PhishMon: A Machine Learning Framework for Detecting Phishing Webpages

Conference Paper · November 2018

DOI: 10.1109/ISI.2018.8587410

3 authors, including:

Amirreza Niakanlahiji
University of North Carolina at Charlotte
10 PUBLICATIONS **55** CITATIONS

SEE PROFILE

Ehab Al-Shaer
University of North Carolina at Charlotte
222 PUBLICATIONS **4,166** CITATIONS

SEE PROFILE

PhishMon: A Machine Learning Framework for Detecting Phishing Webpages

Amirreza Niakanlahiji
Software and Information Systems
UNC Charlotte
aniakanl@uncc.edu

Bei-Tseng Chu
Software and Information Systems
UNC Charlotte
billchu@uncc.edu

Ehab Al-Shaer
Software and Information Systems
UNC Charlotte
ealshaer@uncc.edu

***Abstract*—Despite numerous research efforts, phishing attacks remain prevalent and highly effective in luring unsuspecting users to reveal sensitive information, including account credentials and social security numbers. In this paper, we propose PhishMon, a new feature-rich machine learning framework to detect phishing webpages. It relies on a set of fifteen novel features that can be efficiently computed from a webpage without requiring third-party services, such as search engines, or WHOIS servers. These features capture various characteristics of legitimate web applications as well as their underlying web infrastructures. Emulation of these features is costly for phishers as it demands to spend significantly more time and effort on their underlying infrastructures and web applications; in addition to the efforts required for replicating the appearance of target websites. Through extensive evaluation on a dataset consisting of 4,800 distinct phishing and 17,500 distinct benign webpages, we show that PhishMon can distinguish unseen phishing from legitimate webpages with a very high degree of accuracy. In our experiments, PhishMon achieved 95.4% accuracy with 1.3% false positive rate on a dataset containing unique phishing instances.**

***Index Terms*—Anti Phishing, Machine Learning Framework**

I. Introduction

Today, phishing, a type of social engineering attack, is one of most common attack types used by cyber attackers to lure unsuspecting users to disclose their sensitive information, such as user credentials or credit card information. According to the Anti-Phishing Working Group (APWG), more than 1.2 million phishing attacks are documented in 2016, a 65% increase over 2015 [16]. Moreover, these attacks have evolved over time and become increasingly more advanced as phishers attempt to make the look of their phishing webpages and corresponding URLs as similar as possible to target websites while utilizing various evasion techniques to circumvent existing phishing detection mechanisms.

Phishers use various techniques to convince unsuspecting users by using valid SSL certificates [2], utilizing URL hijacking techniques such as typosquatting [19], and scraping information from target webpages. Phishers also use techniques to mislead existing phishing detection systems. This includes techniques such as using the image of the target webpage instead of reusing its HTML code, using old registered domains, and moving from one domain to another on a regular basis.

978-1-5386-7848-0/18/$31.00 ©2018 IEEE

In recent years, many phishing detection systems have been proposed to combat the increasing number of phishing threats. These systems rely on a combination of features extracted from various sources such as search engines [7], [17], public blacklists [15], DNS records [7], URLs [18], Webpages [18], [24], and SSL certificates [8]. Despite achieving high accuracy, existing phishing detection systems suffer from several shortcomings that limit their applicability: 1) dependency on third party services such as search engines [22], [24] and WHOIS servers [22] introduces concerns of availability, cost, privacy, and performance 2) limited detection scope due to comparing the URLs or webpages under investigation with a whitelist of potential targets [5] or a blacklist of known phishes [23] 3) language dependency as they rely on features extracted from textual content of webpages [24] or URL [17].

In this paper, we propose `PhishMon`, a new scalable feature-rich machine learning framework for detecting unseen phishing attacks in a real-time fashion. It relies on a set of eighteen salient features, fifteen of which are new features that characterize how the webpage is put together, such as certificate validity scope, number of external script blocks, that can be collectively used to discern phishing webpages from the legitimate ones with a high degree of accuracy. `PhishMon` exploits features extracted from HTTP responses, SSL certificates, HTML documents, and JavaScript files when a given URL is loaded by a web browser. These features collectively reveal characteristics of technology used to build and host a webpage. Such characteristics add significant cost obstacles for phishers if they want to evade detection. In summary, our proposed system has the following properties:

- **System independency.** `PhishMon` does not depend on any third-party system to make a decision. Relying on third-party services such as search engines can impose some limitation on the applicability of a phishing detection system. First, it can prolong the decision process; thus making the approach unsuitable for online detection. Second, using a third-party system can be costly in case of requiring a subscription. Third, sometimes finding the right provider is impossible, for example, not all domain registrar provide WHOIS records. Forth, using a third-party system can raise privacy concerns as the queries leak some information about the page under investigation.

- **Broad coverage.**. PhishMon can detect unseen phishing campaigns masquerading as previously unknown targets or brands as it makes a decision based on intrinsic features shared among phishing attacks and not based on features that measure the similarity of a given webpage with a curated list of legitimate or phishing webpages.
- **Language agnostic.** Classifiers that derive features from textual content are bound to a specific language such as English in which they were trained and are not effective against phishing attacks targeting websites in other languages such as Chinese. None of the features used by PhishMon are derived from the textual content of the webpage; hence making the approach language agnostic.

In the rest of paper, first, we present underlying features that PhishMon utilizes to detect phishing pages in Section II. Then, in Section III, we provide a comprehensive evaluation of PhishMon. In Section IV, we compare PhishMon with existing systems. In Section V, we conclude the paper.

II. Feature Selection

In this section, we describe the eighteen features, fifteen of which are new, that PhishMon uses to decide whether the webpage pointed by a given URL is a phish. We start with the observation that the average lifetime of a phishing webpage is short, measured in hours [6], [11]. Phishers need to constantly create new webpages before getting blacklisted either based on URL or content. To reduce cost, phishers tend to focus their efforts on webpages' appeal to victims and do not pay much attention to web technologies and infrastructures used to develop and host their web applications. In contrast, legitimate business websites are increasingly becoming more technology savvy and pay attention to issues such as security of web applications, quality of service, and following best development practices to enhance code maintainability.

We have identified the followings are often used by business websites targeted by phishers: (I) security techniques to protect users from client-side attacks such XSS and CSRF, (II) security techniques to prevent web scraping, (III) security techniques to protect users traffics from sniffing, (IV) tracking techniques to monitor user activities, and (V) techniques to reduce the loading time of webpages. In addition, website developers use best practices, such as separation of JavaScript code from HTML content, to improve the maintainability of their applications. Based on these observations, we propose three groups of features, namely HTTP, code complexity, and certificate features, to characterize the techniques, mechanisms, and technologies that are used by websites. These features indirectly measure the developmental efforts and technological investments associated with websites. They would require substantial resources for phishers to mimic.

To ensure the effectiveness of proposed features in distinguishing phishes from legit websites, we conduct a series of experiments, the results of which shown in Fig. 1, on a dataset containing 2,064 phishing and 17,508 legitimate webpages; a subset of the larger dataset described in Section III-A.

A. HTTP Features

Websites use HTTP headers to pass additional information to web browsers. Exchanged HTTP headers can reveal information about the underlying web application and its technology infrastructure. We observed that the underlying web technology stacks are considerably different between phishing and legitimate websites. Their owners have different objectives that influence their spending on technology and infrastructures. To capture such differences, we define the following features. Please note that a star in front of a feature means that, to the best of our knowledge, we are the first one who propose it for detecting phishing.

HTTP header field names*. We consider all the header names appeared in an HTTP response as a feature. To represent this feature, we use a vector space model [10]. In this model, an HTTP response is represented as a vector. Each dimension in this vector corresponds to a specific header name. If a header name appears in an HTTP response, its value in the vector will be the length of the corresponding field value measured in bytes.

Number of header fields*. We count the number of headers appeared in the header section of the HTTP responses. On average, the number of header fields in a response of legitimate websites is greater than the phishing ones.

Number of non-standard header fields*. We compute the number of non-standard header fields in the header section. These headers are not on the list of standard HTTP header names provided by IANA. As of writing this paper, this list contains 329 permanent header names.

B. Code Complexity Features

These features attempt to capture the fact that a full-fledged website offers a variety of functionalities to its users, some of which are triggered by users' actions, and the rest are triggered by other events such as timeout. Phishers, on the other hand, mainly concern about visual aspect of the webpages by mimicking the target UI as much as possible to mislead the unsuspecting users. As a result, the complexities of the JavaScript codes included in phishing pages are significantly less than the targets. Researchers have proposed various metrics including lines of code, number of functions, and cyclomatic complexity to predict less maintainable [3] or more vulnerable applications [20]. However, in this work, we utilize such measurements to determine the codes that offer more functionalities to end users.

Many modern web applications rely on off-the-shelf libraries, such as JQuery, JQuery UI, and Bootstrap to build their user interfaces. To measure the code complexity of a webpage, we must ignore such libraries as they are implemented by third-party developers. To recognize such libraries, we collected the top 200 JavaScript libraries (8637 versions of them until Oct 2017) listed by cdnjs website and computed the context triggered piecewise hashes (CTPH) for all of their JavaScript files using ssdeep library [13]. PhishMon relies on this list of CTPH values to determine whether a JavaScript file is part of a public library.

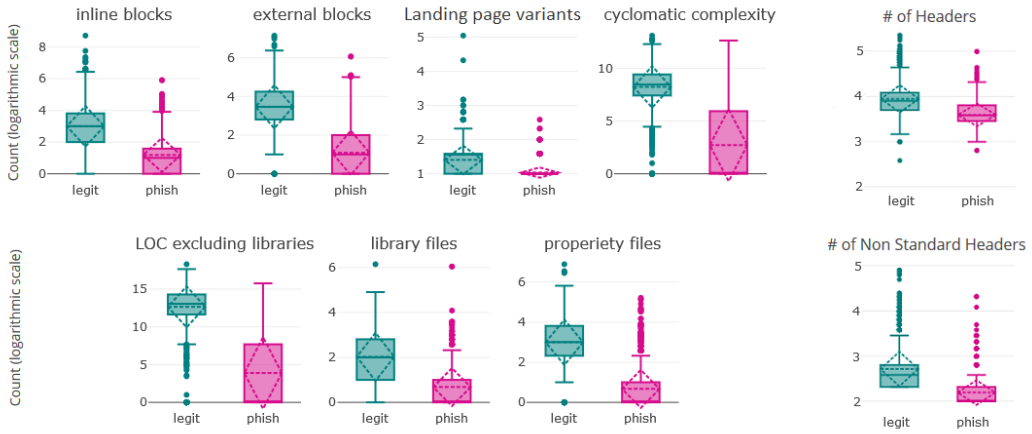

Fig. 1. Comparison of Web UI code complexity and HTTP metrics between phishing and legitimate websites.

Minified/Obfuscated*. A common practice among legitimate websites is the use of code minifiers [21], or obfuscators to both reduce the page loading time and protect the front-end code against web scrapers. As mentioned by several researchers [13], usage of code obfuscation is also fairly common among some types of malicious webpages such as drive-by-download webpages; however, we observed that a large percentage of phishing pages do not use any of such techniques. We use the following set of features to determine whether a script file is minified or obfuscated: the ratio of white spaces to all printable characters, the average length of variable names, and the average length of function names.

Number of external script blocks*. A common practice to enhance code maintainability is to store JavaScript code in separate files from the HTML documents and reference such files in HTML documents using external script blocks. We count the number of script blocks in the landing webpage.

Number of inline script blocks*. Despite objections against usage of inline script blocks due to security concerns, they are still commonly used even in popular websites such as Google and Amazon. As in this way, developers can reduce the number of round trip times (RTTs) required for loading external script files; thus, significantly reducing the page loading time. This feature measures the number of the inline script blocks in the base HTML document.

Number of DOM on-event handlers*. HTML on-event attributes, such as onclick and onload, allow developers to directly register JS event handlers on HTML elements, such as body and image, in an HTML document. These handlers will be invoked when the DOM events of interest have occurred on the specified elements. Unlike inline script blocks, developer best practice refrain from using on-event handlers since placing such handlers inside HTML elements can make the code significantly less maintainable. However, on-event handlers are handy tools for rapid prototyping; hence more often used by mock or phishing webpages.

Number of JavaScript libraries*. We also count the number of JS libraries that are referenced in the HTML document of the landing webpage. As we mentioned earlier, we rely on our JavaScript library detector that recognizes more than 8,600 different versions of popular libraries.

Number of Landing Page Variants*. Many commercial web applications use JavaScript and Ajax calls to dynamically change the look and the content of a webpage without refreshing the page. This feature captures the number of times the base HTML document is changed by Ajax calls during the observation period.

Lines of Code (LOC)*. We compute the average number of lines in external JavaScript files, excluding libraries. This is intended to measure the amount of developer effort spent of building the website. To ensure that the LOC value calculated from different files reflect the number of statements, we use a reformatter, jsbeautifier, to format the code first.

Is URL redirected*. This feature shows whether the initial URL is redirected to other effective second-level domains during the observation period.

C. *Certificate features*

We extract a set of features from X.509 certificates provided by websites hosted over HTTPS.

Has X.509 certificate. This feature indicates whether the website is hosted over HTTPS.

Is passing browser validation*. This feature indicates whether the provided X.509 certificate is valid and is issued for the requested domain. A valid certificate must not be expired or revoked at the time of access. It also must have a valid signature. Moreover, the certificate of CA that signed the certificate must be valid. This chain of valid certificates must end with a valid root certificate that is trusted by the validator. To determine the validity of a certificate, we rely on the internal web browser used by PhishMon.

Valid on multiple 2-level domain names*. This feature shows the number of second-level domain names that this certificate is valid for. To extract this feature, we count the number of distinct second-level domain names declared in the Subject Alternative Names extension of an X.509 certificate.

Extended Validation (EV) Certificate*. This feature indicates whether the certificate is an Extended Validation (EV) certificate. To obtain an EV certificate for a domain name, one needs to go through a standardized vetting process to prove they legally own the domain.

Name of the Issuer*. This feature indicates the names of Certificate Authority (CA) that issued the certificate.

Certificate Longevity Period. This feature indicates the number of days a given certificate is valid.

Certificate Age. This feature indicates the number of days past from the issuance date.

III. Evaluation

In this section, we present our evaluation of PhishMon, in which we seek to answer the following research questions. First, how effective is our approach in detecting unseen phishing webpages? Second, what is the contribution of each feature in the proposed detection model? To answer these questions, we first compare the performance of several well-known classifiers when they are trained on a large dataset consisting of both phishing and legitimate websites. Then, we examine the power of prediction for each of the proposed feature in our system. We also report the performance of our system when a subset of features is considered.

A. Dataset

To study our approach, we collected a large dataset containing $17,508$ legitimate and $4,807$ unique phishing webpages. In this dataset, legitimate pages are homepages of the popular websites selected randomly from the Alexa top one million domain name list, and phishing pages are distinct confirmed phishing instances reported by PhishTank, a community-driven website for sharing and validating phishing URLs.

Initially, we selected randomly 20,000 from Alexa top one million domain name list. We then excluded the domains appeared in malwaredomains.com and networksec.org blacklists. Next, we visited the homepage of the remaining domains with our web scraper to form the dataset of legitimate webpages. Further, we removed webpages that contain certain phrases indicating the site is under construction, not functional, or not supporting the web engine used by our customized web scraper. In this way, we obtained $17,508$ legitimate webpages.

We also collected live phishing webpages by monitoring PhishTank for four weeks between September and November 2017. We fetched urls within five minutes after their publication. To ensure that the collected links are valid phishes, we cross-check them with VirsusTotal about two weeks after their collection. We found many phishing URLs are hosted on the same domains, or their webpages having almost identical HTML documents. Such phishing URLs may belong to the same phishing campaign. We took the following steps to avoid biasing the classifier toward fitting larger phishing campaigns. First, for all phish URLs that share the same domain, we picked only one of them and ignored the rest. Second, we filtered duplicate phishing webpages by comparing fuzzy hash values of their HTML content. In this way, we obtained 4,807 phishing webpages during our monitoring period.

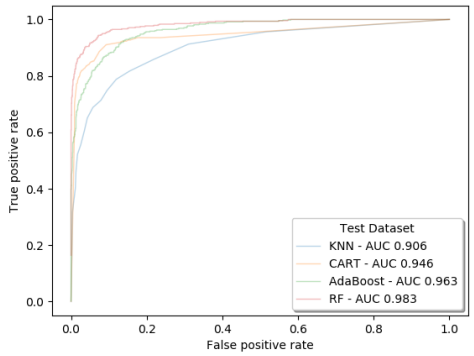

Fig. 2. ROC curves of different classifiers trained on our dataset

TABLE I
RF over all the proposed features (Accuracy: 95.4%, False Positive Rate: 1.3%)

n=22460	Predicted: Benign	Predicted: Phish	
Actual: Benign	TN=17419	FP=231	17650
Actual: Phish	FN=794	TP=4016	4810
	18213	4247	

B. Selection of Machine Learning Algorithm

We formulate the phishing detection problem as a classification problem in which our aim is to determine whether an input URL is a phish. To choose a classifier, we performed a series of experiments on four standard machine learning algorithms, namely CART, K Nearest Neighbors (KNN), AdaBoost, and Random Forest (RF). All of these classifiers were trained on the dataset described earlier in Section III-A.

We adopted the stratified 10-fold cross-validation strategy to estimate the performance of the classifiers under evaluation. The reason that we picked this strategy is twofold. First, this strategy is commonly used by many researchers such as [1], [6], so selecting this strategy makes it easier to compare our results with related works. More importantly, as it is shown in [12], the stratified 10-fold cross-validation, in general, tends to provide a less biased estimation of the accuracy.

We also use the Area Under the ROC Curve (AUC) [9] as a performance measure to compare the classifiers. The ROC curve for a binary classifier is a plot that depicts the relation between false positive rate and true positive rate when different probability thresholds are used by the classifier to make decision. As stated in [9], the AUC of a classifier can be interpreted as the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance by the classifier. Figure 2 depicts the ROC curves [9] of the candidate classifiers when stratified 10-fold cross-validation approach is employed. As it is easily observable, the AUC for RF classifier is larger than the other classifiers; which means it can reach to a lower false positive rate while keeping the true positive rate higher.

TABLE II
RF CLASSIFIER TRAINED ON CERTIFICATE FEATURES.(ACCURACY: 92.6%, FALSE POSITIVE RATE: 0.6%)

n=8400	Predicted: Benign	Predicted: Phish	
Actual: Benign	TN=7035	FP=45	7080
Actual: Phish	FN=574	TP=746	1320
	7609	791	

TABLE III
RF CLASSIFIER TRAINED ON CODE COMPLEXITY FEATURES (ACCURACY: 91.2%, FALSE POSITIVE RATE: 4%).

n=22300	Predicted: Benign	Predicted: Phish	
Actual: Benign	TN=16755	FP=755	17510
Actual: Phish	FN=1190	TP=3600	4790
	17945	4355	

TABLE IV
RF CLASSIFIER TRAINED ON HTTP HEADER FEATURES (ACCURACY: 93.2%, FALSE POSITIVE RATE: 2.7%).

n=22300	Predicted: Benign	Predicted: Phish	
Actual: Benign	TN=17021	FP=489	17510
Actual: Phish	FN=1025	TP=3765	4790
	18046	4254	

Table I shows the detail performance of RF algorithm on our dataset when stratified ten-fold cross-validation is performed. It worth noting that the numbers are the summation of ten test runs. TN, FN, FP, and TP in this table stand for true negative, false negative, false positive and true positive respectively.

C. Impact of Training Size

We also evaluate the accuracy of the RF classifier when it is trained with different percentage of the dataset. We trained the classifier with only 50% to 90% percent of the dataset and kept the remaining 50% to 10% of the dataset for testing. As the percentage grows, the area under ROC curve increases. However, increasing the training size has a marginal effect on the performance of the classifier. In our test, we use 90% for training and 10% for testing. In our dataset the ratio of phishing instances is about 0.2.

D. Performance of Features

In this section we first evaluate each category of features presented in section II individually to determine their power of prediction in detecting phishing URLs. Next, we evaluate the contribution of each feature on the overall performance of PhishMon classifier. In other words, we identify the most effective features in `PhishMon` for identifying phishes. These insights can help future feature development and also help researchers to better understand how might phishers react to evade our detection system. We trained a RF classifier on top of each feature category and measured the performance of these classifiers by using stratified 10-fold cross-validation approach. Tables II to IV show the overall performance of three classifiers trained with certificate, code complexity, and HTTP header features. They achieved 92.6, 91.2, 93.2 percent accuracy on our dataset. These results show that these feature sets roughly have the same power in predicting phishing webpages; in other words, none of these group of features are dominant. To evaluate the contribution of each feature on phish detection, we calculated the Mean Decrease Impurity (MDI) importance [14] of each feature in our classifier when it is trained on our dataset. Table V shows the top 10 most important features in our classification model.

TABLE V
TOP 10 MOST IMPORTANT FEATURE BASED ON MDI

No.	Feature	Feature Group
1	Avg cyclomatic complexity	Code complexity
2	LOC of external blocks	Code complexity
3	Avg number of external blocks	Code complexity
4	Set-Cookie	HTTP response header
5	Number of landing page variants	Code complexity
6	Number of HTTP headers	HTTP response header
7	Proprietary code count	Code complexity
8	Is URL redirected	Dynamic content
9	Avg number of inline blocks	Code complexity
10	Keep-Alive	HTTP response header

IV. RELATED WORK

Zhang *et al.* [24] proposed `CANTINA`, a content-based approach in which the key terms are extracted from a given webpage and fed into a search engine such as Google to examine whether the URL for the page appears in the top N search results. Xiang *et al.* proposed `CANTINA+` [22], a successor of `CANTINA`, which relies on fifteen distinctive features to identify phishing websites. It achieved 90% TP and about 0.4% FP on a dataset with 10% unique training phish instances. However, to achieve this level of accuracy, it relies on third-party services namely search engines and WHOIS servers to compute five of the proposed features. Furthermore, CANTINA+ relies on some HTML-based features such as whether a page contains `bad forms`, `bad action fields` that can be easily manipulated by an attacker; for example, using images of text to avoid their text-based detectors.

Marchal *et al.* [17] proposed PhishStorm, which is a phish detection system that analyzes a given URL by extracting features from the words in the URL and querying `Google Trends` and `Yahoo Clues`. The system achieved a correct classification rate of 94.91% with only 1.44% false positives on a dataset consisting of 96018 phishing and legitimate URLs. Similar to the two previous works, this system relies on external systems.

Chen *et al.* [5] proposed a phishing detection system that calculates the similarity score of a suspicious webpage with a list of legitimate webpages. The system compares the screenshot of the given page with the screenshots of all webpages on a whitelist; in case the similarity score with one of them is above a certain threshold, the page will be marked as a phish for that whitelisted webpage. In their experiment, the system could reach 95% to 98% percent accuracy for different legitimate webpages. The main advantage of their system is that they are not relying on the HTML content of a webpage to determine whether it is a phish; hence evasion techniques such

as using images instead of texts will be ineffective. However, to determine whether a page is a phish, the system needs to compare it with all the webpages in the whitelist which may contain millions of pages. In addition, this system can only detect phishing attacks against legitimate websites on a given whitelist; which means phishing attacks against unlisted legitimate websites can slip under the radar. In our approach, we do not compare with a list of legitimate webpages; yet we can detect a phishing page that use images instead of text due to decreasing of the complexity.

Chang *et al.* [4] proposed a system which utilizes Google Image Search service to identify the website identity based on the website logo. The system is resilient against image-based evasion techniques and can achieve 87% TP and 30% FP on a dataset containing segmented images of webpages. The main limitation of the system is its performance as it sends a number of image queries to Google Image Search and also to Google Search Engine. Also, due to the difficulty of logo extraction, the detection accuracy is less than other approaches.

Dong *et al.* [8] proposed a real-time phishing detection system that can detect phishing webpage host on HTTPS-enabled web servers. It extracts 42 features from X.509 certificates and reaches a recall of 95.5% in detecting phishings category, with a precision of 93.7 on average. The main limitation of the system is that it can only detect phishing websites hosted over HTTPS. We also extract seven salient features, three of which are new ones, from X.509 certificates. However, our system extract features from other sources such as HTML and JavaScript code; which enable it to detect phishing webpages regardless of underlying communication protocol.

V. Conclusion

In this paper, we proposed PhishMon, a feature-rich machine learning system to detect phishing websites. It relies on eighteen salient features, fifteen of which are new, to decide whether the webpage pointed by a given URL is a phish. These features can be efficiently calculated, without requiring interaction with any third-party system, which can prohibitively delay the decision-making process. They capture various characteristics of the web application and its underlying infrastructure. By using these features, we indirectly measure the amount of effort invested in development and deployment of a web application, which is remarkably different between legitimate and phishing websites. We also conducted extensive experiments on a real-world dataset containing confirmed phishing and legitimate instances harvested from the Internet between Sept. and Nov. 2017. Our results show that PhishMon achieves a high degree of accuracy in distinguishing legitimate webpages from unseen phishing pages without raising many false alarms: on the collected dataset, PhishMon reaches a 95.4% detection rate with 1.3% false positive.

References

[1] Maher Aburrous, M Alamgir Hossain, Keshav Dahal, and Fadi Thabtah. Predicting phishing websites using classification mining techniques with experimental case studies. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 176–181. IEEE, 2010.

[2] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *USENIX security symposium*, volume 13, 2013.

[3] Rajendra K. Bandi, Vijay K. Vaishnavi, and Daniel E. Turk. Predicting maintenance performance using object-oriented design complexity metrics. *IEEE transactions on Software Engineering*, 29(1):77–87, 2003.

[4] Ee Hung Chang, Kang Leng Chiew, Wei King Tiong, et al. Phishing detection via identification of website identity. In *IT Convergence and Security (ICITCS), 2013 International Conference on*, pages 1–4. IEEE, 2013.

[5] Kuan-Ta Chen, Jau-Yuan Chen, Chun-Rong Huang, and Chu-Song Chen. Fighting phishing with discriminative keypoint features. *IEEE Internet Computing*, 13(3), 2009.

[6] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security (TISSEC)*, 16(4):16, 2014.

[7] Hyunsang Choi, Bin B Zhu, and Heejo Lee. Detecting malicious web links and identifying their attack types. *WebApps*, 11:11–11, 2011.

[8] Zheng Dong, Apu Kapadia, Jim Blythe, and L Jean Camp. Beyond the lock icon: real-time detection of phishing websites using public key certificates. In *Electronic Crime Research (eCrime), 2015 APWG Symposium on*, pages 1–12. IEEE, 2015.

[9] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[10] Wikipedia Foundation. Vector space model. https://en.wikipedia.org/wiki/Vector_space_model, 2017.

[11] R Gowtham and Ilango Krishnamurthi. A comprehensive and efficacious architecture for detecting phishing webpages. *Computers & Security*, 40:23–37, 2014.

[12] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[13] Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97, 2006.

[14] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[15] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245–1254. ACM, 2009.

[16] Ronnie Manning and G Aaron. Phishing activity trends report. *Anti Phishing Work Group, Tech. Rep. 4th Quarter 2016*, 2016.

[17] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, 2014.

[18] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, pages 323–333. IEEE, 2016.

[19] Tyler Moore and Benjamin Edelman. Measuring the perpetrators and funders of typosquatting. In *International Conference on Financial Cryptography and Data Security*, pages 175–191. Springer, 2010.

[20] Yonghee Shin, Andrew Meneely, Laurie Williams, and Jason A Osborne. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. *IEEE Transactions on Software Engineering*, 37(6):772–787, 2011.

[21] Steve Souders. High-performance web sites. *Communications of the ACM*, 51(12):36–41, 2008.

[22] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.

[23] Guang Xiang, Bryan A Pendleton, Jason Hong, and Carolyn P Rose. A hierarchical adaptive probabilistic approach for zero hour phish detection. In *European Symposium on Research in Computer Security*, pages 268–285. Springer, 2010.

[24] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.