Malicious URL Detection

Alon Firestein, Yosef Danan

Course: Methods for Detecting Cyber Attacks
Lecturer: Dr. Ran Dubin
Department of Computer Science and Mathematics
Ariel University

I. INTRODUCTION

The main idea for this project is to research and create tools to identify and single out malicious URLs from our dataset of hundreds of thousands of URLs using machine learning tools learned from our "Methods for Detecting Cyber Attacks" course. Malicious URL is a link created with the purpose of promoting scams, attacks, and frauds. By clicking on an infected URL, you can download ransomware, virus, trojan, or any other type of malware that will compromise your machine or even your network.
A malicious URL can also be used to persuade you to provide sensitive information on a fake website. Notice that it isn't just links with malware that can be propagated on the internet, after all, there are several types of threats.
The fact is that a short, simple URL can cause a lot of damage. The potential harm is so big that malicious links are considered one of the biggest threats to the digital world, especially when we talk about attacks and threats that arrive by email.

Malicious URLs host unsolicited content (spam, phishing, defacement, etc.) and lure unsuspecting users to become victims of scams (monetary loss, theft of private information, and malware installation).
Thus, causing damages and losses of billions of dollars every year to users and companies worldwide.

Therefore, our aim in this project is to firstly preprocess the data in a unique and informative way to get as many details as possible and features for each URL in our dataset. This stage includes creating lexical features, content-based features, and host-based features. As a result, from our newly detailed dataset, we used different machine learning and deep learning algorithms to get the highest score possible from our models' predictions. Models such as XGBoost, Random Forest, MLP Classifier, Decision Tree, KNN, and an LSTM neural network with char embedding.

II. RELATED WORK AND OUR OBJECTIVE

As explained previously in the introduction (I), this topic is a big issue worldwide especially given the ever-increasing usage and demands in computers and the numbers of users, whom have a wide range of technological knowledge. Meaning that this has been researched quite often and many different approaches have implemented to try and combat this phenomenon.

We have come across several research papers before starting this project to better understand what has been done and what is expected, and in the end each research group sticks to a certain aspect of a machine learning model to focus on it solely. These include Associative Classification, Convolutional Neural Networks, SVM, Logistic Regression... and more.

We tried to "revolutionize" this project by extracting many features solely based on the URL (as explained in the introduction about the preprocess stage) and using an LSTM model in addition to well-known regular machine learning, deep learning and boosting models.

III. DATA EXPLORATION AND DATA PREPROCESSING

In the first stages of this project, it is essential to get a good understanding of how the dataset is organized, labeled, and divided.
To start, we used a public dataset from Kaggle[1] containing 651,191 different URLs and its label (phishing, defacement, malware, benign).
Below we can see the balance of the data and its labels.

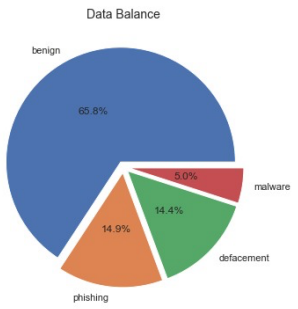

Figure 1.0: Data balance

Before we started the preprocessing stage to create more features for each URL, using plotting libraries in Python (Matplotlib and Seaborn), we were able to get a good visual of how the data is represented so we could take a direct action in organizing the data to prepare it for the training models. This includes handling missing data, meaning we dropped the features that had more than 20% of its total data missing (null) and the rest of the features that have some missing data, we drop the rows with null.

Thereafter, for the initial attempts of this project the preprocessing stage included extraction of lexical features of the URL text itself. This refers to statistical features extracted from the literal URL string.

In the table below, we can see the type of lexical features that we used for our model.

SN	FEATURE NAME	DESCRIPTION
1	URL Length	Total number of characters in URL
2	URL Scheme	The URLs scheme or ***protocol*** represents the set of rules that decide how files would be displayed and formatted and how data would be transported across the web.
3	URL Path Length	The length of the URL path which would also include subdirectories on the web server
4	URL Host length	The total number of characters in the domain or host name. For example 'google.com' will have a character length of 10.
5	URL Host is ip	If the host or domain part of the URL string is an ip address rather than a domain name. For example 'google.com/help' is represented as '8.8.8.8/help'
6	Number of Digits	Number of digits in URL strings [0-9]. For example 'the boot567.au' has a num-digit-count of 3.
7	Number of parameters	The number of values being queried on the site if a search is being performed. For example, 'amazon.com?color=pink&size=20'
8	Number of fragments	Fragments are optional components of a url string that leads to another resource that is a subordinate of a promary resource. This is usually precedded by a '#' sign. for example www.examplewebsite.com/aboutus#whatwedo
9	is Encoded	URL is URL-encoded using a '%' character and a two-character hex value corresponding to their **UTF-8** character.
10	URL Entropy	Shannon entropy of URL string
11	Similar To Alexa	Average similarity of URLs host name to any Alexa site
12	Number of Subdirectories	Number of subdirectories in URL path
13	Number of periods	Number of '.' in URL string
14	Has Keyword 'client'	URL String Contains the keyword 'client'
15	Has Keyword 'admin'	URL String contains the keyword 'admin'
16	Has keyword 'server'	URL String contains the keyword 'server'
17	Has keyword 'login'	URL String contains the keyword 'login'
18	has port	URL string has a port number in it for example examplewebsite.com:8080

Figure 2.0: Lexical Features Description

For example, length of the URL string, number of digits, number of parameters in its query part, if the URL is encoded, etc.… (see [2] for more details).

On the figure below, we can see our model results.

Model Name	Accuracy	F-score 0	F-score 1	F-score 2	F-score 3
Decision tree	81	88	67	82	55
Knn	84	90	77	63	64
Knn PCA	83	90	76	60	64
Logistic	78	87	74	68	27
Logistic PCA	78	86	73	62	26
MLP	92	95	84	86	90
Random forest	79	87	64	72	44
Random forest PCA	87	92	81	86	68
Xgboost	86	91	82	87	55
Xgboost PCA	87	91	83	87	61

Figure 3.0: All model scores and comparisons (PCA=99%)

We can see and infer that MLP had our best result in the initial attempts.

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, 60)                4680
_________________________________________________________________
dropout_8 (Dropout)          (None, 60)                0
_________________________________________________________________
dense_33 (Dense)             (None, 30)                1830
_________________________________________________________________
dense_34 (Dense)             (None, 15)                465
_________________________________________________________________
dense_35 (Dense)             (None, 5)                 80
=================================================================
Total params: 7,055
Trainable params: 7,055
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.0 MLP layer details

Ensuing the initial attempts, the plan was to achieve better scores and precision of our models predictions, thus a deeper work of the preprocessing stage underwent further operations to create a better and more detailed dataset from the URL itself.

This included in addition to the lexical features, content-based features were also taken, these are obtained from the downloaded HTML code of the webpage. These features capture the structure of the webpage and content embed in it.

These will include information on script tags, embedded objects, executables, hidden elements, etc.
Furthermore, host-based features were also extracted, these are the characteristics of the host-name properties of the URL.

These provide information about the host of the webpage, for example, country of registration, domain name properties, open ports, name servers, connection speed, time to live from registration, etc.

Unfortunately, using these feature extraction methods is very time consuming and needs working URLs, and with problems that occur because of rate-limiting on websites from Alexa's top 10000 list and similar URL list websites, we our unable to extract all these features in a normal amount of time.

Therefore, in addition, we worked on an additional dataset that has already includes these previously extracted features for each URL, this dataset included five types of URLs: Benign (~35,300), Spam (~12,000), Phishing (~10,000), Malware (~11,500), Defacement (~45,450).

This dataset included 80 features in advance. In the image below, we can see a table explaining about several features that our data uses, which included lexical features and HTML features of the website itself.

This dataset was provided to us from the Canadian Institute for Cybersecurity from the University of New Brunswick [3]. With this dataset, we were able to achieve accurate predictions even with the "simple" models. Seen on the graph below the feature table, are the results our models obtained from this dataset.

Feature Name	Feature Group	Feature Discription
URL Entropy	URL String Characteristics	Entropy of URL
numDigits	URL String Characteristics	Total number of digits in URL string
URL Lenght	URL String Characteristics	Total number of characters in URL string
numParameters	URL String Characteristics	Total number of query parameters in URL
numFragments	URL String Characteristics	Total Number of Fragments in URL
domainExtension	URL String Characteristics	Domian extension
num_%20	URL String Characteristics	Number of '%20' in URL
num_@	URL String Characteristics	Number of '@' in URL
has_ip	URL String Characteristics	Occurence of IP in URL
hasHTTP	URL domain features	Website domain has http protocol
hasHTTPS	URL domain features	Website domain has http protocol
urlIsLive	URL domain features	The page is online
daysSinceRegistration	URL domain features	Number of days from today since domain was registered
daysSinceExpired	URL domain features	Number of days from today since domain expired
bodyLength	URL page frature s	Total number of characters in URL's HTML page
numTitles	URL page frature s	Total number of HI–H6 titles in URL's HTML page
numImages	URL page frature s	Total number of images embedded in URL's HTML page
numLinks	URL page frature s	Total number of links embedded in URL's HTML page
scriptLength	URL page frature s	Total number of characters in embedded scripts in URL's HTML page
specialCharacters	URL page frature s	Total number of special characters in URL's HTML page
scriptToSpecialCharacterRatio	URL page frature s	The ratio of total length of embedded scripts to special characters in HTML page
scriptToBodyRatio	URL page frature s	The ratio of total length of embedded scripts to total number of characters in HTML page

Figure 4.0 UNB Dataset Feature Description[5]

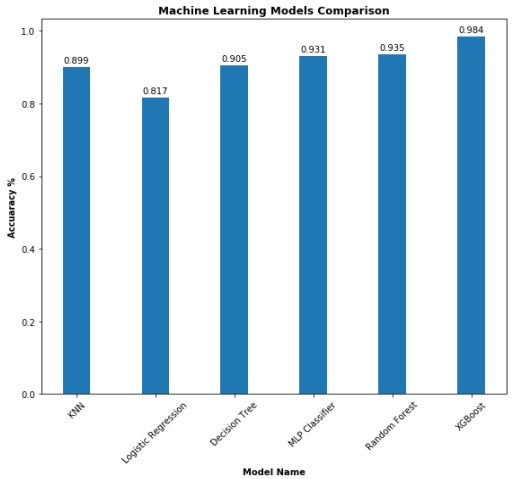

Figure 5.0: Model scores on UNB dataset

IV. OUR FINAL APPROACH

After having a better understanding of our data and our main goal. We ended up not using these features and used an LSTM neural network with char embedding and the results were outstanding.
With emphasis on the f-scores of the LSTM model. This is because model accuracy is used when the true positives and true negatives are more important, while f-scores are used when the false negatives and false positives are crucial.

It is important to clarify that although in our preprocessing stage we were able to extract many different features that could be useful for the classification of the URL, our final approach used only the "lexical features" of the dataset (and not the host-based and content-based features).
This is because we had a lot more data that was lexical, several websites from the dataset were not active anymore, and in terms of time consuming and time complexity was a big factor of using only "lexical features".

For example, in case this project was bounded for deployment and production, it is not reasonable to have a latency of a couple seconds for the crawler to enter the website of each URL and check all its components to extract the required features. Therefore, we concluded that the best choice in this manner was to go with a lighter load of features to avoid these issues.

LSTMs are a special kind of RNN, capable of learning and keeping track of long-term dependencies. It can process not only single data points, but also entire sequences of data.
Char embedding are constructed similarly to word embeddings. However, instead of embedding at the word level, the vectors represent each character in a language.

These types of embeddings do not encode the same type of information that word embedding contain. Instead, char level embedding can be thought of encoded lexical information and may be used to enhance or enrich word level embeddings.

But more importantly, char level embeddings can handle and is a good fit for new slang unknown words and misspelling words (gooogle instead of google…) which are crucial in attempting to classify URLs as malicious or benign. Meaning, every word's vector can be formed even if it is "out of vocabulary" words.
Also, most URLs are known to be short and bounded in order to be readable and accessible, and LSTM is good with a small number of characters therefore it is a good fit for this problem.

In addition, char embedding handles infrequent words better than word2vec embedding as the later one suffers from lack of enough training opportunity for those rare words (and characters) which might appear in URLs. Another reason we decided on using char embedding, is that as there are only a small number of vectors, it reduces model complexity and improving the performance (in terms of speed).

Thus, we can clearly understand why the choice of user char embedding for malicious URL classification is a good choice.
In the graph below, we can see our model train metrics of the LSTM neural network.

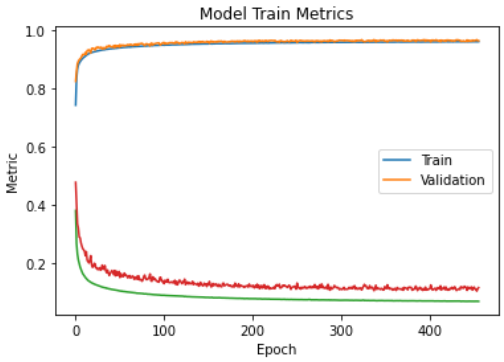

Figure 6.0: LSTM model train metrics with char embedding

LSTM Model Scores (classification report):

	precision	recall	f1-score
0	0.98	0.97	0.97
1	0.97	0.99	0.98
2	0.97	0.95	0.96
3	0.87	0.91	0.89
accuracy			0.96

Figure 7.0: LSTM model classification report

V. CONCLUSION

To summarize, for this project we attempted many different approaches to finding a "different" solution to this issue in comparison to what other researchers have done in the past. This includes different preprocessing stages and feature extraction and data exploration using plots to understand the data and plan our approach.

This report aims to design a new malicious URL detection approach using an LSTM neural network and char embedding.
The many man and machine hours that were put into this, including time training the model until the results were received were worth it to have a better understanding of this issue, and we were able to find a unique solution for this worldwide cyber issue. Performance wise was also a great success, with the whole prediction pipeline process duration taking less than 100ms.

REFERENCES

[1] https://www.kaggle.com/sid321axn/malicious-urls-dataset

[2] https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a

[3] https://www.unb.ca/cic/datasets/url-2016.html

[4] https://en.wikipedia.org/wiki/Principal_component_analysis

[5] https://medium.com/nerd-for-tech/url-feature-engineering-and-classification-66c0512fb34d