**EVENT-DRIVEN MALICIOUS URL EXTRACTOR**

Sajeth Jonathan S.W.   
(IT18071412)

Renuharshatha A.   
(IT18034400)

Ramanayaka A.M.  
(IT18021912)

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**

Sajeth Jonathan S.W.   
(IT18071412)

Renuharshatha A.   
(IT18034400)

Ramanayaka A.M.  
(IT18021912)

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

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

|  |  |  |  |
| --- | --- | --- | --- |
| Student ID | Student Name | Date | Signature |
| IT18071412 |  | 13/10/2021 |  |
| IT18032666 | Wishvajith B. L. D. V | 13/10/2021 |  |
| IT18021912 | Ramanayaka A. M | 13/10/2021 |  |
| IT18034400 |  | 13/10/2021 |  |

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

Signature of the supervisor: Date: 13/10/2021

# Abstract

Cyber-attacks are attacks that are often carried out to obtain sensitive information or to harm internet-based services. Recent events, both internationally and locally, have demonstrated that such attacks are rapidly spreading owing to the deployment of malicious URLs (Uniform Resource Locators). Traditional countermeasures, such as blacklisting malicious URLs, make it difficult to respond to such attacks quickly and effectively. In circumstances where freshly constructed URLs are linked to a recent event, such as Covid-19-related frauds, most existing solutions are limited in terms of providing scalable and proactive user protection. The suggested approach aims to alleviate traditional system constraints by providing an interface for users to protect themselves by detecting phishing/malicious URLs in real time. In this study, we'll look at collecting event-related keywords from user input and using NLP (Natural Language Processing) algorithms to match them with URL (Uniform Resource Locator) token data to see if the URLs are malicious or benign. The report also recommends utilizing a browser extension to automate the process of blocking malicious URLs on endpoint devices, making it a far more user-friendly approach.

Keywords – Machine Learning, Malicious URL detection, Malicious links

**Table of Contents**

[Declaration 3](#_Toc85058396)

[Abstract 4](#_Toc85058397)

[1. Introduction 6](#_Toc85058398)

[1.1 Background Literature 6](#_Toc85058399)

[1.2 Research Gap 6](#_Toc85058400)

[1.3 Research Problem 6](#_Toc85058401)

[1.4 Research Objectives 6](#_Toc85058402)

[2. Methodology 7](#_Toc85058403)

[2.1 System architecture 7](#_Toc85058404)

[2.2. Methodology 7](#_Toc85058405)

[2.2. . Keyword populator 7](#_Toc85058406)

[2.2. . Comparison engine 10](#_Toc85058407)

[2.2 Feature Reduction 13](#_Toc85058408)

[2.2.1 Feature Classification 16](#_Toc85058409)

[2.3 Commercialization 16](#_Toc85058410)

[2.4 Testing and Implementation 18](#_Toc85058411)

[3. Results and Discussion 19](#_Toc85058412)

[3.1. Results 19](#_Toc85058413)

[3.2. Research findings 19](#_Toc85058414)

[3.3. Discussion 19](#_Toc85058415)

[Conclusion 19](#_Toc85058416)

[References 19](#_Toc85058417)

# List of Figures

[Figure 2.1 – Overall System Architecture 6](#_Toc85110156)

[Figure 2.2 - Keyword Populator Workflow 7](#_Toc85110157)

[Figure 2.3 – Comparison Engine Workflow 11](#_Toc85110158)

# List of Tables

[Table 1.1 - Comparison of proposed solution with former researches 3](#_Toc85110256)

[Table 2.1 - Commercialization Versions 17](#_Toc85110257)

# 1. Introduction

## **1.1 Background Literature**

## **1.2 Research Gap**

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** |  |  |  |  |  |

## **1.3 Research Problem**

## **1.4 Research Objectives**

The main objective of this research is to build an event-based system which can detect malicious URLs early in the attack lifecycle. In doing so, we want to increase accuracy of the Malicious URL Classifier by using ensemble modeling and achieve a higher performance using feature reduction. Scalability of the model and system will be built from the core to handle various volumes of URLs.

### **1.4.1. Specific objectives**

#### IT18032666 - Wishvajith B.L.D.V.

1. Identify an efficient strategy to populate keywords from the user-input event keyword.
2. Build a model to find related keywords for an event by aggregating with online sources.
3. Co-relate the populated keyword tokens with the phishing URLs database

# 2. Methodology

The proposed Event-Driven Malicious 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 identified malicious URLs related to an event as a .csv file.
* Ability to prevent users from visiting identified malicious URLs using the browser extension.

## **2.1 System architecture**

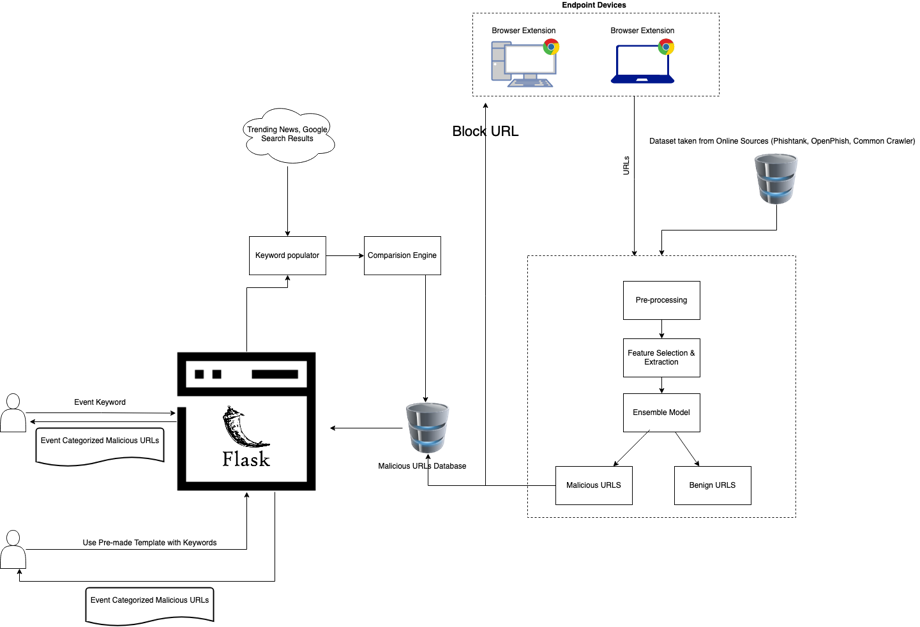


Figure 2.1 – Overall System Architecture

## **2.2. Methodology**

### **2.2.1. Keyword populator**

The keyword populator, as shown in figure 2.1, will be in charge of taking the user's input as an event-related keyword / search query and extracting related keywords from online sources. In this case, we made the decision to extract keywords from real-time tweets on the Twitter website and generate a list of keywords specific to a particular event to be used by the comparison engine for finding potential matches in the malicious URLs database.

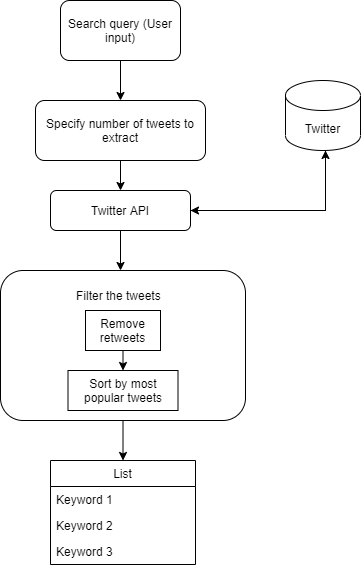


Figure 2.2 - Keyword Populator Workflow

#### **2.2.1.1. Taking event related user input keywords**

To accomplish the objective of populating user input keywords, whenever a user enters a keyword linked to an event through system's web interface, which was developed using the Python Flask framework, the captured keywords are being used as a query by the keyword populator model to discover related real-time tweets. To achieve this objective, the model will be using the Twitter developer API to extract tweets.

#### **2.2.1.2. Extracting tweets through Twitter API**

Once the query has been entered, tweets are extracted as extended tweets with likes and the time of posting using the Tweepy cursor function. Following that, the retrieved tweets are imported into a Pandas data frame for cleaning and filtering, followed by keyword extraction 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) |

#### **2.2.1.3. Filtering and cleaning tweets**

Since the Twitter API also collects retweets, retweets must be filtered as the first step in the filtering process. To filter tweets containing the string "RT," we employ data frames' string matching functionality. The filtered tweets are then sorted by the most popular tweets and the number of likes each has received.

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

As the next step, tweets were split into words for more cleaning and the words are to be appended into 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) |

Emoji, punctuations, and digits are filtered out of the generated word list using the regular expression (RE) library 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) |

These filtered words are then used as related keywords for the user input keyword.

### **2.2.2. Comparison engine**

To match the keywords with the events, the results from the keyword populator will be compared with processed URL tokens generated from the malicious URL tokenizer model in the comparison engine. A sample set of URLs containing event related tokens will be retrieved from the core system's malicious URL database in accordance with those matching keywords, and the user will be offered the option to export the list to a.csv file. These URLs will be categorized with relevant keywords and recorded in a database for future use/processing, as well as to minimize the process time.

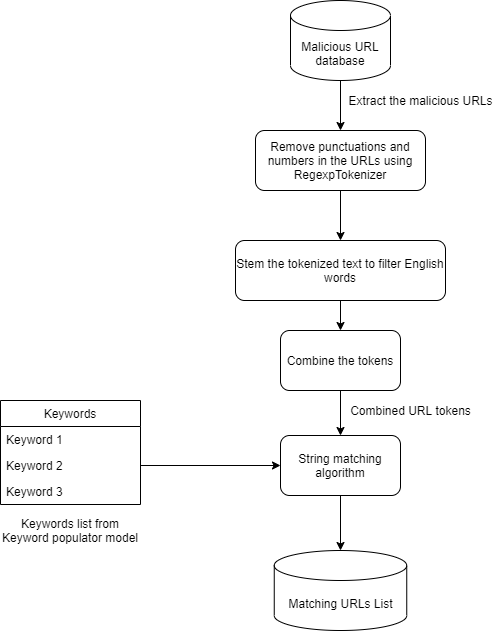


Figure 2.3 – Comparison Engine Workflow

#### **2.2.2.1. Malicious URL tokenizer**

After the initial phase of detecting related URLs that match the user's keywords, we need to standardize the extraction of tokens. It will also enable for the ML model to process these inbound URLs rapidly. As a result, the captured URLs will be transferred from the malicious URL database to the URL tokenizer model for processing. Malicious URLs will be imported into a Pandas data frame prior to getting tokenized. To remove the digits and punctuations, we'll use RegexpTokenizer with the following parameters, just as we performed 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 studies incorporated special characters such as percent (percent), brackets (), or hash (#) in their analyses, we opted not to include those features in order to reduce the time taken to match strings between keywords and URL stem tokens.

#### **2.2.2.2. Stemming the URL tokens**

The tokenized text will then be stemmed, enabling only tokens containing English words to be selected. We'll be utilizing the SnowballStemmer's functionality to stem each of the URLs in the list for this.

|  |
| --- |
| 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]) |

The URL will be segmented into words once these tokens are created. Using those stemmed tokens directly on the comparison stage is inefficient. As a function, the stemmed tokens are grouped together to form a sentence. Finally, these concatenated tokens are exported and sent to a comparison engine for URL matching.

|  |
| --- |
| 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) |

#### **2.2.2.3. Comparing the keywords with URL tokens**

The FuzzyWuzzy levenshtein distance measuring python package is used to compare keywords with tokens. The following function would be used to load and match the exported combined tokens.

|  |
| --- |
| 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 in the URLs list to find the most accurate match. To compute the matching terms, we utilized the token set ratio as the scorer technique and a score of at minimum 70% match ratio. After the matching process is completed, the user will be able to export the matched URLs, along with the keywords and matching score percentage, to a.csv file.

### **2.2.3.** **Feature Reduction**

It's tough to maintain a system effective and up to speed when there are a lot of features to evaluate. It will also lower the system's efficiency and increase the usage of computing resources, both of which are problematic. We need to figure out how to discover patterns in a dataset with a lot of data and a lot of features so that we can minimize the number of features and make the system run faster. There are two forms of feature reduction: dimensionality reduction and feature reduction. Feature reduction retains the most important characteristics, whereas dimensionality reduction looks for a limited collection of features that are a mixture of the fundamental features found. [1]

Feature reduction is one of the main focus areas that is followed by the other areas related to achieving the feature reduction effectively, such as the feature classification, feature extraction and feature analysis which will finally aid in achieving feature reduction much effectively and this will benefit the model, that is built, by providing the speed and efficiency which is important for better and faster identification of the malicious URLs which is the main goal of the system that would be implemented. It is a challenging area due to the number of resources to get information on how to perform feature reduction being low.

Feature classification would be accomplished by comparing and identifying a better list of features that would be much more effective and efficient in comparison to many previous studies conducted by other researchers, and it would be checked for the features that are most effective by comparing and identifying a better list of features that would be much more effective and efficient. Following the feature classification procedure, we can move on to feature extraction, which will be done using Python code to extract the features from the URLs. There will be two datasets utilized to successfully complete this procedure as a test case and to check how efficiently the coding will execute. UNB dataset which contains the legitimate URLs and other malicious URLs, and the dataset from PhishTank which contains the phishing URLs will be used to extract the previously created list of features to understand how the coding would work with the datasets and then it will be used for the feature analysis later on which will further let us understand how effective the feature classification has been and further providing us the confirmation of how the feature analysis would work after the feature extraction. Finally feature reduction will be carried out after both the datasets has been checked for features and after a process of comparison and then we are able to obtain the dataset that would be created after feature reduction.

Feature reduction requires a significant amount of effort, and due to the huge volume and fast speed of the data we get, which is URL data, performing feature classification and feature extraction, which leads to improved feature analysis and feature reduction, has become a significant problem. When training a model for feature categorization, extraction, analysis, and reduction, having high dimensional URL features has become a significant problem. Deep learning approaches are believed to have been created in order to generate improved feature categorization and the feature representation solutions. [2]

URL Feature Classification will be done to understand how many features were found which will first help distinguish between the malicious URLs and the legitimate URLs and then those features will be narrowed down to see which features were thought to be the most common in malicious URLs and to see how effectively we can capture the malicious URLs. URL Feature extraction will be one of the sub objectives of feature reduction that will help us extract those specific features in the URLs so that we can extract the URLs with those specific features related to malicious URLs which then can be used to train the model for better detection of malicious URLs. URL Feature Analysis will be another objective which will be carried out after the feature extraction is done which will help to drop out the URLs which does not have the specific features or which seem to have NAN values. URL Feature Reduction can then be done to improve the efficiency of the system which will reduce the features that were extracted and analyzed by using the datasets which contains both legitimate and malicious URLs.

## **2.3 Commercialization**

Due to the self-evident lower cost and higher usability, small-medium companies (SMEs) is the target user group.

Two versions were planned to be implemented.

• A free version with limited export capability that SMEs and researchers can use to collect URL lists based on events.

• A commercial version that includes a seamless export option as well as a basic endpoint protection via a browser plugin.

*Table 2.1 - Commercialization Versions*

|  |  |
| --- | --- |
| **Free Version** | * Restricted export capabilities and a rate limit on the malicious URL list based on events. |
| **Commercialized Version** | * A browser plugin which safeguards users from malicious websites while also providing basic reporting to the administrator. * No restrictions on exporting malicious URLs based on events. |

## **2.4 Testing and Implementation**

# 3. Results and Discussion

## **3.1. Results**

## **3.2. Research findings**

## **3.3. Discussion**

# Conclusion

Despite the fact that many previous research have been undertaken on recognizing and preventing malicious URLs utilizing various algorithms and techniques, they fall short of capturing the features of malicious URLs early in the attack cycle, as well as overall efficiency. In this paper, we presented an ensemble model as well as a browser add-on to detect fraudulent domain names having event-related keywords in efforts to answer these concerns.

Unlike previous published studies, our approach appears to have become a viable option for decreasing computational time and overhead. This is dependent on its ability to detect malicious URLs with high accuracy using the ensemble model and map them with event-related keywords in real time, early in the attack lifecycle. Future research will emphasize on experimenting with other algorithms and techniques to enhance the models' time efficiency.

# References

|  |  |  |
| --- | --- | --- |
| [1] |  | P. Sharma, "The Ultimate Guide to 12 Dimensionality Reduction Techniques (with Python codes)," Analytics Vidhya, 27 August 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/. |
| [2] |  | C. L. S. C. H. H. Doyen Sahoo, "Malicious URL Detection using Machine Learning: A Survey," 2019. |