ISSN 2278-3091
Volume 9 No.2, March -April 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse122922020.pdf
https://doi.org/10.30534/ijatcse/2020/122922020

Detection of Multi-Class Website URLs Using Machine Learning Algorithms

B.Pandu Ranga Raju[1], B.Vijaya Lakshmi[2], C.V.Lakshmi Narayana[3]
[1]Assistant Professor, Department of IT, AITS, Rajampet, Andhra Pradesh, India, balaraju.pandu@gmail.com
[2]Assistant Professor, Department of CSE, TCSC, Mumbai, Maharastra, India, vijaya.balaraju@gmail.com
[3]Assistant Professor, Department of CSE, AITS, Rajampet, Andhra Pradesh, India, cvlakshminarayana@gmail.com

ABSTRACT

Online phishing is one of the Internet's most widespread crime schemes. A common counter measure includes testing URLs against blacklists of established phishing websites, which are typically collected on the basis of manual verification and are inefficient. As the Internet scale expands, automatic URL detection is increasingly necessary to provide timely security for end-users. In this paper, we propose an efficient and versatile malicious URL detection system with a rich collection of features representing the diverse characteristics of phishing websites and their hosting platforms, including features that are difficult to forge. Using the Random Forests algorithm, our program benefits from both high detection capacity and low error levels. Based on our experience, this is the first research to carry out these large-scale websites / URL scanning and classification experiments, taking advantage of the distributed viewing points for the feature set. The results of the experiment show that our program can be used by the blacklist provider to create automated blacklists.

Key words : Phishing, Malicious, Machine Learning, URL.

1. INTRODUCTION

Phishing is a crime that employs social engineering and technological subterfuge to steal customer identity details and financial account credentials. Social engineering schemes rob unjust victims of their confidence that they negotiate with a trustworthy, legitimate group, for example by the use of fake e-mail addresses and e-mails. They are designed to direct users to bogus websites that divulge financial details including usernames and passwords to beneficiaries. Technical subterfuge programs plant malware on computers to directly steal credentials, typically using devices that intercept customer user names and passwords or misdirect users to falsified websites.

Phishing is an online scam that seeks to manipulate unsuspected users to reveal their confidential (often valuable) personal data to people who do not believe, mostly for malicious intention. It involves usernames, passwords, information on financial accounts, email addresses, SSNs often relationships. In Internet contact, Phishing is typically conducted as a trustworthy individual by mixing social and technological tricks. The techniques used by attackers also involve sending spoofing emails and creating deceptive websites to enable users to reveal details. Spoofing emails are commonly intended by legal companies to direct users to bogus websites that encourage users to enter sensitive information.

The Internet has become an increasingly enticing environment for people who disbelieve. As the 2013 Microsoft Computer Safety Index shows, worldwide phishing may have an annual effect of up to $2.4 billion. Based on APWG's 4th quarter 2019 Phishing Activity Patterns report ("the report"), the total number of phishing sites detected was 162,155. This was down from the 266,387 seen in Q3 and the 182,465 seen in Q2, and up from the 138,328 seen in Q4 2018.

The study also listed the pattern of growing use of HTTPS infrastructure on phishing websites, which means that Internet myths more effectively annoy end-users by converting established protection resources into end-users. The study reported that over 30% of phishing attacks were hosted on Websites with HTTPS certificates by the fourth quarter of 2017. Moreover, over 80 per cent of users find that the green lock icon on the browser URL bar is valid and stable, according to a PhishLabs survey. Even if it is a fake, the confusing "Safe" label provided by modern browsers to verifiable HTTPS sites makes the situation worse.

A variety of existing tools in browsers, search engines or applications such as Google's Safe Browsing and Microsoft's SmartScreen are currently attempting to warn users that a certain URL the user is about to visit has been detected as unsecure or harmful. It is achieved by linking the URL to blacklists generated by the security community. These blacklists are made up using various methods, from user reviews to web crawlers with site content analysis to

automated classification based on heuristics or classifications. However, several malicious websites can still slip through these security schemes, which are the product of several reasons:

1. The website is too new and so no mechanisms have yet been tested or analyzed.
2. The web was not adequately analyzed either because of the systems inadequate or because of counter-measures to identify attackers, such as "cloaking," or because of the misuse of legally accessible short URL services.

Several programs are in place to fix the issue of missing blacklists by reviewing the contents or actions of the website in a real-time client way when the end user uses it. Nevertheless, these devices suffer from overhead runtime. Moreover, consumers may have been exposed to the attacks from these malicious websites depending on the nature of the attack, because the material has been downloaded before an investigation begins. The most important aspect of phishing security is the timely identification of phishing websites. If phishing URLs were effectively identified and blacklisted, end users would receive an alert while they would be fooled for visiting the shelling site.

Within this paper, we therefore propose an efficient sensing device that crawls websites and discovers malicious pages automatically. We plan to use our program from a blacklist provider that can automatically compile and manage a blacklist of malicious URLs. Our program is fitted with several features representing various types of main features of the content or actions of the website that are impossible or difficult to cover through misconceptions. This program will proactively crawl and analyze a URL on the basis of a professional classifier and mark it as phishing / malicious or legitimate. Crawling is often carried out from distributed points of view, allowing the device to acquire new features and to achieve greater exactness and faster recognition speed. The blacklist will achieve greater coverage and timeliness by avoiding manual review. Customer monitoring is therefore avoided, thus eliminating both the overhead latency and the possibility of uploading inappropriate content to personal computers.

Specifically, we suggest a new phishing detection system in this study which views phishing detection as a binary grading problem when the positive and negative grades are legal and use a supervised learning algorithm which uses a range of features which can be hard or impossible to collect from the customer's side. Our end-to-end framework enables the blacklist provider to detect and update the blacklist phishing URL easily and accurately. The number of features used by the learning algorithm are extracted from a wide range of knowledge gathered from the distributed network with widely distributed points of view. Such features are derived from raw test data, including URL string, HTTP headers, DNS records, server status, network traces, WHOIS, and redirect. Our framework is robust to prevent misunderstandings, taking advantage of carefully selected features which indicate diverse aspects of the phishing content distributor, including physical characteristics, network characteristics and programming implementation features.

Although certain aspects of our classification have been addressed in previous works, based on our experience, this is the first study that incorporates so many different characteristics and collects information from more than one million websites to achieve very accurate identification of phishing. More importantly, our system has many features that represent the physical aspects of servers that host difficult to forge malicious content which raise the bar for attackers to sneak through the detection system.

The system with a data set of more than 100,000 phishing URLs and 1,000,000 valid URLs is evaluated. The phishing URLs are downloaded by the free OpenDNS blacklist provider PhishTank, while the legitimate URLs are a snapshot of the top one million Majestic list. According to the experiment findings, 98 percent with very small false positives, while still benefiting from strong classification power for both the phishing and the legitimate classes, could be accomplished with the proposed system.

2. RELATED WORK

Our paper focuses on the identification of machine learning phishing websites. Related issues such as poor domain blacklisting and e-mail spam filtering have been hard to achieve. In addition, machine learning is becoming increasingly common in these areas. Existing approaches to malicious websites [1-13] can primarily be divided in two groups depending on the characteristics used: static approaches [1-8] depending on features and dynamic approaches [9-13] based on functions. Static feature-based approaches [1-8] depend on features extracted from URL, page content, HTML DOM structure, domain-based data, etc. Alternatively, dynamic feature-based solutions [9-13] concentrate mainly on analyzing the captured behaviors when loading and rendering the application, and reviewing device logs when executed with certain scripts. For this article, we focus on using static characteristics.

2.1 Heuristic Based Methods in Proactive Blacklisting

Fukushima et al. addressed the concept of analyzing IP address blocks and registrars used to performatively list unknown malicious websites [1]. They assessed the prevalence of IP address blocks and registrars used by attackers intensively using some 4,000 instances from the Malware domain list [14]. In accordance with registrars, IP

address blocks then calculate the reputational level of websites, whereby individuals with low reputations are stated to be accused.

Their system may be subjected to a high false positive rate, i.e. that benevolent domains are marked as malicious. Many legitimate websites are only deemed malicious if they have the same registrar or IP blocks as other malicious websites.
Felegyhazi et al. suggested a constructive method of blacklisting based on inferences of same miscreant domains, whose domains have been identified in verified blacklists. This inference is obtained using WHOIS and DNS area file information [2].

This approach considers domains that are concurrently registered or that move to the same server as a documented illegal domain at the same time. Based on their analysis, 73% of the domains assumed later appeared in blacklists. Yet attacks based on existing blacklisted domains as seed can be inferred, which means that they are not enough to cope with new assaults that are not connected to existing malicious domains.In comparison, our approaches can be used to catch malicious domains, regardless of the relationship between domains, once the classifier has been equipped.

2.2 Machine Learning Based Methods

2.2.1 Malicious Domain Detection

Dong et al. have reviewed a complete list of features to be derived from X.509 certificates [3]. With the most 100,000 Alexa websites as their valid examples and with Phishing URLs downloaded from phishing examples, 95.5% and 93.7% precision and reminder are achieved for the Random Forests classification group [15]. The list of features includes information not only from licenses, but from a variety of other aspects of a specific website, including server characteristics, DNS responses, network efficiency and so on.

Xiang et al. introduced the CANTINA+ multi-layer learning platform, which uses features such as the URL, HTML DOM, search engines and third-party tools, such as PageRank, for the identification of phishing websites [4]. A minimum of 10 of 15 features are derived from HTML or URL textual patterns and formats. Rosiello et al suggested another similarity method focused on HTML DOM style for the identification of phishing used by browsers [5]. There are, however, several significant disadvantages to the DOM or lexical knowledge methods. First of all, the attacker can easily exploit the lexical characteristics of URL or HTML content to trick the classification scheme. While, the page content needs to be downloaded in order to perform the classification, during which the malicious portion may be added and submitted to the user's website if the system is used in the browser of the user. Furthermore, as many studies have shown, cloaking technologies (clothing means that the site is used to provide specific visitors with different content based on pre-defined requirements for bypassing the security system or deceiving the crawlers) are used by a non-denial able portion of malicious sites for the purpose of misleading crawler services such as Google to place them in a higher position or This phenomenon results in the PageRank or keyword scan, which is less effective for CANTINA+ [1, 15-18].

Ma et al. tried to circumvent this problem by not including content-related classification apps. They also used ten thousand of features created from IP address, WHOIS registrations and domain names, most of which were lexical features developed using "word bag" technology [6].

One downside of their methodology is that due to the large number of algorithmically generated lexical features, the resulting model is difficult to interpret. In comparison, the features of our system are both understandable and explainable.

2.2.2 Email Spam Filtering

Ouyang et al. proposed a multi-stage, machine-learning email detection program that includes an extensive range of network features [7]. They evaluated their approach with more than 1,4 million messages obtained in over two years, and recorded a truly positive rate of between 12% and 77% using the Decision Tree algorithm. Their work on this topic varies in many ways from their study. First, for classification purposes we use an advanced machine learning algorithm, i.e. Random Forests. Second, we used many other features than network features for the classification function, which makes the classification more robust since network quality may affect the network features collected. Thirdly, the focus of our research is on the classification of phishing URLs with a wider range of applications.

Basnet et al. concentrated on exploiting e-mail content functionality for spam detection and explored many different algorithms for machine learning [8]. Yet their approach also has the same problem as above for Xiang et al. [4] and Rosiello et al. [5]. In addition, your examined samples, including only 973 phishing mails and 3027 legitimate emails are small, making your classification accuracy number less credible. Finally, our research provides different viewpoints than current studies in the underlying manner.

First, we compile and check the program with a wide range of example URLs that download the phishing URLs from the PhischTank [19] and download the legitimate URLs from the top one million list of Majestics [20]. This increases the importance of the performance evaluation and the study in question.

Second, the complexity and variety of features tested by the machine learning algorithm was considerably improved. The addition of features representing different aspects of the target URLs ensures a detailed view of the target URLs and of the underlying hosting infrastructure, enhancing the efficiency of the proposed system substantially and increasing the threshold for an incorrect detection system from circumventing such a program. In addition, the study of the studied models contributes to a synthesis of the existing phishing environment.

3. METHODOLOGY

The required steps to construct a precise URL classifier began with the collection and testing of a representative data set for model training. Next, with several algorithms, we established supervised machine learning models. We trained and checked the models using different feature sets once they were implemented. These models and feature sets were then analyzed to define improvement areas. Finally, the performance comparisons of our feature sets and models, to establish more detailed classifications.

3.1. Data Gathering

The first step in the creation of a representative data set was to collect data. We have used many open-source repositories and pages to obtain this data collection. The information comes from five categories of attack: regular, phishing, malware, ransomware and botnet. Representative data from each of the five categories is needed.

Our usual data were obtained from two sources: the Canadian Cybersecurity Institute (CICS) and Frantisek Strasak's researches [21, 22]. The CICS has provided data by passing URLs to a Heritrix web crawler from Alexa Top Websites to retrieve URLs. After extracting the URLs and deleting duplicates, the data set was left with 35.300 URLs labeled as standard [21]. In the Alexa Top 1000, Frantisek Strasak registered its web traffic for 3 days. He created multiple packets capturing web traffic data. After testing his computer for malware, he was able to check that the sites visited in capture files were regular [22]. His capture files have been used to retrieve all URLs. We then removed and restored duplicates to our data collection. The malicious URLs came from four separate sources; the respective threat form is blacklisted. Table 1 indicates the source and date of the data obtained.

Table 1: Malicious Data Gathered

Class	Source	Count	Date Retrieved	Description
Phishing	PhishTank [23]	21,979	July 30, 2019 - August 20, 2019	A blacklist containing phishing URLs
Malware	Abuse.ch, URLhaus [24]	217,818	May 22, 2019 - August 20, 2019	A blacklist containing malware URLs
Ransomware	Abuse.ch, Ransomware Tracker [25]	1,903	May 16, 2019 - August 20, 2019	A blacklist containing ransomware URLs
BotnetC&C	CyberCrime [26]	16,292	August 20, 2019	A blacklist containing botnet URLs

Our data collection effort generated significantly more malicious data than usual data. As a consequence, training and test sets were generated by varying ratios between normal and malicious URLs to make the distribution practical. Even with this measure in place, our training and testing data set is still possible and does not reflect practical traffic.

The normal data from CICS and Frantisek Strasak is expected to consist of entirely legitimate, non-malicious URL data and consist of a representative sample of the general population of URLs.

3.2. Algorithms

In our background research, we found several supervised machine learning algorithms that performed well in URL classification. Most of this previous work was on classification of binary URLs, but we tried to construct a multi-class URL classifier. A binary classifier classifies data into two types, classifying URLs as natural or malicious in this case. A multi-class classifier classifies data into three or more classes, such that URLs are categorized as common, phishing, malware, ransomware and botnet. The advantages of a multi-class grouping are that it gives more information about the URL and possible attacks. It can be very useful in planning and stopping threats from public data sources. The algorithms we implemented were: Vector Machine Support (SVM), Random Forest and Logistic regression.

3.2.1. Support Vector Machine

A support vector machine is a biased classifier, which generates an optimal hyperplane for classifying new data with labeled training data. SVMs use kernels that convert data into a higher dimension space. This is important for classification into complex, non-linear data sets such as URL feature extraction data. This enables the model to produce a mapping function that divides data points into their groups. In our models, we evaluated two separate kernels: the linear kernel and the Radial Basic Function (RBF). The gamma value determines the distance between the boundary data points and the dividing line and has been used to boost the model's accuracy. When these parameters are set, a high gamma value will result in a more complex decision limit and over-adjustment [27, 28, 29].

3.2.2. Logistic Regression

Logistic regression calculates the likelihood of an input belonging to a specific class. The logit function is used to do this. The logit function is set to:

$$logit(p) = log(\frac{p}{1-p}), where\ p\ is\ the\ probability\ of\ classification$$

This gives a value between 0 and 1, which is the probability that an entered data point is part of a class. The closest the value is to one, the more likely a class is to have an object. This is then used to suit a line used to predict new data. The model learns by adjusting the coefficients that represent the data line. As the model learns, the coefficients shift to the highest likelihood of the right category [30, 31] being expected.

The model uses the final coefficients to predict new data after testing. The forecasts use a cluster threshold to assess the type of the data point. For example, if a binary classifier threshold is set to 0.5, any value 0.5 or higher is categorized as normal and any value under 0.5 is categorized as non-normal. Logistic regression is typically a binary classification and we have a multi-class classification and thus we have used the multinomial algorithm. This approach functions in the same way as the binary version, except that it uses multiple binary classification tools. The one-vs-all approach contrasts one class category with the rest; average vs. average, for example, would identify the knowledge as normal or as non-normal [30, 31]. Phishing vs. phishing will be another example. The algorithm then combines the binary classifiers generated by each form into a single model. When the models are executed, every binary classifier is executed on the input and the most probable class is the selected classification.

3.2.3. Random Forest

The Random Forest algorithm is a classification algorithm. That works as follows:

1. The data set is divided randomly into 'L' subsets with 'k' inputs. This is done with substitution, so subsets can contain the same entries. This data sampling approach is known as bootstrapping.
2. Every subset is then used for the decision tree preparation. A decision tree works by providing several splits in which the data is divided by a function. The function is chosen randomly from all data features. The value of the function determines the direction the tree is going down.
3. After training the trees, new inputs go through all of the trees to get a prediction. For classification, the final prediction is based on a majority vote from all of the trees.

This algorithm improves on a single decision tree and creates robustness of the model as it prevents overfitting the data and is able to make splits on randomly chosen features, as opposed to using only the best features to split the data [32, 33]. Random Forest is a bagging type ensemble method. Ensemble methods combine the decisions from other algorithms to give a less biased, more accurate prediction [34]. In particular, bagging methods run several algorithms in parallel and aggregate their results to create a prediction. An overview of bagging can be found in Figure 2.

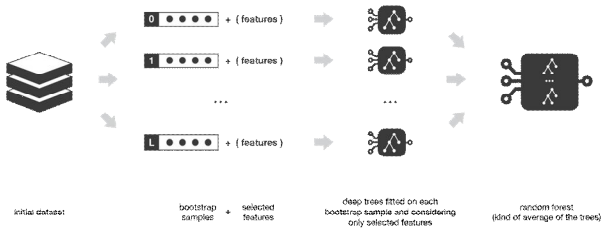

Figure 2: Boosting Overview

3.3. Feature Extraction

Our background research provided us with a wealth of features and feature sets to test. The features we focused on for our implementation were lexical and host-based. Previous research shows that these features are sufficient to create an accurate classifier. Content-based features may provide value but present risk to the integrity of our systems due to potentially malicious web page content related to the URLs; therefore, they were not evaluated. We implemented 32 features in total, 29 of them being lexical. The features can be found in Table 2.

Table 2. Feature List

Type of Features	Feature Name
Lexical Features	1. Length of URL 2. Length of host name 3. Length of path 4. Number of dots in URL 5. Number of @ in URL 6. Number of % in URL 7. Number of underscores in URL 8. Number of Tidles (~) in URL 9. Number of ampersands (&) in URL 10. Number of # symbols in URL 11. Number of hyphens in Host Name 12. Number of dots in Host Name 13. Number of hyphens in Path 14. Number of slashes in Path 15. Number of = symbols in Path 16. Number of semicolons in Path 17. Number of commas in Path 18. Number of dots in path 19. Params in URL 20. Queries in URL 21. Fragments in URL 22. Entropy of Host Name

	23. Check for Non-Standard Port 24. Check Alexa Top 1 Million 25. Check for Puny Code 26. Check Sub-Domains 27. Number of Digits in Host Name 28. IP Based Host Name 29. Check Top Level Domain 30. Username / Password in URL 31. Check URL Protocol
Host based Features	32. Location of IP Address 33. Address Registry 34. Number of Days Registered

3.3.1 Lexical Features

The lexical features are text-based characteristics of the URL. We split the URL into its protocol, host name and path. From there we analyzed the textual features in each. We used the Python libraries tldextract and urllib to parse the URLs and extract features. The lexical features we implemented were based on features described in previous research [27, 35-37].

3.3.2. Host-Based Features

Host-based features are composed of the network information about the URL host. We used a combination of features identified from our background research. The extracted features include: the IP address location, the registered country of the host, and the amount of time the host has been registered. We used the Python package ipwhois and socket to get the host information [27, 35]. The full feature list can be found in Table 2. The last three features in the list are the three host-based features. These features are useful because they can identify URLs with hosts located in suspicious areas and identify inconsistencies between the hosts and where they are registered. Also, malicious URLs tend to be registered more recently, therefore the length of time for domain registration can be a good indicator for detecting malicious URLs [27, 35].

4. DEVELOPMENT

We first determined the set of existing tools and libraries appropriate for our use case. Then, we implemented and trained several models using a training data set. After the models were trained, they were tested with a test data set to determine the models' performance. Finally, we optimized the parameters and train/test data set ratios to maximize the accuracy of the models.

4.1. Tools Used

We decided to use Python as our coding language because it is useful for processing large amounts of data and has readily available open source machine learning libraries. We assessed and selected a suitable machine learning library.

The three main Python libraries for machine learning are: *PyTorch*, *TensorFlow*, and *Scikit- Learn*. *Scikit-Learn* is an easy to use Python library that comes with out-of-the-box algorithm implementations. *Scikit-Learn* is more of a general-purpose machine learning library that includes implementations of many classic algorithms. *TensorFlow* and *PyTorch* are deep learning frameworks. They are more flexible and allow for the integration of custom code. We decided to use *Scikit-Learn* for the beginning implementations of our models because it contained models for all the aforementioned algorithms. *TensorFlow* and *PyTorch* are excellent alternative libraries to *Scikit-Learn*, but due to our algorithms of choice and the ease of use we selected *Scikit-Learn*. Nonetheless, it is possible to replicate what we have implemented using models from *TensorFlow* and *PyTorch*.

We also needed a way to extract the features we discussed previously. Thus, we created our own tool. Our tool takes a URL as input and returns a numerical array containing values for the features previously mentioned

4.2. Training

We split the training data into normal and malicious URLs and varied the split ratios of these two categories. We trained using a 50/50, 60/40, 70/30, and 80/20 normal/malicious splits. We trained the model on each of these split ratios in order to find the optimal training split between normal and malicious data that would produce the best performing model given a real scenario.

4.3. Testing and Evaluating

We used several methods to test and evaluate the models. We evaluated the performance of the features, along with an evaluation of the models' performance. We used built-in *Scikit-Learn* functions as well as some other mathematical tools to test the effectiveness of our features. We used two methods to test the relationship between the feature variables and classes: a chi-square test and ANOVA F-Value test.

The chi-square test is used to test for independence of categorical features. A chi-squared value is calculated for each categorical feature. If the chi-score is greater than or equal to the threshold value, then the feature affects the URL class. Otherwise, if the chi-score is less than the threshold, the feature is most likely not useful. The ANOVA F-Value test is similar to the chisquare test in that it is a test for independence, except it is used for numerical values. Similarly, if the F-Value is greater than or equal to the threshold value, then the feature affects the URL class, and vice versa. We also used a heatmap plot from the Python library, *Seaborn*, to visualize the correlation between features. The heatmap plot required a correlation matrix which was generated using the Python library, *Pandas*. These techniques

reduced the complexity of the data, which led to faster and more accurate classification.

To analyze the performance of our models we examined several metrics. We first looked at the overall accuracy of the model. The accuracy is a percent-value based on the number of true positives over the total number of predictions. A true positive is a correctly classified URL. Table 3 describes true positives in the case of a 'Normal' URL.

Table 3: Example of True Positives and Negatives, and False Positives and Negatives

	Predicted Class	**Actual Class**
True Positive	Normal	Normal
True Negative	Not Normal (ex. Phishing, malware)	Not Normal (ex. Phishing, malware)
False Positive	Normal	Not Normal (ex. Phishing, malware)
False Negative	Not Normal (ex. Phishing, malware)	Normal

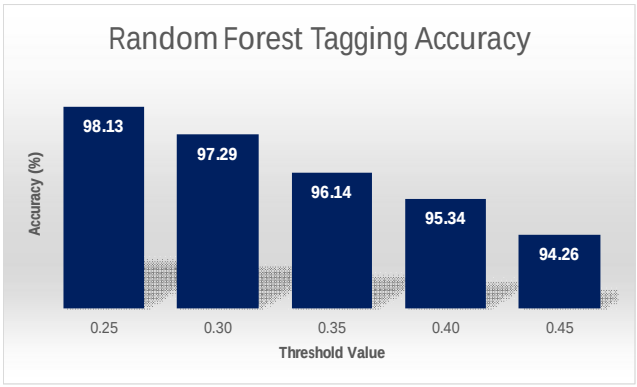

Figure 3: Tagging Accuracy Results

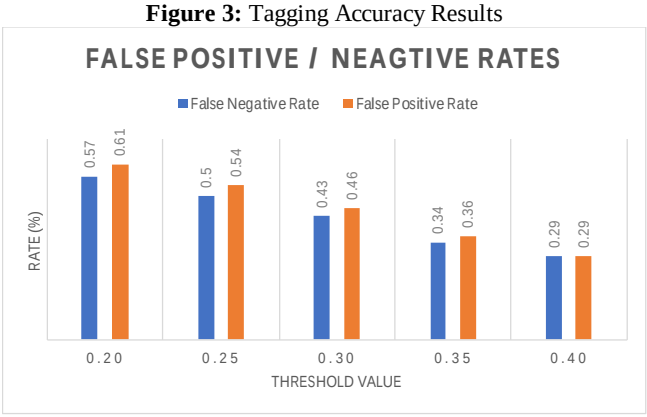

Figure 4: False Positive/Negative Rates

5. PERFORMANCE

5.1. Tagging

We tested the tagging method using the Random Forest algorithm, since it had the highest accuracy among the algorithms in our previous results. We used a 60% Normal / 40% Malicious split and 4 different threshold values. The results of this test and the threshold values used can be found in Figure 3 and Figure 4. In Figure 4, a false positive refers to a 'Normal' URL that was given a malicious tag (e.g. 'malware', 'phish', etc.). A false negative is a malicious URL that was given a 'Normal' tag. The false positive rates were calculated by counting the number of false positives and dividing by the number of URLs. The false negative rate was calculated in the same way but using a count of false negatives. The tagging algorithm produced very high accuracies. With lower threshold values producing higher accuracies. The higher accuracies came with a trade-off as lower thresholds led to higher false positive and negative rates.

5.2. Algorithms Parameters

We performed a 2-dimensional optimization for the Extra Trees and AdaBoost algorithms on a 50% Normal / 50% Malicious split. The results of the AdaBoost optimization can be found in Figure 5. In Figure 5, using a learning rate of 1, we varied the number of estimators from 65 to 75 and stepped by 1. These results showed the ideal parameters for the AdaBoost algorithm which were a learning rate of 1 and the number of estimators equal to 66.

5.3. Training Ratios

Next, we tested the algorithms against the 4 different training ratios we used previously. The results of this test can be found in Figure 6. Similar to our previous results, the 60/40 split yielded the highest accuracies. The Extra Trees classifier had the highest accuracy of 96.10%. The 80/20 split had the lowest accuracies for Random Forest and Extra Trees. The 50/50 split had the lowest accuracy for AdaBoost. These results indicate that the 60/40 training method yields the highest accuracies in malicious URL identification.

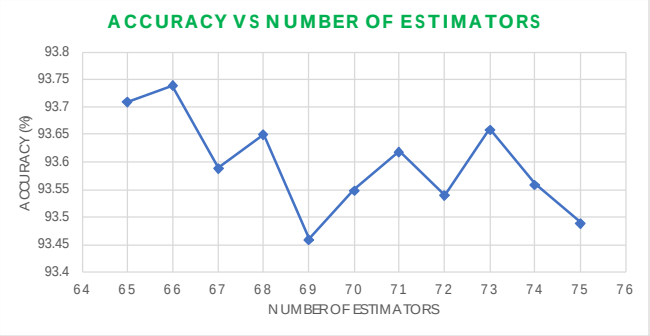

Figure 5: AdaBoost Optimization Line Chart

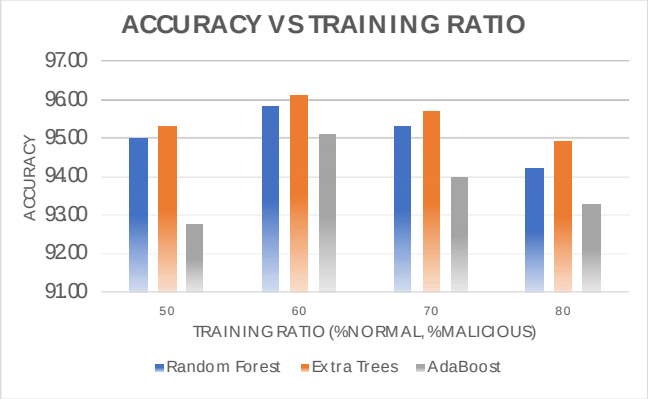

Figure 6: Ensemble Method Accuracies

6. CONCLUSION

An automated phishing detection system is introduced in this paper. The system uses a rich range of features, including hard-to-forge attributes, obtained from various aspects of the corresponding URLs, as well as from distributed points of view. With the use of the top 1 million Majestic list as legitimate URLs and hourly downloaded blacklists from PhishTank as phishing URLs, data collection and machine learning tests were performed to test the proposed technique. In addition, based on our contemporary data collection, a previous study is analysed.

Results of the experiments show that our method achieves good accuracy for phishing detection, which suggests the efficacy of the mechanism proposed. At the other hand, the previous study which is being studied reveals a decline in efficiency due to the evolution of the phishing environment, whereas our proposed methodology and feature set indicates considerable superiority. Meanwhile, variations are exposed between legitimate and phishing websites based on a case study of certain features.

REFERENCES

1. Yoshiro Fukushima, Yoshiaki Hori, and Kouichi Sakurai. **Proactive blacklisting for malicious web sites by reputation evaluation based on domain and ip address registration**. *In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 352–361. IEEE, 2011. https://doi.org/10.1109/TrustCom.2011.46
2. Mark Felegyhazi, Christian Kreibich, and Vern Paxson. **On the potential of proactive domain blacklisting**. *LEET*, 10:6–6, 2010.
3. 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. https://doi.org/10.1109/ECRIME.2015.7120795
4. Guang Xiang, Jason Hong, Carolyn P. Rose, and Lorrie Cranor. Cantina+: **A featurerich machine learning framework for detecting phishing web sites**. *ACM Trans. Inf. Syst. Secur.*, 14(2):21:1–21:28, September 2011.
5. Angelo PE Rosiello, Engin Kirda, Fabrizio Ferrandi, et al. **A layout-similarity-based approach for detecting phishing pages**. *In Security and Privacy in Communications Networks and the Workshops, 2007. SecureComm 2007. Third International Conference on*, pages 454–463. IEEE, 2007.
6. JustinMa, Lawrence K. Saul, Stefan Savage, and GeoffreyM. Voelker. **Beyond blacklists: learning to detect malicious web sites from suspicious urls**. *In KDD*, 2009.
7. Tu Ouyang, Soumya Ray, Mark Allman, and Michael Rabinovich. **A large-scale empirical analysis of email spam detection through network characteristics in a standalone enterprise**. *Computer Networks*, 59:101–121, 2014. https://doi.org/10.1016/j.comnet.2013.08.031
8. Ram Basnet, Srinivas Mukkamala, and Andrew H Sung. **Detection of phishing attacks: A machine learning approach**. *In Soft Computing Applications in Industry*, pages 373–383. Springer, 2008.
9. Alexander Moshchuk, Tanya Bragin, Steven D Gribble, and Henry M Levy. **A crawler-based study of spyware in the web.**
10. Mahmoud T Qassrawi and Hongli Zhang. **Detecting malicious web servers with honey clients**. *Journal of Networks*, 6(1):145, 2011. https://doi.org/10.4304/jnw.6.1.145-152
11. Andreas Dewald, Thorsten Holz, and Felix C Freiling. Adsandbox: **Sandboxing javascript to fight malicious websites.** *In Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1859–1864. ACM, 2010.
12. Kevin Zhijie Chen, Guofei Gu, Jianwei Zhuge, Jose Nazario, and Xinhui Han. Webpatrol: **Automated collection and replay of web-based malware scenarios**. *In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 186–195. ACM, 2011.
13. Bassam Sayed, Issa Traoré, and Amany Abdelhalim. **Detection and mitigation of malicious javascript using**

information flow control. *In Privacy, Security and Trust (PST), 2014 Twelfth Annual International Conference on*, pages 264–273. IEEE, 2014. https://doi.org/10.1109/PST.2014.6890948
14. http://www.malwaredomainlist.com/mdl.php.
15. https://aws.amazon.com/cn/alexa-top-sites/.
16. Sean Ford, Marco Cova, Christopher Kruegel, and Giovanni Vigna. **Analyzing and detecting malicious flash advertisements**. *In 2009 Annual Computer Security Applications Conference*, pages 363–372. IEEE, 2009. https://doi.org/10.1109/ACSAC.2009.41
17. David Y. Wang, Matthew Der, Mohammad Karami, Lawrence Saul, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. Search + seizure: **The effectiveness of interventions on seo campaigns**. *In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14*, pages 359–372, New York, NY, USA, 2014. ACM.
18. Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. **The dark alleys of madison avenue: Understanding malicious advertisements**. *In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14*, pages 373–380, New York, NY, USA, 2014. ACM.
19. https://www.phishtank.com/index.php.
20. https://majestic.com/reports/majestic-million.
21. Canadian Institute for Cybersecurity. (2016). URL dataset (ISCX-URL-2016). Retrieved from https://www.unb.ca/cic/datasets/url-2016.html
22. Strasak, F.Normal datasets. Retrieved from https://www.stratosphereips.org/datasets-normal
23. PhishTank.PhishTank. Retrieved from http://phishtank.org/
24. URLhaus.URLhaus database. Retrieved from https://urlhaus.abuse.ch/browse/
25. Ransomware Tracker.Blocklist. Retrieved from https://ransomwaretracker.abuse.ch/blocklist/
26. CyberCrime.CyberCrime. Retrieved from http://cybercrime-tracker.net/
27. Ma, J., Saul, L., Savage, S., & Voelker, G. (Jun 28, 2009). **Beyond blacklists**. Paper presented at the 1245-1254.
28. Bhattacharyya, S. (2018). **Support vector machine: Kernel trick; mercer's theorem**.
29. Patel, S. (2017). Chapter 2: **SVM (support vector machine) — theory.** https://doi.org/10.1002/9781118548387
30. Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). **Applied logistic regression** John Wiley & Sons.
31. Agrawal, A. (2017, March 31,). Logistic regression. simplified.
32. Breiman, L. (2001). **Random forests**. *Machine Learning, 45*(1), 5-32. doi:1010933404324
33. Liaw, A., & Wiener, M. (2001). **Classification and regression by RandomForest**.
34. Rocca, J. (2019). **Ensemble methods: Bagging, boosting and stacking**. Retrieved from https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stackingc9214a10a205
35. Sahoo, D., Liu, C., & Hoi, S. C. H. (2017). **Malicious URL detection using machine learning: A survey**.
36. Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). **Machine learning based phishing detection from URLs.** *Expert Systems with Applications*, 117, 345-357. https://doi.org/10.1016/j.eswa.2018.09.029
37. Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. C., & Tiong, W. K. (2019). **A new hybrid ensemble feature selection framework for machine learning-based phishing detection system**. *Information Sciences*, 484, 153-166. https://doi.org/10.1016/j.ins.2019.01.064