A

Major Project

On

# VULNERABILITY DETECTION USING MACHINE LEARNING

Submitted to

Jawaharlal Nehru Technological University, Hyderabad

**In partial fulfillment of the requirements for the award of Degree**

**Of**

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By


**G. SAI KARTHIK GOUD (187R1A05K3)**

**P. SHASHIDHAR (187R1A05M8)**

**M. ASHISH (187R1A05L4)**


Under the Guidance of

**G. VINESH SHANKAR**

(Assistant Professor)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CMR TECHNICAL CAMPUS**

**UGC AUTONOMOUS**

**(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)**
**Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.**


**2018-2022**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING


# CERTIFICATE

This is to certify that the project entitled " **VULNERABILITY DETECTION USING MACHINE LEARNING**" is being submitted by **G. SAI KARTHIK GOUD (187R1A05K3), P. SHASHIDHAR(187R1A05M8), M. ASHISH(187R1A05L4)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

**Mr. G. Vinesh Shankar**
**Assistant Professor**
**INTERNAL GUIDE**

**Dr. A. Raji Reddy**
**DIRECTOR**

**Dr. K. Srujan Raju**
**HoD**

**EXTERNAL EXAMINER**

**Submitted for viva voice Examination held on _____**

# ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **G. VINESH SHANKAR,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing,help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **Dr. Mr. J. Narasimharao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

**G. SAI KARTHIK GOUD (187R1A05K3)**
**P. SHASHIDHAR (187R1A05M8)**
**M. ASHISH (187R1A05L4)**

# ABSTRACT

The risk of network information is increasing rapidly in number and level of danger. The methods mostly used by hackers today are to attack end-to-end technology These techniques include social engineering, phishing, pharming, etc. One of the steps in conducting these attacks is to deceive users with malicious Uniform Resource Locators (URLs).

Phishing web pages are one of the most common and dangerous attacks among cybercrimes. The aim of these attacks is to steal the information used by individuals and organizations.The rising cases of phishing, spamming and malware has generated an urgent need for a reliable solution which can classify and identify the malicious URLs. we propose a malicious URL detection method using a Random forest classifier based on our proposed URL behaviors and attributes (Host based, Popularity based, Lexical features)

# LIST OF FIGURES

# LIST OF SCREENSHOTS

# TABLE OF CONTENTS

# 1. INTRODUCTION

# 1. INTRODUCTION

## 1.1 PROJECT SCOPE

The risk of network information in security is increasing rapidly in number and level of danger. The methods mostly used by hackers today is to attack end-to-end technology and exploit human vulnerabilities. In short, we propose a malicious URL detection method using machine learning techniques based on our proposed URL behaviors and attributes

## 1.2 PROJECT PURPOSE

Malicious URLs are known as links that adversely affect users. These URLs will redirect users to resources or pages on which attackers can execute codes on users' computers, redirect users to unwanted sites, malicious website, or other phishing site, or malware download

## 1.3 PROJECT FEATURES

The main feature of this project is that the system prevents users from visiting malicious websites by displaying a pop-up, it also displays the information regarding the website such as domain details and also gives suggestions for user on how to be safe from malicious URLS

# 2. SYSTEM ANALYSIS

# 2. SYSTEM ANALYSIS

## SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

## 2.1 PROBLEM DEFINITION

Compromised URLs that are used for cyber attacks are termed as malicious URLs. In fact, it was noted that close to one-third of all websites are potentially malicious in nature, demonstrating rampant use of malicious URLs to perpetrate cyber-crimes. A Malicious URL or a malicious web site hosts a variety of unsolicited content in the form of spam, phishing, or drive-by download in order to launch attacks.

## 2.2 EXISTING SYSTEM

The Traditional classification techniques like blacklisting, regular expression, and signature matching approach are lacking the ability to detect newly generated malicious URLs

### 2.2.1 LIMITATIONS OF EXISTING SYSTEM

Following are the limitations of the existing system:

- It is hard and expensive
- New types of attacks and vulnerabilities emerge continuously, so they may be misclassified.
- Lack of Security

## 2.3 PROPOSED SYSTEM

Given the URL we extract the following features:

1. Lexical features

2. Host-based features

3. popularity features

using these features we predict if the given URL is a malicious URL or a legitimate URL

### 2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

The following are the advantages of the proposed system:

- Detects if the given URL is malicious
- Displays suggestions if the URL is malicious
- Ensures safe surfing
- Gives the details of the particular Domain, IP rating

## 2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

### 2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on a project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during the preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits are in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also, all the resources are already available, which gives an indication that the system is economically possible for development.

### 2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### 2.4.3 BEHAVIORAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

## 2.5 HARDWARE & SOFTWARE REQUIREMENTS

### 2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system.

The following are some hardware requirements:

- CPU: Intel core i3 and above
- RAM: 4 GB and above
- Hard disk: 8 GB and above
- Input devices: Keyboard, Mouse

### 2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements:

- Operating System: Windows – 8 and above
- Programming Language: Python 3.7, Html, CSS, JS
- IDE: Anaconda - Jupyter notebook and Spyder

# 3. ARCHITECTURE

# 3. ARCHITECTURE

## 3.1 PROJECT ARCHITECTURE

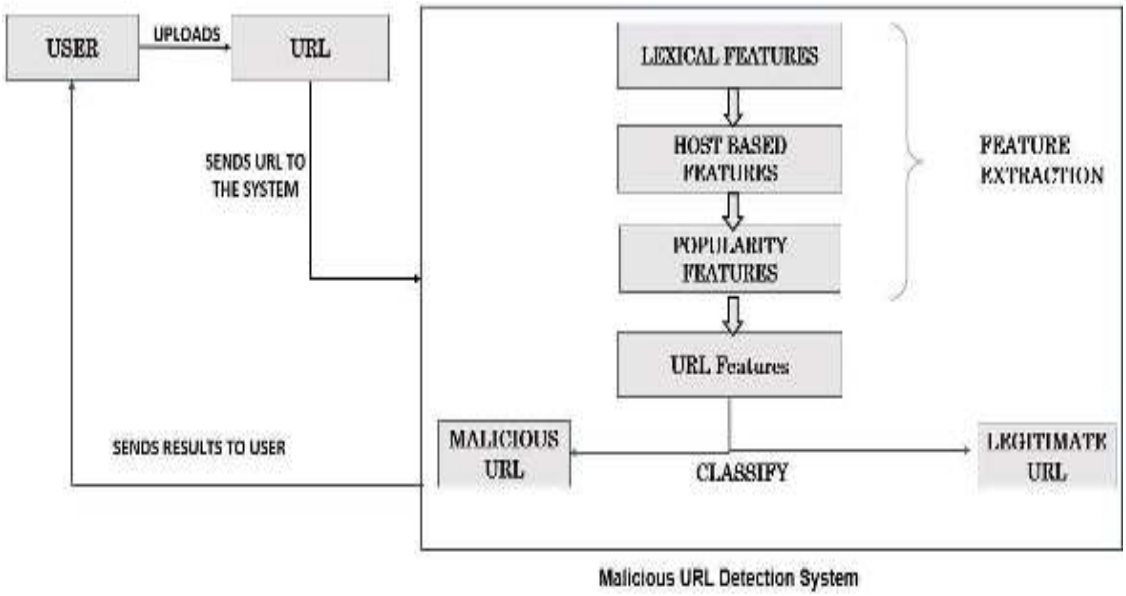



Figure 3.1: Architecture of vulnerability detection using random forest classifier

### 3.2 DESCRIPTION

The most common method to detect malicious URLs deployed by many antivirus groups is the blacklist method.Blacklist are essentially a database of URLs that have been confirmed to be malicious in the past.

Machine learning approaches try to analyze the information of URL and its corresponding websites or webpages, by extracting good feature representations of URLs, and training a prediction model on training data of both malicious and benign URLs. In this we can use static and dynamic features can be used - static features can perform the analysis of a webpage based on information available without executing the URLs that execute the Javascript and includes the lexical and host based feature static analysis techniques have been extensively explored by applying machine learning techniques.

### 3.3 USE CASE DIAGRAM

In the use case diagram, we have two actors who are the user, the admin. The user uploads the URLs in the tool.

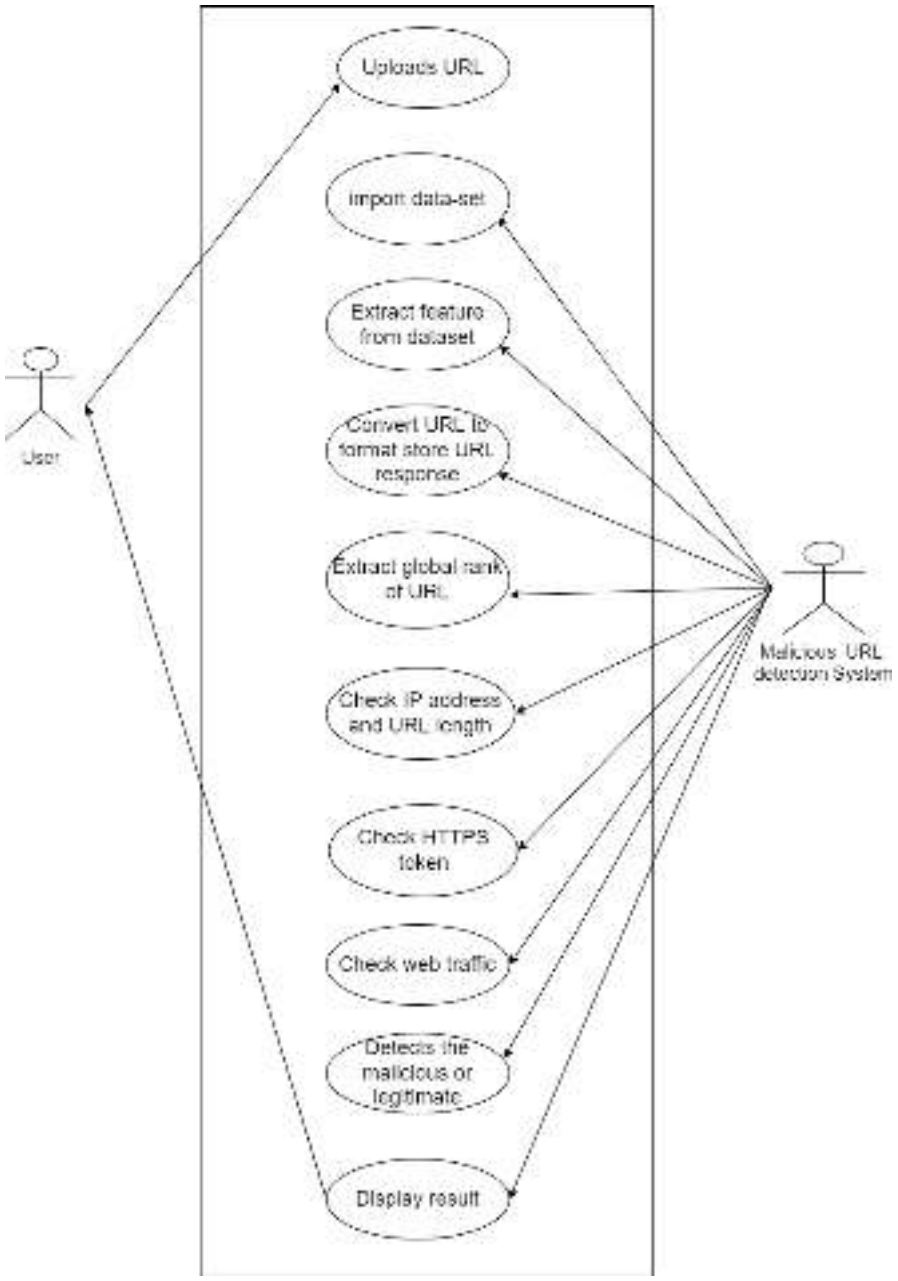



Figure 3.2: Use Case Diagram for vulnerability detection using random forest classifier

## 3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

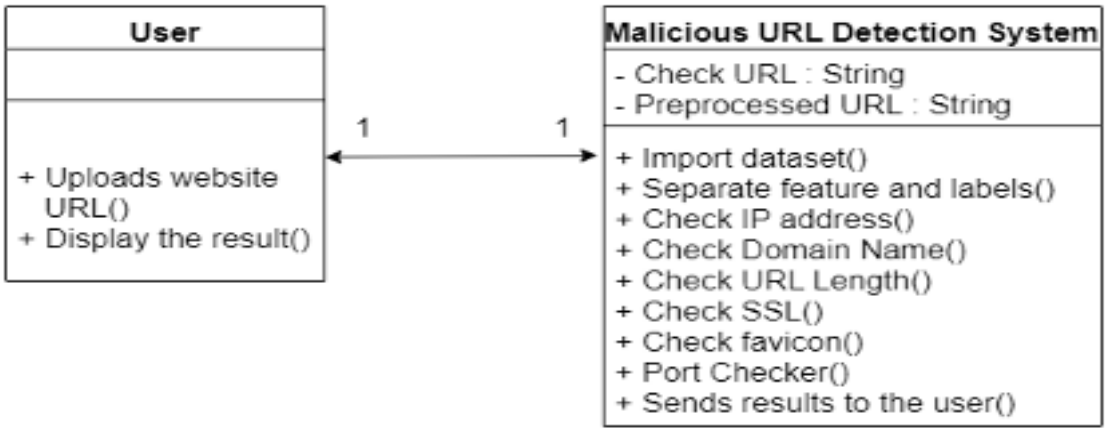



Figure 3.3: Class Diagram for vulnerability detection using random forest classifier

## 3.5 SEQUENCE DIAGRAM

The below Figure 3.4 depicts the Sequence diagram of vulnerability detection using random forest classifier.

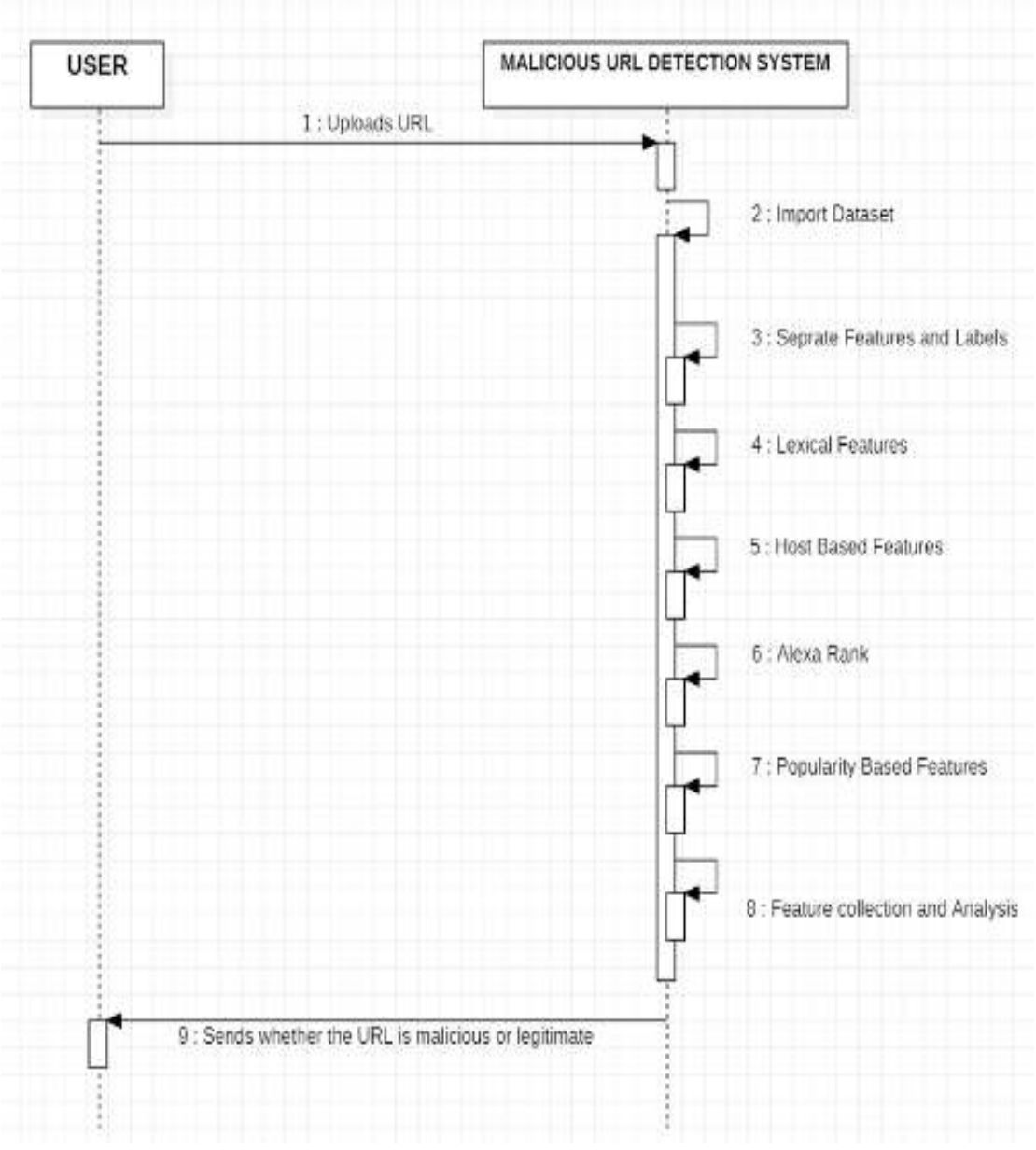



Figure 3.4: Sequence Diagram for vulnerability detection using random forest classifier

### 3.6 ACTIVITY DIAGRAM

The activity diagram describes the flow of activity states.

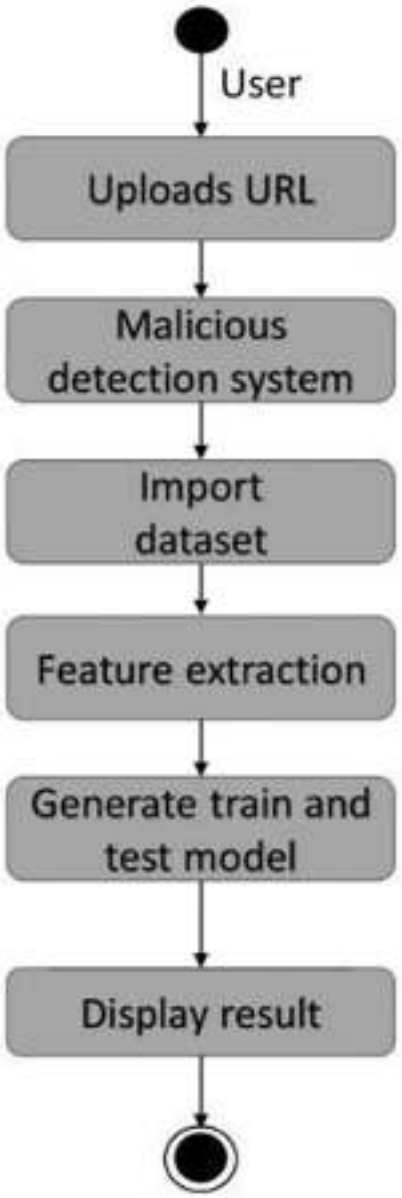



Figure 3.5: Activity Diagram for vulnerability detection using random forest classifier

### 3.7 DATAFLOW DIAGRAM

The data-flow diagram describes the flow of data.

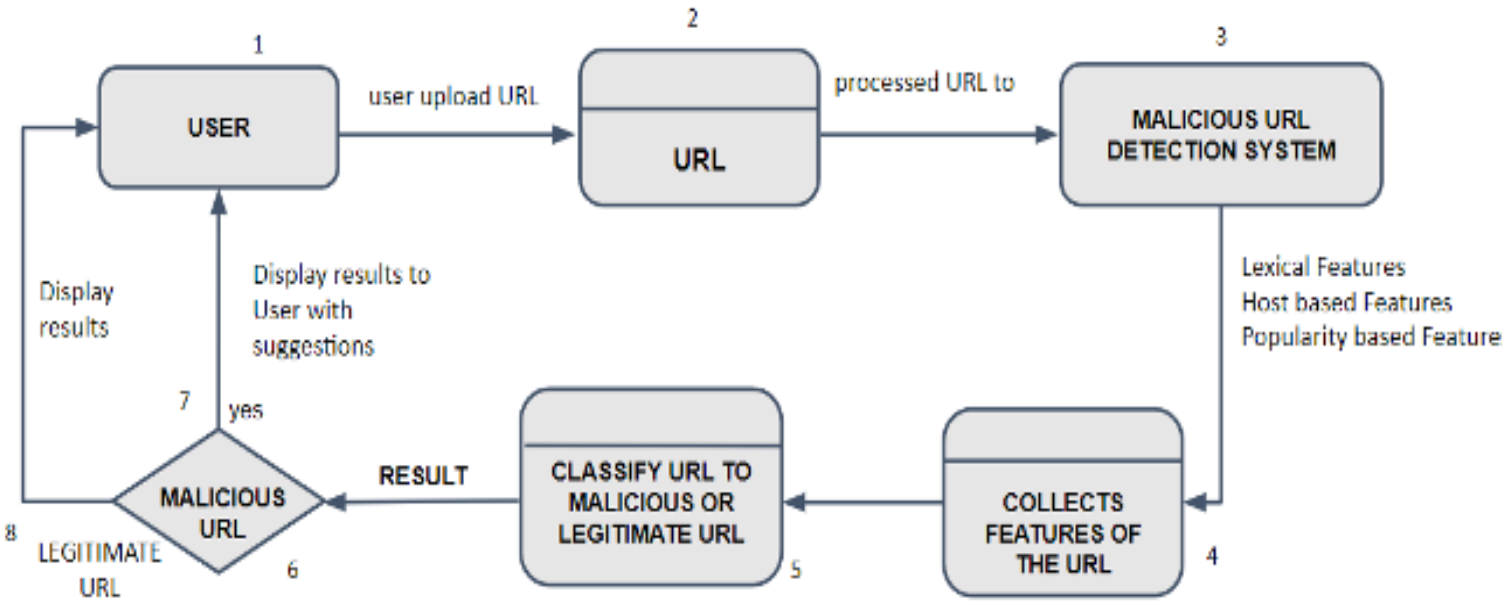



Figure 3.6: Data-flow Diagram for vulnerability detection using random forest classifier

# 4. IMPLEMENTATION

## 4.1 SAMPLE CODE

```
import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import datetime
import time
from dateutil.parser import parse as date_parse

# Calculates number of months
def diff_month(d1, d2):
    return (d1.year - d2.year) * 12 + d1.month - d2.month

# Generate data set by extracting the features from the URL
def generate_data_set(url):

    data_set = []

    # Converts the given URL into standard format
    if not re.match(r"^https?", url):
        url = "http://" + url


    # Stores the response of the given URL
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
    except:
        response = ""
        soup = -999


    # Extracts domain from the given URL
    domain = re.findall(r"://([^/]+)/?", url)[0]
    if re.match(r"^www.",domain):
                domain = domain.replace("www.","")
```

```
    # Requests all the information about the domain
    whois_response = whois.whois(domain)

    rank_checker_response = requests.post("https://www.checkpagerank.net/index.php",
{
        "name": domain
    })

    # Extracts global rank of the website
    try:
                                global_rank = int(re.findall(r"Global Rank: ([0-9]+)",
rank_checker_response.text)[0])
    except:
        global_rank = -1

    # 1.having_IP_Address
    try:
        ipaddress.ip_address(url)
        data_set.append(-1)
    except:
        data_set.append(1)

    # 2.URL_Length
    if len(url) < 54:
        data_set.append(1)
    elif len(url) >= 54 and len(url) <= 75:
        data_set.append(0)
    else:
        data_set.append(-1)

    # 3.Shortining_Service

match=re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|
cli\.gs|'

'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|
'
```

```
'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net',url)
    if match:
        data_set.append(-1)
    else:
        data_set.append(1)

    # 4.having_At_Symbol
    if re.findall("@", url):
        data_set.append(-1)
    else:
        data_set.append(1)

    # 5.double_slash_redirecting
    list=[x.start(0) for x in re.finditer('//', url)]
    if list[len(list)-1]>6:
        data_set.append(-1)
    else:
        data_set.append(1)

    # 6.Prefix_Suffix
    if re.findall(r"https?://[^\-]+-[^\-]+/", url):
        data_set.append(-1)
    else:
        data_set.append(1)

    # 7.having_Sub_Domain
    if len(re.findall("\.", url)) == 1:
        data_set.append(1)
    elif len(re.findall("\.", url)) == 2:
        data_set.append(0)
    else:
        data_set.append(-1)
```

```
# 8.SSLfinal_State
try:
    if response.text:
        data_set.append(1)
except:
    data_set.append(-1)

# 9.Domain_registeration_length
expiration_date = whois_response.expiration_date
registration_length = 0
try:
    expiration_date = min(expiration_date)
    today = time.strftime('%Y-%m-%d')
    today = datetime.strptime(today, '%Y-%m-%d')
    registration_length = abs((expiration_date - today).days)

    if registration_length / 365 <= 1:
        data_set.append(-1)
    else:
        data_set.append(1)
except:
    data_set.append(-1)

# 10.Favicon
if soup == -999:
    data_set.append(-1)
else:
    try:
        for head in soup.find_all('head'):
            for head.link in soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                if url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                    data_set.append(1)
                    raise StopIteration
                else:
                    data_set.append(-1)
                    raise StopIteration
    except StopIteration:
        pass
```

```
    #11. port
    try:
        port = domain.split(":")[1]
        if port:
            data_set.append(-1)
        else:
            data_set.append(1)
    except:
        data_set.append(1)

    #12. HTTPS_token
    if re.findall(r"^https://", url):
        data_set.append(1)
    else:
        data_set.append(-1)
#13. Request_URL
    i = 0
    success = 0
    if soup == -999:
        data_set.append(-1)
    else:
        for img in soup.find_all('img', src= True):
            dots= [x.start(0) for x in re.finditer('\.', img['src'])]
            if url in img['src'] or domain in img['src'] or len(dots)==1:
                success = success + 1
            i=i+1

        for audio in soup.find_all('audio', src= True):
            dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
            if url in audio['src'] or domain in audio['src'] or len(dots)==1:
                success = success + 1
            i=i+1

        for embed in soup.find_all('embed', src= True):
            dots=[x.start(0) for x in re.finditer('\.',embed['src'])]
            if url in embed['src'] or domain in embed['src'] or len(dots)==1:
                success = success + 1
            i=i+1

        for iframe in soup.find_all('iframe', src= True):
```

```
    dots=[x.start(0) for x in re.finditer('\.',iframe['src'])]
    if url in iframe['src'] or domain in iframe['src'] or len(dots)==1
      success = success + 1
    i=i+1

  try:
    percentage = success/float(i) * 100
    if percentage < 22.0 :
      dataset.append(1)
    elif((percentage >= 22.0) and (percentage < 61.0)) :
      data_set.append(0)
    else :
      data_set.append(-1)
```

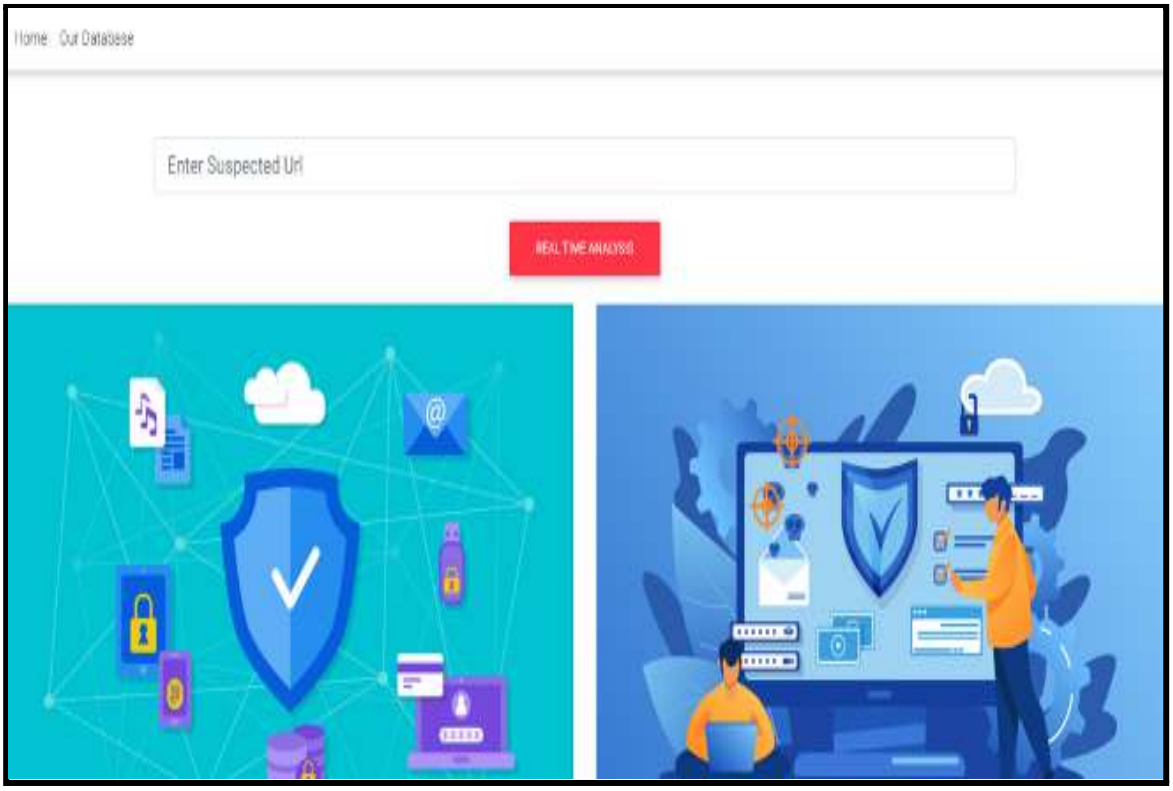
**Drive Link:**

**https://drive.google.com/drive/u/0/folders/1xNlyUfnnXcM6J_SpS5E3gqUWLLGjNPOH**

# 5. RESULTS

## 5.1 HOME PAGE

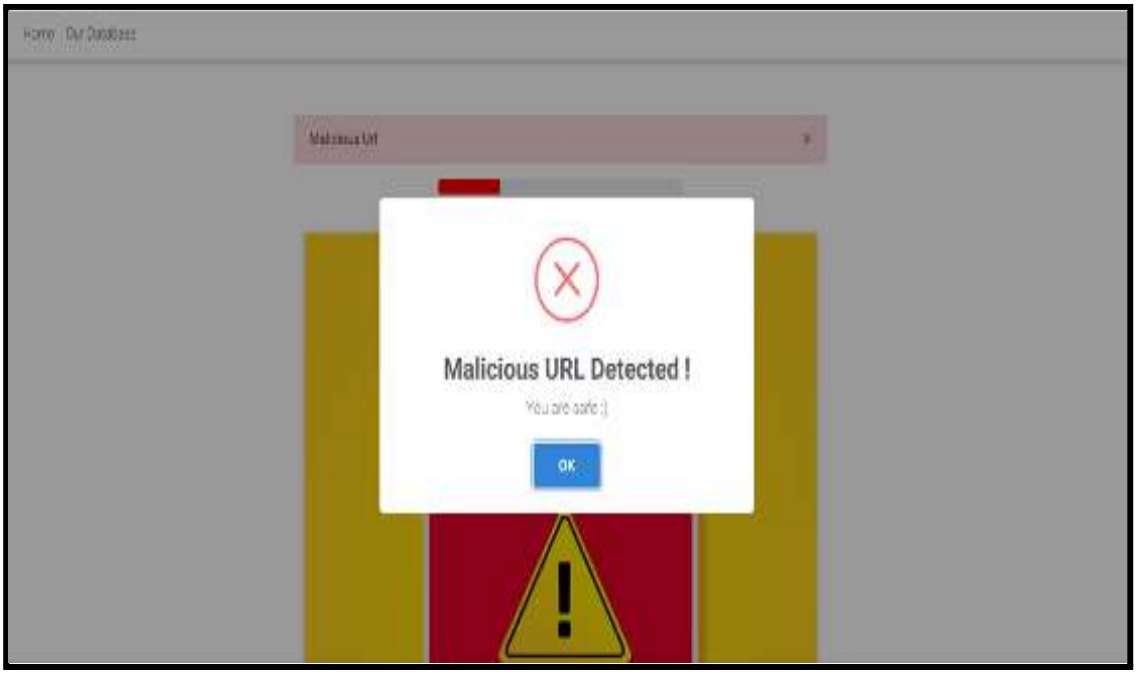
This home page indicates the user uploads the URL to check whether the URL is malicious or legitimate.




Result 5.1: Landing page

## 5.2 MALICIOUS URL

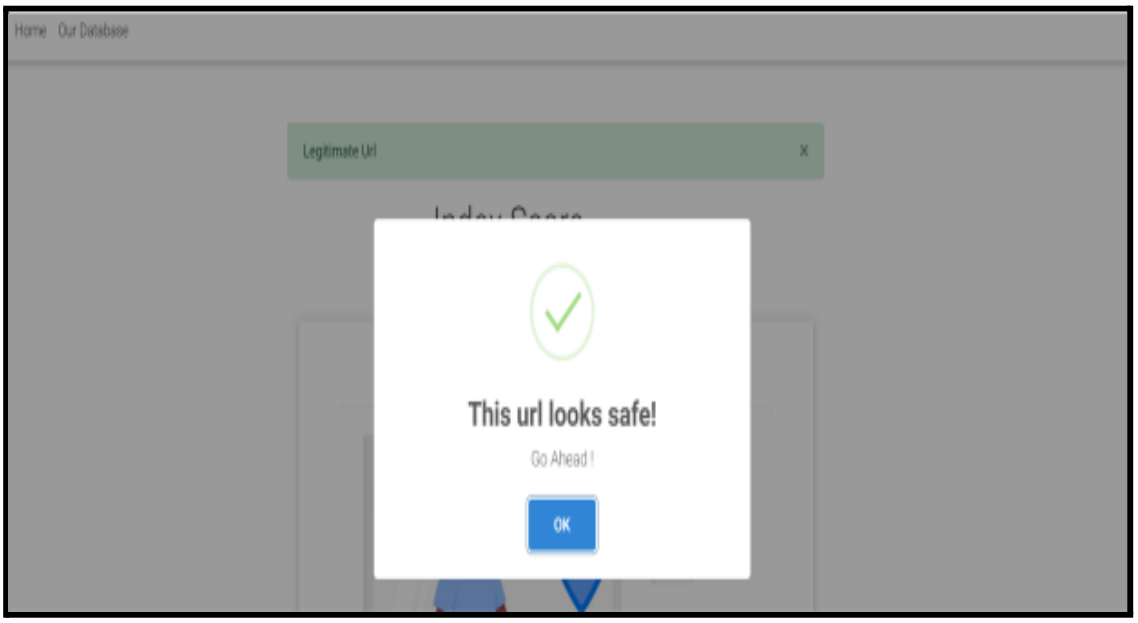
When a user uploads the URL if it is malicious the detection system displays the result.




Result 5.2: Malicious URL Page

## 5.3 LEGITIMATE URL

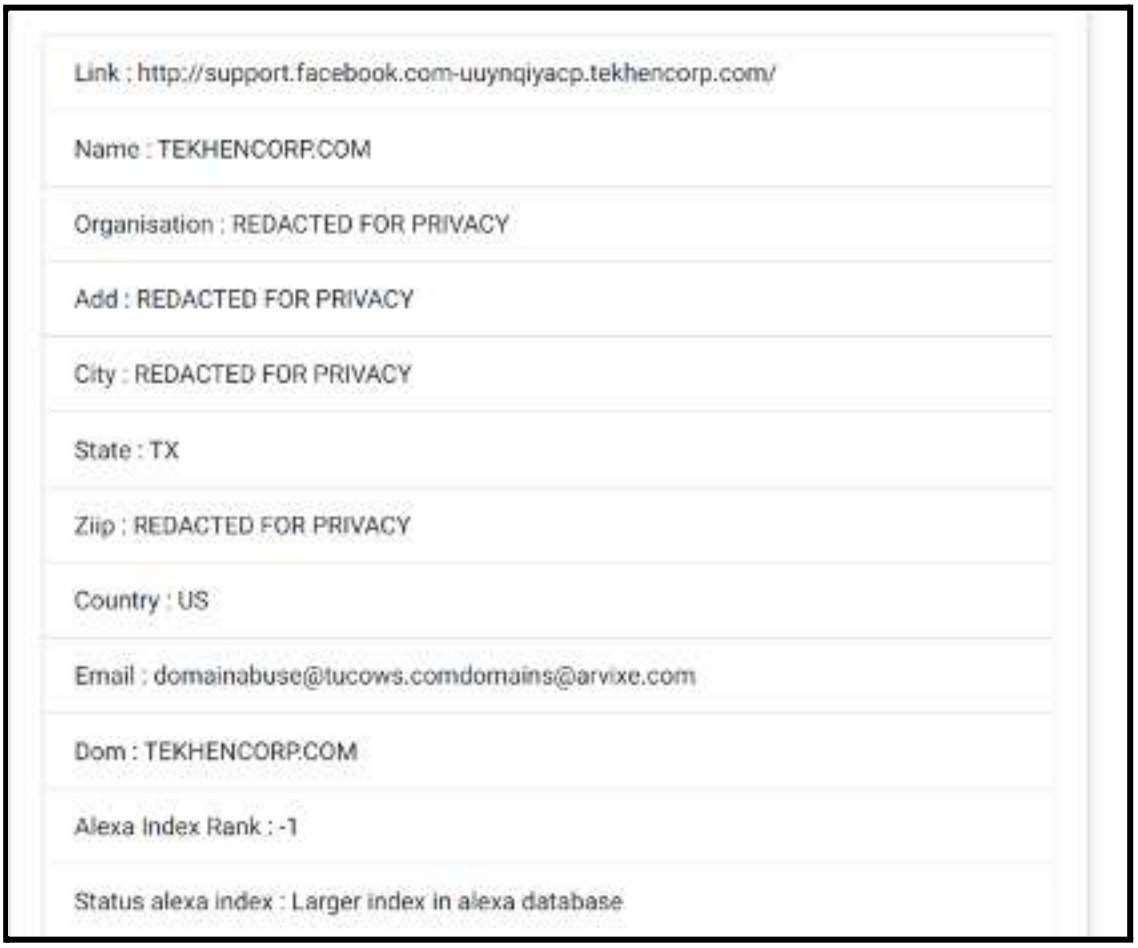
If the URL is not malicious then it displays the url is safe




Result 5.3: Legitimate URL Page

## 5.4 DOMAIN INFORMATION

When the detection system detects the malicious URL and also displays the domain details of the URL.




Result 5.4: Domain Information

## 5.5 HISTORY

Our database contains all the searches that the user has made.

Result 5.5: User History

# 6. TESTING

# 6. TESTING

## 6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

## 6.2 TYPES OF TESTING

### 6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### 6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components are correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

## 6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes

## 6.3 TEST CASES

| Test case ID | Test case name | Purpose | Test Case | Output |
|---|---|---|---|---|
| 1 | Uploading URL | For processing the URL in the detection system | The user uploads the URL and checks whether malicious or not | Uploaded Successfully |
| 2 | Malicious URL | Checking whether the URL is malicious or not | From the feature extraction it checks the URL | Malicious URL detected |
| 3 | Legitimate URL | Checking the URL malicious or not | From the feature extraction it checks the URL | The URL looks safe |
| 4 | Domain details | It displays the URL host details | It displays the host, email, IP address of URL | Deatils of the domain |

# 7. CONCLUSION

# 7. CONCLUSION & FUTURE SCOPE

## 7.1 PROJECT CONCLUSION

The detection model achieves the expected effect in experiments. However, considering that the network traffic in the test environment and the real network are different, and with the development of the Internet, types of malicious URL are more diverse. It is necessary to timely update the model in the actual scenario. Therefore, to better adapt to the requirements of various complex application scenarios, we plan to study how to simplify the detection model's architecture and shorten the training time while keeping the detection performance unchanged in the future.

## 7.2 FUTURE SCOPE

Creating Google-chrome extension so that users can directly interact with the application without any installation process and users can get the results instantly on the same web page on which they are working. Vulnerabilities are rapidly increasing as new technologies are getting evolved so there can be numerous number of loop holes so new algorithms must be implemented for best results.

# 8. BIBLIOGRAPHY

# 8. BIBLIOGRAPHY

## 8.1 REFERENCES

[1] D. Sahoo, C. Liu, S.C.H. Hoi, “Malicious URL Detection using Machine Learning: A Survey”. CoRR, abs/1701.07179, 2017.

[2] M. Khonji, Y. Iraqi, and A. Jones, “Phishing detection: a literature survey,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091–2121, 2013.

[3] M. Cova, C. Kruegel, and G. Vigna, “Detection and analysis of drivebydownload attacks and malicious javascript code,” in Proceedings of the 19th international conference on World wide web. ACM, 2010, pp. 281– 290.

[4] R. Heartfield and G. Loukas, “A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks,” ACM Computing Surveys (CSUR), vol. 48, no. 3, p. 37, 2015.

[5] Internet Security Threat Report (ISTR) 2019–Symantec. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24- 2019-en.pdf [Last accessed 10/2019].

[6] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” in Proceedings of Sixth Conference on Email and Anti-Spam (CEAS), 2009.

[7] C. Seifert, I. Welch, and P. Komisarczuk, “Identification of malicious web pages with static heuristics,” in Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian. IEEE, 2008, pp. 91–96.

[8] S. Sinha, M. Bailey, and F. Jahanian, “Shades of grey: On the effectiveness of reputation-based “blacklists”,” in Malicious and Unwanted Software

[9] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Identifying suspicious urls: an application of large-scale online learning,” in Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 2009, pp. 681–688.

[10] B. Eshete, A. Villafiorita, and K. Weldemariam, “Binspect: Holistic analysis and detection of malicious web pages,” in Security and Privacy in Communication Networks. Springer, 2013, pp. 149–166.

[11] S. Purkait, “Phishing counter measures and their effectiveness– literature review,” Information Management & Computer Security, vol. 20, no. 5, pp. 382–420, 2012.

[12] Y. Tao, “Suspicious url and device detection by log mining,” Ph.D. dissertation, Applied Sciences: School of Computing Science, 2014.

[13] G. Canfora, E. Medvet, F. Mercaldo, and C. A. Visaggio, “Detection of malicious web pages using system calls sequences,” in Availability, Reliability, and Security in Information Systems. Springer, 2014, pp. 226–238.

[14] Leo Breiman.: Random Forests. Machine Learning 45 (1), pp. 5- 32, (2001).

[15] Thomas G. Dietterich. Ensemble Methods in Machine Learning. International Workshop on Multiple Classifier Systems, pp 1-15, Cagliari, Italy, 2000.

[16] Developer Information. https://www.phishtank.com/developer_info.php. [Last accessed 11/2019].

[17] URLhaus Database Dump. https://urlhaus.abuse.ch/downloads/csv/. [Ngày truy nhập 11/2019].

[18] Dataset URL. http://downloads.majestic.com/majestic_million.csv. [Last accessed 10/2019].

[19] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.. In NDSS.

[20] Enrico Blanzieri and Anton Bryl. 2008. A survey of learning-based techniques of email spam filtering. Artificial Intelligence Review (2008).

## 8.2 WEBSITES

[1] https://www.kdnuggets.com/2016/10/machine-learning-detect-malicious-urls

[2] https://www.kaggle.com/datasets/deepsworld/malicous-and-benign-websites

## 8.3 GitHub Link

https://github.com/saikarthiksk/vulnerability-detection-using-random-forest-classifier