the scenario where the true label of the data is not available during the training phase. These approaches rely on anomaly detection techniques where the anomaly is defined as an abnormal behavior. There are several algorithms in literature that can be used for anomaly detection (e.g. clustering, 1-class SVMs, etc.). Unfortunately, due to the extremely diverse set of URLs, it is hard to determine what is "normal" behavior and what is an anomaly. As a result, such techniques have not become very popular for malicious URL detection. However, some techniques have tried to use unsupervised techniques in conjunction with supervised techniques to improve performance. For example, [94] integrate supervised and unsupervised learning in two stages. In the first stage, the supervised prediction model predicts malicious URLs. For those that are classified as benign, a 1-class SVM is used to look for anomalies. [137] follow another style of integration

of supervised and unsupervised where first k-means clustering is performed, and the cluster ID is used as a feature to train a classifier. [138] proposed the usage of an unsupervised hash-based clustering system, wherein specific clusters were labeled as malicious or benign (based on majority of the training data).

2) Similarity Learning: Similarity learning aims to learn how similar two instances are (or in our case, how similar 2 URLs are). This field helps us identify which specific legitimate URLs are being mimicked by the attackers. For this setting, there is a set of protected URLs, and a set of suspicious URLs which are potentially trying to mimic the protected URLs. The aim is to measure the similarity of the suspicions URLs with the protected URLs, and if the similarity is above a specific threshold, we are able to spot a malicious URL. This setting has been largely addressed by extracting visual features [57], [58], [58]–[60], and computing the similarity between the images of the suspicious and the protected URLs.

Another closely related area is learning using kernel functions (which are essentially notions of similarity, and also allow models to learn nonlinear classifiers [103]. The prediction function takes the following form:

$$\mathbf{f}_t(\mathbf{x}_t) = \sum_{i=1}^{t-1} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_t) \tag{13}$$

where α_i is the coefficient for each instance learnt by some learning algorithm, and κ denotes a kernel function (e.g., a polynomial or Gaussian kernel) which measures the similarity between the two instances. Whenever $\alpha_i \neq 0$, the training instance is often known as a support vector. There has been abundant literature on using kernels both in batch and online settings. The batch setting requires a lot of memory and computational resources, which is often intractable in the real world. Online Learning with kernels [139] follows similar approaches as most other online learning techniques except that the pool of support vectors often will increasingly expand whenever there is a mistake in the online learning process. This will result in a very undesired drawback, i.e., the unbounded support vector size, which not only increases the computational cost but also the need for huge memory space for storing all the support vectors in online learning systems. To address this problem, budget online kernel learning has been extensively studied in online learning. Examples include Randomized Budget Perceptron [140], Forgetron [141], Projectron [142], and Bounded Online Gradient Descent [143]. Some of these techniques were empirically studied by [28] for malicious URL detection, but they did not manage to get satisfactory performance. This is not a surprising result, as any small budget size that would enable scalable computation would miss out on most of the instances to be stored as URLs, and will give a very poor kernel approximation. With the recent development of efficient functional approximation techniques for large-scale online learning with kernels [144], [145], it may be possible to obtain competitive performance. Further, Online Multiple Kernel Learning [146], [147] approaches allow for learning with multi-modality, wherein different feature sets correspond to different modalities.

3) String Pattern Matching: As discussed in the previous sections, lexical features are often obtained in the form of bag-of-words representation, which is often abnormally of high dimensionality and could grow over time. Using such predefined features may not be practical for real-world deployment, and such an approach may cause hindrance to interpreting particular common types of attacks. Further, such techniques cannot identify signatures in the form of substrings. To address these issues, [88] proposed a dynamic string pattern mining approach which borrowed the ideas from efficient searching and substring matching. A similar approach based on Trigrams was also developed by [148]. [31]–[33] also designed an approximate string matching strategy to improve over the exact match required by blacklists. While these methods performed string pattern analysis on the URL string, [54] performed string pattern analysis on JavaScript features.

E. Summary of Machine Learning Algorithms

There are a a wide variety of machine learning algorithms in literature that can be directly used in the context of Malicious URL Detection. Due to potentially a tremendous size of training data (millions of instances and features), there was a need for scalable algorithms, and that is why Online Learning methods have found a lot of success in this domain. Efforts have also been made to exploit the sparsity of the data to improve the algorithmic performance. Lastly, there have been efforts in modifying the problem from a typical binary classification algorithm to address class imbalance and multi-class problems. Using these technologies to build live systems is another challenging task. In the following we discuss real systems to demonstrate how Malicious URL Detection can be used as a service. We categorize the representative references according to the machine learning methods applied in Table III.

V. Malicious URL Detection as a Service

Malicious URL detection using machine learning can have immense real-world applications. However, it is nontrivial to make it work in practice. Several researchers have proposed architectures for building end-to-end malicious URL Detection systems and deployed them for real-world utilities. Many have focused on providing services to Online Social Networks like Twitter, where users share plenty of URLs ([10], [69], [72], [73] etc.)

A. Design Principles

When designing and building a real-world malicious URL detection system using machine learning techniques, we aim to achieve the following desired characteristics and goals:

(i) *High Accuracy*: This is often one of the most important goals to be achieved for any malicious URL detection. Ideally, we want to maximize the detection of all the threats of malicious URLs ("true positives") while minimizing the wrong detection of classifying benign URLs into malicious ("false positives"). Since no system is able to guarantee a perfect detection accuracy, a practical malicious URL detection system often has to trade off between the ratios of false positives and false negatives by setting different levels of detection thresholds according to the application needs.

TABLE III
REPRESENTATIVE REFERENCES OF DIFFERENT TYPES OF MACHINE LEARNING ALGORITHMS USED FOR MALICIOUS URL DETECTION

Methodology	Sub Category	Representative References
Batch Learning	SVM	[10], [12], [35], [40]–[43], [47], [48], [50], [69], [70], [79], [81], [92], [93]
	Logistic Regression	[8], [12], [19], [48], [50], [62], [70], [79]
	Nave Bayes	[19], [48], [50], [71], [97]
	Decision Trees	[10], [12], [19], [22], [41], [42], [44], [48], [70], [71], [80], [97]
	Ensembles and others	[20], [46], [83], [84], [87], [106]
Online Learning	First-order algorithms	[13], [28], [37], [73], [109]–[111]
	Second-order algorithms	[28], [36], [39], [76], [85], [113], [114]
	Cost-Sensitive Online Learning	[115], [116]
	Online Active learning	[116], [121], [123]
Representation Learning	Feature Selection	[35], [92], [124]–[131]
	Sparsity Regularization	[73], [132]–[136]
Other Learning	Similarity Learning	[57], [58], [58]–[60]
	Unsupervised Learning	[94], [137], [138]
	String Pattern Matching	[31]–[33], [54], [88], [148], [149]

(ii) *Fast Detection*: The detection speed is another important concern for a practical malicious URL detection system, particularly for online systems or cybersecurity applications. For example, when deploying the malicious URL detection service in online social networks like Twitter, whenever a user posts any new URL, an ideal system should be able to detect the malicious URL immediately and then block the URL and its related tweets in real time to prevent any threats and harms to public. For some cybersecurity applications, the requirement of detection speed could be more severe, which sometimes needs the detection to be done in milliseconds such that a malicious URL request can be blocked immediately in real time whenever a user clicks on any URLs.

(iii) *High Scalability*: Consider the increasing huge amount of URLs, a real-world malicious URL detection system must be able to scale up for training the models with millions or even billions of training data. In order to achieve the high scalability desire, there are two major kinds of efforts and solutions. The first is to explore more efficient and scalable learning algorithms, e.g., online learning or efficient stochastic optimization algorithms. The second is to build scalable learning systems in distributed computing environments, e.g., using emerging distributed learning frameworks (such as Apache Hadoop or Spark) on the clusters.

(iv) *Strong Adaptation*: A real-world malicious URL detection system has to deal with a variety of practical complexity, including adversarial patterns such as concept drifting where the distribution of malicious URLs may change over time or even change in adversarial way to bypass the detection system, missing values (e.g., features that are unavailable or too expensive to compute in required time cost), increasing number of new features, etc. A real-world malicious URL detection system must have a strong adaptation ability to work effectively and robustly under most circumstances.

(v) *Good Flexibility*: Given the high complexity of malicious URL detection, a real-world malicious URL detection system with machine learning should be designed with good flexibility for easy improvements and extensions. These include the quick update of the predictive models with respect to new training data, being easy to replace the training algorithm and models whenever there is a need, being flexible to be extended for training models to deal with a variety of new attacks and threats, and finally being able to interact with human beings whenever there is a need, e.g., active learning or crowdsourcing for enhancing performance.

B. Design Frameworks

In the following, we discuss some real-world implementations, heuristics and systems that have attempted to make Malicious URL Detection as a service in reality.

[73] designed a framework called *Monarch*, and the idea of providing Malicious URL Detection as a service was floated. Monarch would crawl web-services in real-time and determine whether the URL would be malicious or benign. Differences in malicious URL distribution between Twitter and Spam Emails were observed, and accordingly different models were built for both. Monarch, at the time of development, could process 15-million URLs a day for less than $800 per day. The implementation of this system comprises a URL aggregator which collects URLs from some data streams (e.g., Twitter or Emails). From these URLs, features are collected, which are then processed and converted into sparse features in the feature extractor. Finally, a classifier is trained on the processed data to detect malicious URLs. The collected features include both URL-based features and content-based features. In addition, initial URLs, landing URLs, and Redirects are also extracted. For fast classification training from the perspective of efficiency in memory and algorithmic updates, a linear classifier based on logistic regression with the L1-regularizer (for inducing sparse models) is trained on a distributed computing architecture [150], [151]. They are able to process an entire URL in 5.5 seconds on average, most of which is spent on extracting the features. The prediction is relatively very efficient.

[77] applied machine learning techniques to predict malicious URLs, and subsequently attempted to maintain a blacklist of malicious URLs using these predictions. *Prophiler* [19] is another system, where it recommends that the URL classification to be performed in a two-stage process. The first stage would be by analyzing the light-weight URL features, and quickly filtering out URLs for which the classifier has a high confidence. For the low confidence predictions, a more intensive content based analysis can be performed. *WarningBird* [69] is similar to Monarch in that the primary aim is to

detect suspicious URLs in a Twitter streams except that it uses heuristics to obtain new features, and the classifier used was an SVM model using LIBLINEAR [152] and not trained on a distributed architecture. BINSPECT [20] is another system that was developed to take advantage of ensemble classification, where the final prediction was made on the basis of confidence weighted majority voting.

VI. Practical Issues and Open Problems

Despite many exciting advances over last decade for malicious URL detection using machine learning techniques, there are still many open problems and challenges which are critical and imperative, including but not limited to the following.

(i) *High volume and high velocity*: The real-world URL data is obviously a form of big data with high volume and high velocity. In August 2012 [153] disclosed that Google's search engine found more than 30 trillion unique URLs on the Web, and crawls 20 billion sites a day. It is almost impossible to train a malicious URL detection model on all the world's URL data using machine learning. A clever way of sampling effective URL training data (including both malicious and benign ones) and training them using efficient and scalable machine learning algorithms will be always an open question for researchers in both machine learning and cybersecurity communities.

(ii) *Difficulty in acquiring labels*: Most existing malicious URL detection approaches by machine learning are based on supervised learning techniques, which require labeled training data including both benign and malicious URL data. The labeled data can be obtained by either asking human experts to label or acquiring from blacklists/whitelists (which were often also labeled manually). Unfortunately the scale of such labeled data is tiny as compared to the size of all available URLs on the web. For example, one of the largest public available malicious URL training data sets in academia [28] has only 2.4 million URLs. Thus, there is a large room for open research in resolving the difficulty of acquiring labeled data or learning with limited amount of labeled data. One possible direction is to go beyond purely supervised learning by exploring unsupervised learning or semi-supervised learning, such as active learning in some recent studies [116]. Another possible direction is to explore crowdsourcing techniques [154], [155] by facilitating organizations and individuals to label and share malicious URLs, which however is nothing trivial given the practical concerns of cost, privacy and security threats of sharing malicious URLs. More innovations could be explored in the future.

(iii) *Difficulty in collecting features*: As mentioned in the previous section, collecting features for representing a URL is crucial for applying machine learning techniques. However, it is often not a trivial task for collecting many kinds of features for a URL. In particular, some features could be costly (in terms of time) to collect, e.g., host-based features. Some features might be missing, or noisy, or can not be obtained due to a variety of reasons (e.g., IP/DNS addresses of a URL may vary time to time). In addition, real-world URLs may not always be alive. For example, as observed by [11], many malicious URLs may be short lived, and thus accessing its features (e.g., IP address) may not be possible when it is not alive. Besides, some previous benign URLs may be stopped for services, and then were replaced by some malicious URL (or vice versa). All these challenges post a lot of research and development difficulties for collecting features.

(iv) *Feature Representation*: In addition to high volume and high velocity of URL data, another key challenge is the very high-dimensional features (often in millions or even billion scale). This poses a huge challenge in practice when training a classification model with such very high-dimensional data. Some commonly used learning techniques, such as feature selection, dimension reduction and sparse learning, have been explored, but they are far from solving the challenge effectively. Besides the high dimensionality issue, another more severe challenge is the evolving high dimensional feature space, where the feature space often grows over time when new URLs and new features are added into the training data. This again poses a great challenge for a clever design of new machine learning algorithms which can adapt to the dynamically changing feature spaces.

(v) *Concept drifting and new emerging challenges*: Another challenge is the concept drifting where the distribution of malicious URLs may change over time due to the evolving behaviours of new threats and attacks. This requires machine learning techniques to be able to deal with concept drifting whenever it appears. Besides, another recent challenge is due to the popularity of URL shortening services, which take a long URL as input and produce a short URL as an output. Such URL shortening services potentially offer an excellent way for malicious hackers and criminals to hide their malicious URLs and thus creates a new challenge for automated malicious URL detection systems. Last but not least, it is almost for sure that there will always new types of challenges for malicious URL detection since sophisticated malicious hackers and criminals will always find ways to bypass the cyber security systems. How to make effective learning systems which can quickly detect and adapt themselves for resolving new challenges will be a long term research challenge. An additional issue is to identify vulnerable websites which may become malicious in the future [66]. This is important, as URLs deemed to be benign in the past may get compromised and become malicious in the future. This would also help the site administrators to take steps to address such vulnerabilities.

VII. Concluding Remarks and Future Directions

Malicious URL detection plays a critical role for many cybersecurity applications, and clearly machine learning approaches are a promising direction. In this article, we conducted a comprehensive and systematic survey on Malicious URL Detection using machine learning techniques. In particular, we offered a systematic formulation of Malicious URL detection from a machine learning perspective, and then detailed the discussions of existing studies for malicious URL detection, particularly in the forms of developing new feature representations, and designing new learning algorithms for resolving the malicious URL detection tasks. In this survey, we categorized most, if not all, the existing contributions for

malicious URL detection in literature, and also identified the requirements and challenges for developing Malicious URL Detection as a service for real-world cybersecurity applications.

Finally, we highlighted some practical issues for the application domain and indicated some important open problems for further research investigation. In particular, despite the extensive studies and the tremendous progress achieved in the past few years, automated detection of malicious URLs using machine learning remains a very challenging open problem. Future directions include more effective feature extraction and representation learning (e.g., via deep learning approaches), more effective machine learning algorithms for training the predictive models particularly for dealing with concept drifts (e.g., more effective online learning) and other emerging challenges (e.g., domain adaption when applying a model to a new domain), and finally a smart design of closed-loop system of acquiring labeled data and user feedback (e.g., integrating an online active learning approach in a real system).

REFERENCES

[1] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74–81, 2012.

[2] B. Liang, J. Huang, F. Liu, D. Wang, D. Dong, and Z. Liang, "Malicious web pages detection based on abnormal visibility recognition," in *E-Business and Information System Security, 2009. EBISS'09. International Conference on*. IEEE, 2009, pp. 1–5.

[3] D. R. Patil and J. Patil, "Survey on malicious web pages detection techniques," *International Journal of u-and e-Service, Science and Technology*, vol. 8, no. 5, pp. 195–206, 2015.

[4] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 281–290.

[5] R. Heartfield and G. Loukas, "A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 37, 2015.

[6] L. OpenDNS, "Phishtank: An anti-phishing site," *Online: https://www. phishtank. com*, 2016.

[7] S. Sinha, M. Bailey, and F. Jahanian, "Shades of grey: On the effectiveness of reputation-based "blacklists"," in *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*. IEEE, 2008, pp. 57–64.

[8] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on Recurring malcode*. ACM, 2007, pp. 1–8.

[9] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, "Phi. sh/$ ocial: the phishing landscape through short urls," in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*. ACM, 2011, pp. 92–101.

[10] Y. Alshboul, R. Nepali, and Y. Wang, "Detecting malicious short urls on twitter," 2015.

[11] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi." *LEET*, vol. 8, p. 4, 2008.

[12] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: learning to detect malicious web sites from suspicious urls," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1245–1254.

[13] ——, "Learning to detect malicious urls," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 30, 2011.

[14] S. Purkait, "Phishing counter measures and their effectiveness–literature review," *Information Management & Computer Security*, vol. 20, no. 5, pp. 382–420, 2012.

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

[16] R. K. Nepali and Y. Wang, "You look suspicious!!: Leveraging visible attributes to classify malicious short urls on twitter," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 2016, pp. 2648–2655.

[17] M. Kuyama, Y. Kakizaki, and R. Sasaki, "Method for detecting a malicious domain by using whois and dns features," in *The Third International Conference on Digital Security and Forensics (DigitalSec2016)*, 2016, p. 74.

[18] S. C. Hoi, J. Wang, and P. Zhao, "Libol: A library for online learning algorithms," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 495–499, 2014.

[19] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: a fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 197–206.

[20] B. Eshete, A. Villafiorita, and K. Weldemariam, "Binspect: Holistic analysis and detection of malicious web pages," in *Security and Privacy in Communication Networks*. Springer, 2013, pp. 149–166.

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

[22] C. Seifert, I. Welch, and P. Komisarczuk, "Identification of malicious web pages with static heuristics," in *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*. IEEE, 2008, pp. 91–96.

[23] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "Spyproxy: Execution-based detection of malicious web content." in *USENIX Security*, 2007.

[24] K. Rieck, T. Krueger, and A. Dewald, "Cujo: efficient detection and prevention of drive-by-download attacks," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 31–39.

[25] M. T. Qassrawi and H. Zhang, "Detecting malicious web servers with honeyclients," *Journal of Networks*, vol. 6, no. 1, pp. 145–152, 2011.

[26] B.-I. Kim, C.-T. Im, and H.-C. Jung, "Suspicious malicious web site detection with strength analysis of a javascript obfuscation," *International Journal of Advanced Science and Technology*, vol. 26, pp. 19–32, 2011.

[27] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: De-cloaking internet malware," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 443–457.

[28] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: an application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 681–688.

[29] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, "Detection of malicious web pages using system calls sequences," in *Availability, Reliability, and Security in Information Systems*. Springer, 2014, pp. 226–238.

[30] Y. Tao, "Suspicious url and device detection by log mining," Ph.D. dissertation, Applied Sciences: School of Computing Science, 2014.

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

[32] B. Sun, M. Akiyama, T. Yagi, M. Hatada, and T. Mori, "Autoblg: Automatic url blacklist generator using search space expansion and filters," in *2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2015, pp. 625–631.

[33] S. Bo, M. Akiyama, Y. Takeshi, and M. Hatada, "Automating url blacklist generation with similarity search approach," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 4, pp. 873–882, 2016.

[34] M. Felegyhazi, C. Kreibich, and V. Paxson, "On the potential of proactive domain blacklisting." *LEET*, vol. 10, pp. 6–6, 2010.

[35] P. Kolari, T. Finin, and A. Joshi, "Svms for the blogosphere: Blog identification and splog detection," in *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, 2006, pp. 92–99.

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

[37] W. Zhang, Y.-X. Ding, Y. Tang, and B. Zhao, "Malicious web page detection based on on-line learning algorithm," in *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, vol. 4. IEEE, 2011, pp. 1914–1919.

[38] S. Yadav, A. K. K. Reddy, A. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 48–61.

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

[40] H.-K. Pao, Y.-L. Chou, and Y.-J. Lee, "Malicious url detection based on kolmogorov complexity estimation," in *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2012, pp. 380–387.

[41] S. Marchal, J. François, R. State, and T. Engel, "Phishscore: Hacking phishers' minds," in *Network and Service Management (CNSM), 2014 10th International Conference on*. IEEE, 2014, pp. 46–54.

[42] ——, "Phishstorm: Detecting phishing with streaming analytics," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 4, pp. 458–471, 2014.

[43] W. Chu, B. B. Zhu, F. Xue, X. Guan, and Z. Cai, "Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing urls," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1990–1994.

[44] D. Chiba, K. Tobe, T. Mori, and S. Goto, "Detecting malicious websites by learning ip address features," in *Applications and the Internet (SAINT), 2012 IEEE/IPSJ 12th International Symposium on*. IEEE, 2012, pp. 29–39.

[45] T. Holz, C. Gorecki, F. Freiling, and K. Rieck, "Detection and mitigation of fast-flux service networks," in *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS08)*, 2008.

[46] H. Choi, B. B. Zhu, and H. Lee, "Detecting malicious web links and identifying their attack types," in *Proceedings of the 2nd USENIX conference on Web application development*. USENIX Association, 2011, pp. 11–11.

[47] E. Sorio, A. Bartoli, and E. Medvet, "Detection of hidden fraudulent urls within trusted sites using lexical features," in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. IEEE, 2013, pp. 242–247.

[48] L. Xu, Z. Zhan, S. Xu, and K. Ye, "Cross-layer detection of malicious websites," in *Proceedings of the third ACM conference on Data and application security and privacy*. ACM, 2013, pp. 141–152.

[49] D. Chiba, T. Yagi, M. Akiyama, T. Shibahara, T. Yada, T. Mori, and S. Goto, "Domainprofiler: Discovering domain names abused in future," in *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 491–502.

[50] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious web content detection by machine learning," *Expert Systems with Applications*, vol. 37, no. 1, pp. 55–60, 2010.

[51] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 639–648.

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

[53] K. Borgolte, C. Kruegel, and G. Vigna, "Delta: automatic identification of unknown web-based infection campaigns," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 109–120.

[54] Y. Choi, T. Kim, and S. Choi, "Automatic detection for javascript obfuscation attacks in web pages through string pattern analysis," *International Journal of Security and Its Applications*, vol. 4, no. 2, pp. 13–26, 2010.

[55] Y. Wang, W.-d. Cai, and P.-c. Wei, "A deep learning approach for detecting malicious javascript code," *Security and Communication Networks*, 2016.

[56] A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd)," *IEEE transactions on dependable and secure computing*, vol. 3, no. 4, pp. 301–311, 2006.

[57] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," in *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005, pp. 1060–1061.

[58] W. Liu, X. Deng, G. Huang, and A. Y. Fu, "An antiphishing strategy based on visual similarity assessment," *IEEE Internet Computing*, vol. 10, no. 2, p. 58, 2006.

[59] K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen, "Fighting phishing with discriminative keypoint features," *IEEE Internet Computing*, vol. 13, no. 3, pp. 56–63, 2009.

[60] S. Afroz and R. Greenstadt, "Phishzoo: Detecting phishing websites by looking at them," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 368–375.

[61] M. Dunlop, S. Groat, and D. Shelly, "Goldphish: Using images for content-based phishing analysis," in *Internet Monitoring and Protection (ICIMP), 2010 Fifth Intl. Conference on*, 2010, pp. 123–128.

[62] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a bayesian approach," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1532–1546, 2011.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[65] G. Xiang and J. I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 571–580.

[66] K. Soska and N. Christin, "Automatically detecting vulnerable websites before they turn malicious," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 625–640.

[67] F. Maggi, A. Frossi, S. Zanero, G. Stringhini, B. Stone-Gross, C. Kruegel, and G. Vigna, "Two years of short urls internet measurement: security threats and countermeasures," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 861–872.

[68] N. Gupta, A. Aggarwal, and P. Kumaraguru, "bit. ly/malicious: Deep dive into short url based e-crime detection," in *Electronic Crime Research (eCrime), 2014 APWG Symposium on*. IEEE, 2014, pp. 14–24.

[69] S. Lee and J. Kim, "Warningbird: Detecting suspicious urls in twitter stream." in *NDSS*, 2012.

[70] D. Wang, S. B. Navathe, L. Liu, D. Irani, A. Tamersoy, and C. Pu, "Click traffic analysis of short url spam on twitter," in *9th Intl Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*. IEEE, 2013, pp. 250–259.

[71] J. Cao, Q. Li, Y. Ji, Y. He, and D. Guo, "Detection of forwarding-based malicious urls in online social networks," *International Journal of Parallel Programming*, pp. 1–18, 2014.

[72] C. Cao and J. Caverlee, "Detecting spam urls in social media via behavioral analysis," in *Advances in Information Retrieval*. Springer, 2015, pp. 703–714.

[73] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 447–462.

[74] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 133–144.

[75] Y. He, Z. Zhong, S. Krasser, and Y. Tang, "Mining dns for malicious domain registrations," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*. IEEE, 2010, pp. 1–6.

[76] J. Ma, A. Kulesza, M. Dredze, K. Crammer, L. K. Saul, and F. Pereira, "Exploiting feature covariance in high-dimensional online learning," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 493–500.

[77] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phishing pages." in *NDSS*, vol. 10, 2010.

[78] S. Gastellier-Prevost, G. G. Granadillo, and M. Laurent, "Decisive heuristics to differentiate legitimate from phishing sites," in *Network and Information Systems Security (SAR-SSI), 2011 Conference on*. IEEE, 2011, pp. 1–9.

[79] S. N. Bannur, L. K. Saul, and S. Savage, "Judging a site by its content: learning the textual, structural, and visual features of malicious web pages," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*. ACM, 2011, pp. 1–10.

[80] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis." in *NDSS*, 2011.

[81] H. Huang, L. Qian, and Y. Wang, "A svm-based technique to detect phishing urls," *Information Technology Journal*, vol. 11, no. 7, p. 921, 2012.

[82] M.-E. Maurer and L. Höfer, "Sophisticated phishers make more spelling mistakes: using url similarity against phishing," in *Cyberspace Safety and Security*. Springer, 2012, pp. 414–426.

[83] B. Eshete, A. Villafiorita, K. Weldemariam, and M. Zulkernine, "Einspect: Evolution-guided analysis and detection of malicious web pages," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. IEEE, 2013, pp. 375–380.

[84] B. Eshete, "Effective analysis, characterization, and detection of malicious web pages," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 355–360.

[85] M.-S. Lin, C.-Y. Chiu, Y.-J. Lee, and H.-K. Pao, "Malicious url filteringa big data application," in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 589–596.

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

[87] K.-W. Su, K.-P. Wu, H.-M. Lee, and T.-E. Wei, "Suspicious url filtering based on logistic regression with multi-view analysis," in *Information Security (Asia JCIS), 2013 Eighth Asia Joint Conference on*. IEEE, 2013, pp. 77–84.

[88] D. Huang, K. Xu, and J. Pei, "Malicious url detection by dynamically mining patterns without pre-defined elements," *World Wide Web*, vol. 17, no. 6, pp. 1375–1394, 2014.

[89] W. Wang and K. Shirley, "Breaking bad: Detecting malicious domains using word segmentation," *arXiv preprint arXiv:1506.04111*, 2015.

[90] S. Marchal, K. Saari, N. Singh, and N. Asokan, "Know your phish: Novel techniques for detecting phishing sites and their targets," *arXiv preprint arXiv:1510.06501*, 2015.

[91] Y. Sato, Y. Nakamura, H. Inamura, and O. Takahashi, "A proposal of malicious urls detection based on features generated by exploit kits," 2016.

[92] Y. Pan and X. Ding, "Anomaly based web phishing page detection," in *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*. IEEE, 2006, pp. 381–392.

[93] M. He, S.-J. Horng, P. Fan, M. K. Khan, R.-S. Run, J.-L. Lai, R.-J. Chen, and A. Sutanto, "An efficient phishing webpage detector," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 018–12 027, 2011.

[94] S. Yoo, S. Kim, A. Choudhary, O. Roy, and T. Tuithung, "Two-phase malicious web page detection scheme using misuse and anomaly detection," *International Journal of Reliable Information and Assurance*, vol. 2, no. 1, 2014.

[95] G. Canfora and C. A. Visaggio, "A set of features to detect web security threats," *Journal of Computer Virology and Hacking Techniques*, pp. 1–19, 2016.

[96] M. Hara, A. Yamada, and Y. Miyake, "Visual similarity-based phishing detection without victim site information," in *Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on*. IEEE, 2009, pp. 30–36.

[97] A. Aggarwal, A. Rajadesingan, and P. Kumaraguru, "Phishari: automatic realtime phishing detection on twitter," in *eCrime Researchers Summit (eCrime), 2012*. IEEE, 2012, pp. 1–12.

[98] L. Wenyin, N. Fang, X. Quan, B. Qiu, and G. Liu, "Discovering phishing target based on semantic link network," *Future Generation Computer Systems*, vol. 26, no. 3, pp. 381–388, 2010.

[99] J. H. Huh and H. Kim, "Phishing detection with popular search engines: Simple and effective," in *International Symposium on Foundations and Practice of Security*. Springer, 2011, pp. 194–207.

[100] A. N. V. Sunil and A. Sardana, "A pagerank based detection technique for phishing web sites," in *Computers & Informatics (ISCI), 2012 IEEE Symposium on*. IEEE, 2012, pp. 58–63.

[101] D. Gugelmann, M. Happe, B. Ager, and V. Lenders, "An automated approach for complementing ad blockers blacklists," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 282–298, 2015.

[102] 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 *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 5186–5194.

[103] A. J. Smola and B. Schölkopf, *Learning with kernels*. Citeseer, 1998.

[104] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, 2014.

[105] W. Zhang, Q. Jiang, L. Chen, and C. Li, "Two-stage elm for phishing web pages detection using hybrid features," *World Wide Web*, pp. 1–17, 2016.

[106] A. Astorino, A. Chiarello, M. Gaudioso, and A. Piccolo, "Malicious url detection via spherical classification," *Neural Computing and Applications*, pp. 1–7, 2016.

[107] V. Ramanathan and H. Wechsler, "Phishing website detection using latent dirichlet allocation and adaboost," in *Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on*. IEEE, 2012, pp. 102–107.

[108] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006.

[109] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[110] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*. School of Computer Science, Carnegie Mellon University, 2003.

[111] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *JMLR*, vol. 7, pp. 551–585, Dec. 2006.

[112] M. N. Feroz and S. Mengel, "Examination of data, rule generation and detection of phishing urls using online logistic regression," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 241–250.

[113] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 264–271.

[114] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," in *Advances in neural information processing systems*, 2009, pp. 414–422.

[115] J. Wang, P. Zhao, and S. C. H. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, 2014.

[116] P. Zhao and S. C. Hoi, "Cost-sensitive online active learning with application to malicious url detection," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 919–927.

[117] ——, "Cost-sensitive double updating online learning and its application to online anomaly detection," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 207–215.

[118] P. Zhao, F. Zhuang, M. Wu, X.-L. Li, and S. C. Hoi, "Cost-sensitive online classification with adaptive regularization and its applications," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 649–658.

[119] D. Sahoo, P. Zhao, and S. C. Hoi, "Cost-sensitive online multiple kernel classification," in *Proceedings of The 8th Asian Conference on Machine Learning*, 2016, pp. 65–80.

[120] C. K. Maurya, D. Toshniwal, and G. V. Venkoparao, "Online sparse class imbalance learning on big data," *Neurocomputing*, vol. 216, pp. 250–260, 2016.

[121] J. Lu, P. Zhao, and S. C. Hoi, "Online passive-aggressive active learning," *Machine Learning*, vol. 103, no. 2, pp. 141–183, 2016.

[122] S. Hao, S. C. Hoi, P. Zhao, C. Miao, J. Lu, and C. Zhang, "Soal: Second-order online active learning," in *IEEE International Conference on Data Mining (ICDM)*, 2016.

[123] D. Sculley, "Online active learning methods for fast label-efficient spam filtering." in *CEAS*, vol. 7, 2007, p. 143.

[124] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[125] H. Zuhair, A. Selamat, and M. Salleh, "Feature selection for phishing detection: a review of research," *International Journal of Intelligent Systems Technologies and Applications*, vol. 15, no. 2, pp. 147–162, 2016.

[126] D. Zhang, Z. Yan, H. Jiang, and T. Kim, "A domain-feature enhanced classification model for the detection of chinese phishing e-business websites," *Information & Management*, vol. 51, no. 7, pp. 845–853, 2014.

[127] L. Ma, B. Ofoghi, P. Watters, and S. Brown, "Detecting phishing emails using hybrid features," in *Ubiquitous, Autonomic and Trusted Computing, 2009. UIC-ATC'09. Symposia and Workshops on*. IEEE, 2009, pp. 493–497.

[128] F. Toolan and J. Carthy, "Feature selection for spam and phishing detection," in *eCrime Researchers Summit (eCrime), 2010*. IEEE, 2010, pp. 1–12.

[129] M. Khonji, A. Jones, and Y. Iraqi, "A study of feature subset evaluators and feature subset searching methods for phishing classification," in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*. ACM, 2011, pp. 135–144.

[130] R. B. Basnet, A. H. Sung, and Q. Liu, "Feature selection for improved phishing detection," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2012, pp. 252–261.

[131] W. Zhang, R. Huan, and Q. Jiang, "Application of feature engineering for phishing detection," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 4, pp. 1062–1070, 2016.

[132] D. Wang, P. Wu, P. Zhao, Y. Wu, C. Miao, and S. C. Hoi, "High-dimensional data stream classification via sparse online learning," in *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1007–1012.

[133] Y. Wu, S. C. Hoi, C. Liu, J. Lu, D. Sahoo, and N. Yu, "Sol: A library for scalable online learning algorithms," *arXiv preprint arXiv:1610.09083*, 2016.

[134] S. C. Hoi, J. Wang, P. Zhao, and R. Jin, "Online feature selection for mining big data," in *Proceedings of 1st intl. Workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications*. ACM, 2012, pp. 93–100.

[135] J. Wang, P. Zhao, S. C. Hoi, and R. Jin, "Online feature selection and its applications," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 26, no. 3, pp. 698–710, 2014.

[136] Y. Wu, S. C. Hoi, and T. Mei, "Massive-scale online feature selection for sparse ultra-high dimensional data," *arXiv preprint arXiv:1409.7794*, 2014.

[137] M. N. Feroz and S. Mengel, "Phishing url detection using url ranking," in *Big Data (BigData Congress), 2015 IEEE International Congress on*. IEEE, 2015, pp. 635–638.

[138] A. S. Popescu, D. B. Prelipcean, and D. T. Gavrilut, "A study on techniques for proactively identifying malicious urls," in *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2015, pp. 204–211.

[139] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *Signal Processing, IEEE Transactions on*, vol. 52, no. 8, pp. 2165–2176, 2004.

[140] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Machine Learning*, vol. 69, no. 2-3, pp. 143–167, 2007.

[141] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a budget," *SIAM Journal on Computing*, vol. 37, no. 5, pp. 1342–1372, 2008.

[142] F. Orabona, J. Keshet, and B. Caputo, "The projectron: a bounded kernel-based perceptron," in *ICML*. ACM, 2008, pp. 720–727.

[143] P. Zhao, J. Wang, P. Wu, R. Jin, and S. C. Hoi, "Fast bounded online gradient descent algorithms for scalable kernel-based online learning," *arXiv preprint arXiv:1206.4633*, 2012.

[144] J. Wang, S. C. Hoi, P. Zhao, J. Zhuang, and Z.-y. Liu, "Large scale online kernel classification," in *Proceedings of Twenty-Third international joint conference on Artificial Intelligence*, 2013, pp. 1750–1756.

[145] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *JMLR*, 2016.

[146] S. C. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Machine Learning*, vol. 90, no. 2, pp. 289–316, 2013.

[147] D. Sahoo, S. C. Hoi, and B. Li, "Online multiple kernel regression," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 293–302.

[148] C. Xiong, P. Li, P. Zhang, Q. Liu, and J. Tan, "Mird: Trigram-based malicious url detection implanted with random domain name recognition," in *Applications and Techniques in Information Security*. Springer, 2015, pp. 303–314.

[149] B. Wardman, G. Shukla, and G. Warner, "Identifying vulnerable websites by analysis of common strings in phishing urls," in *eCrime Researchers Summit, 2009. eCRIME'09*. IEEE, 2009, pp. 1–13.

[150] Y. Singer and J. C. Duchi, "Efficient learning using forward-backward splitting," in *Advances in Neural Information Processing Systems*, 2009, pp. 495–503.

[151] R. McDonald, K. Hall, and G. Mann, "Distributed training strategies for the structured perceptron," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 456–464.

[152] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[153] D. Sullivan, "Google: 100 billion searches per month, search to integrate gmail, launching enhanced search app for ios," *Search Engine Land. http://searchengineland. com/google-search-press-129925*, 2012.

[154] J. Hu, H. Gao, Z. Li, and Y. Chen, "Poster: Cud: crowdsourcing for url spam detection," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 785–788.

[155] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011.

Doyen Sahoo Doyen Sahoo is a PhD Candidate in School of Information Systems, Singapore Management University. He is supervised by Dr. Steven Hoi. His primary research topic is Online Learning with nonlinear models. He works on theoretical aspects of machine learning with focus on Online Learning, Deep Learning and Multiple Kernel Learning. He also works on applications of machine learning to portfolio optimization, and cyber-security. Prior to starting PhD, Doyen completed his B.Eng. in Computer Science from Nanyang Technological University.

Chenghao Liu Chenghao Liu received his B.Sc. degree from Computer Science and Technology College, Zhejiang University, Hangzhou, China in 2011. He is currently pursuing the Ph.D. degree from the College of Computer Science, Zhejiang University, Hangzhou, China. His current research interests include machine learning and data mining.

Steven C.H. Hoi Dr. Steven Hoi is an Associate Professor in the School of Information Systems (SIS), Singapore Management University (SMU), Singapore. Prior to joining SMU, he was a tenured Associate Professor at the School of Computer Engineering of the Nanyang Technological University (NTU), Singapore. He received his Bachelor degree from Tsinghua University, and his Master and Ph.D degrees from the Chinese University of Hong Kong. His research interests include large-scale machine learning (online learning and deep learning) with application to tackling big data analytics challenges across a wide range of real-world applications, including multimedia retrieval, social media, web search and information retrieval, computer vision and pattern recognition, computational finance, cyber security, mobile and software data mining, etc. He has published over 150 papers in premier conferences and journals, and served as an organizer, area chair, senior PC, TPC member, editors, and referee for many top conferences and premier journals. He is the recipient of the Lee Kong Chian Fellowship Award due to his research excellence. Currently he is the Editor in Chief of Neurocomputing journal.