

## **Mal-advare detection for URL security**

MSc Research Project

MSc in Cybersecurity(MSCCYB)

**Irfan Nayeem**

Student ID: 20252391

School of Computing  
National College of Ireland

Supervisor: Imran Khan

National College of Ireland  
MSc Project Submission Sheet  
School of Computing

**Student Name:** IRFAN NAYEEM.....

**Student ID:** 20252391 .....

**Programme:** MSc in Cybersecurity - MSCCYB..... **Year:** 2022-2023

**Module:** Research in Computing  
.....

**Supervisor:** Imran Khan .....

**Submission Due Date:** 15/12/2022  
.....

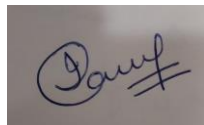
**Project Title:** Malware detection for URL security.....

6120 ..... **Page Count:** 21.....  
**Word Count:**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:**



**Date:** 15/12/2022  
.....

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/> y
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/> y
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/> y

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# **Mal-adware detection for URL security**

Irfan Nayeem

x20252391

## **Abstract**

As Internet technology advances, network security is more and more exposed to various threats. In particular, malicious uniform resource locators (URLs) can be spread by attackers to carry out operations like spamming and phishing. Making progress in identifying malicious URLs is crucial for putting an end to these attacks. However, there are still several severe problems with the current findings. For instance, it could be challenging to identify problems precisely. Some of the detection methods used today are easy for attackers to circumvent. The vector input dimension affects the dynamic convolution technology's middle layer feature mapping width. The pooling layer settings are additionally continuously changed according to the depth of the present convolution layer and the duration of the URL input to obtain more precise features over a wider range. To acquire the embeddings of a URL, the innovative embedding approach bases word embedding on character anchoring.

# 1 Introduction

## 1.1 Background

Malware research and identification have become a problem for cybersecurity due to the growth of ad links on screens that tempt individuals to click on or occasionally download risky payloads that frequently result in significant attacks like ransomware (Oprea et al., 2018). This problem will be addressed by the project, which will create a malware detection program that will scrape each ad link from a parent web page, create a fingerprint for each link, and compare those biometrics to those recorded in the repository to identify each link's classification (whether good or bad one). To deceive people into clicking on risky links that install Trojans on their systems or expose their personal information, hackers frequently utilize spam and phishing. Users can recognize bad URLs and be shielded from attacks by using technology for identifying them. Blacklist-based techniques have been used in the past in studies on harmful URL detection to find harmful URLs. This tactic has a few special benefits. It is quick, easy to understand, and less likely to produce false positive results (Oprea et al., 2018). However, the domain generation algorithm (DGA) nowadays is capable of producing tens of thousands of unique malicious domain names each day that are undetected by conventional blacklist-based techniques.

To detect fake URLs, researchers have started employing machine learning techniques. However, these tactics often require hand-extracting the characteristics since attackers may manufacture these qualities to hide their identities. Research focuses on developing a more precise malicious URL detection system in light of the modern network environment's complexity. This paper proposes a DCNN-based model for detecting malicious URLs (Oprea et al., 2018). To automatically extract properties and understand the language of the URL, it uses word vectors, a character-based technique based on character embedding. The k-max-pooling layer is used in place of the pooling layer. The dynamic convolution technique's intermediate layer feature mapping width is dependent on the vector input dimension. Additionally, based on the previous convolution layer's depth and the URL input's length, the pooling layer parameters are continually modified to aid in the extraction of deeper features over a larger range (Ali et al., 2019).

The method of detection is as follows. By analyzing the URL in order, the domain name, subdomains name, and website domain suffix are first obtained. The first branches of the detection model lengthen each URL and assign a unique number to each word. Numbers are used to represent the complete URL. The embedding layer is then given the sequences and

trained along with other layers. During the training phase, these patterns will learn the right vector expression (Ali et al., 2019). A DCNN is then given the data stream produced by the embedding layer. In two further rounds, the output goes through a convolution layer, a folding surface, and a pooling layer. The data stream is flattened by the flattened layer. To establish whether a piece of software is meant to be hazardous or not, malware detection methods are utilised. A detection system is made up of two processes: analysis and detection. Malware detection software is a method for preventing malware. The methods they employ have an impact on the characteristics of such detectors. Its two inputs are the programme under investigation and the distinctive traits or behaviors of a particular piece of code. Its detecting method can determine if a piece of software is malicious or harmless. To create a method for malware detection that is successful, malware analysis is required (Ali et al., 2019). Malware analysis is the practice of examining malware's functionality and purpose to comprehend how it operates and develop a defence to safeguard the organization's network. Malware analysis, which describes how malware works and its impacts on the system, may be divided into three categories. However, the tools, resources, and expertise needed to carry out each type of examination vary.

Over the last 10 years, a lot of malware researchers have focused on data mining in an attempt to identify new, unknown malware; they have promoted data mining as a fourth recommended malware detection approach. In 2001, the idea of employing data mining and machine learning methods to recognise novel, undiscovered viruses based on each of their unique binary codes was first put out. Then, several investigations were conducted to identify various malware. Data mining facilitates data analysis by employing automated statistical analytic methods to find noteworthy patterns or connections. Since malware may steal personal data, turn off security software, and do other actions, it poses a major risk to a computer's system (Ali et al., 2019).

### **1.2 Research objectives**

To discuss several tools to decrease mal-advare activities.

To evaluate several causes of attacks in mal advare.

### **1.3 Research questions**

1. How to analyse several models related to URL security?
2. What is the purpose of tools in lessening the risks in URLs?

## **2 Related Work**

## 2.1 A malicious URL detection model

At present, outdated algorithms based on blacklisting and modern ones based on machine learning are the two main types of algorithms used to identify dangerous URLs. The first written description of the blacklist-based detection technique was published. Although quick and effective, this approach has considerable drawbacks because it cannot identify newly created malicious URLs. According to Wang et al. (2020), attackers may simply avoid the conventional blacklist-based detection approach by utilizing a random seed to create a wide range of malicious domain names. Using machine learning algorithms, researchers have found dangerous URLs in literature. Using statistical data, machine learning develops a prediction model that categorizes URLs as dangerous or benign. This function looks at the URL and any related websites or web pages to obtain the properties. According to Wang et al. (2020), static features and various properties are frequently used to categorise the properties gathered using this approach. The host information, lexical knowledge in URL strings, and sometimes HTML and JavaScript data are retrieved from literature. The "support vector machine (SVM)" is used for detection after several network traffic-related characteristics are obtained from the URL using literature.

According to Wang et al. (2020), three feature processing techniques are suggested in the literature to enhance classification performance. The aforementioned techniques have been successful, yet they still have several drawbacks. Long-standing machine learning-based detection techniques may need features to be manually extracted. Several researchers have developed several methods based on deep learning models to detect malicious URLs and decide whether a URL is hazardous just by looking at the strings it contains. The idea of text classification is being looked at in this instance. These processes can use the URL to acquire accurate data automatically. For instance, literature Do Xuan et al. (2020) classify URLs generated by DGA using a "character-level cyclic neural network model". A sophisticated machine learning method to identify risky URLs To assess if DGA produces the URL, character-level semantic characteristics may be used with "the n-gram model and deep learning". The single-layer long selective memory structure is one deep learning framework for detecting bogus URLs (LSTM). These characteristics can be used by attackers to avoid detection, making it more challenging to maintain detection systems relying on conventional machine learning. Additionally, a trained algorithm may overlook certain essential information from the URL when detecting dangerous URLs on a big scale.

## **2.2 Malicious URL detection depending on Machine Learning (ML)**

According to Do Xuan et al. (2020), the Uniform Resource Locator of an Internet resource serves as its identification (URL). The resource name, which provides the IP address or web address where the resource is hosted, and the network identifier, which determines the protocol to use, are the features and two crucial parts of the URL. Each URL has a distinct structure and format, as can be seen. Attackers typically try to modify one or more structural elements of URLs to trick people into sharing their malicious URLs. Links that endanger users are known as malicious URLs. These URLs might lead users to harmful, dangerous, phishing, or other web pages where they could download malware or where attackers could take control of users' systems. According to Do Xuan et al. (2020), the number of harmful URLs has increased over the past few years, which has led to the development and use of strategies or procedures for their elimination. When it comes to the difficulty of recognizing malicious URLs, there are now two main trends: harmful URL detection based on indications or sets of rules, and hazardous URL detection based on behavior analysis techniques.

A method based on a collection of signs or criteria may quickly and accurately identify fraudulent URLs. However, this technique cannot be used to find newly identified malicious URLs that do not follow the predefined signs or limits. Identifying risky URLs using blacklisting, machine learning, rule finding, and profound learning-based detection techniques is common practice. Do Xuan et al. (2020) looked at the IP addresses and domain names of various public blacklist sets of data. They created a graph-based approach to locate subsidence in the blacklists after realising that parked domains can account for a sizable portion of entries. To increase the technology's capacity for detection, researchers have increased the blacklists already in use. The fishnet system, developed by Yuan et al. (2021), used five heuristic algorithms to recognise simple permutations of well-known phishing sites to detect new phishing URLs. Since growing the data set takes time and the size of the initial list affects how effectively blacklist detection functions, various academics have suggested a rule-matching-based technique. To locate new, previously unidentified dangerous URLs that fit the matching rule, rule matching examines the text structure of existing harmful URLs. SpyProxy is a proxy-based tool for locating fake websites. Moshchuk developed a set of matching rules for the SpyProxy to match the intercepted data delivered from the web server to the browser.

## **2.3 Utilization of novel optimization algorithm in detecting malicious URL**

The detection of dangerous URLs has historically been the focus of research on internet security protection. Currently, the two main techniques for identifying dangerous URLs are



black-and-white lists and machine learning. A website cannot be reached using the first method once it has been determined that its URL does not exist in the blacklist database. According to Yuan et al. (2021), numerous popular browsers, notably "IE8", "Mozilla Firefox 2.0", "Safari", "Chrome", and others, have made extensive use of them. It distinguishes itself by being both effective and simple. However, the system calls for constant blacklist maintenance, which is costly and might lead to the problem of judgement omission. These issues frequently reduce its appeal. To improve the capacity to identify fake URLs, the second method evaluates the challenges from a variety of angles, covering URL characteristics, website domain character traits, host characteristics, etc. According to Yuan et al. (2021), studies in this field have grown in popularity over time. However, the majority of these study findings pay minimal attention to machine learning research instead and prioritise the extraction and analysis of URL properties. Rumelhart and Hinton published the initial description of the BP neural network (BPNN), a well-liked machine learning approach for URL security testing, in 1986. Recent years have seen a sharp rise in internet usage. According to Yuan et al. (2021), the rise of mobile devices, ad hoc networks, sensor technologies, and the Internet of Things which was driven by the need for a lockdown to combat the COVID-19 epidemic has led to the Internet becoming increasingly important in people's everyday lives and activities. Due to the availability of reliable facilities like cloud storage, affordable platforms, and a sizable target market, the majority of businesses moved online. Internet use carries several risks, such as viruses, spam, scamming, financial fraud, theft of information, and data loss. The main threat source is malicious websites.

## **2.4 Cyber Threat Intelligence-Based Malicious URL Detection**

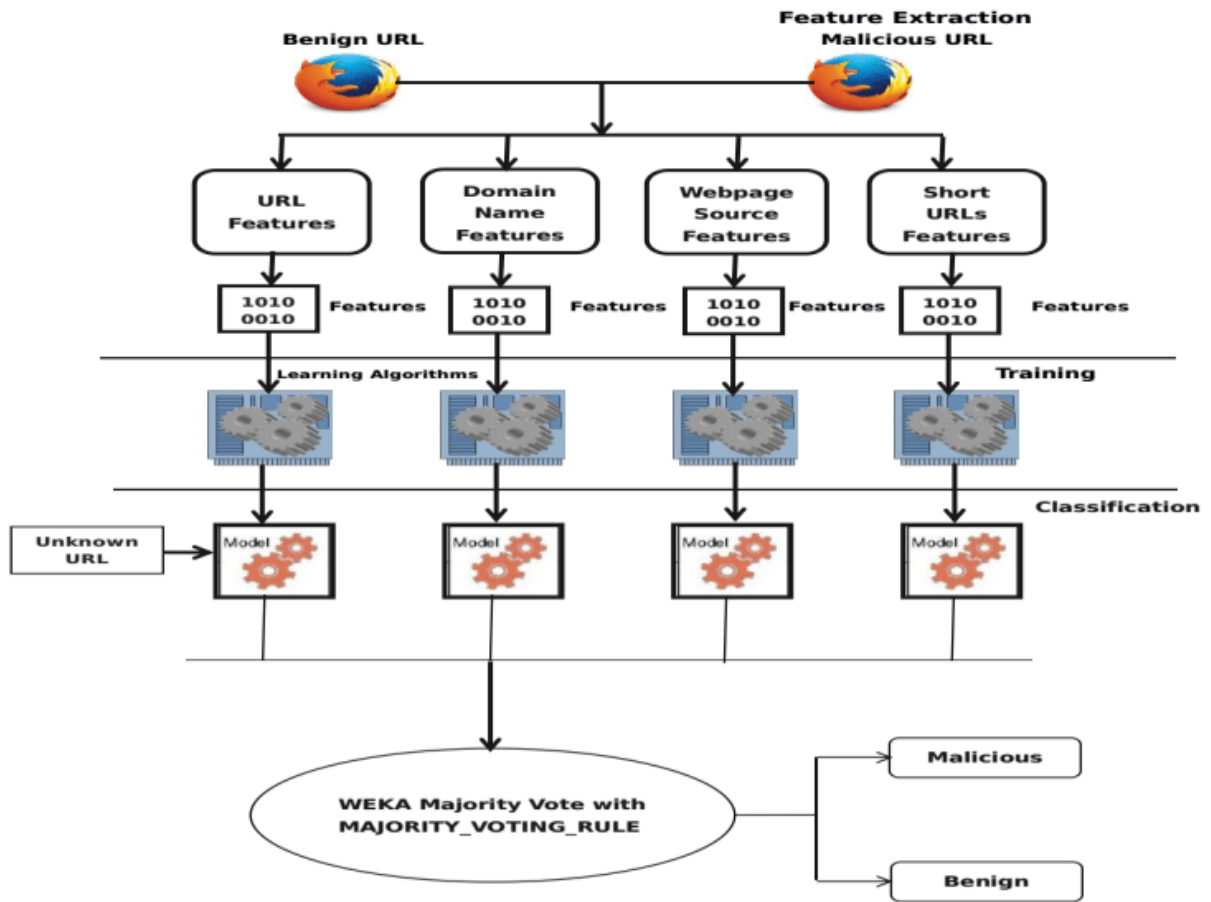
According to Yuan et al. (2021), "18.5 million websites have been compromised with malware". According to Google's safe surfing report, "there were two million phishing websites in September 2020", up more than "280% from the same month in 2010". To persuade people to visit malicious websites, attackers disseminate fake news and adverts. Once a victim accesses a malicious website, attackers can employ various tactics to either trick the victim into aiding them in committing financial fraud or other forms of assaults or install malware payloads on the killer's browsing device. The designers of many bad websites claim that they were not intentionally designed to be destructive. According to Bo et al. (2021), vulnerable websites might be used by nefarious attackers. Identification of malicious websites has been a concern since 2004's first days. To reliably identify these websites, several techniques have been suggested. Depending on where they acquired their information, these methods may be

categorised into three groups: script-based, URL-based, and online content-based. URL-based detection has garnered the most study, trailed by content-based detection, while script-based discovery has gotten less. Since URL-based detection is proactive and secure for the detecting machines and may find harmful URLs before the user accesses them, it was picked as the preferred method of detection.

According to Bo et al. (2021), there are several methods for locating harmful drugs and unfavourable websites, many of which use the extraction of information from the URLs of those sites. Fewer of these procedures mechanized the features using deep learning techniques; the majority of them depended on humans to extract the attributes. Several distinct pieces of information were gathered and used for the detection, including a host questionnaire survey to collect like country name and host sponsor, domain attributes like lexical elements like the length of the URL and the number of dots in it. However, attackers might change URL-based characteristics and manipulate them, rendering them worthless for efficient representation.

### **3 Research Methodology**

In order to detect mal adware for URL security an android application was created using java. The particular application took the URL as input for detecting whether the URL is phishing or not. A decision Tree has been used in order to detect phishing and machine learning has helped to provide a proper description of the detection. In this paper 0 has been considered as “Trusty”, 1 means “Suspicious”, 01 means “Phisy” and 0.5 means “Suspicious”.



**Figure 1: Framework of our proposed of malicious URLs detection system, Source- (Patil, and Patil, 2018)**

The organisational structure of our suggested malicious URL detection system is illustrated in Fig. 1. The feature extraction, training, and classification steps make up this technique. The Java feature extraction programme receives the dangerous and benign URLs directly from the benchmark sources. The 117 static and dynamic properties of the good and bad URLs have been extracted (Patil, and Patil, 2018). These qualities are binary and numeric. When Researchers were generating the dataset, they gave the good URLs the number 1 and the bad URLs the number +1. Six decision tree learning algorithms “J48 Decision Tree”, “Simple CART”, “Random Forest”, “Random Tree”, “ADTree”, and “REPTree”—are taught on our labelled dataset. Six trained models are provided at this stage for the testing stage (Ullah *et al.*, 2020).

### **3.1 Feature extraction**

Researchers have identified four different categories of static and dynamic URL qualities, including URL, domain name, website source, and short URL traits. The Java URL feature extractor has been put to use. The Java URL class is used to extract URL features, and the features are retrieved by lexically scanning the URL string. Depending on domain name recovery and scanning of the web address, the website address features extraction is put into practise. The website is browsed together with a Firefox instance, which collects the webpage's source characteristics, in order to apply a Java feature extraction engine to understand how HTML is constructed (Patil, and Patil, 2018). To ensure a distinct session for each URL viewed for extracting features, a new Firefox browser instance is started for each URL. By getting in touch with the URL shortening service providers, the enlarged URLs may be obtained. The study has established a predetermined threshold of 30 for the duration of URLs; if the submitted URL is longer than 30, it is flagged as malicious when we receive the real URL from URL shortening providers. The retrieved URL string's lexical characteristics were also investigated in the article to assess if it was innocuous or harmful (Patil, and Patil, 2018). This method is often used by many search engines, browser toolbars, etc. to prevent users from visiting the websites that are blacklisted. A database of reported URLs is called Blacklist. These dangerous URLs are not available until successful assaults are identified. However, it is difficult to maintain such a lengthy list up to date because more are always being added. They struggle to identify new threats because new URLs may be generated automatically, enabling them to escape blacklists and zero-hour phishing detection. Despite these disadvantages, blacklists are nevertheless often used by antivirus software and online filtering applications (Ullah *et al.*, 2020).

### **3.2 Decision tree algorithm**

Decision trees can handle scenarios that are both category and numerical/continuous. Numerical decision trees provide a real anticipated value, whereas categorical decision trees only provide a result of 0/1, Yes/No, or True/False. Before analysing the subtrees, decision trees separate the data into subtrees and assign both parent and child nodes to each one. The homogenous clusters of the full dataset are shown in these subtrees. When employing decision trees, overfitting may not be a desirable thing (Ullah *et al.*, 2020). By restricting the variables taken into account for each subtree, the Random Forest classifier may be utilised to address problems. Another method for preventing overfitting is pruning. The model's accuracy might be increased while the learning rate is sped up by altering a few parameters. If the maximum

characteristics option is selected, the model will have more alternatives for evaluation. If there are too many attributes, the technique could run slowly. The number of trees will directly affect the model's accuracy up until the maximum number reaches its threshold. Calculating the accuracy with different tree counts might reveal this (Ullah *et al.*, 2020). Regardless of how many trees are added, accuracy remains constant until it reaches its highest level.

## 4 Design Specification

This engineering-focused phase aims to gather pertinent URL-related data. This information consists of things like popularity statistics, host information, HTML and JavaScript content on the website, if the URLs are on a blacklist, attributes inferred from the URL String, etc. As seen in Figure 2, the feature map may be made by obtaining various kinds of data from a URL.

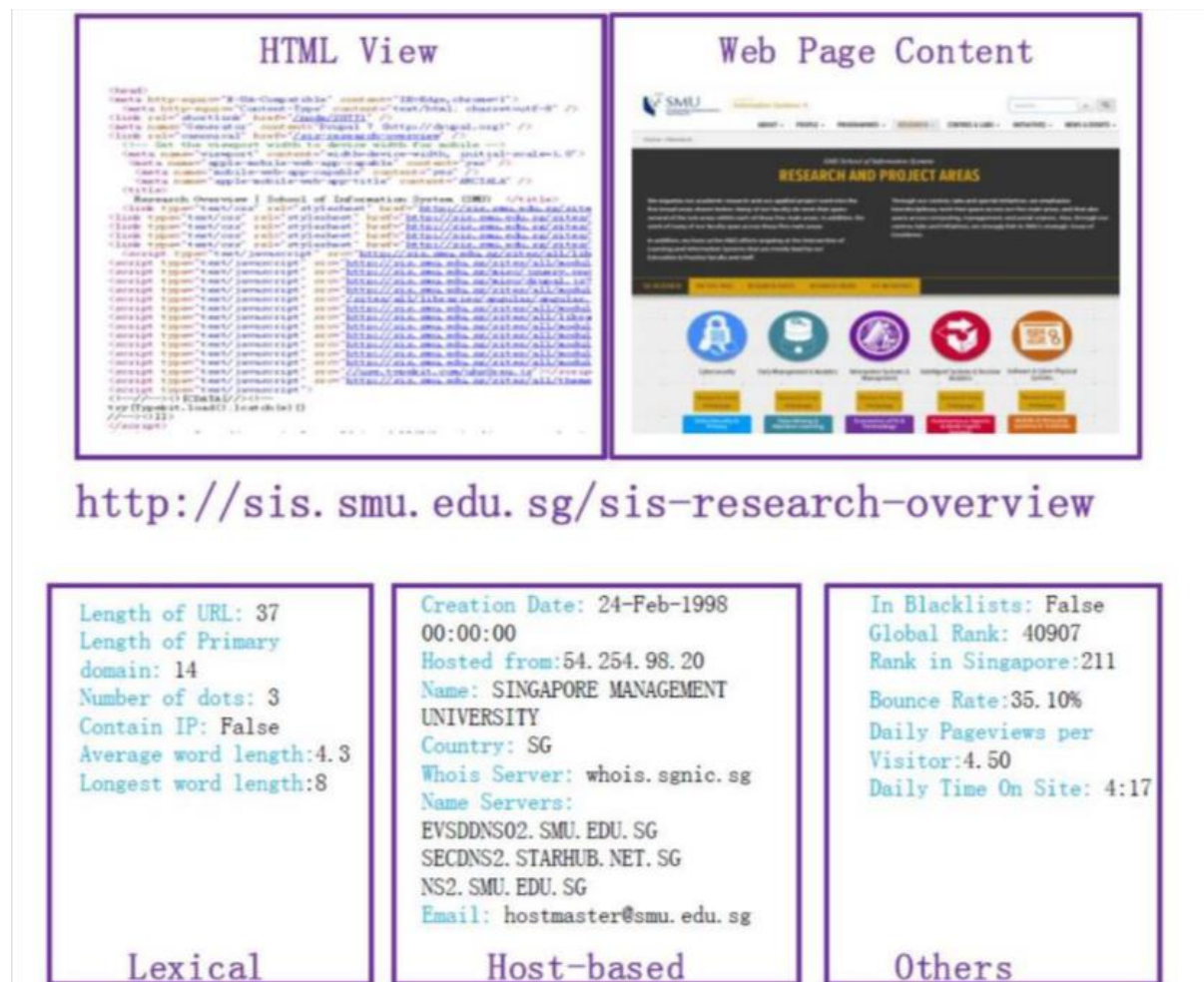


Figure 2: “Example of information about a URL that can be obtained in the Feature Collection stage. Source- (Usman *et al.*, 2021)”

Blacklists may be used, as was already noted, to swiftly identify harmful URLs. All active URLs that have been determined to be harmful are included in the list (either by in-depth investigation or crowd-sourcing). Blacklisting has been shown to suffer from significant high negative cases due to the challenge of keeping comprehensive up-to-date lists, despite its ease of use and simplicity. As a result, blacklist inclusion may be employed as a strong characteristic rather than the only criteria (Usman *et al.*, 2021). Numerous JavaScript techniques are routinely used by hackers to encrypt malicious code or launch unwanted activities without the client's knowledge. Use of the `eval()` and `unescape()` functions, for instance, may indicate that HTML code that has been encoded is being run often. Native JavaScript methods will be used to find risky URLs. During the investigation, a subset (seven) of these native JavaScript methods that are often utilised in cross-site scripting and the spread of web-based malware were found. The "collection complexity" is the technical effort required to obtain accurate data on the attributes (Usman *et al.*, 2021). In contrast to the other characteristics, which require more connections and have greater collection costs, the popularity, contextual, and blacklist attributes are directly retrieved from the URL. The problem of detecting malicious URLs may be solved using a large array of machine learning techniques. Several of these learning techniques may commonly have been used to quickly train a forecasting model after turning URLs into feature vectors (Janet, and Kumar, 2021). In order to more effectively handle the issue, attempts have also been made to develop specialised learning algorithms that either take use of the qualities given by the training data of malicious URLs or that specifically address the issues the application encounters. For this purpose, we categorise and evaluate the learning algorithms used in this area, and we also suggest appropriate machine learning technologies that may be used to solve certain problems.

## **5 Implementation**

### **5.1 Data collection**

This section contains details on where highlighted URLs came from. Several machine learning researchers and online sources, such as PishTank, OpenPish, BlockList, and Yahoo's directory listing, have combined about 4.5M URLs into a sizable dataset. Along with the categorization of URLs, information must be gathered to determine if they are safe or harmful. As a result, they develop software that can draw traits straight from the URL (Janet, and Kumar, 2021). Because the lexical analysis is only designed to collect knowledge from full-length URLs and this study is focused on lexical features, the short URLs must be converted to their original form. This work has been finished with the help of "urlex.org", a URL expander service.

## 5.2 Data pre-processing

The data must first go through pre-processing once the characteristics have been extracted. Pre-processing, sometimes referred to as "data cleaning," involves filling in blank values and formatting data. Each machine learning model requires data in a certain format. For instance, since certain algorithms cannot accept null values, they must handle the raw, unformatted dataset. This stage is crucial for machine learning models since it affects results the most. The performance of the model can be improved by decreasing data noise (Janet, and Kumar, 2021). Before choosing the best strategy, it's crucial to take into account the dataset's organisation so that various machine learning techniques may be used on the same dataset.

## 6 Evaluation

Machine learning may be used to recognise shortened harmful URLs and create a real-world hazardous URL detection method for both short and traditional URLs. To do this, we propose a Chrome plugin that utilises the recommended machine learning model to instantly recognise and block risky URLs. A few studies concentrated on providing services to online social networks. The Chrome addon contains three stages: The URL restriction extension is made, a Python machine learning model is incorporated into Chrome, and the script is changed and attached to Chrome in the first two steps (Yeboah-Ofori, and Boachie, 2019). HTML, CSS, and JavaScript are among the file formats that the Chrome extension is compliant with. Therefore, it is necessary to update and incorporate into the study the written Python feature extraction approaches.

### 6.1 Implementation Explanation Tree in code

- 1) The click on **SUBMIT** button is recorded in **binding.buttonFirst()** in **onViewCreated()** of **FirstFragment**.

A screenshot of a code editor showing Java code for an Android application. The code is as follows:

```
binding.buttonFirst.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        binding.output.setText("");  
        String url = binding.url.getText().toString().toLowerCase();
```

Figure 3: Buildup of submit button

- 2) here first we check if the url is valid with an inbuilt Validation class function **Validations.validURL(url) == false**

```

57     handler.post() -> {
58         if (Validations.isValidURL(url) == false) {
59             mainHandler.post() -> {
60                 binding.output.setText("URL not exists");
61             };
62         }
63     }

```

**Figure 4: Validation of URL is it exist or not exist**

- 3) if the URL is incorrect then show "URL does not exist" and end the process (**return;**) else if url is valid then:
  - a) First, we convert the string url collected from URL format by using line67 **IF** all the conditions of the URL Info class are met.

```

try {
    URLInfo urlInfo = new URLInfo(url);
    String status = urlInfo.getStatus();
    mainHandler.post() -> {
        binding.output.setText(status);
    };
} catch (Exception ex) {
    mainHandler.post() -> {
        binding.output.setText("Invalid URL");
    };
}

```

**Figure 5: Check for URL is live**

- b) Conditions of URLInfo class which are present in parseULR() function – URLInfo.java:
  - i) First, we check if the URL is empty then show error else if it's not empty then we convert the url to LowerCase and then call the 4 main functions which dissect the input url into scheme, domain, port and Ip by using functions extractScheme(); extractDomain(); extractPort(); and extractIP());
  - ii) **extractScheme();** checks **indexOf("://")** line22, if **://** is not found in the url then throw error else go ahead and check store the scheme line27. Again, check if the scheme is empty then throw error. Then check if the scheme is equal to http or https line32. If none then throw error else scheme variable has the correct scheme.
  - iii) **extractDomain();** this function extracts the domain part from the url.
  - iv) **extractScheme();** checks **indexOf("://")** line22, if **://** is not found in the url then throw error else go ahead and check store the scheme line27. Again, check if the scheme is empty then throw error. Then check if the scheme is equal to http or https line32. If none then throw error else scheme variable has the correct scheme.



- v) **extractScheme()**; checks **indexOf("://")** line22, if **://** is not found in the url then throw error else go ahead and check store the scheme line27. Again, check if scheme is empty then throw error. Then check if scheme is equal to **http** or **https** line32. If none then throw error else scheme variable has the correct scheme.
- c) Once all the above conditions are met then we call **urlInfo.getStatus()**; and store its result in the variable **status**. This function is calculating the url phishing score based on the Decision Tree and stores the score in variable **total** as follows:
- i) **getPointUrlLength()** - calculates the length of the full URL and returns the score accordingly.
  - ii) **getDotCount()** - counts the number of dots present in the URL and returns the score.
  - iii) **getCertificateCount()** - checks the validity of ssl certificate using **CertificateUtils** class line177, where we open a connection to capture the certificate in **Certificate** class variable. Next, we check if this received certificate equals **"X.509"** which is a standard to check non-phishing official certificates.
  - iv) **getDomainAge()** - checks the domain's age and scores it accordingly. Age is calculated from line147 where we call the **getDomainInformation(url)** function of the **ApiUtils** class. This function connects to api (<https://ip2whois.com/developers-api>) which is a free service from ip2whois.com for developers to integrate their services in the developer's application. From this api, once we get the response in JSON object format, we extract **domain\_age** from it and return the result.
  - v) **getPointSuffix()** - this function checks for hyphen ("-") and returns the score accordingly.

```

125
126 String result = "unsolved";
127 if (total < 0.1){
128     result = "TrustWorthy";
129 }else if (total < 0.3){
130     result = "Fairly Legitimate";
131 }else if (total < 0.5){
132     result = "Unsolved";
133 }else if (total < 0.75){
134     result = "Suspicious";
135 }else{
136     result = "Phishy";
137 }
138

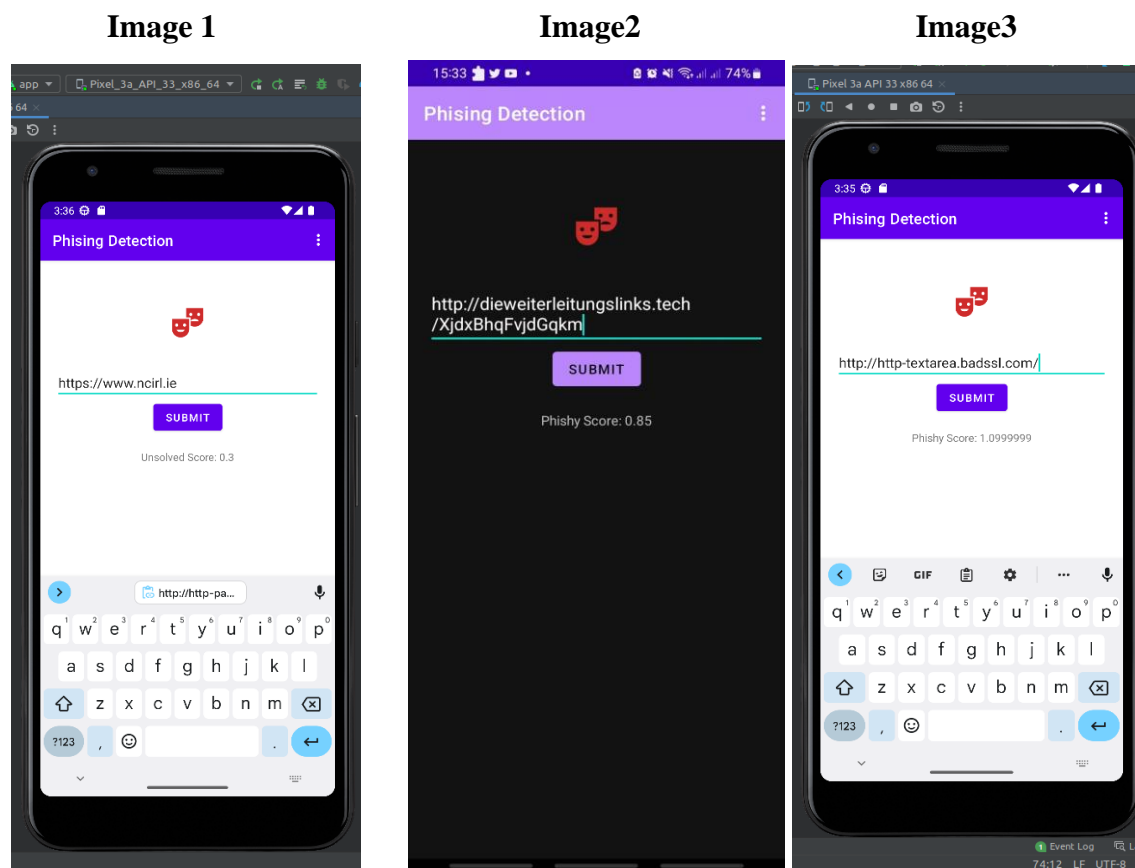
```

**Figure 6: In end give phishy score to URL**

- d) Once we have the total calculated, then we find out the appropriate message to be shown to the user depending on the total.

## 6.2 Final outputs

The test was done on both the application in android phone as well as in the live system Android studio. In order to this many other test were also performed to check the accuracy of the results which is mentioned in the result sections respectively.



**Figure 7: Image1 and Image3 Phishing detection from Android studio.  
Image2 phishing detection from Mobile Phone**

## 6.1 Data source and dataset

The dataset was split into training and testing sets at a ratio of 66:34, or 66% for training and 34% for testing. Researchers collected malicious and helpful URLs from a variety of sources. The list of safe URLs is created using the top Alexa websites. Innocent URLs were gathered for the study from 26,041 different sources (Yeboah-Ofori, and Boachie, 2019). For the dangerous dataset, the study has gathered URLs from three sources, including the spam domain blacklist, the ransomware and phishing blacklist, and the Malware Domains List's list of harmful and injection attack URLs.

## 6.2 Discussion

It's critical to recognise dangerous websites before they cause us damage. In light of this, despite the existence of several URL-shortening services, there are presently no methods that are widely acknowledged for defining short URLs as harmful or benign. This work uses language cues and a straightforward machine-learning technique to find URLs that have been unlawfully shortened. This approach proposes a straightforward method to spot bogus websites: simply examine the URL string. The bulk of the properties that have been found as indications of potentially harmful URLs is categorised in the Feature Representation section (Yeboah-Ofori, and Boachie, 2019). This project has been modified to incorporate a browser plugin that categorises any given short or conventional URL as harmful or benign in order to provide a real-time categorization of shortened URLs. Along with the risk score, this machine learning add-on includes the website's categorization and original URL. The decision tree strategy, which was chosen, provides a wide range of features that will aid the model's accuracy rise. The suggested viable method to identify Android malware in this study uses the Ranker search strategy and Gain Ratio attribute evaluator to find certain qualities (Yeboah-Ofori, and Boachie, 2019). Using the J48 Decision Tree and other machine learning methods, the pre-processed dataset was split into malware and benign data. The study found that the Decision Forest methodology delivered results more rapidly and accurately than the Decision Tree strategy. The study also compared and examined a wide variety of performance measures in respect to different classifiers and feature counts. Users may quickly and correctly discern between hazardous and beneficial programmes using the proposed detection approach.

## 7 Results

Result obtained successfully by different URL. Reading is observed on different scale levels from 0 –0.25, 0.25-0.50 and in between 0.50 – 1. result was obtained on both environments

i.e., android studio as well as by installing the application in mobile devices. In the given images below three tests were done. In the first phase tests were done on NCI website where the result is 0.3 which means it is not suspicious but on the other hand two vulnerable websites were tested on which the phishy score is more than 0.75 which means they are more suspicious.

URL	Phishy Score	Label
https://ncirl.ie	0.05	Trusted
https://amazon.ie	0.0	Trusted
http://www.malwaredomainlist.com/	1.125	Trusted
http://irfannayeem.com	URL not exist	N/A
http://malshare.com/	0.8	Malicious
http://cybercrime-tracker.net/	0.85	Malicious
http://diweiterleitungslinks.tech/XjdxBhqFvjdGqkm	0.85	Malicious
https://google.com	0.0	Trusted
https://gooool.com	URL not exist	N/A
http://vxvault.net//ViriList.php	0.8	Malicious
https://mymoodle.ncirl.ie/	0.3	Fair
https://reliaquest.wd5.myworkdayjobs.com/	0.45	Fairly legitimate
https://www.av-comparatives.org/	0.3	Fairly legitimate
https://bugcrowd.com/	0.0	Trusted
https://transparencyreport.google.com	0.25	Fairly legitimate
http://www.virusign.com/index.html	1.05	High risky

## 8 Conclusion and Future Work

The security of the Internet is at stake due to the increase in harmful website prevalence. When it comes to detecting risky URLs, machine learning techniques are less effective than traditional blacklisting techniques. By examining its attributes, a well-trained machine learning model may be able to detect a dangerous URL with more accuracy than blacklisting, which is unable to discover new problematic URLs (Oprea *et al.*, 2018). There is reason to think that as technology advances, more issues might arise from products like URL shortening, since there is no established method of employing short URLs as a recognition mechanism. In this work, a variety of machine learning techniques are used to detect fraudulent URLs. One may also conduct more study by considering the host-based information, the popularity of the website, and the linguistic characteristics of the URL. The categorization scheme utilised in this study for URLs can be useful for machine learning models. This article classifies the URL as either dangerous or benign (Oprea *et al.*, 2018). By examining this idea in further detail, we may classify harmful URLs into three groups: spam, phishing, and malware. With the help of this

classification, the user will be able to make better decisions. Accuracy will increase while training and prediction durations will decrease when the best machine learning approaches are combined (Zhang, 2018). Consider using an online way to gather real-time data that keeps the model current by training with current malware Attacks that may apply cutting-edge, previously unrecognised tactics. A static plugin was also developed for this project. The machine learning model must be able to just save each URL that the browser accesses on the network and modify the model in real-time if it is to provide us more accurate results (Zhang, 2018). The method may grow more difficult if the trained model is updated, but by doing monthly or weekly upgrades, the study can keep the intricacy to a minimal.

## References

- O. S. A. Aboosh and O. A. I. Aldabbagh, "Android Adware Detection Model Based on Machine Learning Techniques," 2021 International Conference on Computing and Communications Applications and Technologies (I3CAT), 2021, pp. 98-104, doi: 10.1109/I3CAT53310.2021.9629400.
- Ali, R., Ali, A., Iqbal, F., Khattak, A.M. and Aleem, S., 2019, December. A systematic review of artificial intelligence and machine learning techniques for cyber security. In *International Conference on Big Data and Security* (pp. 584-593). Springer, Singapore.[https://link.springer.com/chapter/10.1007/978-981-15-7530-3\\_44](https://link.springer.com/chapter/10.1007/978-981-15-7530-3_44)
- Bo, W., Fang, Z.B., Wei, L.X., Cheng, Z.F. and Hua, Z.X., 2021. Malicious URLs detection based on a novel optimization algorithm. *IEICE TRANSACTIONS on Information and Systems*, 104(4), pp.513-516.<https://www.emerald.com/insight/content/doi/10.1108/JEIM-01-2021-0057/full/html>
- Do Xuan, C., Nguyen, H.D. and Tisenko, V.N., 2020. Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science and Applications*, 11(1).<https://pdfs.semanticscholar.org/2589/5814fe70d994f7d673b6a6e2cc49f7f8d3b9.pdf>
- Janet, B. and Kumar, R.J.A., 2021, March. Malicious URL detection: a comparative study. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 1147-1151). IEEE.<https://ieeexplore.ieee.org/abstract/document/9396014/>
- Oprea, A., Li, Z., Norris, R. and Bowers, K., 2018, December. Made: Security analytics for enterprise threat detection. In *Proceedings of the 34th Annual Computer Security Applications Conference* (pp. 124-136).<https://dl.acm.org/doi/abs/10.1145/3274694.3274710>

Oprea, A., Li, Z., Norris, R. and Bowers, K., 2018, December. Made: Security analytics for enterprise threat detection. In *Proceedings of the 34th Annual Computer Security Applications Conference* (pp. 124-136).<https://dl.acm.org/doi/abs/10.1145/3274694.3274710>

Patil, D.R. and Patil, J.B., 2018. Malicious URLs detection using decision tree classifiers and majority voting technique. *Cybernetics and Information Technologies*, 18(1), pp.11-29.[https://cit.iict.bas.bg/CIT\\_2018/v-18-1/02\\_paper.pdf](https://cit.iict.bas.bg/CIT_2018/v-18-1/02_paper.pdf)

Ullah, F., Javaid, Q., Salam, A., Ahmad, M., Sarwar, N., Shah, D. and Abrar, M., 2020. Modified decision tree technique for ransomware detection at runtime through API calls. *Scientific Programming*, 2020.<https://www.hindawi.com/journals/sp/2020/8845833/>

Usman, N., Usman, S., Khan, F., Jan, M.A., Sajid, A., Alazab, M. and Watters, P., 2021. Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics. *Future Generation Computer Systems*, 118, pp.124-141.<https://www.sciencedirect.com/science/article/pii/S0167739X21000066>

Wang, Z., Li, S., Wang, B., Ren, X. and Yang, T., 2020, September. A malicious URL detection model based on a convolutional neural network. In *International Symposium on Security and Privacy in Social Networks and Big Data* (pp. 34-40). Springer, Singapore.[https://link.springer.com/chapter/10.1007/978-981-15-9031-3\\_3](https://link.springer.com/chapter/10.1007/978-981-15-9031-3_3)

Yeboah-Ofori, A. and Boachie, C., 2019, May. Malware attack predictive analytics in a cyber supply chain context using machine learning. In *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)* (pp. 66-73). IEEE.<https://ieeexplore.ieee.org/abstract/document/9058240/>

Yuan, J., Chen, G., Tian, S. and Pei, X., 2021. Malicious URL detection based on a parallel neural joint model. *IEEE Access*, 9, pp.9464-9472.<https://www.hindawi.com/journals/scn/2021/9994127/>

Zhang, J., 2018. MLPdf: an effective machine learning based approach for PDF malware detection. *arXiv preprint arXiv:1808.06991*.<https://arxiv.org/abs/1808.06991>

