**EVENT-DRIVEN MALICIOUS URL EXTRACTOR**

Wishvajith B.L.D.V.

(IT18032666)

B.Sc. (Hons) Degree in Information Technology Specializing in Cyber Security

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

October 2021

**EVENT-DRIVEN MALICIOUS URL EXTRACTOR**

Wishvajith B.L.D.V.

(IT18032666)

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of

Science in Information Technology Specializing in Cyber Security

Department of Information Technology

Sri Lanka Institute of Information Technology

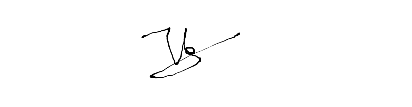
Sri Lanka

October 2021

# Declaration

“I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”



Signature: Date: 08/10/2021

The above candidate has carried out research for the bachelor’s degree Dissertation under my supervision.

Signature of the supervisor: Date: 08/10/2021

# Abstract

Cyber-attacks are attacks that are commonly carried out in order to obtain sensitive information or disrupt internet-based services. Recent occurrences, both internationally and locally, have shown an influx of these attacks expanding rapidly through the use of malicious URLs (Uniform Resource Locators). Traditional measures, including such blacklisting malicious URLs, make it extremely difficult to respond to such attacks in a timely and efficient manner. Most existing solutions remain restricted in terms of scalability and proactive user safeguarding in situations when freshly formed URLs are correlated with a recent event, such as Covid-19 related frauds. The proposed solution is presented with the primary aim of addressing traditional system limitations and offering an interface for users to protect themselves by detecting phishing/malicious URLs in real time. In this research, we will examine extracting user-input event-related keywords and leveraging NLP (Natural Language Processing) algorithms to match them with the accompanying URL (Uniform Resource Locator) token data to determine whether the URLs are malicious or benign. The study further suggests using a browser extension to automate the process of blocking malicious URLs from an endpoint device, making it a much more user-friendly solution.

Keywords— Malicious URL Detection, Machine Learning

# Dedication

My dissertation is dedicated to my family and friends. I owe a special debt of appreciation to my beloved parents, Renuka and Sunil Chandrarathna, and, my sister, Ayesha.

I also want to dedicate this to my friends, which showed assistance for me the entire way. I'll remain grateful for everything they've done to help me reach this point.

# Acknowledgement

This project would not have been achievable without the help of Mr. Amila Nuwan Senarathne, our project supervisor, who read through each revision and assisted with guidance along with the project.

I would also like to show my gratitude to the Sri Lanka Institute of Information Technology for providing the opportunity to conduct this research project.

**Table of Contents**

[Declaration I](#_Toc84536611)

[Abstract](#_Toc84536612) II

[Dedication](#_Toc84536613) III

[Acknowledgement](#_Toc84536614) IV

[List of figures](#_Toc84536615) VII

[List of tables](#_Toc84536616) VIII

[List of abbreviations](#_Toc84536617) IX

[List of Appendices](#_Toc84536618) X

[1 Introduction 1](#_Toc84536619)

[1.1 Background & Literature survey 1](#_Toc84536620)

[1.2 Research Gap 8](#_Toc84536621)

[1.3 Research problem 11](#_Toc84536622)

[1.4 Research Objectives 12](#_Toc84536623)

[1.4.1 Main objectives 12](#_Toc84536624)

[1.4.2 Specific objectives 13](#_Toc84536625)

[2 Methodology 14](#_Toc84536626)

[2.1 Methodology 14](#_Toc84536627)

[2.1.1 System architecture 14](#_Toc84536628)

[2.1.2 Keyword populator 17](#_Toc84536629)

[2.1.3 Comparison Engine 21](#_Toc84536630)

[2.2 Commercialization 25](#_Toc84536631)

[2.3 Testing and Implementation 26](#_Toc84536632)

[2.3.1 Data collection 26](#_Toc84536633)

[2.3.2 Testing setup 27](#_Toc84536634)

[2.3.3 Budget and budget justification 28](#_Toc84536635)

[2.3.4 Implementation 29](#_Toc84536636)

[3 Results and Discussion 31](#_Toc84536637)

[3.1 Results 31](#_Toc84536638)

[3.2 Research findings 35](#_Toc84536639)

[3.3 Discussion 36](#_Toc84536640)

[3.4 Summary of each students contribution 37](#_Toc84536641)

[Conclusion 41](#_Toc84536642)

[Reference List 42](#_Toc84536643)

[Appendix – A 45](#_Toc84536644)

# List of figures

[Figure 1.1 Example of a Uniform Resource Locator [1] 2](#_Toc84536535)

[Figure 1.2 Phishing Incidents in 2020 compared to prior years [3] 3](#_Toc84536536)

[Figure 1.3 File type distribution between 26 Jan 2020 and 24 May 2020 [4] 3](#_Toc84536537)

[Figure 2.1 Complete System Architecture 15](#_Toc84536538)

[Figure 2.2 Overall System Architecture 15](#_Toc84536539)

[Figure 2.3 Keyword Populator Architecture 17](#_Toc84536540)

[Figure 2.4 Malicious URLs Finder Architecture 22](#_Toc84536541)

[Figure 3.1 List of populated keywords prior to cleaning for test case 1 33](#_Toc84536542)

[Figure 3.2 Populated keyword accuracy against search term 33](#_Toc84536543)

[Figure 3.3 Earthquake in Ziarat, Pakistan on 07/10/2021 at 03:31 AM 34](#_Toc84536544)

[Figure 3.4 List of populated keywords for test case 2 34](#_Toc84536545)

[Figure 3.5 Classified URL features [19] 38](#_Toc84536546)

[Figure 3.6 Classified URL features [19] 39](#_Toc84536547)

# List of tables

[Table 1.1 Comparison of proposed solution with former researches 9](#_Toc84536228)

[Table 2.1 Commercialization Versions 25](#_Toc84536229)

[Table 2.2 Experiment of labeling through the classifier 26](#_Toc84536230)

[Table 2.3 Budget Chart 28](#_Toc84536231)

[Table 3.1 Comparison of similarity scores in comparison engine 31](#_Toc84536232)

[Table 3.2 Accuracy of output URLs against the scores 32](#_Toc84536233)

# List of abbreviations

|  |  |
| --- | --- |
| Abbreviation | Description |
| CNN | Convolutional Neural Network |
| IEEE | Institute of Electrical and Electronics Engineers |
| KNN | K-Nearest Neighbor Algorithm |
| LKR | Lankan Rupees |
| LSTM | Long Short-Term Memory |
| UI | User Interface |
| URL | Unified Resource Locator |
| WHO | World Health Organization |

# List of Appendices

|  |  |  |
| --- | --- | --- |
| Abbreviation | Description | Page |
| Appendix - A | Survey Questionnaire Results | 45 |

# Introduction

## **Background & Literature survey**

The rise of new correspondence innovations has had a huge effect on the development and advancement of organizations crossing across numerous applications including internet banking, web based business, furthermore, digital communication. Therefore, the significance of the World Wide Web has been consistently expanding in such various sectors. Lamentably, these advancements come combined with new strategies for attackers to assault and trick users of such internet based services and service models. There is a wide assortment of procedures to execute assaults, for example, drive-by downloads, social designing, phishing, and numerous others. As with the technological advancements, the restrictions of conventional security strategies are getting increasingly more serious given this dramatic development of new security threats, quick changes of new IT innovations, and the lack of security experts. The focused part of this research is focused on preventing and minimizing these attacks and risks by building a feasible security solution. For example phishing attacks, which is a type of attack carried out in order to obtain sensitive information by tricking people into providing them usually by presenting themselves as a legitimate entity. The greater part of these attack methods is acknowledged through spreading traded-off URLs (or the spreading of such URLs forms.)

URL stands for Uniform Resource Locator, and it is the global address of documents and other resources on the Internet. A URL is made up of two parts: protocol identifier, indicating which protocol to use, and the resource/domain name/path. A colon and two forward slashes separate the protocol identification and the resource name. [1]

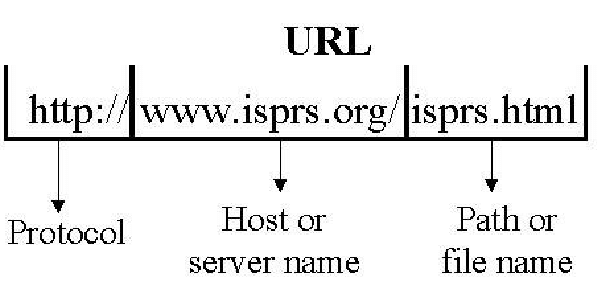


Figure 1.1 - Example of a Uniform Resource Locator [1]

Depending on the type of malicious attack performed, it can either be highly targeted to some organization or a person or it can be random. These highly targeted ones are called ‘Spear attacks’ and are often carried out as an initial step for a massive cyber-attack, which can often result in catastrophic events following the attacks.

The recent COVID-19 pandemic situation has also led to a massive increase of these cyber-attacks which have been leveraging as the world digitally transforms, affecting government, corporate industries, educational organizations, non-profit organizations because they had to shift to online platforms from their traditional on-premises systems as governments across the world have adopted rigorous isolation measures in response to the coronavirus pandemic. [2] As depicted by figure 1.2*,* according to the F5 Labs Phishing and Frauds Report for year 2020 shows a massive 220% increase in phishing activities within the year compared to the last 5 years prior to 2020. [3] Also according to the results of Falcon Sandbox [4], malicious URL is arguably the most critical part of the distribution, followed by the peexe executable, suggesting that during the COVID-19 pandemic, a significant number of malicious URLs transmitted through URLs as depicted in figure 1.3.

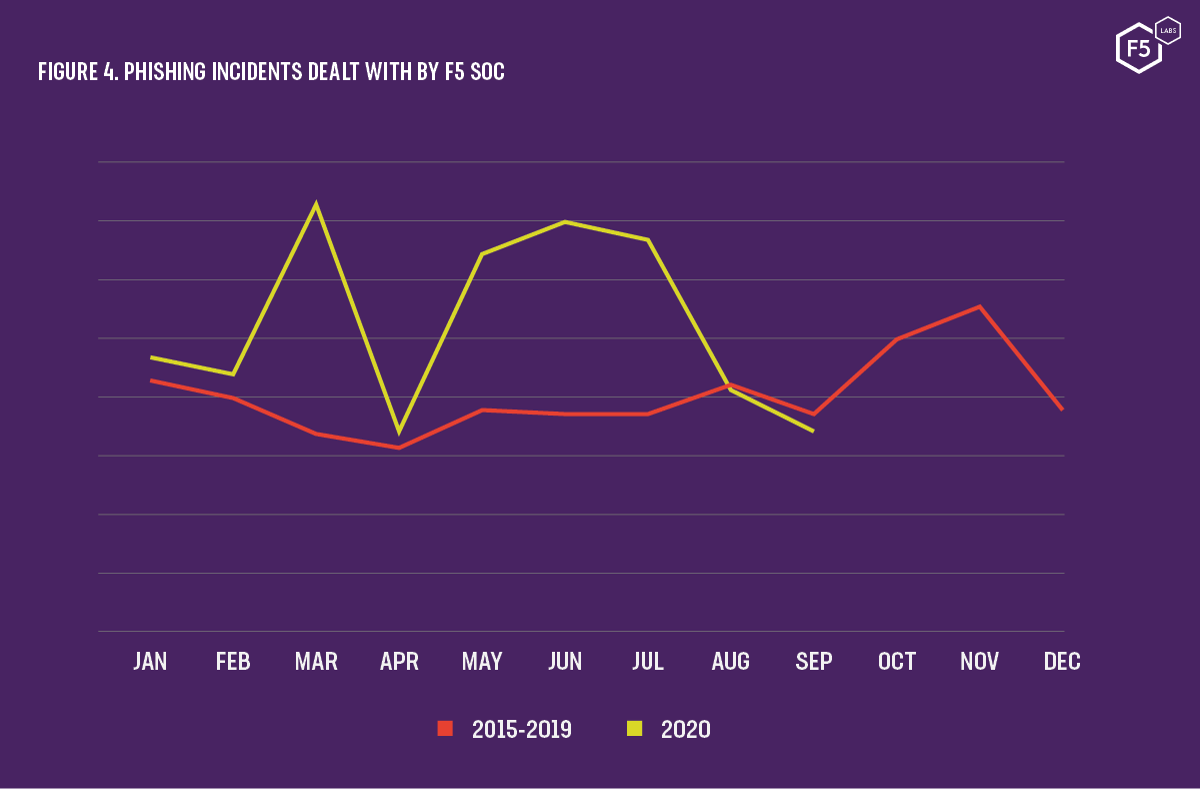


Figure 1.2 - Phishing Incidents in 2020 compared to prior years [3]

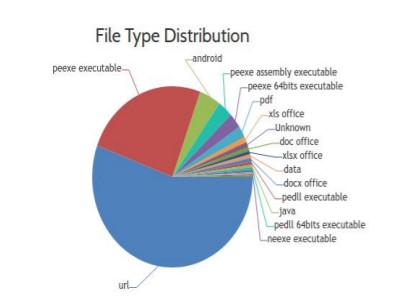


Figure 1.3 - File type distribution between 26 Jan 2020 and 24 May 2020 [4]

The Covid-19 pandemic situation has led to an alarming rate of a 2000% increase in malicious activities since January 2020 [5]. Even before the COVID 19 pandemic situation, cyber criminals have been leveraging natural disasters to launch their phishing/malicious campaigns. According to the conference paper [6], 319 individuals were tested through a survey after the Harvey hurricane, surprisingly enough around 36 % of individuals received unsolicited emails relating to the hurricane and what is even more interesting is the fact that 10% of the surveyed people have clicked on these distributed malicious links.

Unfortunately, as more companies move their workforce online, the prevalence of phishing attacks has also risen. These phishing attacks can then often lead to massive cyber-outages or attacks. Also, cybercriminals since the beginning have had come up with different techniques to make phishing websites look as authentic as possible. As with such incidents like COVID – 19 pandemic situation, attackers always try to exploit this event-related data as an opportunity to trick users into landing on these malicious phishing pages in order to capture sensitive data. A good example would be that during the COVID-19 pandemic, front-line health care workers at UW Medicine relied heavily on tele-health to support patients remotely. Staff noticed a massive spike in phishing emails (spear phishing) persuading employees to download malware via malicious URLs during this time. [7] The propagation of malware or ransom ware on health care infrastructure can delay the diagnosis and treatment of COVID-19 patients. Most of the time when attackers try to target a particular event like this, one popular way is to make the domain name have words similar to a particular situation and user would want to visit the website due to this particular situation. Scammers frequently utilize obfuscation techniques to conceal such malicious URLs. It's predominantly reported in unsolicited emails, ad- related (link shorten services) URLs, and website owners trying to conceal an associated address. Malicious URL parts could use obfuscated benign URL token parts to get around detection systems, specially the blacklisting solutions. Numerous URL obfuscation techniques used in malicious URLs were mentioned in the paper [8]. IP address, use of benign tokens in the URL, using wide host-names, or unknown and misspelled URLs are four of them which are broadly utilized in URLs to avoid detection, as per article, [9]. A recent example would be the COVID-19 phishing URLs which had keywords targeting covid-19, WHO, vaccines in order to trick people into getting them directed to the phishing pages. The report by F5 Labs also showcases the usage of terms “covid” and “corona” in certificates and URLs peaking at nearly 14,940 in the month of March. [3]

Many previous pieces of research have been carried out on the automation of detecting phishing URLs using both machine learning and non-machine learning approaches. Most of these approaches offer options for URL blacklisting, signature blacklisting. For blacklisting to effectively work, the malicious web pages need to be regularly updated and visited by the public. Also, traditional methods like blacklisting approaches are not feasible and fall short in effectiveness as maintaining such large databases of malicious URLs is not practical. In order to solve these problems, researchers have looked into heuristic approaches relying on machine learning algorithms to identify whether a URL is malicious or not.

One of these researches proposes the usage of lexical analysis of the URLs to analyze the hostname length, URL length, and tokens in URL. The authors speculate that often in malicious URLs in order to make it appear authentic, the insertion of additional characters is used. These additional characters are often token parts of benign URLs. This can be used to detect randomly generated malicious URLs using alphabet entropy. After extracting features from URL list the data will go through a multi-class classifier and gets grouped into malicious and non-malicious and attack type used by the web pages by analyzing their URL’s lexical features. In this paper, they present a set of enough features for effective categorization, and test the approach's accuracy on a sample of over 110,000 URLs, and were able to predict the malicious state and the type of attack with a high degree of accuracy of nearly 99 % [10]. This approach has been taken into account, and we propose in our project to match the event keywords with URL list tokens to find matching sets.

In another research, researchers have introduced a framework named, “Monarch” which is propose a real-time web crawling solution to determine if the URLs are malicious or benign. It is estimated that “Monarch” could process up to 15 million URLs per day after the deployment process as per the testing performed on 11 million URLs collected from emails and Twitter inspected during the project. The system performs exactly the same as the previously mentioned research project, by heavily relying on collecting features from the URLs and feeding the data into a classifier to build a model for identification of malicious URLs. What is different is the increased efficiency in memory capabilities and algorithmic updates, which is a linear classifier based on logistic regression along with the L1-regularizer to induce relatively sparse models. [11]

Paper [12] proposes a solution for increasing the scalability of the keyword matching with URL tokens when a larger scale of data is at the analysis phase. This solution, “online learning” is much more efficient than the traditional batch processing methods. Authors categorize the online learning algorithms into two main categories depending on their application at malicious URL detection; First-order online algorithms, and Second-order online algorithms. First order online learning algorithms learn through only the first-order information alongside training data to adjust the weight vector ***w*** for classification in a sequential manner whereas, second order algorithms use second order information to improve the efficacy of learning of algorithm. Second-order algorithms are useful in cases when data is sparse and computationally intensive (due to representations of lexical features) for malicious URL detection whereas first-order algorithms are simple and easier to implement.

In paper [13], they try an alternative approach to detect malicious URLs using natural language processing (NLP) techniques. Using a dataset of over 37000 URLs extracted from Phish Tank, authors suggest extracting features; brand name similarity, word unpredictability and other NLP properties. Testing these NLP features to detect malicious URLs on the dataset shows a significant accuracy rate of 97% using Random Forrest, the model falls short due to the heavy pre-processing the URLs have to undergo making the model significantly slower than the other research models discussed.

As the above researches/case studies point out, the need for a system that is scalable and efficient at handling malicious URL detection is clear. However, most of the papers do not focus on a proper mechanism to sort malicious and benign URLs according to a particular event or provide users with such features for ease of use leading to catastrophic situations in internet related services as above discussed, and even can lead to deaths of individuals due to the fact. To combat this, in this paper, we propose an ensemble model which will have the capabilities to let users sort out malicious URLs relating to an event in real time. And as with paper [10] showcased, malicious URLs using parts of benign URLs as an obfuscation technique. We propose to use this as an advantage to time efficiently tackle the event related tokens in malicious URLs.

## **Research Gap**

There are many established tools and technologies to determine if the website the clients/ users are trying to access is a fraudulent or phishing page based on the previous data fed through the deep learning models such as keyword extractors.

These researches conducted includes,

* Extracting keywords from webpages
* Monitoring for potential malicious matches
* Identifying and warning on malicious pages
* Storage of identified malicious webpages

And machine learning has been extensively examined in the literature for identifying malicious links. The preponderance of these research focus on identifying fraudulent links utilizing features such as the website's content, the the string of characters within the URL which is also known as lexical features. Since it does not require browsing within the malicious link, detection of malicious Hyperlinks by lexical features within the URL has been demonstrated to be quick and low risk. However, most of these studies focus on the use of lexical features from the datasets which they acquire from known blacklisting sources like Phish Tank to train the models. This method can be beneficial for training models as malicious URLs are verified by the source. However, most of these URLs are extracted by these blacklisting sites very late in their attack cycle, hence resulting in machine learning models acquiring features only present in the older links and potentially missing on present ( and newly developed ) features of the newer links. To combat this models need to be capable of identifying malicious links real time or shortly after registration of the links.

If we were to compare the existing solutions [10] [14] [15] and research papers almost all the deep learning /ML systems have been developed on the concept of identifying and categorizing malicious pages by matching keywords related to malicious actions or using a categorization method to identify pages based on set of prior user feedbacks in order to warn the new users of those systems. Particularly in research (A) [14] focuses only on the classification aspect and the accuracy aspect of the data set keywords with the use of a Natural Language Processing (NLP) analysis method, “sentiment analysis” to classify into positive and negative categories given a text set to derive keywords. Paper [15] , research B has concerns set on the detection, feature rich classification and the increasing on detection speed/rate. Lastly, research C [10] focuses on accuracy and the classification of URLs only. As a conclusion there is not a solution which focuses on scalability, accuracy and efficiency along with the user friendliness by providing sorting features for users to select based on a particular event just by entering keywords related.

Table 1.1 - Comparison of proposed solution with former researches

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Paper** | **Scalability** | **Accuracy** | **Classification** | **High rate of detection/ Efficiency** | **Ability to sort through keywords / Events** |
| **Research A** | X |  |  | X | X |
| **Research B** |  | X |  |  | X |
| **Research C** | X |  |  | X | X |
| **Proposed Solution** |  |  |  |  |  |

The ensemble model which has been proposed, is trying to address the common issues of not being able to efficiently filter sets of URLs real-time and with added capacity for researchers and organizational clients to proactively filter malicious URLs based on an event.

## **Research problem**

Digital transformation of the world has paved the way for criminals to launch cyber-attacks. World Wide Web has become a major platform for these attacks to be spread out globally. Due to the recent Covid-19 pandemic situation, almost all the traditional organizations had to shift to online mediums for various reasons. This situation became an opportunity for attackers as most of the shifting personnel lack the knowledge about mitigating phishing attacks as well as cyber-attacks in general. [2] Results of this situation was the exponential growth of phishing attack victims and spawn of phishing URLs in the year of 2020. [3]

Cyber criminals always come up with ways to lure the victims by pretending to be a legitimate source as much as possible. Several techniques like use of keywords embedding to the malicious URLs like for an example covid, WHO, vaccine which of all the words an user will be in search of during this pandemic and use of URL obfuscation techniques to mask the malicious URLs to evade the malicious URL detectors using lexical and static feature analysis is getting popular recently.

These reasons have raised the need of a system where identifying and classification of potential malicious URLs related to each events as soon as possible to encounter the problem of being a victim of such cyber-attacks. In order to solve this problems , use of Machine Learning [ML] approaches are used but the models which are currently being used do not help in a situation like a massive emerge of URLs in a short span of time due to an event because most models do not incorporate scalability. This prevents the system to scale with respect to the processing power given to the system. This results in delayed identification of malicious URLs. [16] Another issue with event related emerging of malicious URLs is that the users do not have the option to sort the results through events to evade corresponding in real time using a simplified system. [10]

## **Research Objectives**

### 1.4.1 Main objectives

The main objective of the system is to make use of trends / events to aid in identifying potential malicious websites and pages. Whenever an entity enters a keyword using our proposed system user interface, they will be presented with pre made lists/templates of event related phishing scenarios. User can either use the pre-defined scenarios or enter keywords in order to begin finding related malicious URLs. The system matches the entered keywords with URL lists tokens/ generated keywords and present those data as a list. Using these data either they can blacklist the malicious URLs or use the populated keyword data to research more on a specific event. These mentioned objective can be achieved easily through machine learning models, However as mentioned in the introduction one of the key problems with these systems is the efficiency or the lack of scalability to the keyword populate module. Therefore, we try to address the issues by having the main objective of increasing the efficiency, scalability and the accuracy of such solutions by introducing a new system.

The methodology of detecting and populating keywords for events real-time is explained further below in the document.

### 1.4.2 Specific objectives

In order for reaching the main objectives described, the following specific objectives needs to be attained,

* + 1. Identify an efficient strategy to populate keywords from the user-input event keyword.

Whenever the user input a keyword using the web interface of the system, the keywords needs to be populated with related keywords to aid in matching with different variations of the keywords to help with efficiently matching malicious URLs with a related event.

* + 1. Build a model to find related keywords for an event by aggregating with online sources.

In order to find keywords related to the user input words, we would need to find datasets to capture sufficient key phrases/words to train the model in order to derive keywords that can be mapped to user inputs, this process can be achieved by collecting related phrases and data from real time tweets. After that, the prediction phase will utilize the populated keywords for analysis of URL data.

* + 1. Co-relate the populated keyword tokens with the phishing URLs database

The system need to map the keyword populated from the user input with the keywords / token data from the URLs in order for the identification of the malicious and benign URLs.

# 2 Methodology

The proposed Event-Driven Phishing URL Extractor has the capability of,

* Generating keywords related to the user input keywords.
* Generating keywords/ tokens from URLs.
* Identifying and classification of matching URLs with keywords for specific events.
* Ability to export the malicious URLs related to an event as a .csv file.

## **2.1** **Methodology**

### 2.1.1 System architecture

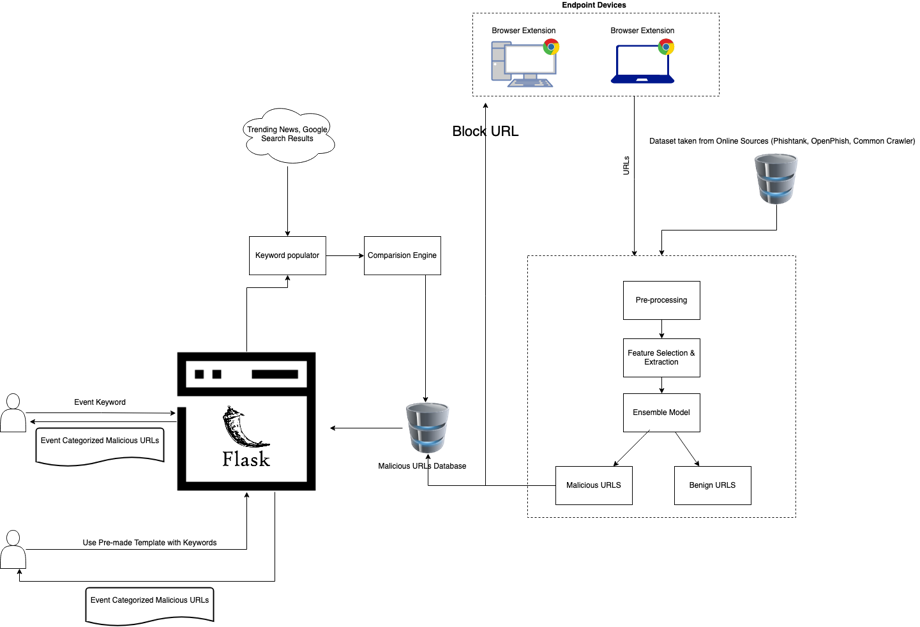


Figure 2.1 – Complete System Architecture

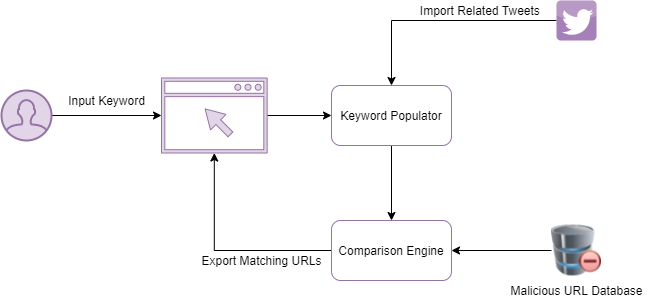


Figure 2.2 – Overall System Architecture

As displayed in the figure 2.1, system consists of three main models;

1. Ensemble model, which consists of a pre processer, feature extractor and the classifier to label the URLs acquired from datasets online and through our extension.
2. Keyword populator model which is connected to the web interface endpoint of the system to take the user inputs and the comparison engine to forward the populated keywords acquired from tweets from the Twitter.
3. Comparison engine which is connected to the malicious URL database, will extract the malicious URLs as a list and capable of generating tokens from the URL to match with the keywords forwarded from the keyword populator model.

Apart from that, there are two endpoints which consists of a device/browser extension, web interface a single database to store the classified malicious URLs.

As depicted in the figure 2.2 my objective is to create the keyword populator and the comparison engine which is a part of the core system which is showcased in the figure 2.1.

### 2.1.2 Keyword populator

Keyword populator as illustrated in the figure 2.3, will be responsible for taking the user input as an event related keyword / search query and extract related keywords from online sources, in this case we came to the conclusion to extract the keywords from real time tweets on Twitter website and generating a list of keywords specific to a particular event in order to be used by the comparison engine for finding potential matches in the malicious URLs database as the data we extract needs to fulfill the objectives of being real time and relevant to the search query.

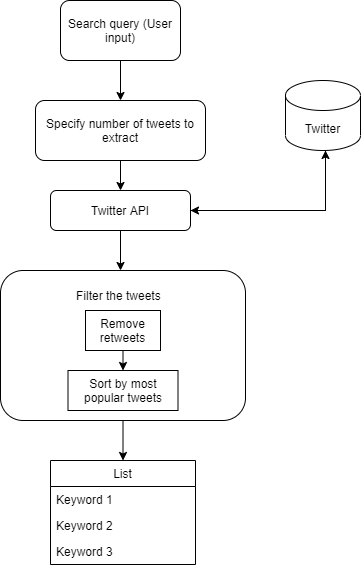


Figure 2.3 – Keyword Populator Architecture

#### **2.1.2.1 Taking event related user input**

In order to achieve the objective of populating user input keywords, whenever a user inputs a keyword related to an event via the system’s web interface which was developed using the python flask framework, those captured keywords will be taken as a query and incorporated by the keyword populator model to find related real-time tweets. To achieve this objective, the model will be leveraging the Twitter developer API to extract tweets.

#### **2.1.2.2 Extracting tweets through Twitter API**

Once the query is taken as an input, using the Tweepy cursor function [] tweets are extracted as extended tweets with likes and the time of the tweet posting. These extracted tweets are then loaded in to a Pandas data frame for cleaning and filtering, and finally extracting keywords from the tweets.

|  |
| --- |
| Algorithm – Tweepy.Cursor to extract tweets from the Twitter |
| number\_of\_tweets = 50  tweets = []  likes = []  time = []  #q = search query  #items=tweeets  #items(how many tweets to pull)  FOR i in tweepy.Cursor(api.search, q ,tweet extraction mode="extended", language =English).items(number\_of\_tweets):  tweets.append(i.full\_text extracted)  likes.append(i.favorite\_count)  time.append(i. tweet\_creation\_time) |

#### **2.1.2.3 Filtering and cleaning tweets**

As the first step of filtering, due to the twitter API collecting retweets as well, retweets needs to be filtered. We use the string matching capability of data frames to filter tweets containing the string “RT”. Afterwards, filtered tweets are sorted by the most popular tweets by the likes each tweet has gotten.

|  |
| --- |
| Algorithm – Removing Retweets to clean the dataset |
| dataframe = dataframe[ REMOVE dataframe.tweets.string which contains ("RT")] |

As the next step tweets are split into words for more cleaning and the words will be appended to a list.

|  |
| --- |
| Algorithm – Splitting tweets into words and append the words to a list called ‘lines’ |
| list\_of\_sentences = [sentence for sentence in dataframe.tweets]  lines = []  FOR sentence in list\_of\_sentences:  words = results from Splitting the sentence ()  FOR w in words:  lines.append(w)  OUTPUT (lines) |

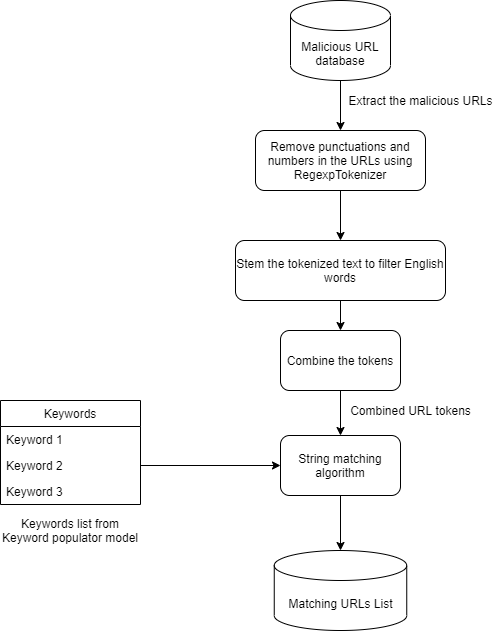
Using the regular expression (RE) library, on the created word list, emoji, punctuations and numbers are filtered out to clean the dataset before processing.

|  |
| --- |
| Function – Filtering punctuations and numbers in the words list (‘lines’) |
| lines = [using regular expression. Subtract function to filter words not containing (A to Z characters PLUS ','', x) FOR x in lines]  lines2 = []  FOR words in lines:  IF words NOT EQUAL to ' ':  lines2.append(words) |

Finally, these filtered words are taken as related keywords for the user input keyword.

### 2.1.3 Comparison Engine

In comparison engine, results from the keyword populator will be compared with processed URL tokens derived from the malicious URL tokenizer model to match the keywords with the events. In accordance with those matching keywords, a sample set of URLs containing the event related tokens will be collected from malicious URL database of the core system and user will be given the options to extract the list to a .csv file. These identified URLs will be categorized with respective keywords and stored in a database for future use/processing as well to cut down the time it takes for the process.



*Figure 2.4 – Malicious URLs Finder Architecture*

#### **2.1.3.1 Malicious URL tokenizer**

We need to standardize to extract the tokens after the initial step of identifying related URLs matching user inputted keywords. It'll also enable the ML model to process these inbound URLs efficiently. As a result, the URLs captured will be delivered from the malicious URL database for processing through the URL tokenizer model.

Before the tokenization process, malicious URLs will be loaded into a Pandas data frame. For tokenization, RegexpTokenizer will be utilized with the following parameters to remove the numbers and punctuations as we also did with the tweets in the keyword populator model.

|  |
| --- |
| Algorithm – Tokenization of the URLs |
| DEFINING tokenizer = Regular expression Tokenizer (Remove characters not containing A to Z characters PLUS')  url\_dataframe named 'tokenizedtext' = url\_dataframe.URL.map(lambda t: tokenizer.tokenize(t)) |

Despite the fact that previous research included special characters such as percent (%), brackets ({}) or hash (#) etc. as part of their analysis, we have decided not to incorporate those features to reduce the time taken for the matching of strings between keywords and the stem tokens of URLs.

#### **2.1.3.2 Stemming the tokens**

As the next step the tokenized text will be stemmed to filter only the tokens containing English words. For this we will be using the SnowballStemmer’s functionality to stem each of the URL from the list.

|  |
| --- |
| Algorithm – Stemming the tokens to filter English words |
| stemmer = SnowballStemmer("english")  url\_dataframe named 'stemmedtokens' = url\_dataframe ['tokenizedtext'].map(lambda l: [stemmer.stem(word) for word in l]) |

However, as these tokens are created the URL will be separated into words. It is not efficient to use those stemmed tokens directly on the comparison stage. Therefore the stemmed tokens are combined together to make the tokens combined together as a sentence. Finally these combined tokens are exported and will be forwarded to the comparison engine for the matching process of URLs.

|  |
| --- |
| Algorithm – Combine the tokens and export the results to a .csv file |
| url\_dataframe['combinedtokens'] = url\_df['stemmedtokens'].map(lambda l:' '.join(l))  url\_dataframe.to\_csv file ('PATH:/STEMMEDURLTOKENS.CSV', export columns ['combinedtokens'], separator used is ‘,’ , Do not print the index) |

#### **2.1.3.3 Comparing the keywords with URL tokens**

To compare the keywords with tokens, levenshtein distance measuring python library, FuzzyWuzzy is used. For this the exported combined tokens will be loaded and matched through the following function.

|  |
| --- |
| Function – Comparing strings (Keywords, Combined tokens) |
| DEFINING matching\_term(term, list\_names, minimum\_score = 0):  max\_score = -1  #Score of -1 will be given if there were no potential matches found above the score the system have specified ( Score = 70 )  max\_name = ""    for term2 in list\_names:  score = Using function fuzz.token\_set\_ratio on term,term2    IF score is larger than min\_score AND score is larger than maximum\_score:  max\_name = term2  maximum\_score = score  RETURN max\_name and maximum\_score  FOR i in lines:  OUTPUT i AND matching\_term from the url\_list |

This function will loop through each keyword to find the most accurate match from the URLs list. We have used the token\_set\_ratio as the scorer method and used a score of minimum 70 percent match ratio to compute the matching terms.

Finally the user will be able to export the matched URLs with the keywords and matching score percentage to a .csv file after the matching process is completed.

## **2.2 Commercialization**

Targeted user base is small-medium enterprises (SME), due to the lower cost and higher usability.

Planned to be implemented in two versions.

* A free version that SMEs and researchers can use to collect URL lists based on events with limited export capability.
* A paid version that will provide seamless export capability plus the basic endpoint protector using the browser plugin.

*Table 2.1 - Commercialization Versions*

|  |  |
| --- | --- |
| **Free Version** | * Rate Limit on event-based malicious URL list and restricted export capabilities. |
| **Paid Version** | * Browser plugin to protect users from malicious pages with basic reporting to the administrator * No limit on exporting event-based malicious URLs |

## **2.3 Testing and Implementation**

### 2.3.1 Data collection

This section describes the collection of malicious URLs and our proposed experimental setup. To achieve my specific objectives of populating keywords and identifying matching malicious URLs, collection of benign URLs were not a requirement. Therefore the data was to be obtained from the malicious URL database created in our core system by the classifier in the ensemble model. However for testing of the malicious URL tokenizer and the comparison engine, domain list was obtained from the Kaggle [17] website containing 549,346 URLs.

The dataset was labeled by the nature of the URLs into good and bad, indicated by a separate column. For testing of our models we removed the label column from the dataset to match with our malicious URL database list format. After experimental analysis of the dataset through the classifier, it was observed that approximately 395,530 URLs were benign and 153,815 were malicious links as depicted in the following table 2.2.

Table 2.2 – Experiment of labeling through the classifier

|  |  |
| --- | --- |
| Dataset - Kaggle URLs | |
| Label | Percentage |
| Benign | 72 % |
| Malicious | 28 % |

### 2.3.2 Testing setup

This section describes the testing of pre collected malicious URLs and our proposed experimental setup specifications. Testing of keyword populator and comparison model of the system was performed initially on our local computer using Jupyter Notebooks prior to integration.

Following were specifications of the system used to conclude the results in the 3.1 section.

Device Specifications

Processor Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz

Installed RAM 8.00 GB (7.85 GB usable)

Device ID 8CB80B88-53DB-49D0-AE88-DAB2803677

System type 64-bit operating system, x64-based processor

Pen and touch No pen or touch input is available for this display

Storage Disk 1 (C:) - NVMe TS256GMTE110S

Capacity: 239 GB

Average response time 1.0 ms

OS specifications

Edition Windows 11 Pro

Version 21H2

Installed on ‎21/‎08/‎2021

OS build 22000.194

Experience Windows Feature Experience Pack 1000.22000.194.0

### 2.3.3 Budget and budget justification

In this section, the budgetary requirements for the deployment of the system and testing environment is described.

Table 2.3- Budget Chart

|  |  |  |
| --- | --- | --- |
| **Task** | **Cost($)** | **Cost (LKR.)** |
| Azure Machine Learning Studio (Implementation and Cloud Deployment) | 100 (Monthly Fee) | **19,900** |
| Web Application Hosting | 72 | **14300** |
| Database for Storing URLs – Mongo DB | 57 | **11400** |
| Plugin registration in Chrome Web store | 5 (One-time fee) | **995** |
| **Total Cost** | | **46,595** |

### 2.3.4 Implementation

This section of the paper will discuss how we implemented the methodologies discussed in 2.1 - methodology section. As depicted from the figure 2.1, our overall model has three main sub models, namely; comparison engine, keyword populator and the ensemble model with classifier to be implemented together.

Section 2.3.4.1 provides the technical information about the system, including the system design decisions taken and Section 2.3.4.2 outlines the structure of the system and the implemented testing environment.

#### **2.3.4.1 Technical information**

The system was developed and coded using the Jupyter Notebooks and using the Azure machine learning cloud suite. The web interface was developed using the Python Flask framework. Chrome plugin was developed using JavaScript. Rest of the sub modules were coded using Python language and the Mongo DB v.4.4 was used for database management purpose.

**Design decision – Website over a software**

We decided to include two endpoints for the customers of our solution can interact with, a web interface (website) and the chrome web plugin. To come to the conclusion of to host the proposed system as a website, following reasons were taken into account.

1. Ease of usage (without installment requirements)
2. Reduce the number of bugs, later in the system.
3. Survey results from the potential users (*Refer Appendix A*)

**2.3.4.2 Implementation structure**

To test the implementation and the deployment of the overall system we decided on using the Azure machine learning studio tools. The implementation structure is represented in figure 2.1.

# 3 Results and Discussion

## **3.1 Results**

This section describes the results of the comparison through using dataset we prepared after omitting the label column. To evaluate the performance of the model, classification results for the algorithm applying distinctive distance scores are shown in Tables 3.1. For the testing 100 keywords were matched against 549,346 URLs.

*Table 3.1 – Comparison of similarity scores in comparison engine*

|  |  |  |
| --- | --- | --- |
| **Similarity Score** | **Time to completion** | |
| 100 keywords | 1 keyword |
| 100 % | 62 mins | 0.62 mins |
| 75 % | 57 mins | 0.57 mins |
| 50 % | 116 mins | 1.16 mins |
| 25 % | 60 mins | 0.6 mins |

As we can see from the results depicted on table 3.1, setting up a similarity score of higher score will result in generating less matching URLs with a minimum time to completion than the lower scores.

However, it is also noted, as the similarity score is set to a lower percentage, outputted results will be less accurate with the inputted search query/keyword. These results are showcased in the table 3.2 below.

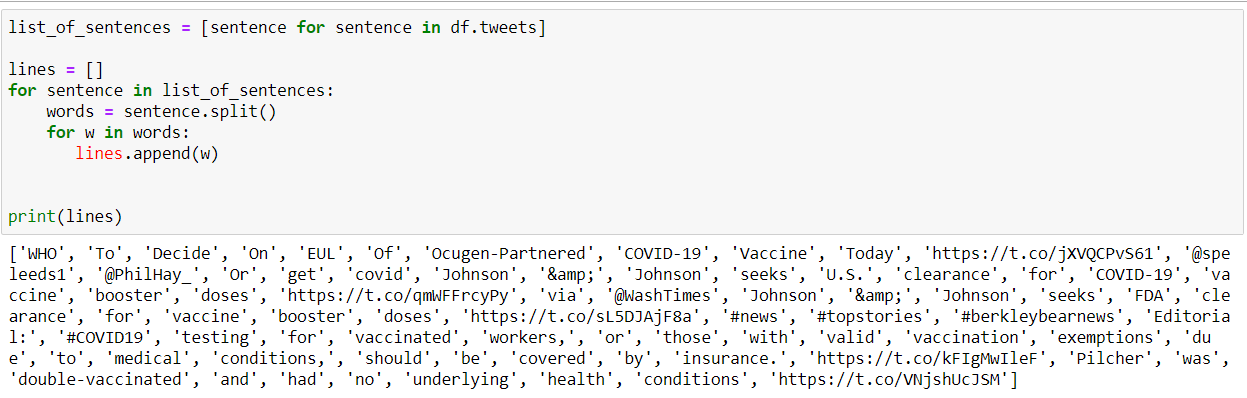
Table 3.2 – Accuracy of output URLs against the scores

|  |  |  |
| --- | --- | --- |
| **Similarity Score** | **Number of Output URLs** | **Accuracy of the output URL list** |
| 100 % | 3 | 98.1% |
| 75 % | 28 | 66.2 % |
| 50 % | 52 | 43 % |
| 25 % | 63 | 19.7 % |

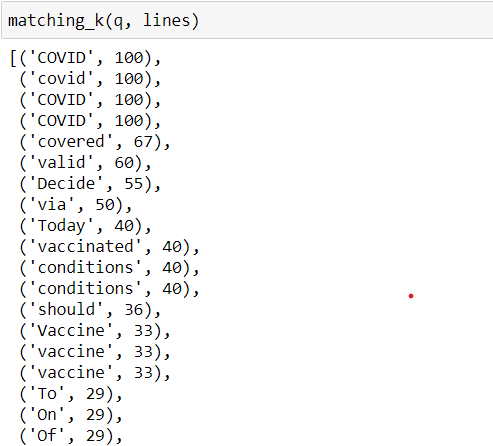
For testing the accuracy of the output URLs in the table 3.2, we have used an online text matching tool [18] to determine if the keywords are contained in the URLs outputted from the comparison model, compared side by side. Same datasets have been used as used for the test of Comparison of similarity scores in table 3.1.

**Test case 1 – using input keyword as ‘Covid’**

By extracting 50 tweets relating to the input keywords, keyword populator model was able to generate a list of populated keywords containing 55 keywords relating to keyword ‘Covid’.



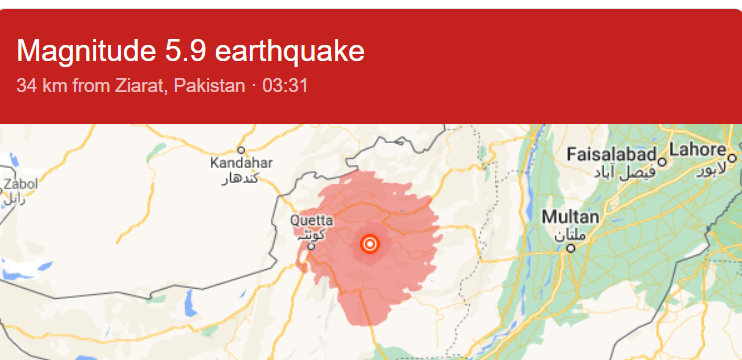
*Figure 3.1 – List of populated keywords prior to cleaning for test case 1*

****

*Figure 3.2 – Populated keyword accuracy against search term*

**Test case 2 – using input keyword as ‘Earthquake’**

At the time (07/10/2021 at 09:00 AM) of performing the test case 2, we have used ‘earthquake’ as the keyword to determine if the keyword populator will be able to capture keywords relating to the recent earthquake happened in Pakistan on the same day. As depicted on the figure 3.3, the model was able to populate the keyword ‘Earthquake’ up to 395 keywords.



*Figure 3.3 – Earthquake in Ziarat, Pakistan on 07/10/2021 at 03:31 AM*



*Figure 3.4 – List of populated keywords for test case 2*

## **3.2 Research findings**

Our research focused on the identification of malicious URLs and the extraction of results related to a specific events early in the attack cycle. During the research, a dataset of 549,346 URLs was tested, and an ensemble model was successfully established to address the issue of efficiency.

According to the testing results of the comparison model of the system with our processed dataset, it is identified that the model is capable of handling up to approximately 600,000 URLs under 1 hour (Calculated from taking the average time from the test case in table 3.1) using a single keyword with a system with specifications as described in 2.3.2 testing setup section.

Upon testing of the system, it was identified that the model’s success to match populated keywords against the URLs accuracy drops drastically if the similarity scores are set too low. However, if the user wishes to block malicious URLs as much as possible, this score has to be lowered in the system due to the model outputting more URLs as the score gets lower. Also, we were able to find from the results from table 3.1 in the 3.1 results section, that setting a lower score doesn’t necessarily impact the time for the operation to complete and it depends on the system itself which does the processing.

Therefore considering these facts, we have concluded to set the similarity score to 75 % in the comparison model in order to produce a balanced and efficient output.

## **3.3 Discussion**

From our conducted survey (Refer Appendix – A) at the proposal stage of the project, we were able to identify the need of a solution which can be easily and readily accessible, user friendly and time efficient enough for users to keep using as a malicious URL blocker / detector. Due to this concept, instead of using a complex software solution suite and an interface, we were able to design and implement the system using two endpoints; browser plugin and web page, which requires little to no effort on the users end with the capability of exporting the event driven URLs to list.

In order to extract the event driven URLs, the model needed to take an input as a search term / query and then extract the URLs. For this, we concluded on populating the keywords using online sources. As a requirement the sources should be able to deliver related key phrases / words in real time. Therefore we found that Twitter tweets can be concatenated to use as relating keywords for match and malicious URLs through our comparison model.

For comparison engine, we tested on few methodologies to acquire the resulting matches of keywords against malicious URLs and concluded on levenshtein distance to calculate the similarities between the strings as to attain the objective of efficiency.

## **3.4 Summary of each students contribution**

**IT18071412 – S.W. Jonathan**

Main responsibility was to build the Ensemble model by creating and testing multiple diverse modeling algorithms to predict malicious URLs with improved accuracy without resulting in decrease in performance of the system. The following algorithms were tested by the student to find the ideal algorithm to be used in the system, which was concluded as the Bidirectional LSTM + CNN.

* Recurrent Neural Networks (RNNs) + Convolutional Neural Networks (CNNs)
* Artificial Neural Network (ANN) + K-Nearest Neighbors (KNN) + Decision Tree (C4.5) + Random Forest Classifier (RFC)
* Long-Short Term Memory [LSTM] + Convolutional Neural Networks (CNNs)
* Monotone Multi-Layer Perceptron Neural Network, Multi-Layer Perceptron, Neural Networks

In order to test these algorithms, datasets were acquired as a sub objective from public open sources; PhishTank, OpenPhish, Common Crawler and Alexa.

Another sub objective includes building the website using the Python Flask Framework with following UI features,

* Provide data visualization to viewers
* Export event-based malicious URL data lists in .CSV Format

**IT18034400 – Renu Harshatha**

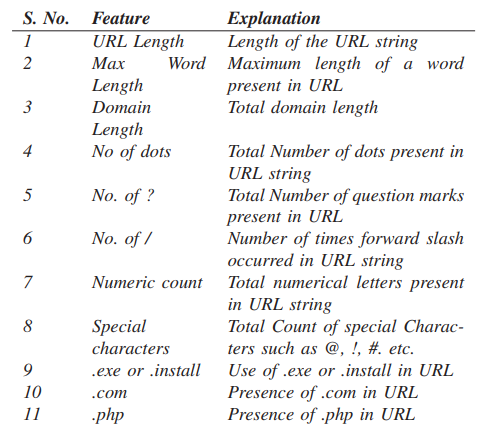
Student’s main objective was to install and integrate the Nomad with the developed malicious URL detection system to scale the overall system.

As a sub objective creation of the browser endpoint extension was carried out, with the objectives of,

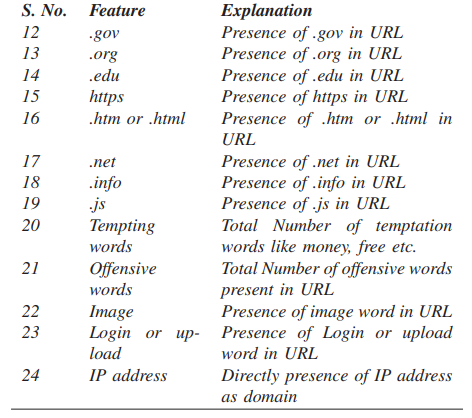
* Monitoring capability to track user behavior ( URLs visited)
* Capturing and storing the URLs in JSON format using JavaScript
* Automated blacklisting of detected malicious URLs upon visiting

**IT18021912 – Ramanayaka A.M.**

As the main objective, feature reduction of the collected URLs from the ensemble model was performed by the student. In order to achieve this, as the first step student identified and tested feature classification of 24 features (Refer figures 3.5 & 3.6) by consideration of previous studies carried out.



*Figure 3.5 – Classified URL features* [19]



*Figure 3.6* *– Classified URL features* [19]

As the next step feature extraction of URLs was performed. Using these extracted features and comparing them with the classified features, URLs are classified as malicious or benign.

**IT18032666 – Wishvajith B.L.D.V.**

As the main objective student’s main responsibility was to build a strategy to populate keywords from the user inputted keyword/query via the web interface of the system and compare those populated keywords with the identified malicious URLs from the ensemble model to find a potential match. In order to achieve this objective, two models were created with the following capabilities,

1. Keyword populator model

* Extracting real time tweets through Twitter API relating to the user input keyword
* Deriving keywords from the tweets
* Exporting of populated keywords as a .CSV file

1. Comparison model.

* Tokenization and stemming of the captured URLs
* Matching of the populated keywords with URL stems
* Export matching sets as a .CSV file

And tested with a sample dataset containing 549,346 URLs to test the accuracy of the models. Testing was also carried out to find the best strategy for the matching of the keywords with URLs by the student and concluded that the use of levenshtein distance to calculate the similarities was efficient.

# Conclusion

Although numerous prior studies have been conducted on identifying and preventing malicious URLs utilizing various algorithms and approaches, they fall short of capturing the features of malicious URLs early in the attack cycle, as well as overall efficiency. In order to overcome these issues, we proposed an ensemble model along with a browser extension in this paper to detect malicious domain names containing event-related keywords.

Our model was able to successfully detect and generate a list of malicious domain names with a 98.1 % average accuracy utilizing populated keywords from the user inputted event keyword. To accomplish this, we employed 549,346 URLs collected and processed from Kaggle to train and test our model.

In contrast to previous research on the subject, our approach appears to offer a feasible alternative for reducing computational time and overhead. This is due to its capacity to use the ensemble model to detect malicious URLs with high accuracy and map them with event-related keywords in real time, early in the attack lifecycle. Future studies will concentrate on enhancing the models' time efficiency by experimenting with other algorithms and technologies.

# Reference List

|  |  |
| --- | --- |
| [1] | ResearchGate, "35 YEARS OF INTERNET, 10 YEARS OF ISPRS ONLINE," 01 October 2021. [Online]. Available: https://www.researchgate.net/figure/An-Internet-address-or-URL-subdivided-in-its-main-parts\_fig3\_242285531. |
| [2] | G. -. G. Technology, "2020: The Year the COVID-19 Crisis Brought a Cyber Pandemic," GT - Government Technology, [Online]. Available: https://www.govtech.com/blogs/lohrmann-on-cybersecurity/2020-the-year-the-covid-19-crisis-brought-a-cyber-pandemic.html. [Accessed 21 02 2021]. |
| [3] | D. Warburton, "2020 Phishing and Fraud Report," F5 Labs, [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/2020-phishing-and-fraud-report. [Accessed 21 02 2021]. |
| [4] | "Free Automated Malware Analysis Service," [Online]. Available: https://www.hybrid-analysis.com/. [Accessed 10 2021]. |
| [5] | STAMFORD, CT, "Gartner CFO Survey Reveals A Dramatic Digital Acceleration Since COVID-19," Gartner, 11 2020. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2020-11-12-gartner-cfo-survey-reveals-a-dramatic-digital-acceleration-since-covid19. [Accessed 10 2021]. |
| [6] | Verma, Rakesh and Crane, Devin and Gnawali, Omprakash, "Phishing During and After Disaster: Hurricane Harvey," in *2018 Resilience Week*, 2018. |
| [7] | Grange, Elisha and Neil, Eric and Stoffel, Michelle and Singh, Angad and Tseng, Ethan and Resco-Summers, Kelly and Fellner, B. and Lynch, John and Mathias, Patrick and Mauritz-Miller, Kristal and Sutton, Paul and Leu, Michael, "Responding to COVID-19: The UW Medicine Information Technology Services Experience.," *Applied Clinical Informatics,* vol. 11, no. 10.1055/s-0040-1709715, pp. 265-275, 2020. |
| [8] | Garera S., Provos N., Chew M., Rubin A., "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on Recurring malcode - WORM '07*, 2007. |
| [9] | Le, Anh and Markopoulou, Athina and Faloutsos, Michalis, "Phishdef: URL names say it all," *Proceedings - IEEE INFOCOM,* 2010. |
| [10] | Mohammad Saiful, Islam Mamun, Mohammad Ahmad Rathore, Arash Habibi Lashkari, Natalia Stakhanova, and Ali A. Ghorbani, "Detecting Malicious URLs Using Lexical," in *University of New Brunswick, Fredericton, NB, Canada*, Fredericton, 2016. |
| [11] | Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song., "Design and evaluation of a real-time url spam filtering service," in *IEEE Symposium on. IEEE*, 2011. |
| [12] | DOYEN SAHOO, CHENGHAO LIU, STEVEN C.H. HOI, "Malicious URL Detection using Machine Learning: A Survey," 2019. |
| [13] | Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, Banu Diri,, "Machine learning based phishing detection from URLs," *Expert Systems with Applications,* vol. 117, no. ISSN 0957-4174, pp. 345 - 357, 2019. |
| [14] | Vishal A. Kharde, S.S. Sonawane, "Sentiment Analysis of Twitter Data: A Survey of Techniques," *International Journal of Computer Applications (0975 – 8887),* vol. Volume 139, no. 11, p. 15, 2016. |
| [15] | Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, Banu Diri, "Machine learning based phishing detection from URLs," *Expert Systems With Applications,* 2019. |
| [16] | Vahid Shahrivari, Mohammad Mahdi Darabi, Mohammad Izadi, "Phishing Detection Using Machine Learning Techniques," in *arXiv:2009.11116v1 [cs.CR] 20 Sep 2020*, 2009. |
| [17] | T. Tiwari, "Phishing Site URLs," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/taruntiwarihp/phishing-site-urls. |
| [18] | "String similarity test," [Online]. Available: https://www.tools4noobs.com/online\_tools/string\_similarity/. [Accessed 10 2021]. |
| [19] | Kumar, Yogendra and Subba, Basant, "A lightweight machine learning based security framework for detecting phishing attacks," in *COMSNETS51098.2021.9352828*, 2021. |
| [20] | Islam, Jamil Ispahany & Rafiqul, "Detecting Malicious Urls of COVID-19 Pandemic Using ML Techniques," Charles Sturt University, NSW, Australia. |
| [21] | Patil, D. R., & Patil, J. B., "URLs detection using decision tree classifiers and majority voting technique," *Cybernetics and Information Technologies,* vol. 18, no. 1, pp. 11-29, 2018. |

# Appendix – A

