Analysis of the system

Functional system requirement: Extension plugin should provide a warning pop-up when they visit a website that is phished; therefore it should strictly follow the following:

a. Extension plugin ability to present the pop-up to the users screen should be quick enough to the point, users will be aware before entering any confidential or sensitive details into a phishing website.

b. Extension plugin should not need the facilities and services from an 3rd party service or APIs, due the reason that those services will always the potential to leak users browsing data and pattern when it gets compromised by hackers

c. Extension plugin will have the capability to also detect latest and new phishing websites

Non-Functional system requirement: Graphical User Interface design Interface developed should be done with the understanding that it must meet the simplicity of what users would like to see when they need an extension for detecting things, and also it needs to adhere to non IT literate users as well. It must also provide the exact information on what the user wants like identifying a phishing website quickly without needing to click on many options. The process of identifying phishing website should be taken directly from the web-page user wants to view through their URL and the result from it should be easily understood by the users. Most importantly, the extension plugin should have a popup that will notify the user regarding the website status of being phished.

Software requirements:

a. PyCharm software

b. Python language

c. Google chrome browser

d. Scikit-learn

e. NumPy

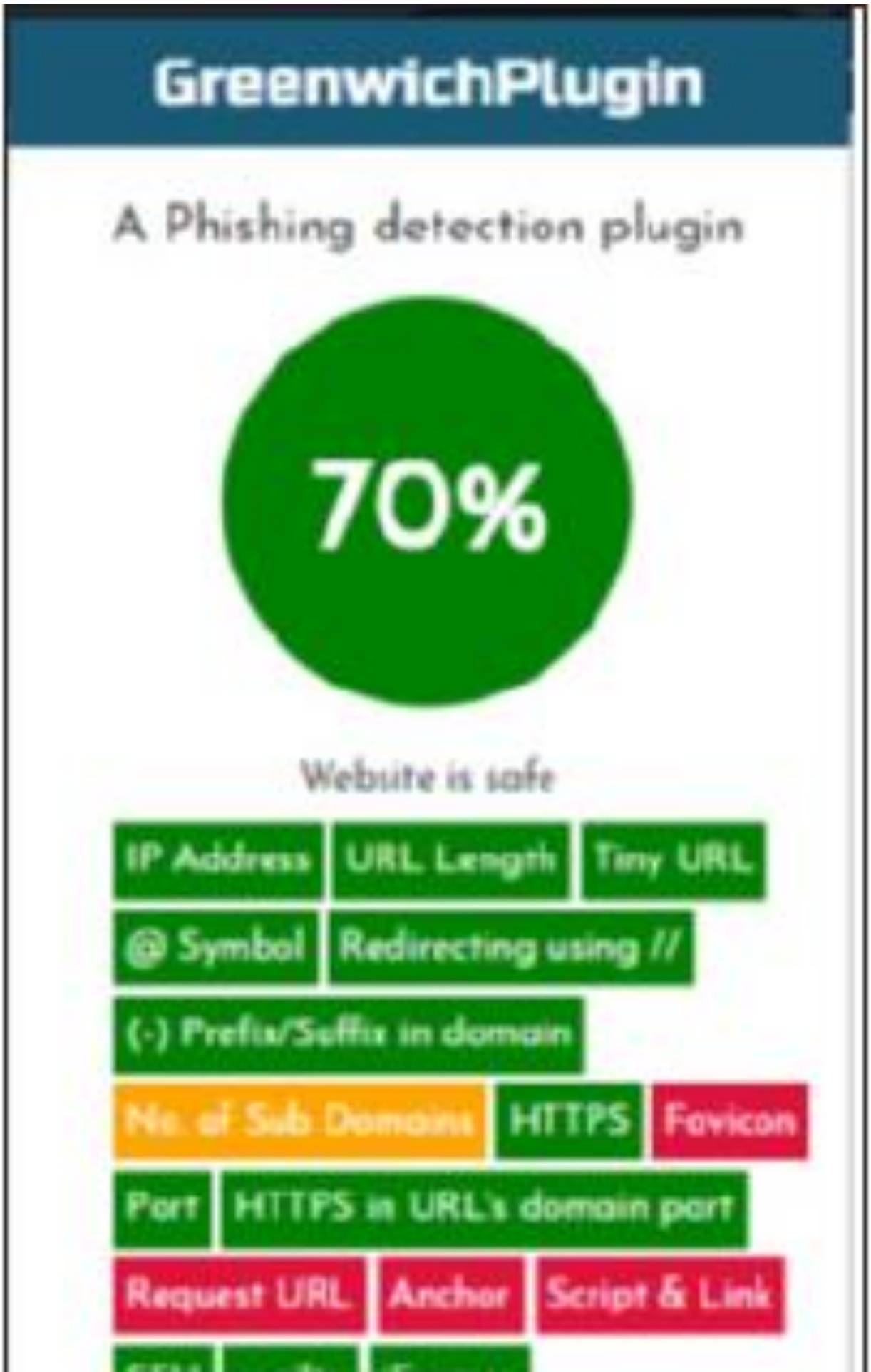
f. Liac-arff for dataset

Designing of the system

The architectural concept of the system begins by training a Random Forest classifier on dataset that contains URL features that can be classified as phishing, legitimate, and suspicious on python using Scikit learn. The result of the random forest classifier is then represented in a JSON format and as well as the classifier that has learnt will be represented in JSON format over

HTTPS. Once this is achieved, a script that is implemented using JavaScript for web browser is developed which will use the exported JSON format model to provide classification of web pages that will be viewed over an users internet browser. The (Figure 1) displayed below will provide an understanding of the system architecture through a system diagram. The main functionality of the plugin extension that is being implemented is to issue a warning notification through a pop up towards the users browser screen in the circumstances if the user accidentally is on the verge of visiting a phishing website. To provide this, classifier should be done on the 17 selected features out of the 30 features that exist in the dataset. Dataset is in an arff file format therefore it needs to be loaded using arff library 1 that python supports. Reason why only 17 features are selected from 30 available features is based on the possibility that these selected features can be extracted without needing to be online from a client side rather than being extracted completely from a 3rd party server or service. Dataset that has been used for this implementation of project is then separated into, a training dataset and to a testing dataset. Using the training data, random forest is then trained on it and the results are exported in a JSON format over HTTPS2. From client side, the google chrome extension then performs an execution of script on every webpage it loads from the browser, and it then simultaneously converts the selected features that have been specified. As these specified features are once converted, google chrome extension plugin then proceeds to verify for the initial JSON model that is stored in cache. Together with the converted features and the model in JSON format, the script on the extension plugin can ideally run a classification. Once this is successfully achieved, a warning notification through a pop-up can be displayed towards the user, if the web-page the user is visiting is considered phishing. The process implementation of this extension plugin is very lite towards users computers and as well it gives the capability to detect phishing website in a quick effective manner.

Figure 1: Graphical user interface.


GreenwichPlugin
A Phishing detection plugin
70%
Webuite is safe
IP Address
URL Length
Tiny URL
@ Symbol
Redirecting using //
(-) Prefix/Suffix in domain
No. of Sub Domains
HTTPS
Favicon
Port
HTTPS in URL's domain part
Request URL
Anchor
Script & Link

User interface design

The designing of the GUI of the plugin was made simple and feasible to attract all audience to use it, and also understands the contents it will be displaying needs to be easy to understand for all users, and all this is achieved by using mix of HTML3 and as well CSS4. The user interface will provide the main indication to user on how legitimate the web-page they are viewing through a large circle. This circle will change colour depending on how the websites are being classified, if its phished or legitimate. The results that are derived from the classification are visually represented in the large circle with the following colour code:

a. Dark Green - Legitimate website and safe to view
b. Golden Orange - Suspecting possible phishing website
c. Crimson - Phished website

The percentage score represented in the chart is calculated in the frontend.js section of the development where legitmate Count(phishingCount+suspiciousCount+legitimateCount)*100

```
i. function classify (tabId,result) {
ii. var legitimate Count=0;
iii. var suspicious Count=0;
iv. var phishing Count=0;
v. for(var key in result) {
vi. if (result[key]==”1”) phishing Count++;
vii. else if(result[key]==”0”) suspicious Count++;
viii. else legitimate Count++;
ix. }
x. LegitimatePercents [ tabId]=legitimateCount/
(phishingCount+suspiciousCount+legitimateCount)*100;
```

Listing 1: code for obtaining and calculating legitimate Percent. This helps us to provide a justifiable score based on feature categorization into phishing or legitimate sites. One of the existing issue due to how this is calculated is, the percentage representation of sites can be confusing to users at initial glance since, without knowing how it is being calculated they might question the authenticity of the percentile score of classifying websites. This is something that is being constantly emphasized on for future work progress.

Extension plugin will also have the function to alert users when they are about to view a website if it is a possible phishing website, in-order to prevent any entry of confidential or sensitive information's from user into the web-page. Accuracy score, together with recall and precision results will also be available for users to view on a separate tab that can be accessed from the main plugin interface. The designed concept of the graphical user interface can be seen from (Figure 1) that is displayed below.

Design of model

Dataset that is being used for this project is initially downloaded from UCI dataset repository and then it is imported into an array with NumPy. The dataset that is loaded has 30 features with it, but it needs to be re-evaluated to figure out which specific features can be used and extracted on the browser extension plugin. This is done by manually testing each features on the plugin to identify which features are capable of working without needing a 3rd party service to give results. Accomplishing this, helped us identify the 17 specific features from the initial dataset that gives results without drastic loss in accuracy value coming from test data. While having more features being used from the dataset can easily provide better results value for accuracy but this would require more processing time to produce the result and this would not be following the gene scope of the project that was to provide a quick and effective phishing detection. It needs to be understood, the features that were chosen specifically will be a compensation for quicker result than for accurate results. (Table 1) will show the 17 features that have been selected. Once this is completed, dataset is then split into training and testing data where its 30% for testing data and the remaining 70% will be for training data.

Table 1: Features that were selected to identify phishing websites.

IP Address	No. of Sub Domains	Anchor
URL Length	HTTPS	Scripts & Link
Tiny URL	Favicon	SFH
@ symbol	Ports	mailto
Redirecting using//	HTTPS in URLs domain part	iFrames
(-) Prefix/Suffix in domain	Request URL	

Training of model

Training of the data obtained from pre-processing are loaded and random forest is then trained on the training data using scikit-learn. As it is known, Random Forest is part of an ensemble machine learning 5, which allowed us to use 10 estimators for our classification. The decision tree estimators works under CART algorithm6 and impurity of gini is reduced in each decision tree to provide the result. Cross Validation score is also done on the training data while the F17 score is calculated on the testing data. Lastly, the model that has been trained with results and parameters, it will be exported using JSON format.

Development of the System

Project that has been developed is divided into 2 different category, namely the backend which consists of classifier and dataset and frontend of the system. The work of backend is to preprocess the dataset that was used and also, train the models with Random Forest with chosen parameters and values. Front end of the system development mainly consists of JavaScript, also scripts to enable the contents to be functional, and scripts that's running on the backend like our Random Forest JavaScript. It also contains the relevant HTML files needed to create the graphical user interface (GUI).

Results

Testing with dataset

Test set that was used consisted data from the initial dataset was split into a 70 to 30 ratio. To make and implement a fully functional extension plugin, it was tested with many various phishing website that are obtained and listed from phish tank website1. Since phish tank is a large active community, it always has new phishing websites being listed in a frequent manner. With this in mind, it should be noted that the extension plugin that has been implemented for the project has the capability to detect new phishing websites that are added to the website as well. Initial dataset contains 1105 data points with 30 features. For pre-processing, we are splitting the Dataset 70 for training and 30 for testing and selecting 17 features specifically that can be used without needing a 3rd party service. Once the pre-process is successful, the results are saved into a JSON format file.

Feature extraction on extension plugin

For understanding how features are classified on websites, we extracted portal.gre.ac.uk for the 17 features which are logged in google chrome console. (Figure 2) displayed below will display the result that is logged on the console. It should also be noted that the features are always stored in pair values and these values are encoded as 1 to -1 where 1 is phished and -1 is legitimate websites.

Figure 2: Features that are extracted from portal. gre.ac.uk.

Classification in extension plugin GUI

Result of the classification is displayed on the extension plugin on the (Figure 3) below, through an indication that is displayed on a circle where dark green represents a legitimate website and crimson represents a phished website.

Figure 3: Classification result on the GUI.

Testing phase sample screenshots

The result that was displayed on the extension plugin while visiting a phished PayPal website obtained from Phish tank domain as shown in (Figure 4). As you can notice, the website has low 32% value of trust hence why the indication is in crimson, stating it is a phished website.

Figure 4: PayPal website.

Testing phase

Testing for this project was done manually by taking websites directly from phish tank and also from web history to find out if the classification is done properly without any fails. The reason why this was done rather than testing the model on a dataset with URLs was, since the existing dataset that was used in this project classifies URL features as either 1 as phishing, 0 as suspicious website, and -1 as legitimate on URL features on websites that is being viewed. This way, the extension provides a functionality to also predict new phishing websites as well because we are classifying websites based on URLs features rather than a complete URL. There were several problems that needed to be addressed after testing was done. When testing was completed, it came to our understanding that the accuracy score has dropped while porting the JSON format of the parameters and testing of random forest from Python to JavaScript. While this reason is unknown, the loss of accuracy was a trade with how quick the phishing detection was done on the browser. Another thing that came to attention was, while testing email URLs, URL for Gmail accounts were being prompted as a potential phishing website by the implemented popup feature in this extension. The reason for this is uncertain but based on current investigation; it may be due to how certain features are tracking information. While this isn't proven, it was merely an observation that was done by comparing the analytic result from DuckDuckGo tracking extension 4. For testing purposes, 5 websites were chosen from phish tank website to see if the model correctly classifies them as phishing. About 50 websites were chosen from the website and tested against the plugin manually, the success rate was 35/50 while the remaining websites were falsely reported to the website. Due to high number of tested website, only the 5 latest websites are listed. Website are listed below:

a. https://online-billing-llc.net/993a36dc7b50be416f3...
b. https://kundenservice-umstellung.live/
c. https://shortinieri-fast.life/HwfiG
d. https://allegro.pl-nowe-regulamini3758.cho274.pl/6...
e. http://superchange.site/

Conclusion

This research illustrates and explains the development of a phishing detection on URLs as an extension plugin on google chrome which aids in privacy matters as its implemented on a client side with the capability to detect phishing in efficient rapid manner for users to be notified

before accessing potential phished websites or entering confidential information's. The implementing method of exporting Random Forest from python to JavaScript is the most essential part in this project as JavaScript for Random Forest had to be done natively with proper understanding of how the classifier works and performs. Many of past related work done by several other researchers often prefers to use website features with the aid of 3rd party services to provide a better accuracy measure when it comes to predicting phished websites. This won't be a viable option for us since it results in security measures regarding privacy of data browsing and also it will be dependent on latency of the networks. As our development extracts features through client side, this helps vastly in providing fast reliable detection together with the possibility of providing privacy towards users browsing. But it needs to be kept in motion that while not using all the features provided, the accuracy result gets effected minimally but it does increase the functionality of the extension being built. This is achieved by choosing specific subset of features on web-pages that can be used to implement without huge loss of accuracy from the client-side. Exporting the classifier from python to JavaScript for extension plugin development, with the development of Random Forest in JavaScript gave us a foundation to provide an efficient and quick detection plugin for phishing as the model was represented in JSON format, together with scripts for classification was development with the idea of the understanding of timing is most vital when it comes to providing awareness to users. With this development, it is possible to detect phishing websites before the page even loads, as this gives the awareness to users before they decide to provide any confidential information into the phished website. While it was noted that using minimal features to adapt to client side functionality will reduce the accuracy score of the model, but it wasn't severe to the point of many false positive results. Precision score that was calculated on the testing set is 0.8724385245901639. While the accuracy score is rather low, it didn't have much impact on true positive results for detecting phished websites. It was a trade-off between loss of accuracy against efficient and quick detection.

Acknowledgement

I would like to firstly thank University of Greenwich for allowing me to be part of their education system, also to all the lecturers who have taught me in the process. I would also like to provide my sincere gratitude to my supervisor Dr Tuan Vuong for assisting me and providing a good foundation of understanding in the field of machine learning. Thank you to Gloria

Meneses, for all her lovely, priceless companionship through tough times. Thanks go to Dr. Khan, UCSI University, Malaysia for his assistance in writing my research paper. Lastly, my gratitude and acknowledgement towards my parents, there were not only my parents but also lecturers providing essential and important perspective into life and as well into academics.

References

1. Patil D, Patil J (2015) Survey on malicious web pages detection techniques. International Journal of u- and e-Service, Science and Technology 8(5): 195-206.
2. Wardman B, Shukla G, Warner G (2009) Identifying vulnerable websites by analysis of common strings in phishing URLs. IEEE pp. 1-15.
3. Hong J, Rose C, Xiang G, Cranor L (2011) Framework on identifying phishing websites using machine learning. ACM Transactions on Information and System Security 14(2): 1-28.
4. Akanbi O, Abunadi A, Zainal A (2013) Feature extraction process: A phishing detection approach. 13th International Conference on Intellient Systems Design and Applications.
5. Aydin M, Baykal N (2015) Feature extraction and classification phishing websites based on URL. IEEE Conference on Communications and Network Security (CNS).
6. Nappa D, Wang X, Abu Nimeh S, Nair S (2007) A comparison of machine learning techniques for phishing detection. Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit on–eCrime '0 pp. 60-69.
7. Sandhya L, James J, Thomas C (2013) Detection of phishing URLs using machine learning techniques. International Conference on Control Communication and Computing (ICCC).
8. Chang Y, Chen T, Laih C, Hou Y, Chen C (2010) Malicious web content detection by machine learning. Expert Systems with Applications 37(1): 55-60.
9. Lu G, Debray S (2012) Automatic simplification of obfuscated javascript code: A semantics based approach. IEEE Sixth International Conference on Software Security and Reliability.
10. Benjamin L, Benjamin Z, Curtsinger C, Christian S (2011) Zozzle: Fast and precise in-browser javascript malware detection.
11. Aldwairi M, Alsalman R (2012) Malurls: A lightweight malicious website classification based on URL features. Journal of Emerging Technologies in Web Intelligence 4(2): 128-133.
12. What is multithreading?-definition from techopedia 2019.