Declaration on Plagiarism

Names:	Aruna Bellgutte Ramesh (18210858) Salman Khaleel Sab (18210266)
Programme:	MCM in Computing (Cloud Computing)
Module Code:	CA645
Assignment Title:	URL Classification
Submission Date:	15th April 2019
Module Coordinator:	Darragh O'Brien

We declare that this material, which we now submit for assessment, is entirely our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of our work. We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. We have read and understood the Assignment Regulations. We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by us or any other person for assessment on this or any other course of study.

We have read and understood the referencing guidelines found at

http://www.dcu.ie/info/regulations/plagiarism.shtml , https://www4.dcu.ie/students/az/plagiarism

and/or recommended in the assignment guidelines.

Names: Aruna Bellgutte Ramesh, Salman Khaleel Sab

Date: 15th April 2019

URL Classification Using whois and Lexical Features

Aruna Bellgutte Ramesh
MCM Computing (Cloud Computing)
Dublin City University
Dublin, Ireland
18210858

Salman Khaleel Sab
MCM Computing (Cloud Computing)
Dublin City University
Dublin, Ireland
18210266

I. INTRODUCTION

Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it [1]. By accessing a URL, one can retrieve a web page or achieve a connection to database or achieve file transfers (based on URL protocols), located at some server within the computer network.

But not all the URLs are safe to access. Some of them may lead to unintentional downloading of viruses; some URLs may serve a web page that may look very similar to your banking website – ending up fraudulently accessing your bank credentials or other sensitive information. Such URLs that aren't safe to access are known as **Malicious** URLs and the ones safe to access are known as **Benign** URLs.

The malicious URLs look very similar to benign URLs - duping users and tricking them to trust malicious URLs just as if they were benign. This is perhaps the most common and most vicious characteristic of a malicious URL. In this paper we attempt to breakdown and study various other characteristics / features of a malicious URL; use them to train our Machine Learning (ML) model and achieve classification of URLs into malicious or benign.

The rest of the paper is organized as follows: in Section 2, we briefly overview related work of previous researches in the same field. In Section 3, we talk about the dataset used for our work. Section 4 describes the Feature Extraction and Implementation details. Section 5 shows the results obtained. Finally, Section 6 provides the conclusion and discussion on future improvements.

II. RELATED WORK

This section describes the related techniques and methods implemented by previous researchers for URL classification.

A. Sujata et al. [2]

Sujata et al. [2] designed a machine learning framework which can detect phishing URL. They worked with the following features:

Page-Based: is a numeric value which indicates the page rank of a website. Benign websites tend to have higher page rank; whereas since phishing websites do not exist for longer periods their page rank will either be less or page rank wont exist at all [3].

Domain-Based: indicate if any URL's domain is present in white domain table.

Type-Based: They classified the phishing URL's into four categories which describes how attackers can make victims to click any link which leads to phishing website.

Word-Based: Phishing URLs are found to have certain common words such as 'banking, 'login'. They extracted all of these words from phishing URLs and used those words as library to detect if any new URL contains these words.

Using all of these features they built a machine learning framework which determines the average number malicious URLs which were redirected from google toolbar. Results showed that on average 8.24% of the people who visits phishing websites are potential victims.

B. Mohammed et al. [4]

Mohammed et al. [4] in their work achieved 98.46% in detecting phishing URL. They developed a machine learning framework which detects any phishing URL based on the results of **Microsoft Reputation Services** (MRS) [5] and other three features extracted from a URL which are **Lexical, Host-Based** and **Cluster Label.** MRS returns one or more labels for any given URL such as benign, moderate and severe. End user should determine if any URL is malicious based on the results returned from MRS. In this paper they used results of MRS and other three features mentioned above to determine if any URL is malicious.

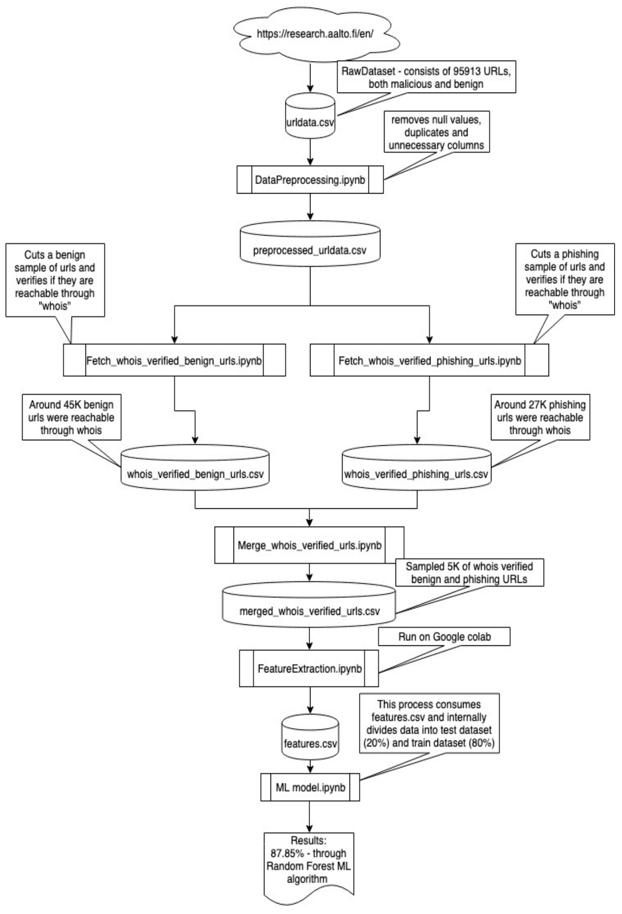
III. DATASET

Dataset was obtained from Aalto University's research portal [6]. Fig. 1. describes all the steps involved in dataset management process; starting from downloading the data to fitting it into a Machine Learning model.

Major steps involved were:

- **Data Preprocessing** : removed nulls and duplicate records since they affect our Machine Learning (ML) model. Also, removed unnecessary columns and retained only URL and Label columns.
- **whois Verification** : the preprocessed data was split into benign sample set and malicious sample set. These two sample sets were individually checked for reachability of whois to ensure our features data is not filled with meaningless features and throwing ML model off balance.

- **Feature Extraction** : the whois verified benign and malicious URLs were further sampled to 5000 URLs each and merged to get a final dataset for extracting features. Features were extracted.
- **Machine Learning**: Features extracted were used to train our ML model and prepare our machine for URL classification.



https://research.aalto.fi/en/
RawDataset - consists of 95913 URLs, both malicious and benign
urldata.csv
removes null values, duplicates and unnecessary columns
DataPreprocessing.ipynb
preprocessed_urldata.csv
Cuts a benign sample of urls and verifies if they are reachable through "whois"
Cuts a phishing sample of urls and verifies if they are reachable through "whois"
Fetch_whois_verified_benign_urls.ipynb
Fetch_whois_verified_phishing_urls.ipynb
Around 45K benign urls were reachable through whois
Around 27K phishing urls were reachable through whois
whois_verified_benign_urls.csv
whois_verified_phishing_urls.csv
Merge_whois_verified_urls.ipynb
Sampled 5K of whois verified benign and phishing URLs
merged_whois_verified_urls.csv
Run on Google colab
FeatureExtraction.ipynb
This process consumes features.csv and internally divides data into test dataset (20%) and train dataset (80%)
features.csv
ML model.ipynb
Results: 87.85% - through Random Forest ML algorithm

Fig. 1. Dataset management process

IV. FEATURE EXTRACTION AND IMPLEMENTATION

We divided the features into two categories: Dynamic Features and Lexical Features.

A. *Dynamic Features*

1. **whois features**: whois can be performed through python's whois module. whois python module gives us many information about the URL such as – domain name, registrar, whois_server, creation_date, updated_date, expiration_date, name_servers, emails, etc. Amongst these, the feature on dates i.e., creation_date, updated_date and expiration_date were found to be the **most salient**. Reasons being:
   - **creation date** validates most of the benign URLs. The older the URL more the chances of it being benign.
   - Most hackers don't keep their domains running for longer periods. Hence, **expiration date** will be useful.
   - Similarly, **updated date**, in combination with creation / expiration date, can show genuinity of a URL.

   Other features like zip code and address, that could have potentially been used for classification, were returned as null for most of the URLs, both benign and malicious, for the dataset we used. Hence, only the **date features** were finally **extracted** and used for classification.

   whois module of python, in our opinion, still has scope for development. For example, whois on google.com doesn't return **Registrar URL** feature at all since the python module doesn't hold the key for it. Had this key / feature been available in python whois module, we could have performed a whois on that Registrar URL value to find out if there was a back reference to "google.com" or explored other results. Even a simple absence of the Registrar URL feature could classify some URLs maybe. Likewise, there are many such features that are available in other whois websites but not available in the whois python module. A fully developed python module can be more powerful in dynamically classifying URLs.

2. **nslookup features:** nslookup can be performed with the help of python's DNS resolver module. **DNS resolver** helps us get results such as the type of **IP** address (IPv4 or IPv6), **MX** details (provides information on mail server for the zone), **NS** details (provides information on name servers associated with the zone) and **PTR** details (for reverse lookups) among others. However, in our work we found that many URLs that were **reachable to whois** were **not reachable to DNS resolver** and vice versa - leading to **shrinkage** of workable **dataset** size. Since most of the URLs were whois responsive, we had to trade using only whois features for using both whois and DNS resolver features. Also, DNS resolver mostly yields features about the zone; which is more general compared to the whois results. Hence, in our study we only used whois features and dropped DNS resolver features.

B. *Lexical Features*

Malicious URLs and domains show certain characteristics that cannot be seen in any benign URLs. For an instance the length of the URL. Phishing websites tend to have a lengthy URL consisting of a long query string which will lead users to phishing websites. These phishing websites may ask user to enter credit card details. We also noticed that few phishing URLs contain more than one protocol - which is basically a link to some other malicious website appended as a query string. It was useful that we extracted this feature, **Protocols Count,** which has never been done in any of the previous research works so far. We captured such features of a URL by studying the lexical content of each URL. Table. 1. represents the lexical features we captured.

Lexical Features
Dot Count

URL length
Digits count
Special Characters count
Hyphen Count
Double Slash Count
Single Slash Count
@ sign count
Protocols Count

Table. 1. Lexical feature extracted

V. Results

We performed this study using the following protocol. We divided the data into two sections, 20% of the data, that is 2000 URLs, were used for testing and rest 80% of the data, that is 8000 URLs, were used for training. We trained our Machine Learning model with the following Machine Learning Classification algorithms: Decision Trees, KNN, Kernel SVM, Logistic Regression, Naïve Bayes, Random Forest and Support Vector Machine. Table. 2. shows the results achieved by different classification algorithms; sorted by Accuracy in a descending order. It is obvious from the table that Random Forest, Decision Tree and KNN are the top three performers amongst all evaluated classifiers. Random Forest achieved the best results by gaining accuracy of 87.85%. It misclassified only 243 URLs out of 2000 URLs on which we predicted.

However, the results achieved by top four algorithms are almost same which is above 84% with no such huge difference. It is notable that KNN and Decision Tree achieved almost same accuracy with a difference of 0.3%. Decision Tree and Random Forest achieved almost similar accuracy having good balance between false positives and false negatives. From the results it is notable that Logistic Regression, Naïve Bayes and SVM are the bad performers amongst all the evaluated classifiers achieving accuracy score less than 80%. Naïve Bayes obtained the least accuracy of 69% which sits at the last position. The bad performing algorithms owes its bad results to the virtue of the algorithms being basic and naive. The top performers owes its good results to the recursive approach of their algorithms.

Algorithm	**Accuracy %**	**False Positives**	**False Negatives**
Random Forest	87.85	121	122
Decision Tree	86.8	129	135
KNN	86.55	137	132
Kernel SVM	84.5	116	194
SVM	77.65	282	165
Logistic Regression	75.4	277	215
Naïve Bayes	69.0	9	611

Table. 2. Results achieved by various classifiers

VI. Conclusion and Future work

Based on our results we can conclude that Random Forest obtained the best baseline accuracy for further comparison.

We did not get the recent verified benign URLs data for our work. So, we made use of an older dataset where, as previously discussed in the paper, most of the URLs were not reachable through DNS resolver (nslookup) and / or whois python modules. Due to this we were not able to capture more features. In future, we intend to find better a dataset and develop a better whois / DNS resolver python module where we can extract more salient features efficiently and evaluate them with different Machine Learning classifiers.

References

[1] "URL - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/URL. [Accessed 13 04 2019].

[2] N. P. a. M. C. Sujata Garera, "A Framework for Detection and Measurement of Phishing Attacks," in ACM workshop on Recurring malcode (pp. 1-8), 2007.

[3] "PageRank - Wikipidea," [Online]. Available: https://en.wikipedia.org/wiki/PageRank. [Accessed 12 04 2019].

[4] S. M. Mohammed Nazim Feroz, "Phishing URL Detection Using URL Ranking," in IEEE, New York, NY, USA, 2015.

[5] "Microsoft Reputation Services," [Online]. Available: http://www.microsoft.com/security/portallmrs/. [Accessed 12 04 2019].

[6] "Aalto University," [Online]. Available: https://research.aalto.fi/portal/en/. [Accessed 13 04 2019].