Machine Learning Features for Malicious URL Filtering – The Survey

Arunita Das
Global Data And Analytics Platform
Walmart Labs
Bengaluru, India
arunita.das@walmart.com

Eshan Jain
Global Data And Analytics Platform
Walmart Labs
Bengaluru, India
eshan.jain@walmart.com

Smaranya Dey
Global Data And Analytics Platform
Walmart Labs
Bengaluru, India
smaranya.dey@walmart.com

Abstract – **Malicious URL is the URL created for harmful purposes which contains spam, phishing, misleading applications like fake antiviruses or fake codecs. The use of this kind of URLs might lead to monetary loss, theft of sensitive information such as personal details or corporate data, disruption of operations, unauthorized access to system resources etc. Often these websites are built to look like a genuine website to deceive the users in installing malicious content in their systems. As per NetCraft January 2018 web server survey, there are 1.8 billion sites across 213 million unique domain names. According to Symantec Internet Security Threat Report 2018, 1 in 13 web requests lead to malware which is up 3% from 2016. Sudden rise of cyber-attacks in recent years makes this problem indispensable for both private and public organizations.**

The primary objective of the paper is to provide a near exhaustive set of meaningful features that can help professionals and practitioners to facilitate their own research and practical applications on malicious URL filtering. These features are systematically classified and described in keyword-based features, lexical features, content-based features (HTML and JavaScript), IP Address Properties based features, web-rank and score-based features. This paper also briefly discusses on how URL filtering techniques have evolved in the past. The paper talks about traditional techniques like blacklisting URLs, heuristic approaches while also highlighting the shortcomings of these approaches. We then touch upon newer machine learning based techniques like cosine similarity-based URL classification, Support Vector Machines and Neural Network based models.

Keywords - **Cyber Security, URL filtering, machine learning, feature engineering, algorithms, cosine similarity, support vector machines, neural network**

I. INTRODUCTION

Earlier internet was mostly used for searching information but today it has touched and impacted variety of domains like online banking, ecommerce, social networking. The globalization of the Internet has occurred faster than anyone could have imagined. It was not until March 2016 that the 1 billion websites mark was broken. Today the number has reached 1.8 billion sites across 213 million unique domain names [1].

With the increasing importance of internet usage, the events of cyber-attacks, phishing, spamming has also increased tremendously. According to Symantec Internet security threat report 2018 [2], one in thirteen web requests lead to malware which is up 3% from 2016. With Internet of things and other distributed applications, a single threat or attack could lead to a chain reaction of system failures, data loss or monetary loss.

Some of the most common types of attacks seen today include a) Malware: It's a harmful software such as viruses and ransomwares. Once it gets foothold in one's machine it can take form of executable code, script or active content which can create all sorts of havoc from taking control of the attacked machine to monitoring its action and keystrokes. It can also send confidential data from attacked machine or network to attacker's home base, b) Phishing: It's a creative tactic used by cyber attackers to compel the user to download and install malicious content on their machine. An attacker may send this content using a legitimate looking email or a website. Since phishing attacks leverage human curiosity and impulses, these are hard to prevent, c) SQL Injection Attack: Many of the servers store critical data for websites and services and use SQL to manage the data in their databases. SQL Injection Attack works by exploiting any one of the known SQL vulnerabilities that allow SQL code to run malicious code. For example, an attacker knowing the vulnerability, may inject a SQL code in website's search box that would force the site's SQL server to dump all of its stored usernames and passwords for the site, d) Denial-of-Service: During denial-of-service (DoS) attack, a website is flooded with more traffic than it was built to handle which overloads the website's server. It almost becomes impossible for website to serve its content to unexpected high number of visitors trying to access it. These DoS attacks are in many instances performed by multiple computers at the same time. This scenario of attack is known as a Distributed Denial-of-Service Attack (DDoS). Identifying source of DDoS attack is even more difficult as many different IP addresses around the world simultaneously attack the network, e) Session Hijacking and Man-in-the-Middle Attacks: While a user is browsing on internet or when user is logged into a website with username and password, a unique session id is created which should stay private between the two parties. In Session Hijacking an attacker hijacks the session by capturing the session ID and posing as the computer making a request, allowing them to log in as an unsuspecting user and gain access to unauthorized information on the web server. An attacker can also opt to hijack the session to insert themselves between the requesting computer and the remote server, pretending to be the other party in the session. This allows them to intercept information in both directions and is commonly called a man-in-the-middle attack.

Few prominent cyber-attacks in the past a) WannaCry: It was a ransomware attack that spread rapidly in May of 2017. It took over the infected machines and encrypted all the contents of their hard drives, then demanded a payment in Bitcoin in order to decrypt them [3], b) Equifax: The massive credit rating agency announced in July of 2017 that "criminals exploited a U.S. website application vulnerability to gain access to certain files," getting personal information for

almost 150 million people[4]. c) Yahoo: Yahoo's email system hack actually happened way back in 2013 — but the severity of it, with all 3 billion Yahoo email addresses affected, only became clear in October 2017. Stolen information included passwords and backup email addresses, encrypted using outdated, easy-to-crack techniques, which is the sort of information attackers can use to breach other accounts[5]. d) GitHub: On February 28, 2018, the version control hosting service GitHub was hit with a massive denial of service attack, with 1.35 TB per second of traffic hitting the popular site[6]. e) Target: Hackers stole data from up to 40 million credit and debit cards of shoppers who visited its stores during the first three weeks of the holiday season in the second-largest such breach reported by a U.S. retailer[7][8].

With ever increasing use of internet, supported devices, IOT and its combined users it has become crucial to have a robust system in place to prevent cyber-attacks. One crucial aspect of preventing cyber-attacks is to have an efficient and accurate URL Filtering system in place for both public and private organizations as well as personal users. Users spend increasing time on the web on various activities like surfing favorite sites, clicking on email links or utilizing web-based SaaS applications for both personal and business use. This kind of non-monitored web activity exposes organizations as well as individuals to a range of security and business risks, such as propagation of threats including malwares and ransomwares, data loss and potential lack of compliance. Web-filtering systems are either client or server based. A client-based system performs web content filtering solely on the computer where it is installed, without consulting remote servers about the nature of the web content that a user tries to access. A server-based system provides filtering to computers on the local area network where it is installed. It screens outgoing Web requests, analyzes incoming Web pages to determine their content type, and blocks inappropriate material from reaching the client's Web browser [9]. Many URL Filtering systems are commercially available in the market. However, the techniques used by these systems are not accurate enough and do not adapt well to ever changing web attacks. Hence, there is a constant need and opportunity to build a better URL-Filtering system.

In this paper we throw light on how URL filtering techniques have evolved from traditional techniques like blacklisting URLs, heuristic approaches to newer machine learning based techniques like cosine similarity-based URL classification, Support Vector Machines and Neural Network based models. The primary focus of this paper is to provide a near exhaustive set of informative features that can help professionals and practitioners to facilitate their own research and practical applications on URL filtering.

II. MACHINE LEARNING FEATURES

The data that a machine learning expert use and the way it is used, will likely define the success of predictive modeling problem. Data and the framing of a machine learning problem is point of biggest leverage on one's problem. The term data in machine learning context has two dimensions to it, first the depth that is the number of observations you have. One should perform sensitivity analysis to identify how much (or little) data is needed. Second is spread, that is the measurable property or characteristic of different patterns and phenomena being observed across observations. These numeric and sometimes string coded information are called Features. One should find relevant information to create variety of features and test each of them. Without testing these features in a model one won't know what variables will be helpful in your predictive modeling problem. Based on extraction process, features could be further classified as static and dynamic. Static features are those which can be obtained directly from the URL information like lexical and host-based features, whereas to get the dynamic features, one has to crawl and process the web content.

A. Blacklist Features

Blacklist process is easy to use and implement, though it is difficult to maintain an exhaustive list of blacklists which results to significant number of false positive cases. Anyway, if used in collaboration with other features, blacklist features can lead to significant improvement in result of URL filtering [14].

- Blacklist sources- Phishtank, operated by OpenDNS is a blacklist of phishing URLs [15] which is free to use. Other sources could be the list provided by Google, Cisco, Yahoo, Alexa
- To avoid detection via blacklisting, many attackers modify the original URL a little bit. The problem can be avoided by extending the blacklist. This can be achieved by deriving new URLs based on the following five heuristics [16]:
    - Replacing top-level domains
    - IP address equivalence
    - Directory structure similarity
    - Query string substitution
    - Brand name equivalence

B. Lexical Features

These features can be extracted from the URL string itself. URL is Uniform Resource Locator which is the global address of the documents and other resources in the world wide web. There are two main segments of a URL: (a) the protocol: indicates which protocol to use, http:// or https://, (b) the resource name: specifies the IP address or the domain name where the resource is located.

- Length of the URL
- Length of each of the components of the URL- It considers taking the length of hostname, top-level domain name, primary domain name etc. separately.
- Number of special characters present in the URL
- Bag of words model- Based on all possible words, could be present in an URL string, a dictionary can be created, and each word can be regarded as feature. If the word is present in the URL then the value of the feature can be 1, else 0.
- Distinction of tokens- This approach talks about the same bag of words model but separate dictionary for hostname, path, top-level-domain, primary domain. 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 word "com" in the top-level domain compared to the other parts of the URL.

- Bi-gram features- The presence of set of two words along with the single words in the URL is considered to be a feature [17].
- N-gram features- It is a feature selection scheme based on relative entropy to reduce dimensionality where n > 2 [18].
- Analyze character level strings to obtain features- Hackers can generate new URLs algorithmically, detection of which becomes difficult using only bag of words. The idea is algorithmically generated URLs and human generated URLs have different alpha-numeric distribution [19]. The number of features, thus obtained, are small as the number of characters is small. A few methods like KL-divergence, Jaccard coefficient and edit-distance using unigram and bigram distribution of characters.
- Directory related features- length of directory, number of subdirectory tokens can be considered as features for this category [20].
- File name features- length of the filenames, number of delimiters [20].
- Argument features- length of the argument, number of variables [20].

C. Web Content Based Features

Instead of using bag of words for each document, web content and structure analysis suggests representing each web page by a limited number of content and link features, which reduces the dimensionality of the classifier. The relevance and quality of a webpage can be reflected in the following aspects [21]: 1) the content of the page itself, 2) the content of the neighboring documents of the page, 3) link information of the page.

1) Page content- All the terms extracted from the title and the body of a webpage, say p. These terms will be compared with the domain lexicon [22]. Hence two feature scores can be determined from here:

- Title(p): number of the terms present in the title of the webpage p, found in the domain lexicon
- TFIDF(p): sum of TFIDF scores of the terms in the webpage p, found in the domain lexicon

2) Page content of the neighbors- Three types of neighbors for a webpage p, are considered- (a) incoming: Suppose a, b are two webpages which have hyperlink pointing to p. Then, a and b are called incoming (parent) neighbors for p, (b) outgoing: Suppose, e, f are the webpages mentioned in p as hyperlinks. Then, e and f are outgoing neighbors for p, (c) siblings: Sibling pages are those pages that are pointed by any of the incoming neighbors of p. Suppose, webpage a points to webpage p and h as hyperlinks. Then p and h are known to be siblings. Two scores of the neighboring documents are calculated similar to those created in the previous aspect:

- InTitle(p): average of number of terms in the title of webpage a found in domain lexicon, for all a which are incoming neighbors of p
- InTFIDF(p): average of the sum of TFIDF of the terms in webpage a found in domain lexicon, for all incoming a of p
- OutTitle(p): average of number of terms in the title of webpage e found in domain lexicon, for all e. which are outgoing neighbors of p
- OutTFIDF(p): average of the sum of TFIDF of the terms in webpage e found in domain lexicon, for all outgoing neighbors e of p
- SiblingTitle(p): average of the number of terms in webpage h found in domain lexicon, for all h which are siblings of p
- SiblingTFIDF(p): average of the sum of TFIDF of the terms in the webpage h found in domain lexicon, for all h which are siblings of p

3) Link Analysis- Web link structure is useful to understand the quality of a webpage. The webpage a has a link pointing to another webpage b means, the author of a believes b is a kind of webpage similar to a. Usually, the higher the number of in-links, the better a page is considered to be. Several methods have been developed in order to determine the relevance and importance of the page. Pagerank and HITS algorithms are the two most widely used.

Pagerank algorithm [23] is used by google search to rank the websites in their search engine results, which is a recursively defined result with the underlying assumption that, a page becomes important if other important pages are linked with it. Suppose, a random surfer on the web, is following links from page to page. The probability that the random surfer will land on a particular page is the pagerank of that page. The behavior of the random surfer is an example of Markov process, which depends only on the current state of a system and not on its history. Pagerank of a page p can be determined as,

$$Pagerank(p) = (1-d) + d \sum_{all\ q\ linking\ to\ p} \left(\frac{Pagerank(q)}{c(q)}\right)$$

Where d: the damping factor between 0 and 1. The idea is at any moment, the random surfer can stop searching and, c(q): number of outgoing links in q.

The pagerank score of each webpage has to be calculated iteratively which makes it computationally complex.

HITS (Hyperlink-Induced Topic Search) algorithm was proposed by Jon M. Kleinberg [24] is another link analysis algorithm which rates webpages. According to this algorithm, there are high quality pages related to any topic which are known as authority pages and there are a few pages which are not as rich in information as the authority pages are, but they point to other authority pages. These are called hub pages. A page is represented as a good authority page if it is pointed by many different hubs and a good representation of a hub page is that points to other authority pages.

On this idea, hub score and authority score for each webpage can be formulated as below:

$$AuthorityScore(p) = \sum_{all\ q\ linking\ to\ p} (HubScore(q))$$

$$HubScore(p) = \sum_{all\ r\ linking\ to\ p} (AuthorityScore(r))$$

To incorporate the above ideas in the link analysis, a few scores for a webpage say, p, could be thought of which are as follows:

- Pagerank(p): pagerank score of p
- Hub(p): Hub score of p calculated by the HITS algorithm
- Authority(p): Authority score of p calculated by the HITS algorithm
- Inlinks(p): Number of incoming links pointing to p
- Outlinks(p): Number of outgoing links from p
- Anchor(p): Number of terms in the anchor texts in a page p found in the domain lexicon

D. HTML Content Features

HTML Content based features are extracted from the HTML object the URL is pointing. These features could be very useful in determining presence of malicious contents but could be little difficult to generate these features on real time. Following table summarizes the different HTML content-based features that could be generated from the HTML content.

TABLE I: HTML Content Based Features

S.No	Feature Description
1	Length Of the document
2	Average Length of the words
3	Word Count
4	Distinct Word Count
5	Word Count in a line
6	Number of null characters
7	Usage of string concatenation
8	Unsymmetrical HTML tags
9	Link to remote source of scripts and invisible objects
10	Number of hyperlinks
11	Number of elements with a small area
12	Number of elements with suspicious content
13	Number of out of place elements
14	Presence of double documents

- Length of the document- Given an URL and the page pointed by the url, this feature calculates the length of the document for the page. Malicious web pages may have very different distribution of document lengths as compared to benign web pages.
- Length of words- Given the content of the web page, summary statistics of word length could be computed. Average word length, median word length and 3rd quartile value of word length could serve as important features. Certain obfuscation techniques that are used in malicious pages results in unusually large and concatenated words, thus median or mean word length in a malicious vs. a non-malicious page could be significantly different.
- Word count in a line- Average number of words per line in the web page and distinct word count in a web page could also reflect if the page is benign or malicious. These features are easier to calculate and widely used and should be considered as baseline features for any such model aiming to perform malicious webpage filtering.
- Number of NULL characters- Researches have observed the distribution of NULL space in a benign webpage could be very different from that of malicious web page. Average number of NULL spaces per line, total number of NULL space in a page could serve as content based features in this context.
- Usage of string concatenation- Paper [25] proposes creation of features to measure extent of string concatenation in web pages. The paper also demonstrates examples of encryption methods where hackers store certain malicious code within some string variable and when they need to execute those encrypted parts unescape() function is invoked within. Such encryption methods often result in large concatenated strings; thus extent of string concatenation could serve as an important feature while distinguishing between malicious Vs. non-malicious web pages
- Number of elements with small area- In [26], the researchers explained when the attackers aim at launching a drive -by-download infections they try to hide most of the elements on purpose. In this kind of attacks, visibility attributes are not used to hide the elements rather they set width and height of elements used to deliver the attack to very small values. In this paper [26] the researchers propose a feature that counts the number of elements of type div, iframe or object whose dimension is less than a certain threshold, in their study they experimented with – 30 square pixels for the area or 2 pixels for each side.
- Number of elements containing suspicious content- In [26] the researchers tried to come up with a measure of suspiciousness. The presence of shellcodes between the start tag and end tag is considered as suspicious if it is no longer than a certain threshold (128 characters) and contains less than 5% of whitespace characters. Count of such suspicious elements in a web page could serve as an important feature here.
- Number of out of place elements- In the paper [26] researchers have talked about another interesting feature which could be created based on the distorted positioning of certain objects. This feature counts the number of elements that reside out of their natural positioning in the HTML document. This feature is useful to detect web pages that have become malicious as the result of a stored XSS or SQL injection attack. In these cases, it is common to see scripts or iframes included in strange positions, such as between title tags or after the end of the document (outside the body or html elements). iframe, frame, form, script, object embed element positions are checked according to the

allowed positioning, as defined by the HTML DTD specifications.

- Number of included URLs- This feature proposed in [26] counts the number of elements which, being not inline, are included specifying their source location. Elements such as script, iframe, frame, embed, form, object are considered in computing this feature, because they can be used to include external content in a web page. The img elements and other elements are not considered, as they cannot be used to include any executable code
- Presence of double documents- This feature [26] indicates whether a web page contains two or more html, head, title, or body elements. This is not allowed by the HTML specification but can be seen in certain malicious web pages as a side-effect of the com-promise of a web site.

E. JavaScript Content Features

JavaScript [27] is a dynamic client-side scripting language used to create content for the Web along with HTML and CSS. It is used by the most of the Websites and supported by all the modern Web browsers. It is widely used to develop the interactive Web pages. However, in recent years JavaScript has become the most common and successful attack construction language. The malicious JavaScript can be inserted in a Web page and will run when the page is loaded in any browser. It will evade security tools such as a firewall and antivirus software. Cyber criminals regularly manipulate the code on countless websites to make it perform malicious functions. JavaScript is such a dynamic programming language that its improper implementations can create backdoors for attackers. When users visiting a website, JavaScript files are downloaded automatically. Due to the users' strong habits of online browsing, cyber criminals easily target such users for exploitation.

A malicious JavaScript consists of suspicious functions and patterns which tend to certain attacks like drive-by-downloads, XSS and malware distribution. Table II has features used by different researchers for the detection and analysis of benign and malicious Java Scripts [28][29][30][31].

TABLE II: JavaScript Features

S.No	JavaScript Function	Description
1	eval()	The number of eval() functions
2	setTimeout()	The number of setTimeout() functions
3	Iframe	The number of strings containing "iframe"
4	unescape()	The number of unescape() functions
5	escape()	The number of escape() functions
6	Classid	The number of classid
7	parseInt()	The number of parseInt() functions
8	fromCharCode()	The number of fromCharCode() functions
9	ActiveXObject()	The number of ActiveXObject() functions
10	No. of string direct assignments	The number of string direct assignments
11	concat()	The number of concat() functions
12	indexOf()	The number of indexOf()functions
13	substring()	The number of substring() functions
14	replace()	The number of replace() functions
15	document.addEventListener()	The number of document.addEventListener() functions
16	attachEvent()	The number of attachEvent() functions
17	createElement()	The number of createElement() functions
18	getElementById()	The number of getElementById() functions
19	document.write()	The number of document.write() functions
20	JavaScript word count	The number of words in JavaScript
21	JavaScript Keywords	The number of JavaScript keywords
22	No. of characters in JavaScript	The number of characters in JavaScript
23	The ratio between keywords and words	The ratio between keywords and words
24	Entropy of JavaScript	The entropy of the script as a whole
25	Length of Longest JavaScript Word	The length of the longest JavaScript word
26	The No. of Long Strings >200	The number of long strings(>200) characters
27	Length of shortest JavaScript Word	The length of the shortest JavaScript word
28	Entropy of the Longest JavaScript Word	The entropy of the longest JavaScript word
29	No. of Blank Spaces	The number of blank spaces in the JavaScript
30	Average Length of Words	Average length of words in the JavaScript
31	No. Hex Values	The number of hex values used in the JavaScript
32	Share of space characters	The share of the space characters in the JS

Number Malicious JavaScript's are mostly in the obfuscated form. Attacker uses obfuscation techniques to hide the real identity of JavaScript from the user and browser. Obfuscated malicious JavaScript mainly uses combination of digits (0-9), hex values and special characters like % , (,), ;, #, |, [,], {, }, ., etc. Also, such scripts uses suspicious JavaScript functions like split(), setAttribute(), charAt(), charCodeAt(), decode(), toString() etc. To explore such scripts, researchers [26] have identified a set of new features given below in Table III,

TABLE III: JavaScript Based Advanced Features

S.No	JavaScript Function	Description
1	search()	The number of search() functions
2	split()	The number of split() functions
3	onbeforeunload	The number of onbeforeunload events
4	onload	The number of onload events
5	onerror()	The number of onerror() functions
6	onunload	The number of onunload events
7	onbeforeload	The number of onbeforeload events
8	onmouseover	The number of onmouseover events
9	dispatchEvent	The number of dispatchEvent events
10	fireEvent	The number of fireEvent events
11	setAttribute()	The number of setAttribute() functions
12	window.location()	The number of window.location() functions
13	charAt()	The number of charAt() functions
14	console.log()	The number of console.log() functions
15	.js	The number of external JavaScript files
16	.php	The number of .php files
17	var	The number of var keywords used in the JavaScript
18	function	The number of function keywords used in the JavaScript
19	Math.random()	The number of Math.random() functions
20	charCodeAt()	The number of charCodeAt() functions
21	WScript	The number of WScript used in the JavaScript
22	decode()	The number of decode() functions
23	toString()	The number of toString() functions
24	No. of Digits	The number of digits used in the JavaScript
25	No. of Encoded Characters	The number of encoded characters used in the JavaScript
26	No. of backslash Characters	The number of backslash characters used in the JavaScript
27	No. of Pipe Characters	The number of pipe(|) characters used in the JavaScript
28	No. of % Characters	The number of % characters used in the JavaScript
29	No. of '(' Characters	The number of '(' characters used in the JavaScript
30	No. of ')' Characters	The number of ')' characters used in the JavaScript
31	No. of ',' Characters	The number of ',' characters used in the JavaScript

S.No	JavaScript Function	Description
32	No. of '#' Characters	The number of '#' characters used in the JavaScript
33	No. of '+' Characters	The number of '+' characters used in the JavaScript
34	No. of '.' Characters	The number of '.' characters used in the JavaScript
35	No. of ' Characters	The number of ' characters used in the JavaScript
36	No. of '[' Characters	The number of '[' characters used in the JavaScript
37	No. of ']' Characters	The number of ']' characters used in the JavaScript
38	No. of '{' Characters	The number of '{' characters used in the JavaScript
39	No. of '}' Characters	The number of '}' characters used in the JavaScript
40	Share of Encoded characters	Share of encoded characters in the JavaScript
41	Share of Digits characters	Share of digits in the JavaScript
42	Share of Hex/Octal characters	Share of hex/octal characters in the JavaScript
43	Share of Backslash characters	Share of backslash (\) characters in the JavaScript
44	Share of Pipe (|) characters	Share of pipe (|) characters in the JavaScript
45	Share of % characters	Share of % characters in the JavaScript

F. Host Based Features

These features describe properties of the Web site host as identified by the hostname portion of the URL. They allow us to approximate "where" malicious sites are hosted, "who" own them, and "how" they are managed. We examine the following sets of properties to construct host-based features:

- WHOIS information- This includes domain name registration dates, registrars, and registrants. So, if a set of malicious domains are registered by the same individual, we would like to treat such ownership as a malicious feature.
- Location - This refers to the host's geography, IP address prefix and autonomous system (AS) number. So, if malicious URLs tend to be hosted in a specific IP prefix of an Internet service provider (ISP), then we want to account for that disreputable ISP when classifying URLs.
- Connection speed - If some malicious sites tend to reside on compromised residential machines (connected via cable or DSL), then we want to record the host connection speed.
- Membership in blacklists - Over our experiments, 55% of malicious URLs were present in blacklists. Thus, although this feature is useful, it is still not comprehensive.
- Other DNS-related properties - These include time-to-live (TTL), spam-related domain name heuristics (Rudd, 2007), and whether the DNS records share the same ISP.

III. TRADITIONAL APPROACHES

Various web filtering approaches have been taken to prevent the malicious attacks in the past like blacklisting techniques, keyword blocking etc. Blacklisting method requires to have an exhaustive list of suspicious webpages which is costly to maintain. Keyword blocking also, is prone to give dubious results. These methods are commonly used by most of the antivirus systems as they are easy to implement, simple and efficient in most of the cases.

A. Blacklist and Whitelist Approach

This approach maintains a database with the list of URLs which are identified as malicious or benign respectively. The moment a user requests a URL, a lookup task is triggered to check that URL within the blacklist database. A previously identified malicious URL will thus get blocked[10]. As many new URLs and webpages are being created every day, the database needs to be updated periodically to identify every other malicious URLs accurately.

B. Keyword Blocking

This approach uses a list of keywords identified in the known malicious URLs. If a URL contains certain number of such keywords, it will be considered as malicious[11]. The problem with this method is the meanings of the words depend on the context. For example, few words like breast, penis can be used in medical as well as pornographic context. Also, this approach can be easily bypassed by misspelling certain words which might be present in the list.

C. Heuristics Approach

This approach is an extension of blacklisting techniques and study also shows that heuristics-based techniques outperform blacklisting based techniques used by most web browsers [12]. Instead of creating a list of blacklisted URLs, this approach creates a signature of these URLs. These signatures could be identified in new URLs as well and hence, this technique is more generalized than traditional blacklisting techniques. But such methods can only be designed for a limited number of attacks and cannot generalize to all types of (novel) attacks.

IV. MACHINE LEARNING BASED APPROACHES

Since heuristics-based approaches have their shortcomings, machine learning based approaches described below are gaining more importance nowadays in order to come up with a robust system of URL filtering in place. These kinds of approaches require a good set of features, extracted from either the URL string itself or from crawling the webpage and web content. Formulation of the URL filtering problem, as discussed in this paper, is a two-class classification problem, a webpage is either classified as 'malicious' or 'benign'. For training such a classifier, a training data consisting of both known malicious and benign URLs are needed.

A. Cosine Similarity Based Approach

This approach uses cosine similarity algorithm for text classification which is eventually used to classify the URLs as malicious or benign. We use samples of known malicious URLs to characterize the class of webpages that must be blocked as our training set. A new URL that is "close" or "similar" to members of this class, is blocked and those that are "dissimilar" will be allowed. Each document in the training set is represented by a vector of frequency of the most frequent words. The similarity between two documents is calculated by the cosine of the angle of the corresponding vectors. If these two vectors are similar to each other, the angle between them would be smaller, which indicates the larger cosine value [13].

Let us assume, $\Im$: set of known malicious URLs. This set needs to be updated at a regular interval. For a document $T \in \Im$, v_T : vector of relative frequencies is calculated after removing the stop words, low frequency words, words equal or shorter than two letter like 'to', 'of' etc.

An appropriate threshold cosine value needs to be calculated to classify the test pages. In order to do this, we consider another set $\Im'$, consists of samples both inside and outside of the forbidden or malicious class. After measuring the similarity τ between each element of $\Im'$ to $\Im$, we calculate the percentage of documents that are correctly classified in $\Im'$. We choose the threshold for which we have the highest percentage of correctly classified document.

For a test webpage W, we will calculate the cosine similarity with each member of $\Im$ by $\cos(v_W, v_Y)$, $X \in \Im$. S, the set of n% highest similarity values are found. Now the class coefficient σ_W is obtained by calculating the average of the n% similarity values. If $\sigma_W \geqslant \tau$, the threshold, then the page will be classified as malicious.

If the page comes out to be a benign webpage, then the same exercise can be conducted on the randomly selected hyperlinks present in that webpage. If majority of the class coefficients are coming out to be malicious, then the webpage will be decided to be blocked.

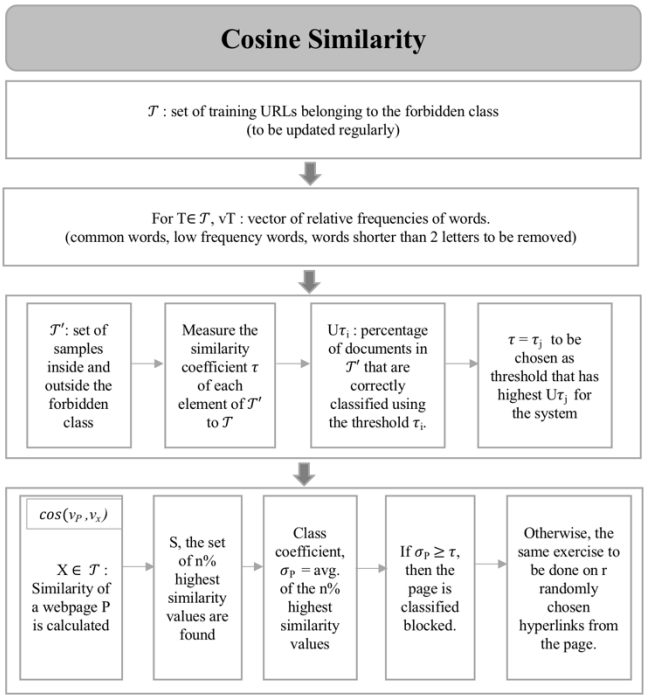

Fig. 1: Cosine Similarity Process Flow

B. Support Vector Machines

Support Vector Machines are widely used supervised learning algorithms. In case of two class classification problems, the algorithm tries to find a linear hyper-plane that separates the examples of different classes and maximizes the distance of the hyper-plane and the closest examples from different classes. When the data is not linearly separable in the given feature space, SVMs use a kernel function to map the data into a higher dimensional space and separate the data on the mapped dimension where it is possible to achieve a hyperplane separating the two classes. Given the non-linearity involved in the data, one can test out the different available kernel functions (radial basis function, polynomial, gaussian etc) and choose the one which exploits the non-linearity in the data in the best possible way.

For the case of malicious URL classification, as we will see in section no IV, the set of features are computed from various sources – the HTML content based features, lexical content based features etc. and in most cases the decision boundary that separates the malicious web pages from that of the non-malicious pages are non-linear in terms of the feature space. Thus SVMs could produce considerably good accuracy on this kind of problems. However, it has been observed for large datasets SVMs would require very long training hours and the memory requirement to store the kernel matrix is proportional to the number of training examples. So, these algorithms aren't feasible for large data sets. In the URL filtering problem, the feature set could be huge which requires considerable amount of training examples, which in turn would require long training hours and large amount of memory.

C. Artificial Neural Network

Artificial neural networks (ANNs) can learn and adapt according to training cases fed to them. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables. As every URL can be characterized using many variables including content based, rank variables, java/html code etc., ANN models prove effective in learning, modeling and generalizing non-linear and complex relationships among high dimensional input space with given output variables. ANNs can thus achieve high classification accuracy, however ANN training should involve a sufficiently large number of training examples, including both positive and negative cases. To train a robust ANN classifier one must help the network decide on the number of neurons in each layer, the network topology, the initial weights and the hyper parameters.

V. Potential Framework

URL filtering problem is formulated as a two-class classification task for predicting "malicious" versus "benign" URLs. Specifically, given a data set with N URLs $\{(u_1,y_1),...,(u_N,y_N)\}$, where u_i for i = 1,...,N represents a URL from the training data, and $y_i \in \{0, 1\}$ is the corresponding label where $y_i = 1$ represents a malicious URL and $y_i = 0$ represents a benign URL.

The two major components for this system are, extracting the appropriate feature representation: $u_i \rightarrow x_i$ where $x_i \in R^d$ is a d- dimensional feature vector representing the URL; and learning a classifier $f : R^d \rightarrow R$ which predicts the class assignment for any test URL.

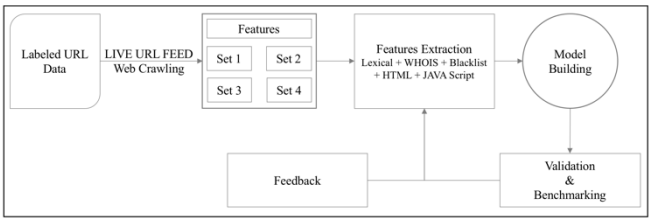

Fig. 2: Architecture of the URL filtering system

A. Data Preparation

Static features like lexical and host-based features can be directly computed just using URL information. However, dynamic features require executing the URL and crawling the page content. The crawled raw data needs to be processed with methods like one-hot encoding, TFIDF, bag-of-words, bi-gram or n-gram techniques etc. The end objective of the data preparation stage is to generate a feature vector coded numerically, representing all the characteristics of each URL.

B. Model Building

The goal of the machine learning model is to maximize the predictive accuracy. Accuracy should be defined as per the use case of an organization. For example, a highly data sensitive organization like commerce or banking would like the URL filtering system to decrease the false negative cases as much as possible.

Another important metric of URL filtering system is the total run time required to make a prediction using machine learning model. This metric changes depending on where the system is installed, client-side or server-side. Machine learning experts need to take both predictive accuracy and run-time into consideration while choosing a classifier for the system.

C. Level of Complexity in the Feature Set

Similar to classifier selection, feature selection will also play a role in determining the prediction accuracy and run time of the system. All the features discussed in section IV has the potential to improve classification accuracy but many of them need to be excluded for various reasons. Classifying a URL with a trained model built on static features is computationally less expensive compared to a model built on both static and dynamic features as for the latter case, the page content needs to be crawled and processed for running the classifier. Also, downloading page content could make the user's machine vulnerable to threats. In Table IV, we are discussing the collection difficulty and time complexity for different feature types.

Table IV.

Features	Collection Difficulty	Collection Time
Blacklist	Moderate	Moderate
Lexical	Easy	Low
Web-Content Based Features	High	High
HTML Content Features	Easy	High
Java Script Based Features	Easy	High
Host Based Features	Easy	High

REFERENCES

[1] Netcraft Web server Survey, "https://news.netcraft.com/archives/2018/01/19/january-2018-web-server-survey.html", 19th Jnuary 2018

[2] Symantec Internet Security Threat Report Volume 23, March 2018

[3] F. Bamber, A. Bowyer, N. Leung, F. Lopes, L. Mills, D. Williams under direction of R. White, "Investigation: Wannacry cyber attack nd the NHS", Report by Comptroller and Auditor General, Department of Health, 25th April, 2018, www.nao.org.uk

[4] Equifax, "2017 Cybersecurity incident & important consumer information", 1st March, 2018, www.equifaxsecurity2017.com

[5] Nicole Perlroth, "All 3 billion Yahoo accounts were affected by 2013 attack", New York Times, 3rd October, 2017

[6] Sam Kottler, "February 28th DDoS Incident Report", Github Engineering, 1st March 2018

[7] Xiaokui Shu, Ke Tian, Andrew Ciambrone and Danfeng Yao, member IEEE, "Breaking the Target: An analysis of Target data breach and lessons learned", arXiV:1701.04940v1, 18th January 2017

[8] Aorato Labs, "The untold story of the Target attack tep by step", August 2014

[9] Pui Y. Lee, Siu C. Hui, A. C. M. Fong, "Neural Networks on web content filtering", IEEE Intelligent Systems, Vol. 17, Sep/Oct 2002

[10] Faronics, "Blacklist versus Whitelist Software solutions", August, 2005

[11] Vrushali Sanjay Kharad, Prof. S. S. Kulkarni, "Design model on website filtering and blocking", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, Issue 4, April 2005

[12] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, C. Zhang, "An emperical analysis of phishing blacklists", 6th Intl. Conference on Email and antispam CEAS, Mountain View, California, 2009

[13] R. Du, R. Safavi-Naini, Willy Susilo, "Web filtering using text classification", 11th IEEE international conference on networks, 2003

[14] J. Ma, L. K. Saul, S. Savage, G. M. Voelkar, "Beyond Blacklists:learning to detect malicious web sites from suspicious urls", Proceedings of the 15th ACM SIGKDD international conference Knowledge discovery and data mining, 2009

[15] Phishtank, "https://www.phishtank.com/faq.php"

[16] P. Prakash, M. Kumar, R. R. Kompella, M. Gupta, "Phishnet: predictive blacklisting to detect phishing attacks", IEEE 2010

[17] A. Blum, B. Wardman, T. Solorio, and G. Werner, "Lexical features based phishing url detection sing online learning", Proceedings of the 3rd ACM workshop on artificial intelligence and security, 2010

[18] P. Kolari, T. Finin, A. Joshi, "SVMs for blogoshpere: Blog identification and splog detection", AAAI Symposium,: Computational approaches to analyse weblogs, 2006

[19] S. Yadav, A. K. K. Reddy, A. Reddy, S. Ranjan, "Detecting algorithmically generated malicious domain names", ACM 2010

[20] A. Le, A. Markopoulou, M. Faloutsos, "Phishdef: Url names say it all", IEEE 2011

[21] Michael Chau, Hsinchun Chen, "A machine learning approach to web page filtering using content and structure analysis", Elsevier 2007

[22] O. Baujard, V. Baujard, S. Aurel, C. Boyer, R.D. Appel, "Trends in medical information retrieval on the Internet", Computers in Biology and Medicine 28 (1998) 589–601

[23] Sergey Brin, Lawrance Page, "The anatomy of a large-scale hypertextual web search engine"

[24] Jon M. Kleinberg, "Authoritative source in a hyperlinked environment", Proceedings in ACM-SIAM Symposium n discrete algorithms, 1998

[25] 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

[26] 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

[27] Dharmaraj R. Patil, J. B. Patil, “Detection of Malicious JavaScript code in web pages”, Indian Journal of Science and Technology, Vol 10(19), May 2017

[28] Seshagiri P, Vazhayil A, Sriram P., “AMA: Static code analysis of web page for the detection of malicious scripts”, Procedia Computer Science. 2016 Dec 31; 93:768-73.

[29] Cova M, Kruegel C, Vigna G., “Detection and analysis of drive-by-download attacks and malicious JavaScript code”, Proceedings of the 19th International Conference on World Wide Web; 2010. p. 281-90

[30] Fraiwan M, Al-Salman R, Khasawneh N, Conrad S., “Analysis and identification of malicious JavaScript code”, Information Security Journal: A Global Perspective. 2012; 21(1):1-1

[31] Wang WH, Yin-Jun LV, Chen HB, Fang ZL., “A static malicious JavaScript detection using svm”, Proceedings of the International Conference on Computer Science and Electronics Engineering. 2013. p. 21-30