Computer Science Department
The University of Hong Kong
Final Year Project 2019-20

SMART EMAIL CLIENT TO DETECT MALICIOUS URLS

Group: 2
Members: Trisha Gupta (3035342419)
Shreya Palit (3035346556)

Supervisor: Dr. S.M. Yiu

2nd February 2020

Table of Contents

Abstract

Phishing is a type of identity theft that conglomerates social engineering practices and complex attacks to harvest financial information from unsuspecting consumers. Malicious emails are a form of phishing attack, which redirects the user to visit unexpected sites or cause computer viruses to be downloaded on the user's system. In this project, we try to create a smart email client that will try to combat this problem and provide solutions to consumers by having a tracker that will be able to detect whether the link included in the email is malicious or not.

The methodology describes several features that can be used to distinguish a phishing URL from a benign one. These features are used to model a Random Forest filter that is efficient and has a high accuracy. Since the preliminary research phase is over, the current stage is about building the machine learning model with various classifications and performing comprehensive measurements on millions of URLs to quantify the occurrence of phishing on the internet today.

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the guidance to complete this project and report. A special thank you to our supervisor Dr. S. M. Yiu whose contribution in stimulating suggestions and encouragement helped us greatly with our project. We would also like to thank our CAES Professor, Mr. Cezar Cazan for his support and guidance on writing professional technical reports. Lastly, we are also sincerely grateful to the Computer Science Department of The University of Hong Kong for giving us the opportunity to work on this final year project.

List of Figures and Tables

Abbreviations

HTTP – Hyper Text Transfer Protocol

IP – Internet Protocol

ML - Machine Learning

MLE – Maximum Likelihood Estimation

SFH - Server from Handler

SVM – Support Vector Machine

TLD – Top Level Domain

TME - Targeted Malicious Email

URL - Uniform Resource Locator

UTME - Un-Targeted Malicious Email

WHOIS – Who is (Query and Response Protocol)

1. Introduction

The following chapter introduces the progress report and what it is about. Firstly, there is a brief background of the project which includes the importance of this topic especially in today's world, as well as definitions and explanations of technical terms that have been used throughout the report. The main objectives and deliverables have also been included followed by an outline of the report content and structure.

1.1 Project Background

Email is one of the most vulnerable vectors for cyber-attacks owing to its highly aimed and tailored essence. Every day, we receive a plethora of emails which may contain URLs. These emails may be malicious. In fact, it was noted that close to one-third of all websites are potentially malicious in nature [1], demonstrating rampant use of malicious URLs to perpetrate cyber-crimes. A malicious email may redirect one to unexpected sites or cause harmful software to be downloaded. To give an example of a malicious email, assume an email containing special information is directed at a recipient that will help with his health issues or some common problems. Since these emails do not provide any details, the URL needs to be clicked in order to know its content. Once this malicious URL has been clicked, the recipient is redirected to harmful websites which steal sensitive information like credit card details, passwords etc. It is very easy to be misguided by the appearance of any email and not see what is hidden in the URL, but eventually it is just complicated programming. With a little bit of knowledge, it can be understood how this works and leveraging this knowledge, a better alert system can be built to avoid these potential problems.

URL is the abbreviation of "Uniform Resource Allocator" that is the global address of documents and other resources on the World Wide Web. There are two parts [2] of a URL: (i) resource name – specifies the IP address or domain name (ii) protocol identifier. The above two components are separated by a colon and two forward slashes. A graphical representation of the URL is shown below in Figure 1.

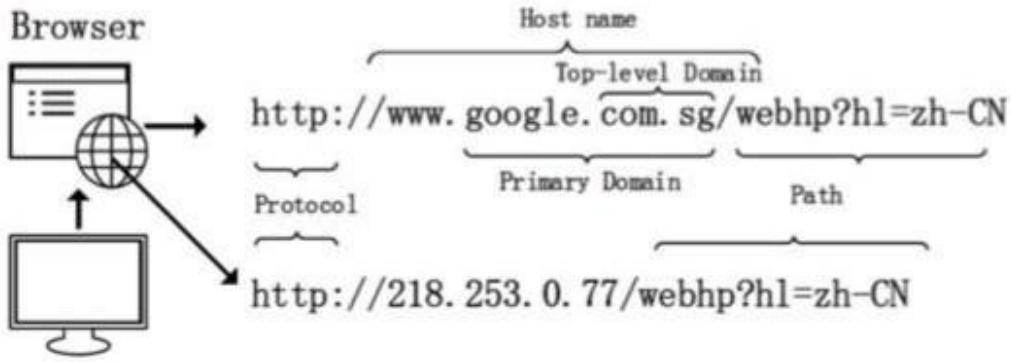

Figure 1: Graphical Representation of a URL

The malicious URLs that are present in emails are compromised URLs.

1.2 Objective and Deliverable

The senders are determined to convince the legitimacy of the malicious URLs in the email and it is incredibly difficult to not fall victim to such a trick. It is easy to look at an email and not see what is hidden behind the display but actually it is just a complex software that is lurking behind that very simple looking message. Our main objectives are as follows:

- Detect a malicious URL much before user clicks on it, thereby saving time
- Reducing the risk of phishing attack on the user, thereby saving money and information

Our main deliverable will be in the form of a warning message that advises the user whether the URL in the email is safe to open or whether it is harmful.

1.3 Report Outline

The remainder of the progress report is structured as follows. Chapter 2 includes the methodology that will be implemented in order to combat malicious URLs. There is a brief chapter 3 which contains the findings, current status and future work. A summary of the whole process is then included in Chapter 4.

2. Methodology

This project is separated into two phases. Firstly, the research phase which went into seeing the different methods that are already present for detecting malicious URLs in an email and whether those could be modified, or selected parts of the methods could be used and incorporated into our own smart email client. For that it was necessary to learn about the different types of attacks that are present, what are usually the contents of such emails that contain malicious URLs and the existing methods.

Then is the implementation phase wherein we scrape the last unread email in a user's inbox and use the Machine Learning model to predict whether the URL in it is malicious or not.

2.1 Target People

The target of a malicious email may either be for one specific recipient – TME (also known as spear fishing) or may be for multiple recipients - Untargeted Malicious Email (UTME). The email targeting multiple recipients can be more easily identified as malicious as opposed to the one targeting one specific person. This is because of the cost trade-off to sender and is usually only used when a high level of information is to be extracted. For example, a high technology enterprise may be vulnerable to this attack as the person may hold database access such as plans of marketing, client details or other sensitive data.

In order for senders to camouflage their identities, they use certain hiding techniques in order to mask the email source, such as copying someone on the recipient's contact list, or a celebrity.

2.2 Existing Method

Blacklist method is the most common method of detecting malicious URLs embedded within emails and which has been used by several antivirus groups [2]. Blacklists are fundamentally a database of URLs that have been found to be malicious previously. Over time this database is then compiled such that the database would have grown to contain lots of examples of malicious URLs. Figure 2 below gives a very basic flow of how a Blacklist method is used to detect malicious URLs. This technique is very fast as it uses a simple overhead request and is therefore very easy to implement. However, this strategy has a very low false positive rate, although it is said that blacklisting usually suffers only non-trivial false-positive rates [2].

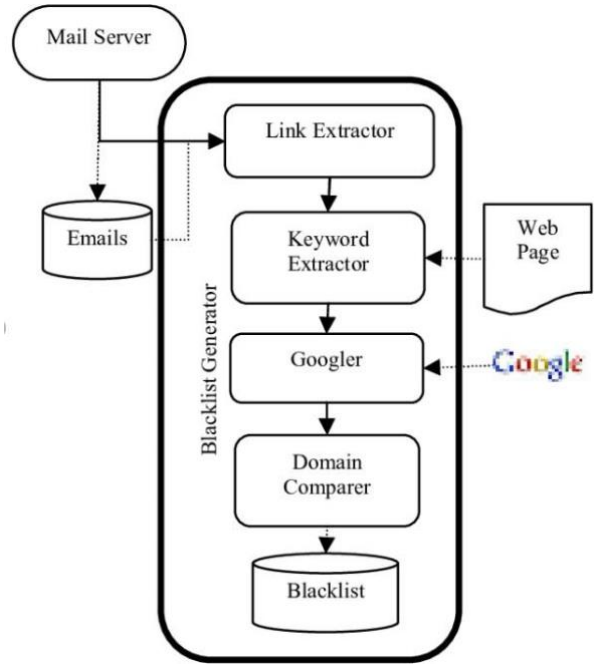

Figure 2: Basic Architecture of Blacklist method

Another drawback of this method is that an extensive record of malicious URLs is almost impossible to maintain, especially as new URLs are created every day. In addition, attackers now use ingenious techniques to escape blacklists and fool users by modifying the URL to make it seem legitimate through obfuscation [4].

2.3 Research and Implementation

We first used Jupyter Notebook to run our python script and statistical learning knowledge to ascertain which of the three shortlisted models was the most accurate for our project.

2.3.1 Dataset

The most important step was to come up with an appropriate data set. After careful research, we obtained ours from UCI (University of California, Irvine) machine learning repository. The dataset contains 31 columns, with 30 features and 1 target. The dataset has 2456 observations.

2.3.2 Technology Used

We coded the whole project in Python3 using MacOs.
The **major python libraries** used:

- BeautifulSoup
- Sci-kit Learn
- tdlextract
- imaplib
- pythonssl

2.3.3 Manipulation

We then ran a Python script to determine if the following features were present or not. In some cases, the presence of the features corresponded to phishing, and in the other, the absence did. For example, given below are some features which were checked.

- The use of URL Shortening services
- The use of Long URL to hide suspicious part

- Presence of HTTP vs HTTPs
- Length of Domain Registration
- Subdomain
- Addition of Prefix or Suffix
- Redirection tactic
- Contains symbol "@"
- The use of IP Address
- The use of favicon
- The use of non-standard port

Furthermore, we also went on to observe certain abnormalities in URL:

- Anchor Text in URL
- If the email contains information submitted
- Abnormal URL
- Website forwarding activated
- Right Click disabled
- Server from Handler (SFH)
- Customization of Status Bar
- Links in <Meta>, <Script> and <Link> tags

The following figure (Figure 3) shows one of the graphs that depicts that IP address was used in malicious rather than benign URLs

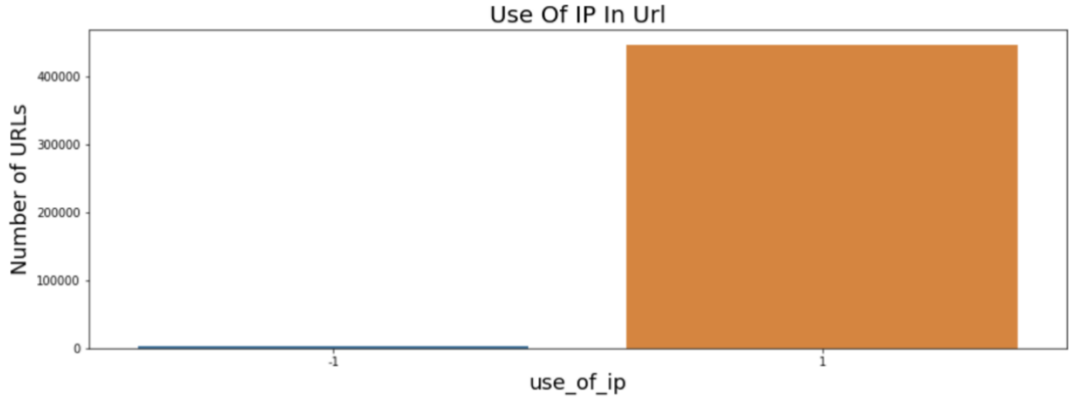

Figure 3: Use of IP Address in Benign or Malicious URL

2.3.4 Data Training

We split the data into training and testing in the ratio 75-25. The following classifiers were shortlisted for our project after careful research:

- **Logistic Regression** - It is used when target variable is categorized. It hinges on Maximum Likelihood Estimation (MLE) and is a qualitative choice model. It is used to predict whether the risk factor increases the odds of a given outcome by a specific factor. Logistic Regression can be used to model binary classification problems. The mathematical representation is given as -

$$F(x) = \frac{1}{1 + exp(-w^T x)}$$

where F can take values in the range 0 to 1.

  We created a confusion matrix of real and predicted values to obtain an accuracy of 92.3%

- **Support Vector Machine (SVM)** – It is a linear classifier algorithm based on supervised learning. It helps to create a boundary between the variables to classify them. It creates a hyper plane boundary with maximum margin to separate the variables. This algorithm is robust to outliers. The coordinates of individual observations are called support vectors. SVM creates a hyperplane separating support vectors with the maximum possible margin. This with a “gridsearchcv” and “rbf kernel” for prediction gave an accuracy of 96.47%
- **Random Forest Classification** – It is a supervised learning algorithm used in both classification and regression problems. In the random forest classifier, to get high accuracy results we need to create large number of decision trees. The prediction obtained from a Random Forest is prone to be far better than the predictions obtained by an individual decision tree. Random Forest utilizes the concept of bagging for creating several minimal correlated decision trees. Advantages of Random forest is its ability to handle missing values and to avoid overfitting the model. We obtained features’ importance and used “gridsearchcv”

to obtain accuracy of 97.26% which was the highest of all. This is why we decided to choose this for our project.

2.3.5. Working Explanation

1. We used the "imap library" of python to log into user's email account. Further, we filtered our extraction to procure only the unread email from the inbox and wrote our code to further only save the body of the email into a file in the project.
2. Then, we wrote code to read the contents from the saved file and detect the presence of a URL. In the event that it is present, we run it through our python script which predicts whether the URL is malicious or not using our models.
3. The results are accordingly displayed to the user

The gist of our whole logic and implementation can be easily understood from the model below (Figure 5):

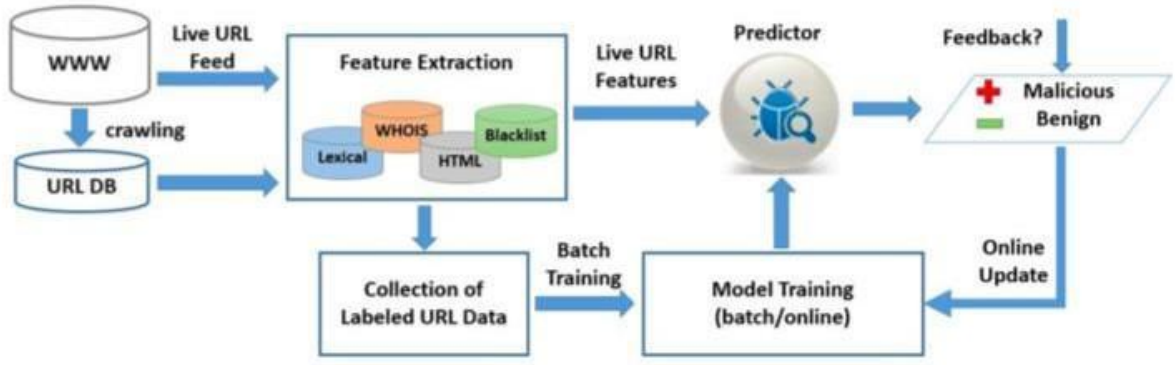

Figure 4: Framework for using Machine Learning to detect malicious URL

3. Findings

This section briefly talks about the results, the current status of the work and the limitations or difficulties that has been faced while completing the project. This is followed by a brief discussion of what can be done in the future.

3.1 Current Status

The research phase has been completed and the implementation phase is also almost done. Essential features have been extracted from the dataset that was found online. These can be seen from the figure below:

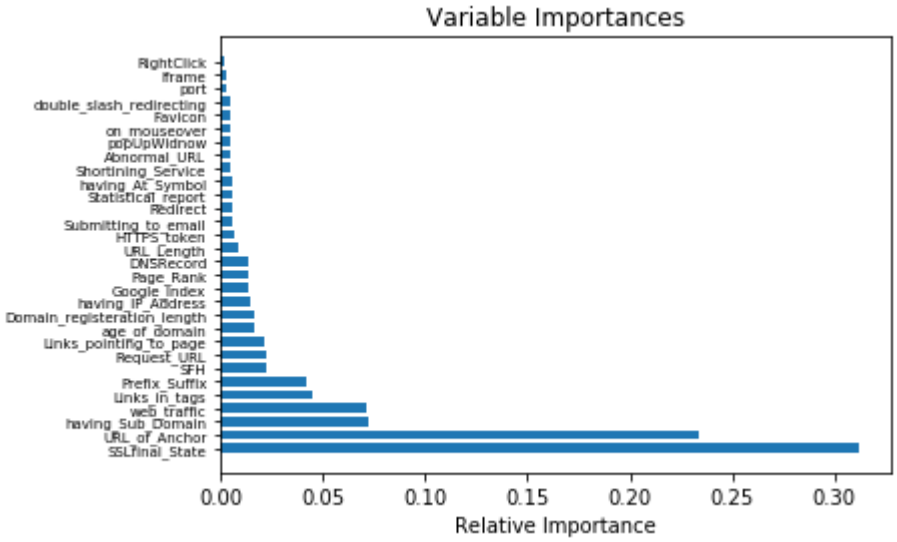

Figure 5: Relative importance of features

Following that the data has been preprocessed and visualized after which the machine learning model has been built. Then, the email scraper was built to scan URLs on top of which the business logic would run to show the results.

3.2 Results

We have now created a system to scrape the last email received and extract its body to detect whether it contains a URL or not and thence ascertain if it is a phishing attack or not with an accuracy of 97.26%.

Our code can be accessed on GitHub: https://github.com/guptatrisha97/Phishing-URL-Detector

The deliverable that we have right now gives the following output (Figure 6) after running through the whole logic.

```
tgupta-4whv2t:URL-Phishing-Detector tgupta$ python3 -W ignore index.py
/usr/local/lib/python3.7/site-packages/sklearn/externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functiona
lity directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
enter url
https://web.whatsapp.com/
URL seems safe
tgupta-4whv2t:URL-Phishing-Detector tgupta$ python3 -W ignore index.py
/usr/local/lib/python3.7/site-packages/sklearn/externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functiona
lity directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
enter url
https://www.netflix.com/browse
URL seems safe
tgupta-4whv2t:URL-Phishing-Detector tgupta$ python3 -W ignore index.py
/usr/local/lib/python3.7/site-packages/sklearn/externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functiona
lity directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
enter url
https://www.google.com/
URL seems safe
tgupta-4whv2t:URL-Phishing-Detector tgupta$ python3 -W ignore index.py
/usr/local/lib/python3.7/site-packages/sklearn/externals/joblib/__init__.py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this functiona
lity directly from joblib, which can be installed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
enter url
https://www.goggle.com
URL may be a phishing attack!
```

Figure 6: Example of output after running our system

3.3 Difficulties

It has been quite difficult to create the model from scratch using machine learning. The first difficulty was to obtain a vast database of URLs. However, this challenge was overcome after “Kaggle” was introduced which provides free online datasets. The next difficulty was that we first used the “blacklist” methodology but then it had high false positive rate and was not the best method to use. This was overcome by changing the method completely. Another difficulty was how to extract the content of the emails but the “imap library” in python was found to be the solution.

3.4 Future Works

The future work shall include further revamping our code. A major last step that needs to be done is adding a plug-in on the browser so that the script need not be run on a terminal but directly on the email account of the user, such that a popup declares whether the email containing a URL is malicious or not. Once this is implemented, it shall take us to the end of our expected deliverable.

The table below (Figure 7) shows the timeline for the completion of the project.

Timeline	Milestones	Deliverables
September	Proposal	• Project plan • Project Webpage
October - November	Design and Research	• Improve project webpage • Research and compare algorithms
Late November – Mid-March	Research	• Obtain data sets • Create ML models
January	Development	• Train data • First Presentation • Get model working • Display output
February-March	Development	• Detailed interim Report • Improve UI • Finish deliverable • Revamp Website
April	Testing	• Finalized tested implementation • Final Report
End April	Deployment	• Final Presentation

Figure 7: Table showing the timeline

4. Conclusion

The detection of malicious URLs plays an important part in numerous cybersecurity applications and undoubtedly machine learning methodologies are an encouraging direction to follow [5]. This project aims to explore the application of machine learning in the context of detecting malicious URLs in emails.

We highlight several useful issues for the application domain in this document and have even identified few relevant techniques for further study. Despite extensive studies and impressive progress in recent years, automated detection of malicious URLs using machine learning still remains a very challenging open issue. As researchers design techniques to combat phishing attacks, hackers manipulate their ways to exploit any possible vulnerability. There is always a strong race between developers and such hackers. However, the future directions should take this into account, and hopefully our smart email client helps alleviate the phishing attack risk to save users' time, information and money.

References

[1]A. Rubin, M. Chew, N. Provos, and S. Garera, "A framework for detection and measurement of phishing attacks", presented at ACM workshop, Alexandria, 2007.

[2]B. Liang, D. Dong, D. Wang, F. Liu, J. Huang, and Z. Liang, "Malicious Web Pages Detection Based on Abnormal Visibility Recognition," presented at IEEE International Conference on E-Business and Information System Security, 2009.

[3]C. Liu, D. Sahoo, and S. Hoi, "Malicious URL Detection using Machine Learning: A Survey" vol. 1, pp. 2-5, August 2019.

[4]C. Kamachi and J. Ryan, *Detecting and Combating Malicious Email*. Massachusetts: Syngress, 2015.

[5]E. Buber. (2018, Feb 8*). Phishing URL Detection with ML* [Online]. Available: https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5 [Accessed: 2019, Oct 8].