PhishFry – A Proactive Approach to Classify Phishing Sites using SCIKIT Learn

Daniel Brites and Mingkui Wei
Sam Houston State University
Huntsville, TX

***Abstract*—Phishing is a type of malicious attack that involves the fooling of unsuspecting victims into providing or sharing personal information such as names, addresses, and banking information, which may lead to damages to the individual such as identity theft and financial losses. To combat phishing attacks, there have been many strides toward the use of newer technologies instead of conventional approaches such as personnel training and physical security. These technologies involve a proactive approach towards identifying Phishing websites that utilize machine learning and have become more and more efficient. In this paper, a more proactive and online machine learning approach is proposed that utilize features that have been well-accepted among industries and academia. Within the algorithm, prioritizing of features will be broken up into layers, and the output of the tool will be a digital tag that could be included in web browsers for quick identification and classification. If a site is tagged, the website owner will have the opportunity to legitimize the website through a detailed informational session and will allow them to fix any features that may be classified as malevolent in nature.**

Keywords— phishing, social engineering, machine learning, Phishtank, openfish, Alexa

I. INTRODUCTION

The impact of Phishing schemes has more than doubled in the recent years, and with the ongoing entanglement of the Internet makes this an even more challenging endeavor to undergo. Since 2014, phishing incidents that were reported has exploded by nearly 250% [1]. Phishing is essentially a subset of social engineering: Phishers (the ones who launch phishing attacks) utilize many different tricks in order to attain private information from users. Victims, for the most part, are wholly unsuspecting and may not have the same literacy in computers as a computer professional. The answer, though, has to be more than just conventional approaches such as personnel educating. In this regard, many other approaches have been taken in the recent decade.

Notwithstanding, there are challenges that need to be addressed, and limitations to many of the existing algorithms that still need updates and more efficient implementation [2]. Some of those challenges involve utilizing proper scanning techniques, internet etiquette, and understanding Phishing from the standpoint of the attacker instead of the victim [3]. These are just some of the challenges in the computer security industry, whereas attackers are not focused on simple etiquette techniques. Attackers utilize these limitations as a vulnerability, and often use whatever methods available to attain their goals. Phishing is just one of those that involve multiple vectors and angles onto which to use against the victim.

IN this paper, we propose to utilize a machine learning based approach to identify and classify potential phishing webpages. For this purpose, we compose our own dataset, and based on which we comprehensively evaluated various machine algorithms and features sets. Our study carries insights in combating phishing attack by leveraging machine learning algorithms.

The rest of this paper is organized as follows. In Section II we briefly introduce background and related works to this study. The main approach of the study is demonstrated in Section III through V, and in Section VI we summarize our work and propose future directions.

II. BACKGROUND AND EXISTING ALGORITHMS

Phishing, in the most common form, begins from the sending of an email by the attacker to multiple victims' email addresses. Within the email the attacker attempts to spoof itself as something that would be otherwise interested and/or trusted by the recipient of the email. Such emails may include those from financial institutions, friends, or shopping websites in which many people frequently visit. These emails are meant to either illicit personal information in a reply or direct the recipient to click on a link that would take them to the website where they will then ask for them to input their private information [4]. Although seems intuitive, Phishing has somewhat become an art form and the techniques are becoming more and more sophisticated less identifiable. For every countermeasure, there almost always seems to be a workaround. Phishing techniques can range in the forms of link manipulation, filter evasion, website forgery, and covert redirects [5].

Most of the approaches towards combatting Phishing can be classified into two main categories: proactive and reactive. In proactive approaches, the algorithm is meant to find newly created sites and classify them as either legitimate, suspicious, or malicious. Reactive approaches utilize blacklisting, which endeavor to take on reported sites and classify them very much the same way. According to Varshney et. al [4] detection is the most effective at battling Phishing, with user training and security schemes being other less effective approaches. It is within the detection section that includes algorithms that can be

considered either reactive or proactive while utilizing differing approaches in either direction.

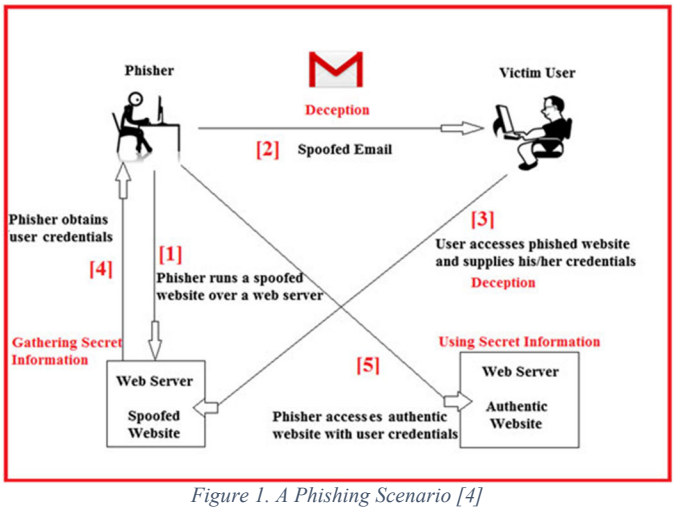

Figure 1. A Phishing Scenario [4]

There are several tools that utilize differing techniques, however the focus will be on Machine Learning, and some other techniques such as filtering which include two main directions: visual filtering and internal filtering. In 2012, Maurer et al. [6] went down the road of identification and classification through the testing of four visual detectors that included colors, layouts, color histograms, and color space division in order to differentiate between legitimate and suspicious websites. On the internal side, in 2015 Kumar et al. [7] proposed the CANTINA, which is a filter that mainly focuses on the structure and contents of the HTML, CSS, and JavaScript within a page and identifies markers that may seem malicious. This tool also takes into consideration of the URL and uses web design standards as another filtering tool within CANTINA. Works on this level had also been done in 2009, when Suriya et al. [8] identified a multi-faceted approach into the identification and classification of sites utilizing script checking and a tool named CODE SCRIPT that checks the validity of the website while tracking through a suspect email. Even earlier, in 2007, Fette et al. [9] looked at targeting emails from questionable sources and identifying through the use of the WHOIS database [10].

In the recent years, however, with the advent and popularity of machine learning, anti-phishing techniques have gone from user education and filtering techniques and moved on to a more sophisticated way of targeting Phishing websites while still branched in a form of filtering. Miyamoto et al. [11] began down the heuristic approach through the creation of a tool known as PhishCage, which is essentially an emulated version of the larger internet, in order to take a heuristic approach onto URL filtering. In 2015, Fang et al. [12] devised another heuristic approach onto filtering that added features of Phishing websites and utilized machine learning to analyze and filter through. The approach included a blacklisting method as an output from a visual filtering tool (RAY SCAN through spatial filtering).

Soon after, the need for efficiency in processing larger datasets lead to the breakthrough of designing tools that can iterate through features that were analyzed from datasets from popular locations such as Alexa, phishtank.com, and openfish.com [13] [14] [15] [16]. It is with this analysis in which the tools such as Fresh-Phish [1] and Phisher [15] utilize in order to filter through and classify through the identification of these features. Marchal et. al [15] created Phisher that included 5 separate modules that each conducted a step in the classification system: A webpage scraper, feature extractor, classifier, key-terms extractor, and target identifier. Fresh-Phish dives in deeper and creates a system that breaks into five feature categories: URL based, DNS based, External Statistics, HTML, and JavaScript based [1]. The next approach features a classification off of a large set of 212 features [17].

In 2017, Weiss et al. [18] devised an approach that featured Heterogenous Transfer Learning called Canonical Correlation Analysis. This method utilized a different and mutually exclusive data set for both the training and testing dataset from a baseline dataset. In order to break down the feature set, the Chi-Square measure was utilized to help identify relevance to the dataset and feature space. Once the feature space was aligned, they were able to utilize 11 algorithms, four of which were basic machine learning, and were not able to differentiate an advantage from tradition machine learning versus homogenous or heterogenous transfer learning.

In other works, while still utilizing machine learning, authors worked towards identifying different methods of data mining and varying sources of datasets. One method involved the use of financial industry webserver logs [19]. From the logs, URL's were extracted and brought into the system to compare and blacklist while utilizing a manual verification technique. Another method introduced SEAHOUND [20], which utilized semantic analysis centered around Natural Language processing. In 2016, Li et al. [16] used the global massive domain name registration to identify a dataset that can be filtered almost autonomously.

III. Machine Learning

In order to identify the proper algorithm in machine learning techniques, it is important to understand the overall goal of the end model. There are multiple algorithms for multitudes of uses and approaches in research, both supervised and unsupervised. These algorithms can be used for predictions, personal assisting, development, automation, and many other processes that can be made more accurate or efficient. In this case, since we are attempting to identify and classify phishing sites as such, we will utilize supervised classification algorithms. For testing and research purposes there were a partially supervised algorithms utilized. The research, however, was centered around the accuracy of the supervised classifiers.

The database employed for initial testing was from the UCI center for Machine Learning [21]. The data set from the repository is titled "Phishing Websites Data Set" which

includes 2456 instances to process through. The dataset applies a set of features that are common to phishing sites, and since the features will also be utilized in this research, the dataset fits onto the goal of detection and classification. The features, with their value possibilities labeled thereafter, are as follows:

1. having_IP_Address { -1,1 }
2. URL_Length { 1,0,-1 }
3. Shortining_Service { 1,-1 }
4. having_At_Symbol { 1,-1 }
5. double_slash_redirecting { -1,1 }
6. Prefix_Suffix { -1,1 }
7. having_Sub_Domain { -1,0,1 }
8. SSLfinal_State { -1,1,0 }
9. Domain_registeration_length { -1,1 }
10. Favicon { 1,-1 }
11. port { 1,-1 }
12. HTTPS_token { -1,1 }
13. Request_URL { 1,-1 }
14. URL_of_Anchor { -1,0,1 }
15. Links_in_tags { 1,-1,0 }
16. SFH { -1,1,0 }
17. Submitting_to_email { -1,1 }
18. Abnormal_URL { -1,1 }
19. Redirect { 0,1 }
20. on_mouseover { 1,-1 }
21. RightClick { 1,-1 }
22. popUpWidnow { 1,-1 }
23. Iframe { 1,-1 }
24. age_of_domain { -1,1 }
25. DNSRecord { -1,1 }
26. web_traffic { -1,0,1 }
27. Page_Rank { -1,1 }
28. Google_Index { 1,-1 }
29. Links_pointing_to_page { 1,0,-1 }
30. Statistical_report { -1,1 }

Explanation of features are located in the index. All of these features have been identified in previous works and are accepted within the community.

This dataset was then fit into the following algorithms utilizing the SCIKIT Learn tool on Python [22]:

- Supervised:
  - K-neighbors
  - SVC
  - Gaussian Process and Naïve Bayes
  - Decision Trees
  - Random Forests
  - AdaBoost
  - Quadratic
  - MLP
- Semi-supervised:
  - Label Propagation

Dataset optimization happened at a 25% sample rate splitting. All algorithms were set at the base solvers (where there were multiple solvers) in order to attain an initial accuracy in predicting utilizing the base method. The accuracy scores were as shown in figure 2.

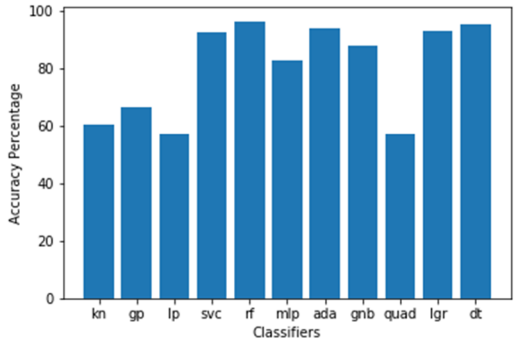

Figure 2. Base Accuracy Scores for Classifiers.

K-neighbors was adjusted for two different weight adjustments: uniform and distance. Both only reached a 60% prediction rate so this algorithm was removed. Label propagation was removed temporarily as it requires more data for learning and more iterations for uncertain predictions. SVC, Random Forests, Decision Trees, MLP (Multi-Layer Perceptron), Gaussian NB, and Adaboost (Which is an effective boost onto Decision Trees) were all kept. Removed were Quadratic and Gaussian Process due to lack of initial prediction and model fitting.

In the next round of testing, SVC was adjusted for a different kernel (linear) which has been identified to be more accurate against binary classes, rather than a multi-class dataset. This kernel was not efficient as its processing time increased by almost a minute, however it proved to be more accurate in predicting than before reaching a high of 92%. Figure shows the classifiers that will be utilized based off of their base scores and will be tuned for further accuracy.

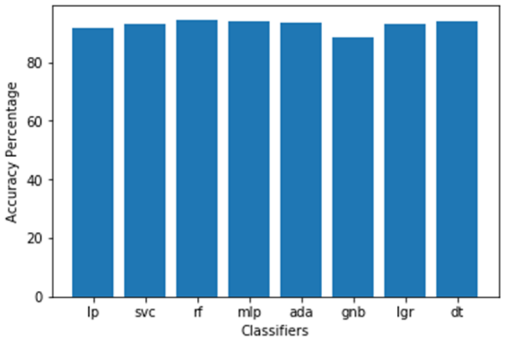

Figure 3. Accuracy Scores for Classifiers (2nd iteration).

The Logistic Regression models were only ever able to get to 93% through tuning. The MLP models through tuning achieved an accuracy of 93.33%. Gaussian (Naïve Bayes model) had a high score of 89%. The highest accuracy from the classifiers on this dataset not considered to be an ensemble were Decision Trees at 96%. The ensemble version of Decision Trees Random Forests posted a higher accuracy at 97.03%. The Decision Tree and subsequent Random Forest are included in

the index. To include another ensemble, Adaboost with Decision Tree as base estimator showed an accuracy of 95.58% after tuning, and with a Random Forest base estimator, 97.28% proving to be the algorithms of choice. Moving forward, only Decision Trees and the ensemble algorithms Random Forests and Adaboost will be used.

IV. Methodology

To continue identifying a machine learning model that maintains accuracy through multiple iterations, a newer dataset needed to be built with utilizing more URLs. Since this tool also requires an efficient processing strategy, it was ideal to split the feature list and remove the least weighted features in order to provide quicker iteration and classification without loss on accuracy. In previous works such as the Fresh-Phish framework [1], a method of dataset building was built in order to process through URLs and provide the same type of information utilized in the UCI dataset. For this tool, the base framework was used as an outline, and then rebuilt because of methods being outdated and/or permissions of tools changing (such as Google indexing). The biggest challenges lies in that many of the modules producing false positives within the dataset. Many were due to newer user-type rules, out-of-date or extinct Python modules, user agreements, connection-type errors, request time-outs, etc. Some modules were kept intact, the rest were updated to provide proper classification.

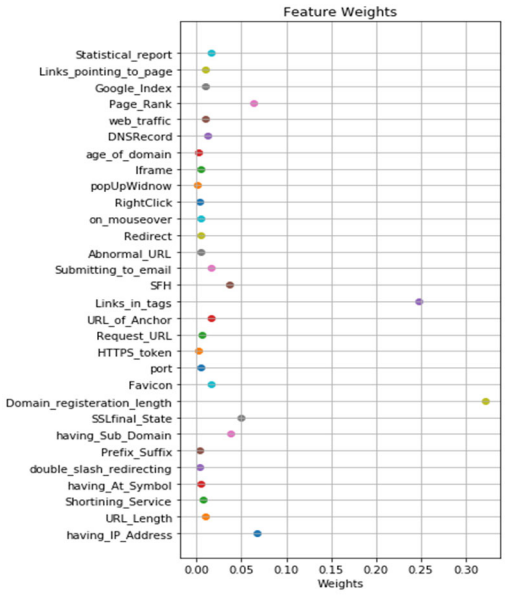

Figure 4. Feature Weights

Once rebuilt with workarounds, a dataset with 17,381 was created utilizing inputs from Phistank, Openfish, Alexa Top Sites(ATS) and 5000Best [23] [24] [25] [26]. Just like in previous works utilizing these inputs, it was assumed that Phishtank and Openfish included URLs that were reported to be phishing sites, and ATS and 5000Best were benign and considered not-phishing. 5000Best utilizes information and analytics from multiple sources such as Google, ATS, and various other platforms for determining ranking.

In determining feature weights, the most accurate performing model from the UCI dataset testing was extracted. The features are shown in figure 4. The most important feature in determining phishing was the length of the URL, while the least important feature was Pop-Up Window.

V. Design

The goal of the PhishFry tool is to create an efficient multi-layered approach for the classification of a phishing site. This approach includes being proactive; that is not waiting for a report, but to reach out and crawl the web on a regular basis while trying to locate phishing sites. This means that while many solutions are reactive in their detections, this tool is free-standing, and will utilize the models built in order to classify with pre-existing large-scale web-crawlers such as that of the AWS Common Crawl (AWSCC) [27].

AWSCC crawls the web and saves the data into separate indices that can then be utilized for research purposes. Recently, that have included a new functionality that includes a listing of just the URLs, which can then be used in other programs without having to crawl the entire crawl and saves time by allowing the tool to process the URL instead of building another tool to search through the crawl data. Utilizing the index will be important in the creation of the online version of the tool.

The tool that we are proposing is sectioned into three layers. Each layer will be focused on a different process within the overall classification. The classification model for layer one (PhishFry1) will be the same model that was utilized in building the prediction model. The second layer (PhishFry2) will involve another stage of classification that can remove any false positives. The third will be final classification and includes a future proposal for a digital signature type of tag to suspected URLs.

A. Layer One

In this layer, the classification tool named PhishFry 1 (PhishFry consists of two classification modules) which is built upon part of the final feature set. As well as utilizing features that were identified as either unimportant in the decision tree process or were not able to be built based upon various restrictions, such as Google user agreement violations and blocking of bulk requests. (Amazon also does not offer the Alexa information for free anymore, and thus lead us to find a different way of attaining the information while staying in line with the UCI feature set). The features in the most recent iteration are in Figure 4 with the added weights. Some features were rendered irrelevant even in this cycle due to not being utilized or other errors such as inaccessible or processing errors that allow for parsing through the website data.

Processing improved upon removal of the features, and some of the features that were unimportant utilizing the UCI model are now important in this feature set, mainly due to the changing of the software classification. In order to mimic the UCI set, a feature classification tool had to be built. Some features were either not utilized anymore, or some methods in

retrieving that feature may have changed. One of the most noticeable differences was the importance in having a pop-up window, which increased in the new model, but was nearly nonexistent (weight-wise) in the UCI model. Another item of import is the MLP model increased in accuracy, which joined the Decision Tree and Random Forest albeit still lower. The models by accuracy are in Figure 5. The Random Forest models are included in the appendix.

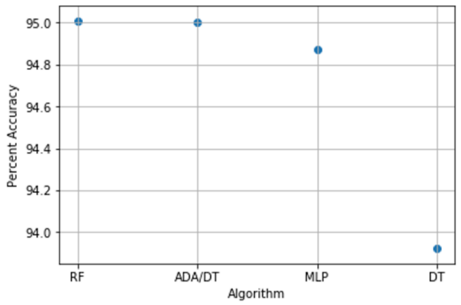

Figure 5. Models by Accuracy

This (Random Forest, based on accuracy) model will be utilized as the first layer in the tool, which then modeled with another part of the tool that grabs all of the URLs from the Common Crawl URL Index and runs through the model. The model will then predict the URLs that appear to be suspicious, and then push them to the second layer.

B. Layer Two

In this layer, some of the more process-heavy features were chosen to be in this layer to remove processing times, while also increasing accuracy. The processing quips happened when having to use Selenium (a browser emulation module for Python) in order to access specific traits of a webpage through emulation, while also utilizing workarounds (Google has come down on entities whom utilize automated software that make requests in bulk as a violation of user agreements) to attain other information that either Google removed altogether, or privacy and request permission changed. Selenium allows us to process a page through simulation, which allowed us to utilize other resources to gather that information. Those features are:

1. Google Index – Indexing in a website shows its popularity and whether Google has indexed this page.
2. iFrame – Used for cross site scripting, iFrame tag in HTML can be useful but also malicious depending on its use.
3. Google Search – Returns the results from searching the page. If Google can't find anything, it's suspicious.
4. Phishing site search – Checks the existing databases to see if the URL was already reported.

Once data has been retained for the suspicious site, the features get added to the index, and then run through the secondary prediction based off of the new data. If the site gets tagged again, it moves to Layer Three.

C. Layer Three

In this final layer of the tool proposal, the model has already been utilized to predict on whether the site is considered to be phishing or not and will be utilized mainly as a tagging layer. The tag will be a digital signature that is embedded into the site, which will then flag the site as phishing. Once the tag is applied, the owner of the page will have a process to follow that will get the tag removed. The flagging can be built into browsers and can either block the site altogether or warn the visitor of the issues involved with visiting the site. The tagging helps browsers keep unnecessary software from their code, while also warning the end-user of its malicious nature.

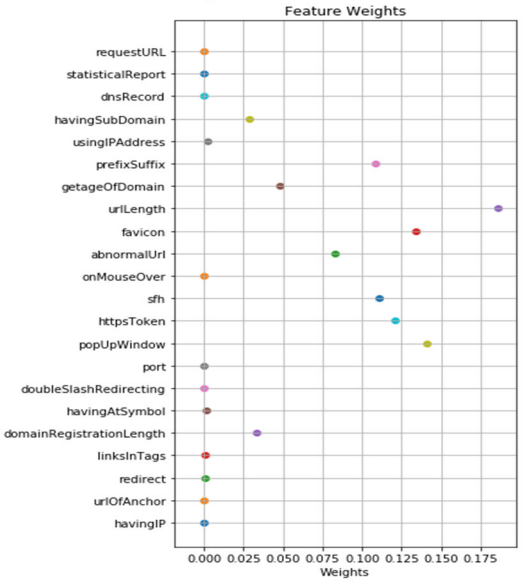

Figure 6. Feature Weights for Latest Dataset

If the tag cannot be mutually agreed upon, then this layer can be transformed into a feeder tool, which feeds the data into repositories such as Phishtank and Openfish in order to assist in preventing phishing and for future development.

VI. Future Work and Recommendations

We consider this tool to be in the research stage, and there is still a lot of work to be done, which includes increasing effectiveness of initial classifier to include newer phishing technologies and features, utilizing more classification models and tuning for a higher accuracy, and adding the digital signature tag or funnel to the end of the tool. The overall intention of the design is an intelligent learning system that can maneuver and update itself while providing a service that can protect end-users from the plight of phishing attempts. Work can also be done to add the work done by Li et al. [16] that involves utilizing URLs for the mechanism through newly registered domains which can also have a positive effect on overall processing times and reduce equipment.

In continuation to the work done by Fresh-Phish, this added design increases the overall accuracy of determining whether a site is a phishing one or not and adds other important mechanisms that will assist in the creation of a more proactive approach. The layer deployment is important because of

processing times and we were able to maintain a high degree of accuracy while eliminating features that may have been over utilized in previous iterations.

In this research, we found the Random Forests and Multi-Layer Perceptron models were the most accurate which identifies the power of ensemble versions of machine learning algorithms and may indeed help us increase accuracy with more combinations of algorithms in the near future.

References

[1] H. Shirazi, K. Haefner and I. Ray, "Fresh-Phish: A Framework for Auto-Detection of Phishing Websites," in *2017 IEEE International Conference on Information Reuse and Integration*, Fort Collins, USA, 2017.

[2] I. W. Jr., "Website Forgery: Understanding Phishing Attacks & Nontechnical Countermeasures," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, 2015.

[3] D. Kumar, Z. Ma, Z. Durumeric, A. Mirian, J. Mason, J. A. Halderman and M. Bailey, "Security Challenges in an Increasingly Tangled Web," in *2017 International World Wide Web Conference Committee*, Perth, Australia, 2017.

[4] G. Varshney, M. Misra and P. K. Atrey, "A survey and classification of web phishing detection schemes," *Security and Communications Networks,* pp. 6266-6284, 26 October 2016.

[5] R. A. Halaseh and J. Alqatawna, "Analyzing CyberCrimes Strategies: The Case of Phishing Attack," in *Cybersecurity and Cyberforensics Conference*, 2016.

[6] M.-E. Maurer and D. Herzner, "Using Visual Website Similarity for Phishing Detection and Reporting," in *CHI '12*, Austin, TX, 2012.

[7] B. Kumar, P. Kumar, A. Mundra and S. Kabra, "DC Scanner: Detecting Phishing Attack," in *2015 Third International Conference on Image Infonnation Processing*, Jaipur, 2015.

[8] R. Suriya, K. Saravanan and A. Thangavelu, "An Integrated Approach to Detect Phishing Mail Attacks A Case Study," in *SIN '09*, North Cyprus, Turkey, 2009.

[9] I. Fette, N. Sadeh and A. Tomasic, "Learning to Detect Phishing Emails," in *WWW 2007*, Alberta, Canada, 2007.

[10] NTT Communications America, "WHOIS.net," 2018. [Online]. Available: https://www.whois.net/. [Accessed 24 August 2018].

[11] D. Miyamoto, Y. Taenaka, T. Miyachi and H. Hazeyama, "PhishCage: Reproduction of Fraudulent Websites in the Emulated Internet," in *EMUTools Workshop*, Japan, 2013.

[12] L. Fang, W. Bailing, H. Junheng, S. Yushan and W. Yuliang, "A Proactive Discovery and Filtering Solution on Phishing Websites," in *IEEE International Conference on Big Data*, China, 2015.

[13] H. Shirazi, B. Bezawada and I. Ray, "“Kn0w Thy Doma1n Name”: Unbiased Phishing Detection Using Domain Name Based Features," *SACMAT '18,* pp. 69-75, 13 June 2018.

[14] G. Armano, S. Marchal and N. Asokan, "Real-Time Client-Side Phishing Prevention Add-on," in *2016 IEEE 36th International Conference on Distributed Computing Systems*, Helsinki, 2016.

[15] S. Marchal, K. Saari, N. Singh and N. Asokan, "Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets," in *2016 IEEE 36th International Conference on Distributed Computing Systems*, Helsinki, 2016.

[16] X. Li, G. Geng, Z. Yan, Y. Chen and X. Lee, "Phishing Detection Based on Newly Registered Domains," in *2016 IEEE International Conference on Big Data (Big Data)*, China, 2016.

[17] D. R. Ibrahim and A. H. Hadi, "Phishing Websites Prediction Using Classification Techniques," in *2017 International Conference on New Trends in Computing Sciences*, 2017.

[18] K. R. Weiss and T. M. Khosgoftar, "Detection of Phishing Webpages using Heterogeneous Transfer Learning," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing*, Florida, 2017.

[19] J. Hu, X. Zhang, Y. Ji, H. Yan, L. Ding, J. Li and H. Meng, "Detecting Phishing Websites Based on the Study of the Financial Industry Webserver Logs," in *2016 3rd International Conference on Information Science and Control Engineering*, China, 2016.

[20] T. Peng, I. G. Harris and Y. Sawa, "Detecting Phishing Attacks Using Natural Language Processing and Machine Learning," in *2018 12th IEEE International Conference on Semantic Computing*, USA, 2018.

[21] R. M. A. Mohammad, L. McCluskey and F. Thabtah, "Phishing Websites Data Set," UCI Machine Learning Repository, Irvine, 2017.

[22] "scikitit learn," scikit learn developers, 2018. [Online]. Available: https://scikit-learn.org/stable/index.html. [Accessed 2019].

[23] Amazon Corp., "Alexa Top Sites," Amazon, 2019. [Online]. Available: https://www.alexa.com/topsites. [Accessed 2018].

[24] OpenDNS, "Phishtank," 2019. [Online]. Available: http://phishtank.com/. [Accessed 2018].

[25] A. Paterek, "5000Best," 2019. [Online]. Available: http://5000best.com/websites/. [Accessed 2019].

[26] OpenPhish, "OpenPhish," 2019. [Online]. Available: https://openphish.com/.

[27] Amazon Web Services, "Common Crawl," Amazon, 2019. [Online]. Available: http://commoncrawl.org/the-data/get-started/.