POPSTAR: MALICIOUS WEBSITE DETECTION USING SOCIAL REPUTATION AND SITE AGILITY

MALCOLM G HAYNES QIUXIANG JIN ARUNPRASANNA S P
HENGFU WU AKASH GUPTA PRANAV DILIP BHEDI
HAOCHEN ZHAO JIAWEI ZHANG

GEORGIA INSTITUTE OF TECHNOLOGY

Abstract-**Detecting unknown malicious websites is a difficult problem. Despite years of research, the problem continues to exist. In this paper, we introduce *PopStar* a system for classifying a site as benign or malicious based on a combination of static and lightweight dynamic analysis. We calculate the social reputation of a site based on the activity in several popular social networks. Additionally, we define and calculate a site's agility based on features extracted from the screenshot and source code. Based on this information, *PopStar* is able to classify a site as benign or malicious with high accuracy. *PopStar* produced up to 97.8% true positive rate with a low 1.5% false positive rate. To the best of our knowledge, *PopStar* is the first system to use social reputation extensively to classify a website as benign or malicious.**

We also study the problem of detecting malicious links in search engine results. Specifically, we focus on detecting water-holing links, or links designed to lure people from a specific business or industry. We present a novel way to check for water-holing links with PopRank. PopRank takes search engine results for the most popular keywords in an industry and returns the links most likely to be water-holing sites. Up to 38% of domains identified by PopRank as potential water holes were also listed on one or more blacklists. Additionally, PopRank provides a social reputation score for each site that can be used to classify a site as benign or malicious.

Keywords: Website agility, social reputation, malicious web page analysis, efficient web page filtering

1. INTRODUCTION

The World Wide Web is the primary means for many people to access the Internet. Typically, access occurs when a user opens a browser and visits a web page. User interaction with the webpage may include transmitting personal and financial information. Such an environment provides a vast ground for criminals who attempt to steal users' information and subvert users' computers. These criminals use malicious websites to achieve their nefarious goals.

Our goal in this paper is to classify a site as either benign or malicious. Specifically, we render a website and record the resulting source code as well as capture an image of the site. We also record social reputation information. This information is then used as input into our system, *PopStar*, to label a site as either benign or malicious. Our system is based on two observations: first, a link to a malicious site is unlikely to be shared on online social networks such as Facebook and Twitter. Second, a malicious site is likely to change its visual appearance and underlying source code in a different manner than a benign site.

In addition to detecting malicious web pages, we study the problem of detecting malicious links in search engine results. Specifically, we focus on detecting *water-holing* links, or links designed to lure people from a specific business or industry. Our algorithm, *PopRank* takes search engine results for the most popular keywords in an industry and reorders those results based on social metrics.

Previous work on static detection of malicious websites mainly focused on analyzing elements of the URL and page source code [5, 6, 7]. Eshete et. al [7] also included some minimal social reputation metrics in their analysis. To the best of our knowledge, this is the first paper that builds a malicious webpage and link detection system around the concept of site agility and social reputation. Our main contributions are as follows:

- We defined and measure site agility based on three metrics: visual agility, source code agility, and social reputation. These metrics are used as inputs to a detection engine that classifies a site as benign or malicious.
- We built *PopStar* a malicious website detection system that takes as input, site agility metrics and classifies a site as benign or malicious. During testing, the system correctly classified sites with 97.8%

accuracy and a 1.5% false positive rate. These results indicate *PopStar*'s suitability for use as part of a malicious site detection engine.

- We developed *PopRank,* a system to rank search engine results based on the social popularity of a website.

The rest of the paper is organized as follows: Sections 2 and 3 provide background and related work. Readers familiar with this field may skip to Section 4, which lays out our approach to the problem. In particular, Section 4 and 5 discusses our approach to calculating social reputation and measurements of a webpage's visual agility, source code agility. To develop and evaluate the performance of our system, we collected data from thousands of sites. Section 6 describes the data collection process. Section 7 discusses our machine learning classifier and analyzes the results. Section 8 describes *PopRank* in detail. This section lays out the approach for detecting potential *water-holing* sites that appear in search engine results. Section 9 presents some limitations of our system areas of future work. Section 10 presents our final conclusion.

2. BACKGROUND

One-way cyber criminals illegally profit from the web is by running malicious web pages designed to exploit unwary visitors. A favorite technique used by these criminals is the *drive-by download*. In this kind of attack, the cybercriminal crafts a script designed to exploit some vulnerability in a browser. If successful, the attacker is ultimately able to gain high-level (often root) access to a system and thereafter incorporate it into a botnet or otherwise illegally use the system (and the information stored in it) for personal gain. What is so insidious about the *drive-by download* is that the victim does nothing other than visit the malicious page. In rendering the page, the victim's browser will automatically execute the malicious script.

The drive-by download is not the only attack available to cybercriminals. Another popular technique is phishing. A phishing site is designed to trick the victim into willingly transmitting personal information to the attacker. This information may include name, social security numbers, addresses, login credentials, credit card numbers, and/or other personal identifying information. In a similar vein, the attacker may convince the victim to click on a link to download some malicious software masquerading as benign software.

These attacks are only possible when the victim visits the malicious site. As such, criminals spend a great deal of time figuring out how to lure people to their sites. Popular techniques include spam, posting on forums, posting comments on websites or social network sites, and achieving high rankings in search engines. Of these techniques, ranking in search engines may be the most difficult to defend against because users have been trained to trust the links provided by search engine results. However, a study by Provos et al found that 1.3% of all search engine results are links to malicious pages [11].

This is possible because PageRank or a similar algorithm is a key factor in the results returned by a search engine. PageRank is based on the premise that the more sites that link to a page, the more important the page is. It uses the related concept that a link from an important site carries more weight than a link from an unimportant site. Understanding the PageRank logic, malicious actors attempt to manipulate search engine results in order to artificially increase the ranking of their website. This is commonly referred to as *web spam*. Wu et. al [10] studied how link farms can artificially increase the ranking of a site. Attackers may also exploit vulnerabilities to install hidden links on popular sites.

A malicious site that ranks well in the search engines may be used for a *water holing* attack. *Water holing* is the practice of using a third party site trusted by a victim to launch an attack. If an attacker can compromise a site frequently visited by employees of a business, there is a good chance the attacker can find a way to compromise the business' site. This becomes much easier if an attacker owns the third party site. Knowing the popular search terms for an industry, attackers can use *web spam* on industry related keywords to rank highly in the search engines.

3. RELATED WORK

Malicious website analysis falls into two major categories: dynamic and static. Dynamic analysis consists of running code associated with a URL and observing what it does. Provos et. al [2] used dynamic analysis as part of their study of

billions of URLs. From within a virtual machine (VM), they visited each suspect URL and classified it as malicious or benign based on a combination of execution heuristics and antivirus detection results. Their execution heuristics measured the following changes caused by visiting the URL: the number of processes created, the number of registry changes, and the number of file system changes.

Static analysis consists of analyzing features associated with a URL without executing any code. These features primarily consist of the URL analysis, WHOIS analysis, DNS analysis, and html source code analysis. Ma et. al [6] derived 4 features from the URL and WHOIS information that could predict the nature of a website with approximately 90% accuracy. When they added additional features (all based on the URL and WHOIS information) they were able to achieve 95%-99% accuracy.

Canali et al. [5] developed Prophiler which used 77 static features from the URL, page source code, WHOIS information, and DNS information to reliably classify a site with 0.77% False Negative. Prophiler was designed as a filter to reduce the number of sites required for dynamic analysis. Prophiler was able to reduce the number of sites requiring additional analysis by 85%.

Eshete et. al [7] combined dynamic and static features with BINSPECT. They used lightweight emulation to get rendered source code. They then analyzed 39 features of the URL, source code, and social reputation to achieve 97.8% accuracy with .19% false positive and .01% false negative. Of note, they used 3 social reputation features: # of Facebook shares, # of Twitter shares, and # of GooglePlus shares.

Karunakaran et. al [15] published a review of web spam detection techniques. Egele et. al [13] attempted to detect web spam by classifying links based on featues of the web page most likely to be manipulated (such as title, H1 headers, etc.). Araujo et. al [12] used a language model to detect web spam by analyzing relatedness of a page to the page it links to. Castillo et. al [14] used web topology to identify web spam based on the observation that spam links tend to link to other spam sites while non-spam links tend to link to other non-spam sites.

To our knowledge, we are the first to attempt to identify water-holing web spam links in search engine results or use social reputation for link classification. Additionally, no one has used as many social reputation features to classify a website.

4. SOCIAL REPUTATION

Work in social sciences suggests that the collective wisdom of a large group can provide better answers to a problem than any of the individuals could provide on their own [16]. This observation is the motivation behind the use of social reputation for classifying websites. Intuitively, we know that people do not share malicious sites with their friends.

We use the following metrics to arrive at a social reputation score: number of Facebook likes and comments; Twitter tweets, Google Plus One +1s, Pinterest pins, Linkedin shares, and StumbledUpon stumbles.

The Social Reputation score is calculated as follows. First, each social reputation metric is normalized for all the websites. Then, a weighted sum is used to calculate the overall social reputation score.

4.1 Individual Metric Normalization

The normalization is done using the following formula:

$$SR_i = \left(\frac{value\ of\ current\ element}{sum\ of\ all\ elements}\right)^{\frac{1}{100}} \qquad (1)$$

For each social reputation parameter add all the values and divide individual value by the sum (for all websites) and raise it to the power 0.01 to bring the value close to 1 in such a way that the scaling remains proportional and the value does not exceed 1.

Now calculate SRi for all the 8 social reputation parameters (i.e. $SR_{iFbLikes}$ for Facebook Likes, $SR_{iStumble}$ for StumbleUpon stumbles, $SR_{iTwitter}$ for Twitter tweets, etc.)

4.2 Weighted sum

$$\text{Reputation Score} = ((70.84*SR_{Stumble} + 96.08*SR_{Twitter} + 80.41*SR_{Linkedin} + 84.35*SR_{Pinterest} + 92.13*SR_{FbComment} + 90.6*SR_{FbLike} + 96.07*SR_{Google}) / 7) * 1.1466 \qquad (2)$$

The weights are obtained from machine learning algorithm discussed in 7 and represent the predictive accuracy of the metric when considered in isolation. For example, Twitter is our most reliable parameter with 96.08% accuracy (true

positive rate) and hence has the highest weight (96.08), while StumbleUpon Stumbles is the least reliable parameter with 70.84% accuracy and has the least weight (70.84). We scale the final result by 1.1466 in order to derive a score between 0 and 100.

5. SITE AGILITY

The web is a dynamic place and websites change every day. One-way to measure a website is by how much it changes over time (i.e. between some time, *t*, and some other time, *t+i*). This change can be considered a measurement of the site's agility over the specified time period.

To measure the change in a website is in essence measuring how much the source code changes. Intuitively, if the code doesn't change at all, then the site hasn't changed. Conversely, significant changes in the source code should result in significant changes to the website. However, this intuition doesn't always match with experience. For example, the code for an ad will not change with each visit to a site, however the ad served may change. This change is reflected in the rendered appearance of the website. Thus, we derive two key metrics for site agility: 1) change in source code over time or source code agility and 2) change in rendered appearance over time or visual agility.

We use the Levenshtein distance to quantify how much the source code of website has changed over a time period. The Levenshtein distance between two documents is the minimal number of words that need to be deleted or inserted to transform one document to the other. The Levenshtein computational complexity between two documents is O(*a***b*), where *a* and *b* are the length of the two documents respectively. If the two documents are similar in length, the time complexity of the algorithm is O(n2). In practical application, the smaller the difference between two documents, the less time it takes to calculate this value. Since the daily change of a website will normally be small, the time to do the calculation is normally small. To further accelerate the calculation, we strip the HTML tags from the website source code.

In addition to the Levenshtein distance, we also calculate the relative Levenshtein distance between the source codes over two time periods:

$$RellevDist(A,B) = LevDist(website1, website2) / (len(website1) + len(website2)) \quad (3)$$

Where *LevDist* is the Levenshtein distance and *len(website)* is the length of the website source code.

To measure visual agility, we compute the difference between screenshots using four metrics derived from the image histograms: Correlation, Intersection, Chi-Square and Bhattacharyya Distance. Given two histograms H1 and H2, these metrics are calculated as follows :

Correlation:

$$d(H_1,H_2) = \frac{\sum_I (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}{\sqrt{\sum_I (H_1(I) - \overline{H_1})^2 \sum_I (H_2(I) - \overline{H_2})^2}} \quad (4)$$

Chi-Square:

$$\overline{H_k} = \frac{1}{N}\sum_J H_k(J) \quad (5)$$

Intersection:

$$d(H_1,H_2) = \sum_I \min(H_1(I), H_2(I)) \quad (6)$$

Bhattacharyya Distance:

$$d(H_1,H_2) = \sqrt{1 - \frac{1}{\sqrt{\overline{H_1}\overline{H_2}N^2}}\sum_I \sqrt{H_1(I)\cdot H_2(I)}} \quad (7)$$

For correlation and intersection metrics, the higher the value, the more similar the two images are. For chi-square and Bhattacharya distance, the smaller the value, the more similar the two images are.

6. DATA COLLECTION

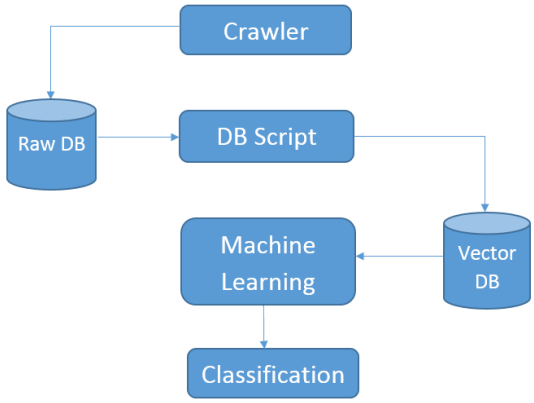


Figure 1: Data Collection Process

Data Extraction

To collect data, we crawled sites in the Alexa list as well as malicious sites from MalwareDomains. These represented our ground truth for benign and malicious sites. We implemented the crawler using a multi-threaded Python script designed around the Selenium framework. Selenium is a tool designed for automated website testing, however, it also provides a novel way to crawl websites. Using a PhantomJS headless webkit as the 'browser', our script visits the selected websites, scrolls through the site, and then takes a snapshot of the site. The snapshot, page source code, and other associated data are piped into a SQLite database for further analysis.

Our decision to use this method of crawling was driven by observations made when using a Scrapy-based webcrawler. We input domains into the Scrapy crawler and expected it to resolve the name to the correct website. We know that some websites should redirect to SSL enabled pages to ensure secure connections. However, using the default settings on Scrapy, sites that should resolve to 'https' were resolving to 'http'. We determined that this was a result of the user agent that Scrapy sent (i.e. the default is none). When a site does not detect an SSL enabled agent, it often responds with a non-secure, non-JavaScript enabled page. Since we wanted to study some differences between malicious and benign sites, we determined to emulate a modern, JavaScript enabled browser. Therefore, we modified the crawler to indicate that it was running Internet Explorer 9.0.

However, when we ran Scrapy with this setting, we noticed that Scrapy would have a connection timeout on sites that we knew were valid. Further analysis showed that, by emulating a JavaScript enabled browser, JavaScript enabled sites were attempting to send JavaScript code to Scrapy for execution. Scrapy cannot run JavaScript and therefore, sites would timeout. The solution was the PhantomJS headless kit running through Selenium.

Selenium enables full rendering to include JavaScript. We felt this would give us the most accurate results since most users visit websites using a modern browser and selenium essentially enabled lightweight dynamic analysis. In fact, many drive by downloads and other malicious sites take advantage of JavaScript for their exploits. Therefore, to get accurate results we needed to execute JavaScript. Further, we realized that some JavaScript enabled sites (for example, AJAX sites) do not fully display the page until a user scrolls down some amount. Since we wanted the full site, we used the selenium framework to enable the crawler to scroll down through the entire rendered page. This method is slow, requiring on average 20 seconds to completely render and scroll through a page, thus limiting the number of sites we could crawl per day.

Data Storage

All data obtained during the crawl was stored in raw form in an SQLite database. This database rapidly grows in size. Crawling all 1 million sites resulted in a 80GB database.

When choosing a database to store date, there are many different types available, classified by the number of users, the database location, and the expected type and extent of use [11]. We chose to use SQLite because it is native to Python and our database requirements are relatively simple. Although SQLite only allows one write at a time, it allows multiple concurrent reads. Since we were crawling relatively few sites per day, we determined that a slight delay in writing to the database was insignificant. At the same time, the entire team of researchers could search the database and pull data for further analysis, even while the web crawler was collecting additional data.

Data Manipulation

Once stored in the database, a separate process retrieves the data and prepares it for input into *PopStar*. This process calculates the source code and image agility as described in section 5. Additionally, this process calculates the number of internal and external links. Other analysis of the rendered source code is possible, but not implemented at this time.

The results of the data manipulation are stored in a separate SQLite database for *PopStar* to use for website classification.

7. *POPSTAR*: A WEBSITE CLASSIFICATION SYSTEM

PopStar is implemented as a binary classifier, assigning a value of either benign or malicious to each website it evaluates. It uses 13 features to evaluate a website. These features consist of five code features, one image feature, and

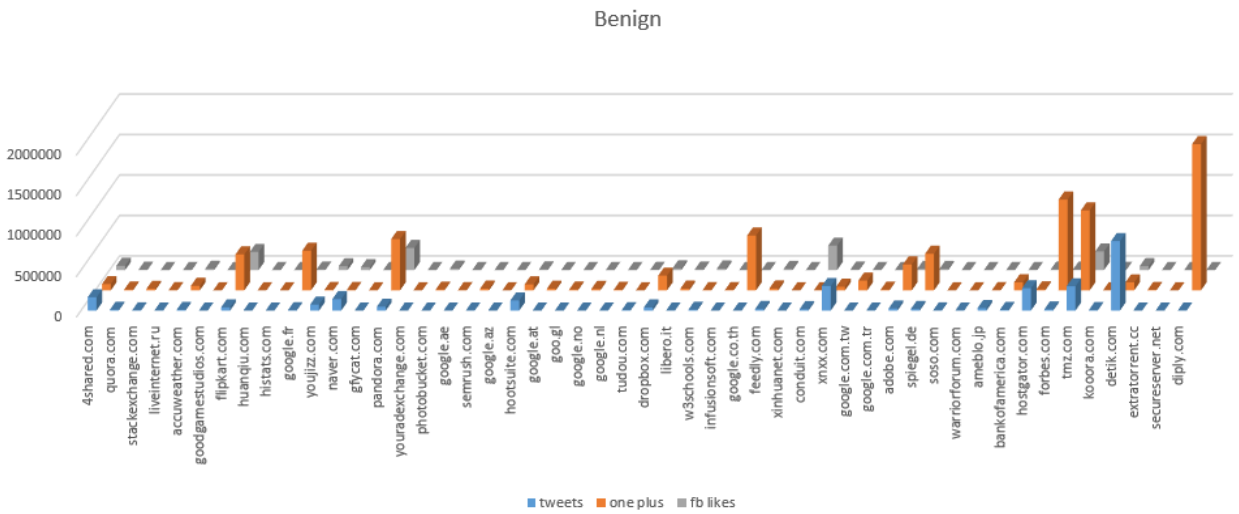
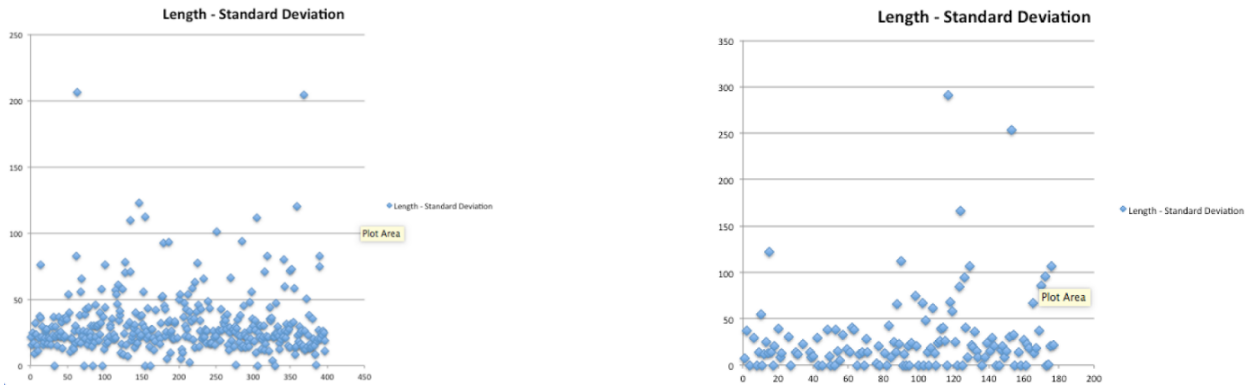


Figure 2: Comparison of Social Features



(a) Legitimate Websites (b) Malicious Websites

Figure 3: Standard Deviation of Source Code Length

seven social reputation features to evaluate a website. We trained *PopStar* using Random Forest as it is core algorithm and validated with 5 fold cross validation. In detection mode, *PopStar* identified sites with greater than 97.8% accuracy and 1.5% false positives. On a set of unpopular websites, *PopStar* achieved 88.0% accuracy.

7.1 Feature Selection

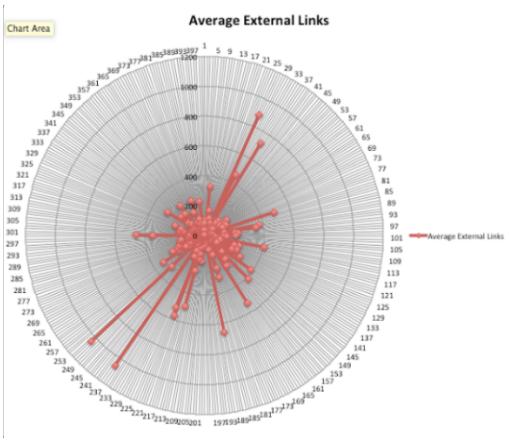
We use three kinds of features in our detection system to differentiate between malicious and benign websites: code features, image features, and social reputation features.

Code features

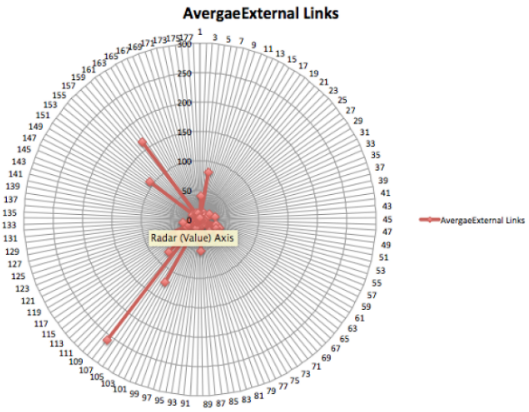
PopStar uses five code features. These include two code agility features and three link featuers.

The code agility features are the (absolute) Levenshtein distance (LevDistabs) and relative Levenshtein distance (LevDistrel) as described in section 5. The Levenshtein distance is a numerical value representing the source code change, while the relative Levenshtein distance reflects the percentage of the change.

Figures 3(a) and 3(b) are graphs of the standard deviation of the length of the source code. These plots show that malicious sites do tend to show changes in their source code. However, there is wide variation between benign sites of different categories.

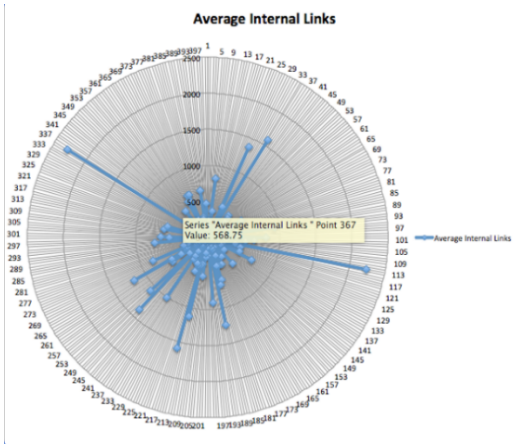


(a) Benign Websites

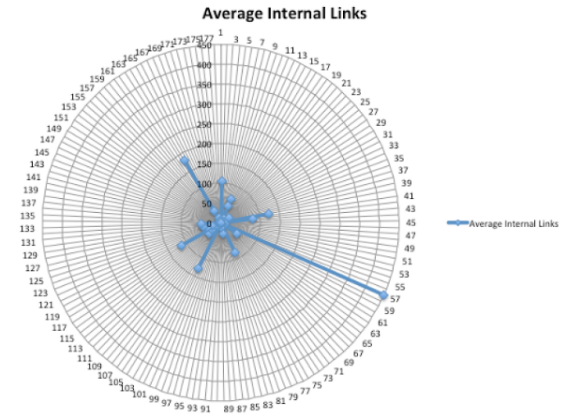


(b) Malicious Websites

Figure 4: Average External Links



(a) Benign Websites



(b) Malicious Websites

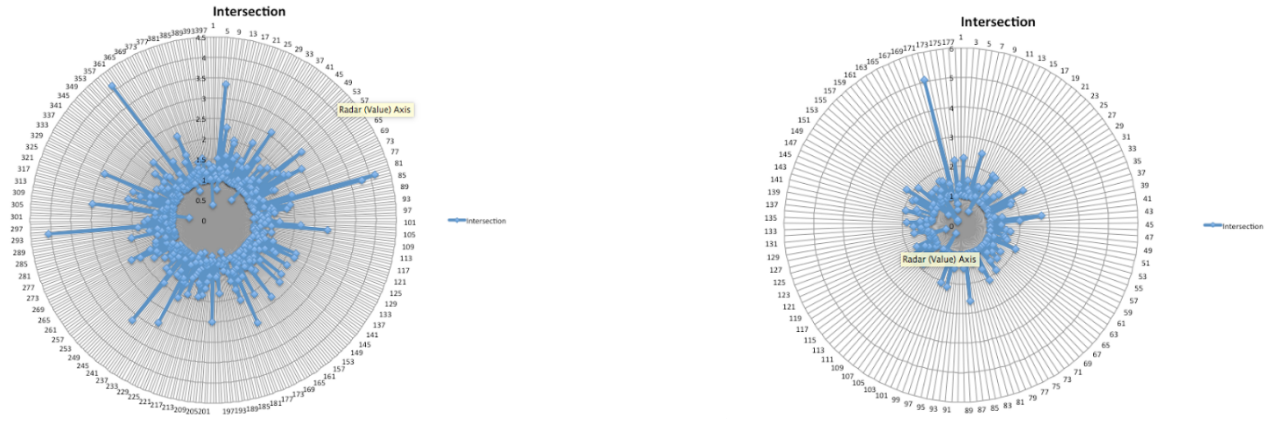
Figure 5: Average Internal Links

The link based features include the total number of links found on a web page (Linktotal), the number of external links (Linkexternal), and the number of internal links (Linkinternal). The intuition behind selecting the links figure is shown in figures 4(a) and 4(b).

These figures show that benign sites tend to have more external links than malicious site. There are notable exceptions to this rule. For example, '*www.cskarepublications.com*' was a malicious website that had numerous external links to illegal downloads. A similar observation can be made for the number of internal links, with benign sites having more internal links than malicious sites as shown in figures 5(a) and 5(b).

Image features

PopStar uses one image feature which is the visual agility as measured by the Bhattacharyya Distance. Initially, we calculated four visual agility measures - Image Correlation (ImageCor), Intersection(ImageInter), Chi-Sqaure (ImageChi) and Bhattacharyya Distance (ImageBha) as described in Section 5. Correlation and Intersection measure the similarity between two images, while Chi-Square and The Bhattacharya Distance measure the difference between two images. Analysis shows that any pair of these metrics is linearly dependent. Therefore, we chose to plot only one metric, the Image Intersection, as the image-based feature. Figures 6(a) and 6(b) show that malicious websites tend to change more than benign ones, however, there is large variation between different benign websites.



(a) Benign Websites (b) Malicious Websites

Figure 6: Plot of Image Intersection

Social Reputation Features

PopStar uses seven social reputation features. These include the total number of StumbleUpon stumbles, Twitter tweets, Linkedin shares, Pinterest pins, GooglePlusOne +1s, Facebook likes, and Facebook comments.

These features collectively reflect the reputation of a website in the social network. Facebook shares are purposely not included as a feature. Rationale for this is discussed in 7.4 Results. Figure 6 shows the social reputation for selected websites. Generally, benign websites have higher social reputation scores than malicious sites.

7.2 Model Selection

Using the features described in 7.1, each website, w, is characterized as a feature vector defined as follows:

$$V(w) = [V_{code}(w), V_{image(w)}, V_{social}(w)] \quad (8)$$

Where

$$V_{code}(w) = [LevDist_{abs}(w), LevDist_{rel}(w), Link_{total}(w), Link_{external}(w)] \quad (9)$$

$$V_{imgage(w)} = [\mathrm{Im}age_{Bha}(w)] \quad (10)$$

$$V_{social(w)} = [SR_{Stumble}(w), SR_{Twitter}(w), SR_{Linkedin}(w), SR_{P\,\mathrm{int}erest}(w), SR_{FbLike}(w), SR_{FbComment}(w), SR_{Google}(w)] \quad (11)$$

The feature vector was input into several models with Random Forrest and Bernoulli Naive Bayes returning the best results as shown in Figure 7. We choose the random forest algorithm for *PopStar* because it produced the best results when used with all features. Additionally, it has several advantages over other available classification algorithms.

	True Positive	False Positive	Accuracy
Random Forest	96.6%	1.7%	97.8%
Naïve Bayes	92.1%	2.6%	95.7%

Figure 7: Accuracy of Different Models

First, the ranges of different features in $V(w)$ are wildly different, and many of them have totally different orders of magnitude. Distance-based classification methods like K nearest neighbor (KNN) require all the parameters of the feature vector to be normalized. While we can normalize all the parameters of $V(w)$, different methods of normalization can cause different final results, and the normalization method itself can remove valuable information. The random forests algorithm, on the other hand, doesn't require these features to be normalized, so we are able to use these features without modification and avoid the possible influence and information loss caused by normalization.

Second, the dimensions of our feature vector $V(w)$ are large compared to the number of websites we use to train the machine learning algorithm. In this situation, classification methods involving the use of Euclidean space will cause dimensionality which means in order to obtain a statistically sound and reliable result, the amount of data needed grows exponentially with the dimensions of the vector. Due to the limited resources we have, we are not able to provide the amount of website data required by the number of dimensions we use. The random forest algorithm doesn't involve the use of Euclidean space, so we avoid the problem of dimensionality.

Moreover, while each individual feature itself is a good indicator, we are not sure how important each feature should be in comparison to the others. The random forests algorithm can automatically emphasize the features that can better differentiate between malicious and benign websites, so we are able to get an optimal performance to some degree. While the decision tree algorithm can also achieve that, the fact that random forests combine the results of several randomly generated decision trees will prevent us from over fitting, that is, over-emphasizing some features according to our training sets.

Our detector implements a random forest algorithm with 10 trees in it. The detector is trained off-line. The feature vectors of known benign and malicious are computed from the crawled data, and then used as ground truth to train the detector. After the training is complete, we can use it to identify whether a new website is malicious or not.

7.4 Experiment and results

In our experiment, we use websites from the Alexa and malicious websites from malware domainlist.com as ground truth to train and test our detector. We labeled the top 500 websites from Alexa as benign websites and 200 websites in the blacklist as malicious websites. We remove websites with invalid data, and use the remaining websites (398 benign and 178 malicious) to perform the experiment.

We use 5-fold cross-validation to evaluate the performance of our detector. It randomly shuffles the websites, divides them into 5 equally sized parts, uses four parts to train the detector and tests it with the fifth part. The results of this experiment show that the true positive rate is 96.42%, the false positive rate is 1.52%, and that the accuracy is 97.80%.

Moreover, to evaluate the role different features play in the algorithm; we repeat the experiment using only part of the features. Table 1 shows these results.

Feature used	**True Positive Rate**	**False Positive Rate**	**Accuracy**
Social Reputation, Source Code Agility, Image Agility, Link Analysis	96.42%	1.52%	97.80%
Social Reputation, Source Code Agility, Image Agility	96.29%	1.71%	97.63%
Social Reputation, Link Analysis	96.13%	1.25%	97.89%
Social Reputation	95.58%	1.61%	97.46%
Source Code Agility, Image Agility, Link Analysis	74.79%	15.91%	81.02%
Source Code Agility, Image Agility	70.45%	14.55%	80.50%

Table 1: Performance comparison with different combination of features

As we can see in the table, social-based features V_{social} play a very important role in our detector performance. To see how much each individual social reputation parameters can differentiate between malicious and benign websites, we conduct experiments using only one of these social reputations at a time. Table 2 shows these results.

Social parameters used	True Positive Rate	False Positive Rate	Accuracy
Google Plus One +1s	95.05%	3.43%	96.07%
Twitter Tweets	94.71%	3.25%	96.08%
Linkedin Shares	92.11%	25.36%	80.41%
Pinterest Pins	97.37%	22.08%	84.35%
Facebook Comments	90.00%	6.82%	92.13%
Facebook Likes	92.92%	10.55%	90.60%
StumbleUpon Stumbles	88.58%	37.91%	70.84%
All Social Parameters	**95.58%**	**1.61%**	**97.46%**

Table 2: Performance comparison with individual social reputation

As we can see from the results, when using only one social reputation as the feature, the result either has a relative low true positive rate or a very high false positive rate. That indicates that relying on just one source of social reputation is not reliable, probably due to the fact that even some malicious websites have very high social reputations in some social network platforms. However, when we combine these social reputation scores together, the detector is able to give us very accurate results.

To test whether our detector will have acceptable performance with less popular benign websites, we tested websites that rank after 999000 in Alexa. We treat these sites as benign. This test shows that the performance of the system, although still acceptable, decreases significantly with less popular sites. This test had a true positive rate of 87.59%, a false positive rate of 11.90%, and an accuracy of 87.97%.

We excluded Facebook Shares from our feature vector. The reason is that the detector's accuracy actually decreased when including Facebook shares. This may be because malicious actors use Facebook, the most popular social network, to promote their websites in order to make them seem legitimate. However, it is unlikely other people will like or comment on pages associated with a malicious website, which explains why Facebook likes and shares still add value to the detector.

8. POPRANK: A DETECTION SYSTEM FOR WATER-HOLING WEB SPAM LINKS

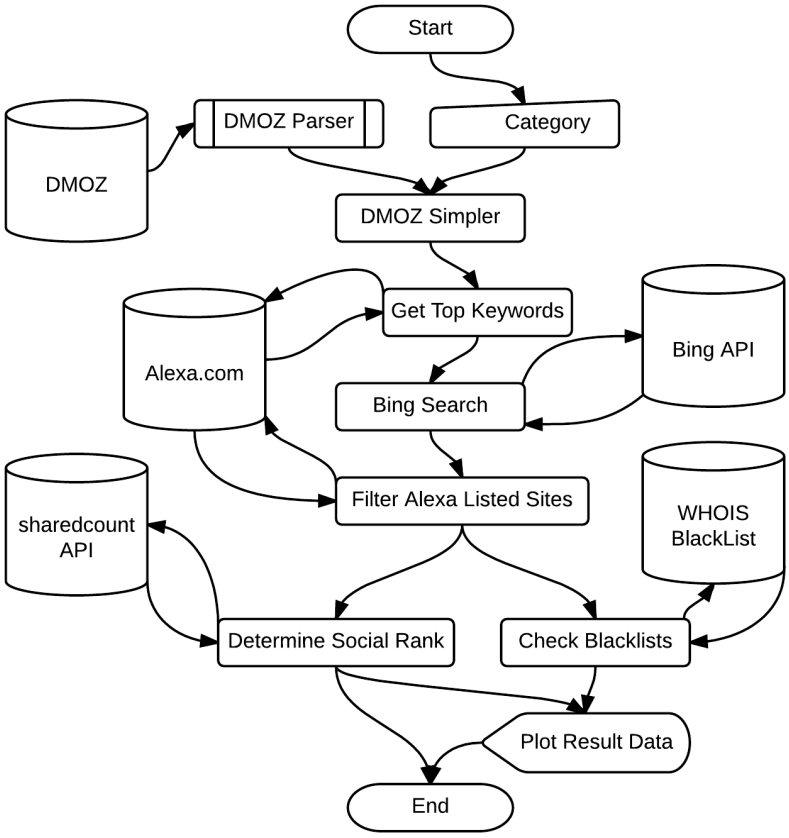


Figure 8: *PopRank* System Overflow

PopStar determined that social reputation is a key metric that can help detect malicious web pages. To test the performance of social reputation in the wild, we implemented *PopRank*, a system to rank websites that are not listed on Alexa.com and check for water-holing links. The *PopRank* system, ranks search engine results by social reputation and provides the social reputation scores together with their rankings in search engine to help the user defend against watering Attacks.

Figure 8 shows the flow of *PopRank* system. *PopRank* takes a user input industry and finds the most popular sites for that industry as determined by Alexa. It then gets the top keywords for those sites and runs searches in a popular search engine using those keywords. *PopRank* then filters the results, removing sites found in the Alexa ranking. The remaining sites are first given a rank, according to their calculated social reputation score. Then they are checked against blacklists and given a social reputation score.

To determine the top sites in an industry, *PopRank* uses the DMOZ open web directory project. DMOZ provides a publicly downloadable database of industries and links to websites within an industry. This database is in RDF/XML format. *PopRank* parses the DMOZ database, pulling links associated with the desired industry. It then use these links to fetch the Alexa page associated with each domain. Alexa is a useful site not only because it provides a ranking, but also because it provides a list of the top keywords for a website according to the percent of search traffic. Unfortunately, Alexa does not have a free API to get the keywords for a domain.

Therefore, *PopRank* scrapes the top 5 keywords from the source code returned when it fetches the associated Alexa page. It then conducts a frequency count on all the returned keywords to determine the 3 most import keywords for the industry. Table 3 shows the top keywords associated with each industry.

Industry	Football	Food	Energy	Education	Environment
#1 Keyword	ps4	food	energy	university	green
#2 Keyword	one	dog	oil	oxford	building
#3 Keyword	xbox	pet	gas	architecture	cleaning

Table 3: Top Keywords

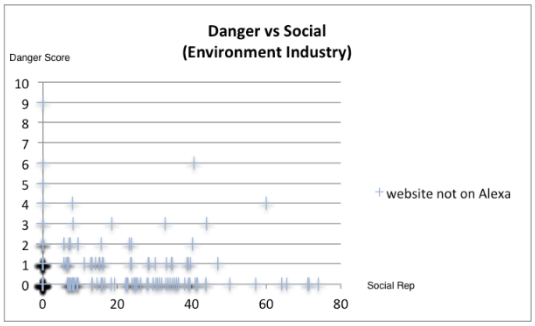


Figure 9: Danger Score versus Social Rep (for Environment Industry)

Using these top keywords, *PopRank* executes a web search for the top 1000 results using Bing, a popular search engine. It removes sites that appear in Alexa's current ranking by querying Alexa for each site and taking off the site if Alexa returns a rank. The remaining sites are then ordered based on average rank in the search engine results.

We calculate the remaining sites' social reputation score, as mentioned in 4.1 and 4.2, and then rank them descending according to it, as *PopRank* result differing from the old Bing search ranking. To test the *PopRank* system, we refer to the WHOIS blacklist check and define the danger score of a domain as how many blacklist databases that domain appears in. Our result shows that, among the websites not listed on Alexa, those sites with lower social reputation score are more likely to receive a higher danger score. See Figure 9.

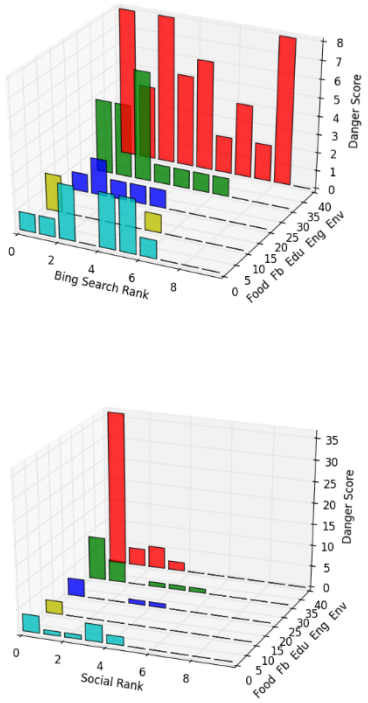


Figure 10: Distribution of BlackListed Sites in Bing Search Rank and *PopRank*

Figure 10 provides intuitive sense of difference between *PopRank* system and the Bing search engine. The overall trend of five industries' websites in *PopRank* system is that potential water-holing sites (with high danger score) tend to have low social reputation scores, which mean low ranks in *PopRank* system. However, in Bing search results, there is a significant amount of

potential water-holing sites appears in high-ranking positions. This comparison between Bing search rank and *PopRank* proves that *PopRank* system has a better prediction on dangerous websites.

9. DISCUSSION

9.1. Limitations

An attacker could evade detection from *PopStar* and *PopRank* by artificially increasing his social reputation scores. This could be done through creation of fictitious accounts on various online social network platforms and using those accounts to increase the social reputation of the malicious site. While this is not a major threat now (Facebook estimates that less than 1.5% of their accounts are fictitious), it must be taken into account. Likewise, benign sites with low social reputation scores may appear as malicious. For this reason, we recommend using *PopStar* in series with other filters.

9.2 Future Work

This paper provides a proof-of-concept for *PopStar*. However, more extensive testing is required. An ongoing project is to remove the agility features while retaining the social reputation features and adding URL, WHOIS, DNS, and BGP features to create a robust system that does not need to visit a page to rank a site as benign or malicious with high probability.

The *PopRank* system holds much promise. 38% of sites identified as potential waterholes appeared on one or more blacklists. Future development on *PopRank* can be dichotomized. First, *PopRank* can be further developed to reduce false positive rate by identifying the commonalities between the malicious water-holing sites. If these sites have attributes in common, it would be possible to develop a classifier focused specifically on detection of industry-focused water-holing sites. On the other hand, *PopRank* is potentiated to develop into a fully functioned search engine with incorporation with not only social reputation but also more user behaviors, such as average page dwell time.

10. CONCLUSION

This paper presented *PopStar* a system for labeling web pages as benign or malicious based on the page's social reputation and site agility. We defined site agility with two metrics. The first is visual agility, which denotes how much the rendered appearance of a web page changes over time. The second is source code agility, measured as the change in a page's source code over time. The final metric is social reputation measured as the weighted value of online social engagement in five popular social websites. Based on this information, *PopStar* is able to classify a site as benign or malicious with high accuracy. In tests, *PopStar* produced up to a 97.7% true positive rate with a low 1.7% false positive rate. To the best of our knowledge, *PopStar* is the first system to use social reputation and site agility to classify a website as benign or malicious.

Additionally, we present a novel way to check for *water-holing* links returned by a search engine results page (SERP). *PopRank* checks the most popular search terms for an industry and then analyzes the returned sites' social reputations to return whether or not a site is likely benign or malicious.

REFERENCES:

[1]: Provos, Niels, et al. "The ghost in the browser analysis of web-based malware."Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. 2007.

[2]: Mavrommatis, Niels Provos Panayiotis, and Moheeb Abu Rajab Fabian Monrose. "All your iframes point to us." 17th USENIX Security Symposium. 2008.

[3]: Nadji, Yacin, Prateek Saxena, and Dawn Song. "Document Structure Integrity: A Robust Basis for Cross-site Scripting Defense." NDSS. 2009.

[4]: Kapravelos, Alexandros, et al. "Revolver: An Automated Approach to the Detection of Evasive Web-based Malware." USENIX Security. 2013.

[5]: Canali, Davide, Marco Cova, Giovanni Vigna, and Christopher 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, pp. 197-206. ACM, 2011.
[6]: Ma, Justin, et al. "Beyond blacklists: learning to detect malicious web sites from suspicious URLs." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.
[7]: Eshete, Birhanu, Adolfo Villafiorita, and Komminist Weldemariam. "Binspect: Holistic analysis and detection of malicious web pages." Security and Privacy in Communication Networks. Springer Berlin Heidelberg, 2013. 149-166.
[8]: Lim, Lipyeow, et al. "Characterizing web document change." Advances in Web-Age Information Management. Springer Berlin Heidelberg, 2001. 133-144.
[9]: Chau, M., & Chen, H. (2008, 12). A machine learning approach to web page filtering using content and structure analysis. Decision Support Systems, 44(2), 482-494.
[10]: Wu, Baoning, and Brian D. Davison. "Identifying link farm spam pages." Special interest tracks and posters of the 14th international conference on World Wide Web, pp. 820-829. ACM, 2005.
[11]: Cadish, Boris, and Zinovy Diskin. "Algebraic Graph-Oriented Category Theory Based." Manifesto of categorizing database theory. TechReport 9506 (1994).
[12]: Araujo, Lourdes, and Juan Martinez-Romo. "Web spam detection: new classification features based on qualified link analysis and language models." Information Forensics and Security, IEEE Transactions on 5, no. 3 (2010): 581-590.
[13]: Egele, Manuel, Clemens Kolbitsch, and Christian Platzer. "Removing web spam links from search engine results." Journal in Computer Virology 7.1 (2011): 51-62.
[14]: Castillo, Carlos, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. "Know your neighbors: Web spam detection using the web topology." In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 423-430. ACM, 2007.
[15]: Karunakaran, K. Priya, and Seema Kolkur. "Review of Web Spam Detection Techniques." International Journal of Latest Trends in Engineering and Technology (IJLTET) 2, no. 4 (2013).
[16]: Surowiecki, James. The wisdom of crowds. Random House LLC, 2005.