EasyChair Preprint

№ 2855

Detecting Banking Phishing websites using Data Mining Classifiers

M. Kanchana, Prabodhan Chavan and Arjun Johari

March 3, 2020

Detecting Banking Phishing Websites Using Data Mining Classifiers

Dr. M Kanchana
Computer Science Engineering
Chennai, India

Prabodhan Chavan
Computer Science Engineering
Chennai, India
prabodhanwork@gmail.com

Arjun Johari
Computer Science Engineering
Chennai, India
arjun201997@gmail.com

***Abstract*—Phishing is a malicious, criminal activity executed by obtaining the credentials of system user by unethical means. Phishing has always been a menace in world of internet as it threatens the privacy as well as security of the user. It is executed by multiple means like creating unauthentic webpages, user logins, made-up emails targeting banking sector as well as the ecommerce sector of digital industry. Since the booming of digital market in society the threat of Phishing is eminent. This document explains the existing work and additional work done to counter such conditions and secure the use of internet for the users. Modern day data mining techniques and use of machine learning is used to counter such attacks. We are putting dynamic extension to use for easy access and user protection is enabled for user and is highly optimal.**

Keywords— Phishing, Security, Machine learning, Dynamic Extension.

I. INTRODUCTION

Phishing is defined as malicious, criminal attack carried out on users for the sole purpose of obtaining their credentials unethically and using it to steal their information or mess with their bank accounts. It is done by various methods and many tricks in books are used to carry out such attacks. Phishing is basically creating fake webpages or website that resemble the original websites and then redirecting the credentials to personal database of attacker. This can be done by many means such as creating fake links or URLs that redirect to phoney webpages and rest is history. Attackers choose areas such as ecommerce websites, gaming websites or webpages that show unbelievable deals on various things that will attract users. Once the user fills the detail information and tries to log in the information is relayed back to database of attacker which is accessible to him then he uses that same information to log in to your account and steal whatever is available to him. Detecting phishing websites often include lookup in a directory of malicious sites since most of the phishing websites are short lived the directory cannot always keep track of all including new phishing websites. The attackers can use these credentials to even steal money from bank accounts or steal important information from your accounts. Only way for an end user to benefit from this is to implement detection in a browser plugin. So that the user can be warned in real time as he browses a phishing site. However, browser extensions have restrictions such as they can be written only in JavaScript and they have limited access to page URLs and resources. Existing plugins send the URL to a server, so that the classification can be done in the server and the result is returned to the plugin. With this approach, user privacy is questioned and also the detection may be delayed due to network latency and the plugin may fail to warn the user in right time. As it is an important security problem and also considering the privacy aspects, we decided to implement this on a chrome browser plugin which can do the classification without an external server. To develop a browser plugin which once installed, should warn the user on the event of he/she visiting a phishing website. The plugin should not contact any external web service for this which may leak the user's browsing data. The detection should be instant so that the user will be warned before entering any sensitive information on the phishing website. In the first half of 2019 businesses and residents of India were hit with more than 93,570 phishing events in a month span. With increase in number of internet users, there is a prominent need for security solutions again attacks such as phishing. Hence this plugin would be a good contribution for the chrome users. This is the first implementation of phishing website detection in browser plugin without use of an external web service. This makes use of existing works done on phishing detection and implements them in a manner that it will benefit end users. This involves porting the existing python classifier (random forest) to JavaScript. The plugin with a one-time download of the learned model, will be able to classify websites dynamically. This involves developing such a model (random forest) in JavaScript, as browser plugin supports only JavaScript. Thus, this project contributes to better privacy and rapid detection of phishing.

II. LITERATURE SURVEY

A. Direcory Based Approaches

This chapter gives a survey of the possible approaches to phishing website detection. This survey helps to identify various existing approaches and to find the drawbacks in them. The difficulty in most of the approaches is that they are not implemented in real time so that an end user will benefit from it. Most popular one of this kind is Phish Tank. According to Phish-Tank , it is a collective group of data and information about phishing on the Internet. Phish Tank also gives an open Application Programming Interface for developers as well as researchers to install anti-phishing data into their Apps for free. Thus Phish Tank is a directory of all phishing websites that are found and reported by people across the web so that developers can use their API for detecting phishing websites. Google has a Application Programming Interface called Google Safe Browsing API which also follows directory based approach and also provides open API similar to Phish Tank. This kind of approach clearly can't be effective as new phishing web sites are continuously developed and the directory can't be kept up to date always. This also leaks users browsing behaviour as the URLs are sent to the Phish Tank API.

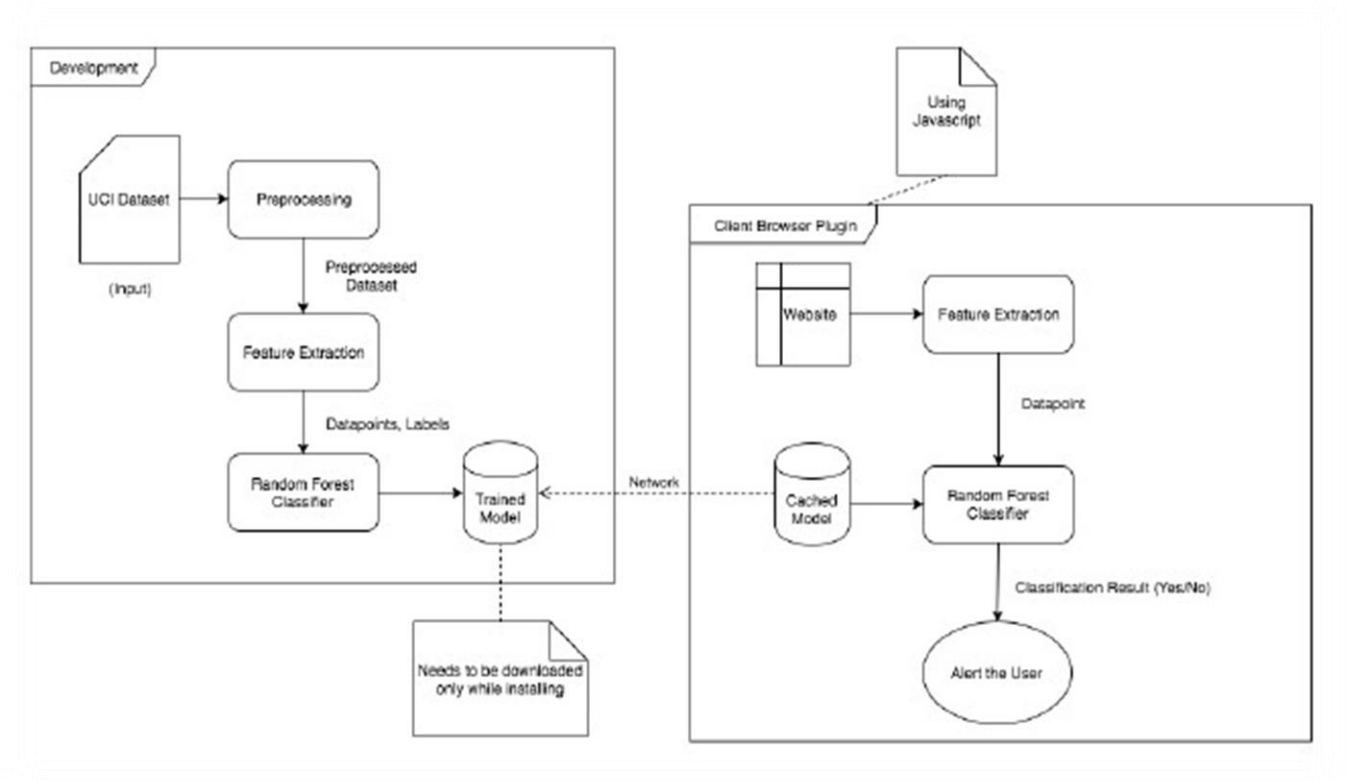

Figure 1:System Architecture

B. Rule Based Approaches

An existing chrome plugin named PhishDetector uses a rule based approach so that it can detect phishing without external web service. Although rule based approaches support easier implementation on client side, they can't be accurate compared to ML based approaches. Similar work by Shreeram.V on detection of phishing attacks using genetic algorithm uses a rule that is generated by a genetic algorithm for detection.PhishNet is one such Predictive blacklisting approach. It used rules that can match with TLD, directory structure, IP address, HTTP header response and some other. SpoofGuard by Stanford is a chrome plugin which used similar rule based approach by considering DNS, URL, images and links.

C. ML Based Approaches

Intelligent phishing website detection using random forest classifier (IEEE-2017) by Abdulhamit Subasi, Esraa Molah, Fatin Almkallawi and Touseef J. Chaudhery discusses the use the random forest classifier for phishing detection.[2] PhishBox: An Approach for Phishing Validation and Detection (IEEE-2017) by Jhen-Hao Li, and Sheng-De Wang[5] discusses ensemble models for phishing detection. As a result, the false-positive rate of phishing detection is dropped by 43.7% in average. They were able to come up with a detection mechanism that scans various types of phishing attacks maintaining a low rate of false alarms. Netcraft is one popular phishing detection plugin for chrome that uses server-side prediction.

D. Drawbacks

Based on the above-mentioned related works, it can be seen that the plugins either use rule-based approach or server-side ML based approach. Rule based approach doesn't seem to perform well compared to ML based approaches and on the other side ML based approaches need libraries support and so they are not implemented in client-side plugin. All the existing plugins send the target URL to an external web server for classification. This project aims to implement the same in browser plugin removing the need of external web service and improving user privacy.

III. PRPOSED WORK

A. Functional Requirements

The plugin warns the user when he/she visits a phishing website. The plugin should be fast enough to prevent the user from submitting any sensitive information to the phishing website. The plugin should not use any external web service or API which can leak user's browsing pattern. The plugin should be able to detect newly created phishing websites. The plugin should have a mechanism of updating itself to emerging phishing techniques.

B. Non Functional Requirements

There must be a simple and easy to use user interface where the user should be able to quickly identify the phishing website. The input should be automatically taken from the webpage in the current tab and the output should be clearly identifiable. Further the user should be interrupted on the event of phishing. No special hardware interface is required for the successful implementation of the system. The plugin should be always available and should make fast detection with low false negatives.

C. Constraints and Assumptions

Certain techniques use features such as SSL, page rank etc. Such information cannot be obtained from client-side plugin without external API. Thus, those features can't be used for prediction. Heavy techniques can't use considering the processing power of client machines and the page load time of the website. Only JavaScript can be used to develop chrome plugins. Machine learning libraries support for JavaScript is far less compared to python and R. The plugin is provided with the needed permissions in the chrome environment. The user has a basic knowledge about phishing and extensions.

IV. Implementation

Random Forest classifier[10] is trained on phishing sites dataset using python scikit-learn. The implementation of Architecture is shown in Figure 1. A JSON format to represent the RFC has been devised and the learned classifier is exported to the same. A browser script has been implemented which uses the exported model JSON to classify the website being loaded in the active browser tab. The system aims at warning the user in the event of phishing. Random Forest classifier on features of a website is used to classify whether the site is phishing or legitimate. The dataset arff file is loaded using python arff library and features are chosen from the existing features. Features are selected on basis that they can be extracted completely offline without being dependent on a web service or third party. The dataset with chosen features are then separated for training and testing. Then the Random Forest is trained on the training data and exported to the above mentioned JSON format. The JSON file is hosted on a URL. The client side chrome plugin is made to execute a script on each page load and it starts to extract and encode the above selected features. Once the features are encoded, the plugin then checks for the exported model JSON in cache and downloads it again in case it is not there in cache. With the encoded feature vector and model JSON, the script can run the classification. Then a warning is displayed to the user, in case the website is classified as phishing. The entire system is designed lightweight so that the detection will be rapid.

Pre-processing dataset is downloaded from UCI repository and loaded into a NumPy array. The dataset consists of 8 features, which needs to be reduced so that they can be extracted on the browser. Each feature[1] is experimented on the browser so that it will be feasible to extract it without using any external web service or third party. Based on the experiments, 8 features have been chosen out of 30 without much loss in the accuracy on the test data. More number of features increases the accuracy and reduces the ability to detect rapidly considering the feature extraction time. Thus a subset of features is chosen in a way that the trade-off is balanced. Then the dataset is split into testing set and training with 30% for testing. Both the training and testing data are saved to disk

A. Trainig

The training data from the preprocessing module is loaded from the disk. A random forest classifier is trained on the data using scikit learn library. Random Forest is an ensemble learning technique and thus an ensemble of 10 decision tree estimators is used. Each decision tree follows CART algorithm and tries to reduce the Gini impurity.

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

The cross-validation score is also calculated on the training data. The F1 score is calculated on the testing data. Then the trained model is exported to JSON using the next module.
The formula for the f1 score is f1 = 2 * precision * recall / precision + recall the precision recall and f1 score of the phishing classifier is calculated manually using JavaScript on the test data set

B. Exporting Model

Every machine learning algorithm learns its parameter values during training. In Random Forest, each decision tree is an independent learner and each decision tree learns node threshold values and the leaf nodes learn class probabilities. Thus, a format needs to be devised to represent the Random Forest in JSON. The overall JSON structure consists of keys such as number of estimators, number of classes and etc. Further it contains an array in which each value is an estimator represented in JSON. Each decision tree is encoded as a JSON tree with nested objects containing threshold for that node and left and right node objects recursively.

C. Classification

The feature vector obtained from the content script is ran through the Random Forest for classification. The Random Forest parameters JSON is downloaded and cached in disk. The script tries to load the JSON from disk and incase of cache miss, the JSON is downloaded again. it contains an array in which each value is an estimator represented in JSON More number of features increases the accuracy and reduces the ability to detect rapidly considering the feature extraction time. A JavaScript library has been developed to mimic the Random Forest behavior using the JSON by comparing feature vector against the threshold of the nodes. The output binary classification is based on the leaf node values and the user is warned if the webpage is classified as phishing.

V. Results and Discussion

We used Kaggle Datasets in this study to check the performance of our project to test various accuracies with different datamining techniques. The result is displayed in figure 2. The test set consists of data points separated from the dataset by ratio 70:30. Also the plugin is tested with websites that are listed in dataset. New phishy sites are also added to dataset as soon as they are found. The 8 features[1] extracted for the webpage are logged in to the console. The features are stored as key value pairs and the values are encoded from -1 to 1

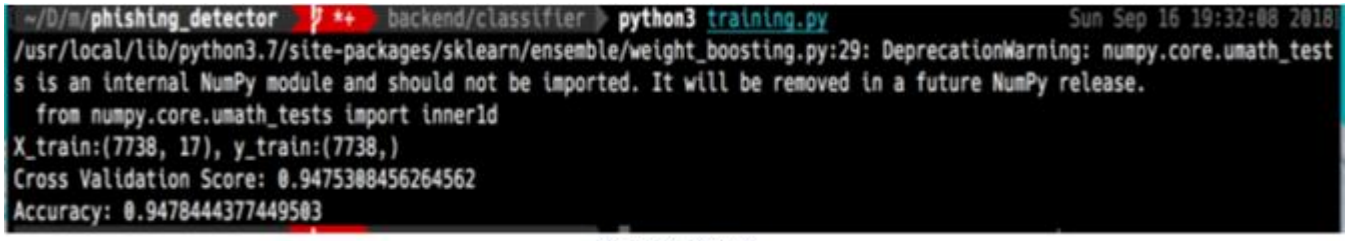

Figure 2: Result

VI. Conclusion

A. Summary

This is a phishing website detection system that focuses on client side implementation with rapid detection so that the users will be warned before getting phished. The main implementation is porting of Random Forest classifier to javascript. Similar works often use webpage features that are not feasible to extract on the client side and this results in the detection being dependent on the network. On the other side, this system uses only features that are possible to extract on the client side and thus it is able to provide rapid detection and better privacy. Although using lesser features results in mild drop in accuracy, it increases the usability of the system. This work has identified a subset of webpage feature that can be implemented on the client side without much effect in accuracy. The port from python to JavaScript and own implementation of Random Forest in JavaScript further helped in rapid detection as the JSON representation of the model and the classification script is designed with time complexity in mind. The plugin is detecting the phishing even before the page loads completely.

B. Criticism

The system has a lower accuracy but it is more usable and the trade-off between accuracy and rapid detection is handled well enough. The chrome extension API restrictions has a small effect on the plugin. Since the features are extracted in content script which is injected on page load, this plugin can't prevent a malicious javascript code from executing. Further the accuracy reduces while porting from python to javascript and this needs to be investigated. Javascript doesn't support multithreading and browser execute only javascript. Thus the classification can't be made faster by using parallel threads. Currently the results are not cached on the plugin and it's computed repeatedly even for frequently visited sites.

C. Future Work

The classifier is currently trained on 8 features which can be increased provided that, they don't make the detection slower or result in loss of privacy. The extension can made to cache results of frequently visited sites and hence reducing computation. But this may yield in phishing attack being undetected. A solution needs to be devised for caching of results without losing the ability to detect phishing. The classification in JavaScript can be done using Worker Threads which may result in better classification time. Thus, a lot of improvements and enhancements are possible this system offers a more usable solution in the field of phishing detection.

References

[1] Phishing page detection via learning classifiers from page layout feature, Mao et al. EURASIP Journal on Wireless Communications and Networking (2019) 2019:43 https://doi.org/10.1186/s13638-019-1361-0

[2] A. Subasi, E. Molah, F. Almkallawi, and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Nov. 2017.

[3] Anindita Khade Detection of Phishing Websites Using Data Mining Techniques (IJERT) 2013 Dept. of CE, Lokmanya Tilak College of Engineering Koparkhairane, Navi Mumbai 421302.

[4] "UCI Machine Learning Repository: Phishing Websites DataSet," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/phishing websites..

[5] J.-H. Li and S.-D. Wang, "PhishBox: An Approach for Phishing Validation and Detection," 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2017.

[6] A. A. Ahmed and N. A. Abdullah, "Real time detection of phishy websites," 2016 Annual Information Technology, Mobile and Electronics Communication Conference (IEMCON), 2016.

[7] R. Aravindhan, R. Shanmugalakshmi, K. Ramya, and S. C., "Certain investigation on web app security: Phishing detection and phishing target discovery," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS),2016.

[8] J. Mao,W.Tian, Phishing-alarm: efficient and robust phishing detection via page component similarity. IEEE Access. **5**,17020–17030 (2017)

[9] P. Likarish, E. Jung, D. Dunbar, T. E. Hansen, J. P. Hourcade, in Proceedings of IEEE International Conference on Communications, ICC'08. (IEEE, Beijing, 2008), pp. 1745–1749.

[10] Wikipedia, Random forest (2018). https://en.wikipedia.org/wiki/Random_forest/, [Online]. Accessed 26 July 2018.

[11] Y. Zhang, J. I. Hong, L. F. Cranor, in Proceedings of the 16th International Conference on WorldWide Web. Cantina: a content-based approach to detecting phishing web sites (ACM, Banff, Alberta, Canada, 2007)

[12] N. Abdelhamid, F. Thabtah, H. Abdel-Jaber, in Proceedings of IEEE International Conference on Intelligence and Security Informatics. Phishing detection: a recent intelligent machine learning comparison based on models content and features (IEEE, Beijing, China, 2017)

[13] Ammar ALmomani, G. B. B, Tat-Chee Wan, Altyeb Altaher, and Selvakumar Manickam, "Phishing Dynamic Evolving Neural Fuzzy Framework for ...," Jan-2013. [Online]. Available: https://arxiv.org/pdf/1302.0629.

[14] S. Gupta and A. Singhal, "Phishing URL detection by using artificial neural network with PSO," 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), 2017.

[15] "An Efficient Approaches For Website Phishing Detection Using Supervised Machine Learning Technique," International Journal of Advance Engineering and Research Development, vol. 2, no. 05, 2015.