Synopsis Report on

WAVD: WEB APPLICATION VULNERABILITY DETECTOR

Submitted in partial fulfillment of the requirements of the degree

of

Bachelor of Engineering

by

ROHAN SHARMA (34)
SHUBHAM YADAV (35)
DHIRAJ MISHRA (36)

Supervisor:

PROF. AKSHAYA PRABHU

Computer Engineering Department

VIVA Institute of Technology

University of Mumbai

2022-2023

CERTIFICATE

This is to certify that the project entitled **"WAVD: WEB APPLICATION VULNERABILITY DETECTOR"** is a bonafide work of **"Rohan Sharma, Shubham Yadav, Dhiraj Mishra"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering in Computer Engineering"**

Prof. Akshaya Prabhu
Guide

Prof. Ashwini Save
Head of Department

Dr. Arun Kumar
Principal

Project Synopsis Approval for B.E.

This project report entitled **WAVD: WEB APPLICATION VULNERABILITY DETECTOR** by **Rohan Sharma, Shubham Yadav and Dhiraj Mishra** is approved for the degree of **Bachelor of Engineering in Computer Engineering.**

Examiners

1.----------

2.----------

Date:

Place:

Abstract

In today's world, Cyber security has become an important leap in the form of jobs, education. But the reality is that only a few are aware of the major web vulnerabilities. Some statistical studies show that small scale industries are directly and indirectly connected to the world of the internet, but they are not aware of the major web vulnerabilities of their web application. Since website hosting has become common nowadays, most of the web applications are prone to attacks and malicious attacks of web applications. Assessing and avoiding these vulnerabilities require deep knowledge of these vulnerabilities. There are numerous online scanners available on the Internet that provide only paid limited service. The tools are made in a way that it can only operate in command line interface or in any programming language. So, it is a difficult task for a normal person to operate the scanners without previous knowledge.

The web applications are the most common interface for security-sensitive information and functionality available. As web applications are sources of sensitive data, they are prone to vast numbers of web-based attacks. The majority of these attacks happen because of vulnerabilities resulting from input validation problems. Although these vulnerabilities are easy to understand and mitigate, many web developers are unaware of these security aspects. Which results in more vulnerable web applications on the Internet. We implemented a system which will scan the web application for the most frequent vulnerabilities in an automated manner. Our system detects flaws in web applications and presents a comprehensive report

Table of Contents

List of Figures

List of Tables

Declaration

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will result in disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

ROHAN SHARMA

SHUBHAM YADAV

DHIRAJ MISHRA

Date:

Chapter 1

Introduction

Web applications have become an integral part of everyday life, but many of these applications are associated with vulnerabilities. In this era, where website hosting has become cheap and easy, the security has failed to keep up. Such vulnerabilities can risk small scale to large scale industries. Exploitation of vulnerability by an unauthorized person demands for quick recovery of these flaws so that reputation of the organization can be recovered. Therefore, vulnerability scanners can be widely used to evaluate the known weakness and vulnerabilities in a website. Many applications are becoming online, but how secure these applications are a matter of concern. Thus, it becomes necessary to find vulnerabilities that may cause severe risk to user's security. Vulnerability assessment means detecting the vulnerabilities before they could be used by an attacker. It is not only performed on a particular application, but it can be run on any platform on which the application is run. This strategy only takes into consideration all the factors that can provide the correct answer for assessment of the vulnerability and security of the system. Therefore, vulnerability scanners are used to scan the network and software application.

We can infer that Security plays an important role in developing websites. Unfortunately, web developers are not aware of these security aspects resulting in more vulnerable websites. Some of the most commonly occurring ones being SQLi, XSS, RFI, Open Redirection and

Phishing. So, we are developing a system that will find these vulnerabilities in given web applications and report them to the user of the system. We are developing a system that will accept the target URL from the user. The system will then crawl the target URL in an Automated way using AI techniques and collect all the connected URLs. Then it will scan all collected URLs and it will test different payloads to exploit the vulnerabilities. Finally, a report will be generated which will contain the detected vulnerabilities and payloads used.

Chapter 2

Literature Survey

The following chapter is a literature survey of the previous research papers and research whichgives detailed information about the previous system or previous methods along with its advantages and disadvantages.

2.1 Survey of the existing system

A survey was done on the existing literature, products, and technologies to find out their shortcomings, to learn about the working flow of architecture, the impact of effective scanner, and research gaps in various case studies or in their systems.

Haibo Chen, et. al [1], proposed a web vulnerability scanner that is designed based on python programming language. They have also adopted Browser/server architecture for realizing. The information collection module and the vulnerability detection module are deployed on the database created by them. The main modules presented are 1) Database and management module 2) Information collection module 3) Vulnerability detection module 4) Tasks and targets management module. The database is established based on 4 SQLite in version 3.24.0. The scanner ran on a machine serving Windows 10 Home, and equipped with 16 GB RAM, i5 CPU in 2.3 GHz. There are 2 detection modules provided by the proposed scanner including comprehensive detection and special detection and it is mainly focused on common web

security vulnerabilities such as SQL injection, XSS, framework vulnerabilities etc. The special vulnerability module is developed based on the pocsuite3

Mr. Kalyan D Bmane, et. al [2], the proposed system has used waterfall model for the project, Demand Gathering and analysis, System Design , Implementation , Testing , Reading system , Maintenance Which researchers on numerous internet vulnerabilities that comes under linguistics uniform resource locator, XSS, RFI, LFI, SQLi, CMDi . So, system mainly concentrates on detection and prevention of web application by using various techniques such as Dynamic Allocation, File Size Verification, Digital Signature and Sanitization of Input

Pranav Gadekar, et. al [3] The system will crawl the target URL in an Automated way using AI techniques and collect all the connected URLs Then it will scan all collected URLs and it will test different payloads to exploit the vulnerabilities. A report will be generated which will contain the detected vulnerabilities and payloads used. Various Scanner used in the system Acunetix Vulnerability Assessment Engine: It is a security testing solution used for both standalone and as a part of complex environments. Burp Suite Web Vulnerability Scanner: It hunt out an honest range of vulnerabilities Qualys Web Application Scanner: It also covers public cloud instances and provides you instant visibility of vulnerabilities like SQLi and XSS. Nessus Vulnerability Scanner: Nessus is the vulnerability assessment solution for security practitioners

Bin Wang, et. al [4] the proposed system uses Web crawler module which can crawl from a URL to any URL associated with it. Core vulnerability detection module enhance the scalability of the system. Scanner interface and overall function integration, the interface of the system is implemented by pyqt4 technology. The interface component is written through the xml file, and the specific components in the relevant xml file are retrieved in the python core file. It also scans multiple target websites at the same time. In this system XSS vulnerability; SQL injection vulnerability and File upload vulnerability.

Binny George, et. al [5] proposed a system which scans SQL injection, cross site scripting and Broken Authentication If a situation arrives where the scanner cannot detect the vulnerability, then the attacker can easily crawl into the system and exploit the data and resources. So Nmap, Nessus, Acunetix, Nikto, Burp Suite are compared. Burp Suite scans SQL Injection, Improper Error, Cross Site, Scripting, Insecure, Cookies, Session Token, URL, Password Auto Enabled.

Vahid Shahrivari, et. al [6] proposed in this paper, a comparative evaluation of different machine learning methods provided on detecting the phishing websites. The machine learning methods studied are Logistic Regression, Decision Tree. Random Forest, AdaBoost, SVM, KNN, Artificial Neural Networks, Gradient Boosting, and XGBoost. They evaluated the accuracy, precision, recall, F1 score, training time, and testing time of these models and they used different methods of feature selection and hyperparameters tuning for getting the best results. XGBoost gives the best accuracy of 98.32% using the phish tank dataset.

Rishikesh Mahajan, el. at [7] in this paper deals with machine learning technology for detection of phishing URLs by extracting and analysing various features of legitimate and phishing URLs. Decision Tree, random forest and Support vector machine algorithms are used to detect phishing websites. The Python program is implemented to extract features from URL like Address bar bashed, Domain Bar Based, HTML and JS based feature. Here Random Forest gives the best accuracy of 97.14% by using alexa.com dataset. Scikit-learn tool has been used to import ML algorithms.

P.S.Sadaphule, et.al[8] proposed System to prevent web application from various malicious attacks of RFI and LFI by using PHP language and CSS, preventing them using Dynamic Allocation, File Size Verification, Digital Signature and Sanitization of Input prevention methods. Since RFI and LFI are very rare attack but can harm the whole system so this project Focus on the RFI and LFI vulnerabilities detection.

Mita Patil,et. al [9] proposed an approach that allows to automatically identify whether above vulnerabilities present in Web application or not. System uses black box approach for the analysis of the targeted application. First to find out all notion of injections. To avoid false negatives Author maintains a state while crawling. By applying clustering algorithm system detect vulnerabilities with assurance of increasing performance.

Table 2.1: Analysis Table

Title	Summary	Advantages	Open Challenges
An Automatic Vulnerability Scanner for Web Application. [1]	The proposed scanner includes comprehensive detection and special detection, and it is mainly focused on common web security vulnerabilities such as SQL injection, XSS, framework vulnerabilities etc. The special vulnerabilities etc. The special vulnerability module is developed based on the pocsuite3	The vulnerability scanning system can improve the efficiency of web application vulnerability	Python, JavaScript
Web Vulnerability Scanner. [2]	The project comes underneath linguistics uniform resource locator. System tends to be studied numerous vulnerabilities like Remote File Inclusion, Locate File Inclusion, SQLI, Cross-Site Scripting.	RFI (Remote File Inclusion) and LFI (Local File Inclusion) are truly vulnerable. Though, these kinds of attacks are rare, but the unauthorized access can harm the whole system.	Python, JavaScript, Burp Suite
Automated Web Application Vulnerability Scanner. [3]	application is having any of these vulnerabilities: SQL Injection and Cross Site Scripting The report will be generated consisting of endpoint	Supports automated and reliable crawling. Optimized use of the number of threads to control the load on the target application.	Python, JavaScript

	affected, payload used, and generalized remediation.		
Research on Web Application Security Vulnerability Scanning Technology. [4]	In this paper, the characteristics of various types of vulnerabilities is analyzed, Such as XSS Detection, SQL Injection Detection and File upload Vulnerability A web application vulnerability detection method is proposed, combining the fuzzy test method to enable the scanning system to automatically detect vulnerabilities.	The automated vulnerability scanning system can improve the efficiency of web application vulnerability scanning compared to traditional manual detection vulnerabilities that are time-consuming, labor-intensive, and inefficient.	Python, JavaScript
Web Application Security Scanner for Prevention and Protection against Vulnerabilities. [5]	Proposed method is a vulnerability scanner which detects the vulnerabilities like SQL injection, cross site scripting, broken authentication, payload, email disclosure. It uses Burp Suite which gives best scanning results.	he above proposed scanner is best suited for beginners who are not aware of the complex steps of scanning Vulnerability scanning identifies the security. vulnerabilities in an organization. The advantage of using vulnerability scanner is that it identifies known security	Burp Suite, fuzzy testing, Python.

		exposures before attackers find them..	
Phishing Detection Using Machine Learning Techniques. [6]	In this paper, a comparative evaluation of different machine learning methods provided on detecting the phishing websites. They used different methods of feature selection for getting the best results.	he main advantage of XGBoost is its fast speed compared to other algorithms, such as ANN and SVM, and its regularization parameter that successfully reduces variance	Python , JavaScript
Detection Using Machine Learning Algorithms. [7]	machine learning technology for detection of phishing URLs by extracting and analyzing various features of legitimate and phishing URLs. Decision Tree, random forest and Support vector machine algorithms are used to detect phishing websites.	XGBoost is its fast speed compared to other algorithms, such as ANN and SVM, and its regularization parameter that successfully reduces variance.	Forest classifier gives best accuracy with lowest false negative rate than other two classifiers
Prevention of Website Attack Based on Remote File Inclusion. [8]	RFI and LFI are very rare attack but can harm the whole system so this project Focus on the RFI and LFI vulnerabilities detection.	identifies Remote File Inclusion and Local File Inclusion which are rare vulnerabilities.	CSS and PHP.
Design of Efficient Web Vulnerability Scanner. [9]	proposed clustering approach to efficiently detect the SQL Injection,	Automatically identify whether above vulnerabilities	Python

	Xpath Injection and Cross Site Scripting attacks. The objective is to improve and its evolution hist detection efficiency of vulnerability scanner maintaining low false positive and false negative rate.	is present in web applications of no. .	

2.2 Research gap

There are tools that detects vulnerabilities, but they are not open source and also expensive. 10 Most of the scanners detects few vulnerabilities generally the most common SQL Injection and XSS injection. One of the most common attacks the is phishing is not detected in most Vulnerability systems.

2.3 Problem statement and Objective

There is unavailability of automated scanners for detecting vulnerabilities in web applications. And some scanners which are present are highly expensive. This leads to defacement, hijacking, stealing data from the server. This creates a security problem for all businesses as well as government people. So, making an affordable scanner is important. Scanners first crawl the web pages of a particular domain and scan each URL using different payloads and find out if a URL is vulnerable or not. The aim here is to develop WAVD (Web App Vulnerability Detector), a tool to scan & test URLs for certain vulnerabilities & security issues by simply inspecting the corresponding client-side website. The overall system would include a virtual server with modules for detecting the different vulnerabilities, along with a proxy server, to direct requests from a browser to the virtual server first while visiting a website. The proxy could warn the user before redirecting to the website if some vulnerabilities are found during the scan done by our virtual server.

We intend to identify & assess the following classes of vulnerabilities that a website may possess:

• Absence of Valid TLS Certificates

• Cross-Site Scripting (XSS)

• Potential Phishing Attempts

• SQL Injection

• Remote File Inclusion & Local File Inclusion

2.4 Scope

We are going to scan rare vulnerabilities which affects the whole system. We have added phishing detection using machine learning with best accuracy 98.32% using XGradient Boost. We are going to add rare vulnerabilities like RFI and LFI which can harm whole 11 system if got unauthorized access. SQLi, XSS, File Upload Vulnerability and TLS Certificate Validation vulnerabilities will be added in the system.

Chapter 3

Proposed system

This chapter gives an overview of the Proposed system.

3.1 Algorithm

The algorithm for the proposed system is as follows:

Step 1: Start

Step 2: Configure Brower

Step 3: Start Web Brower

Step 4: Start Intercept Proxy

Step 5: Enter URL

Step 6: Send URL to restful API component

Step 7: Restful API will call services

Step 8: Servers will perform security check result

Step 9: Restful API fetches security check result

Step 10: Proxy will fetch results

Step 11: Result will be formatted in HTML

Step 12: It will return the result on the web browser

Step 13: Show results

Step 14: END

3.2 Details of System

3.2.1 Software Requirements

1. Proxy Server: NodeJS
2. Virtual Server: Python & Fast API
3. Visual Studio Code.
4. WAVD Dashboard: HTML, CSS, JS & Bootstrap Framework
5. Brower: Google Chrome
6. Operation System: Window & Linux
7. Jupyter notebook

3.2.1 Hardware Requirements

1.Processor: intel i5 (7th gen).
2. Ram: 8GB +.
3. Graphic Card: 2GB.
4. Storage: 120GB

3.3 Design Details

In this section flow diagrams and block diagrams of the system are explained.

3.3.1 Flow Chart Diagram

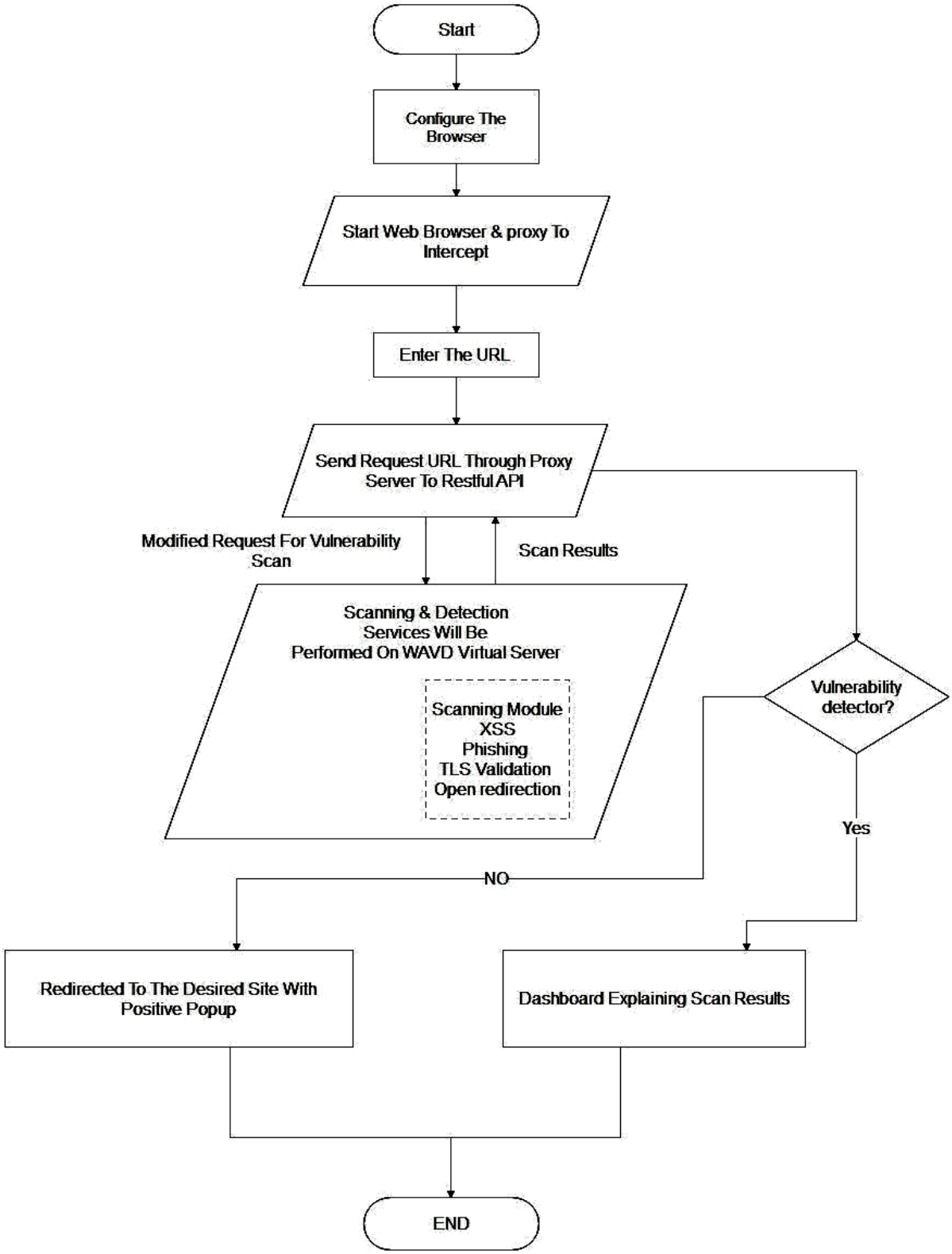


Fig 3.3.1 WAVD flow diagram

Figure 3.3.1 represents a flow diagram for the proposed system. A flow diagram is a collective term for a diagram representing a flow or set of dynamic relationships in a system. Flow diagrams are used to structure and order a complex system, or to reveal the underlying structure of the elements and their interaction.

3.3.2 Block Diagram

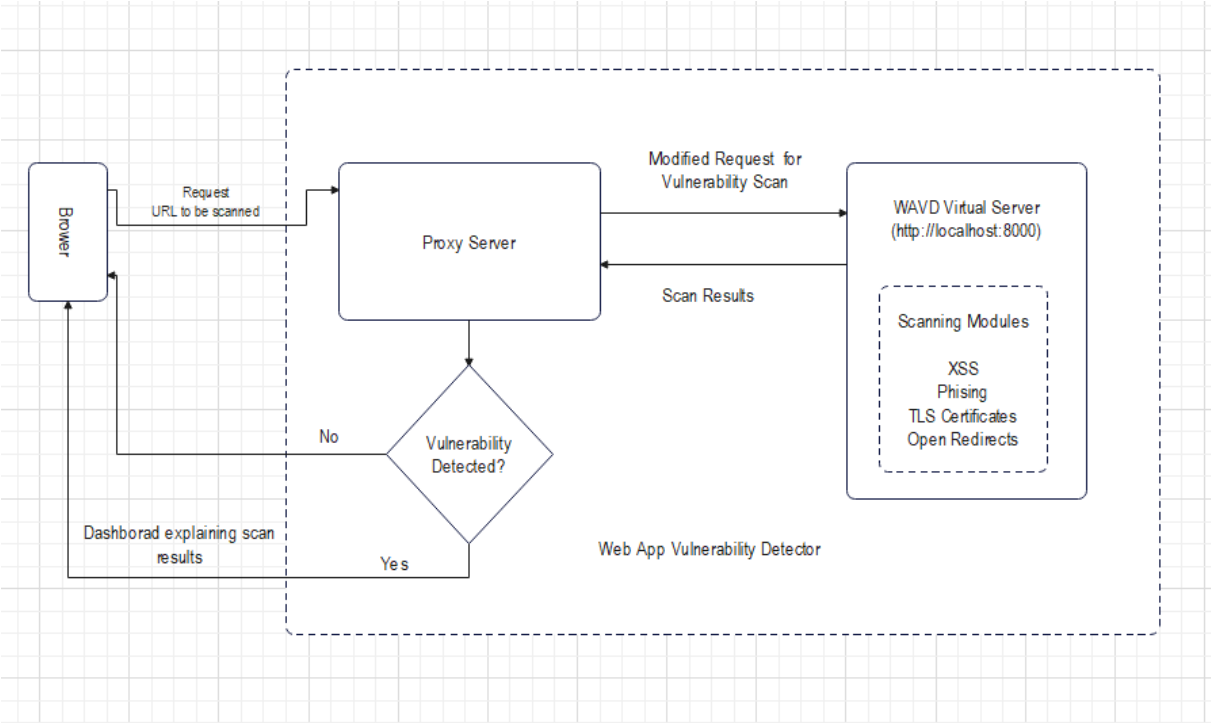


Fig 3.3.2 WAVD block diagram

In Figure 3.3.2 Block diagram for the complete system of WAVD is demonstrated. The application starts with a dashboard where user can enter the URL then it is connected to Proxy Server. The proxy passes the URL to virtual server where the models are shown each models are different for different vulnerabilities.

3.3 System Architecture

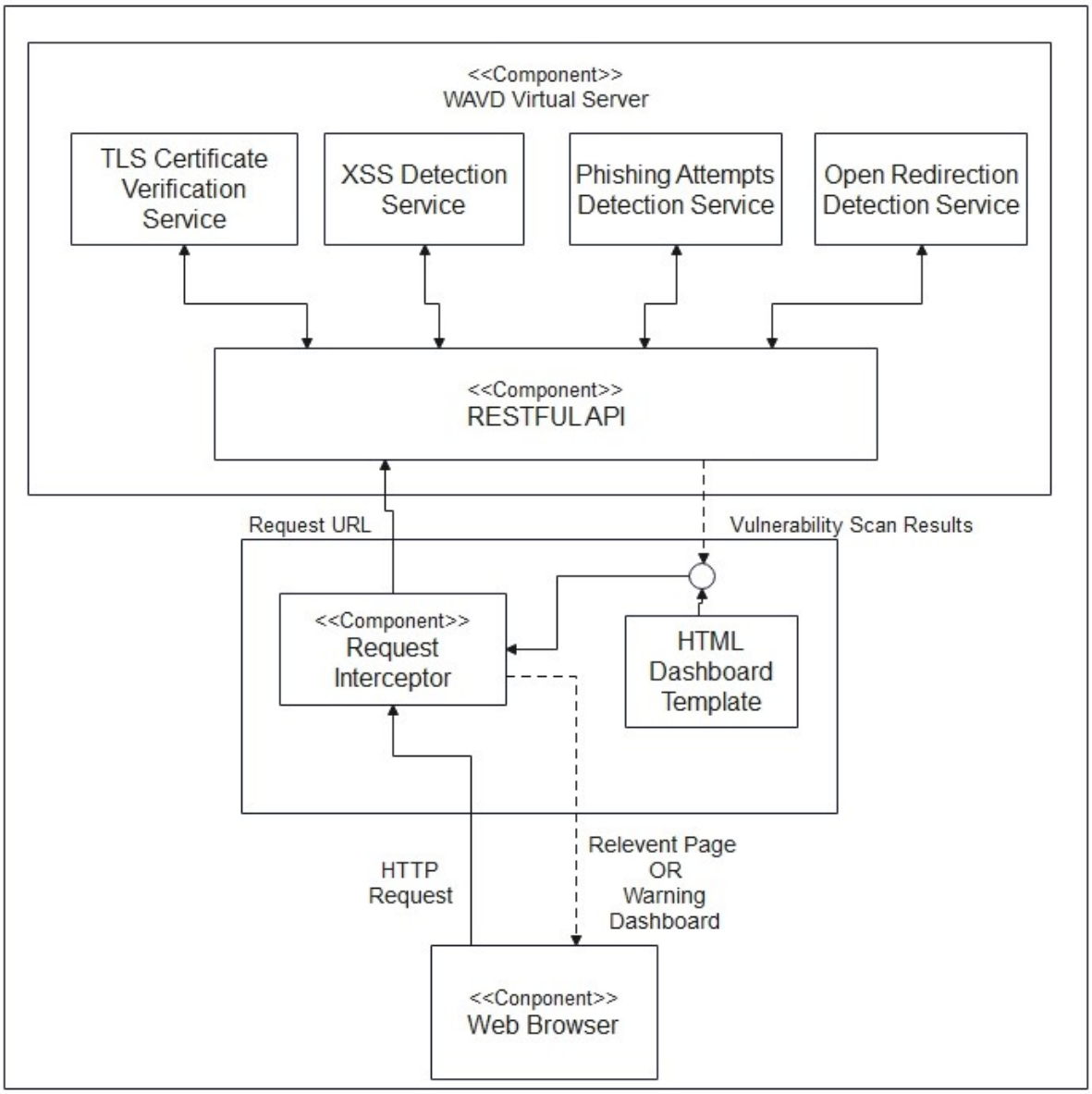


Fig 3.3.3 WAVD System Architecture

In Figure 3.3.2 System architecture for the complete system of WAVD is demonstrated. The application starts with a dashboard where user can enter the URL then it is connected to Proxy Server. The proxy passes the URL to virtual server where the models are shown each models are different for different vulnerabilities.

3.4 Methodology

The proposed system is based on the concept of making an overall development and improving the system to check the vulnerability. In order to develop such a system that should be different from the rest of the existing apps an analysis of existing systems was done. We need to develop Proxy Server for intercepting the URL and redirecting it to our virtual server. To develop this Server, we need to learn how to inject relevant information from JSON objects into DOM elements of an HTML template. Familiarize with node-http-proxy for developing the proxy server. After intercepting the request, URL is being send to the virtual server where all the modules are present and the URL gets served to all different module simultaneously and module performs assessment on the given URL. Each module performs different task based on their detection service. There are total six module in the virtual server and to develop virtual server we need to familiarize with FastAPI & Heroku for developing & deploying the virtual server. We have to set up the server & integrate the various microservices such as TLS Certificate Verification Service where we have to gain background knowledge about TLS certificates, the security they offer & ways to detect their absence. Also, familiarize with the working of available CLI python tools like check-tls-certs & use a similar approach to develop the service for checking the existence & validity of a website's TLS certificate. After this microservice, the system is going to check XSS, SQL Injection, Open Redirection, RFI & LFI by implementing the payloads for particular detection service & inject them into DOM elements that are susceptible for the particular attacks. After checking all this microservices, the system is going to check for the phishing attempts by delving deeper into phishing attacks & study the paper Intelligent rule-based phishing website rule-based phishing websites classification to understand the relevant features to identify phishing websites. It is going to use UCI phishing websites datasets to train & test some traditional ML classifiers (DT, SVM & RF) and identify the most important features indicating phishing & build functions to extract them.

3.5 UML Diagram

3.5.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

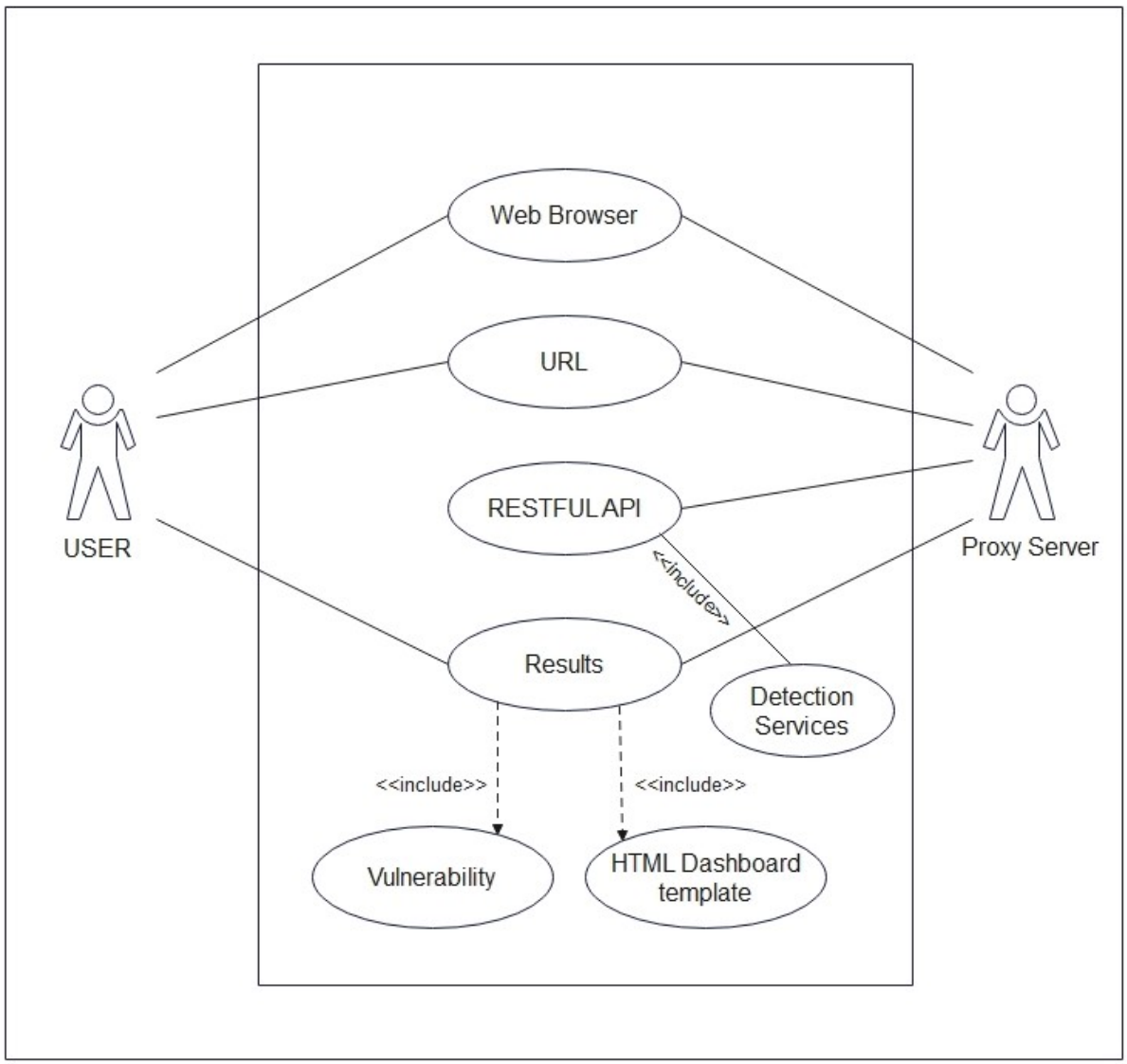


Fig 3.5.1 Use case diagram

Chapter 4

Implementation plan for next semester

This chapter covers the activity table and Gantt chart for the proposed system. The main focus is to create the Proxy Server & Virtual Server, as it will be the one to handle the intercepting the URL and redirecting it to the microservices. tasks for all. Integration of all the modules is another part. After all, this will test all the modules together and will fix identified bugs.

Table 4.1: Planned Activity

Sr.No	Task Name	Start	Finish	Days
1	Proxy Server	28-11-2022	10-12-2022	10
2	Virtual Server	11-12-2022	24-12-2022	10
3	SQL Injection Detection	25-12-2022	02-01-2023	05
4	XSS Detection Service	03-01-2023	10-01-2023	09
5	Open Redirection Detection	11-01-2023	15-01-2023	13
6	TLS Validation Certificate	15-01-2023	18-01-2023	04
7	RFI & FLI	18-01-2023	21-01-2023	03
8	Developing GUI for Dashboard	21-01-2023	25-01-2023	04
9	Integrating all the modules	26-01-2023	28-01-2023	03

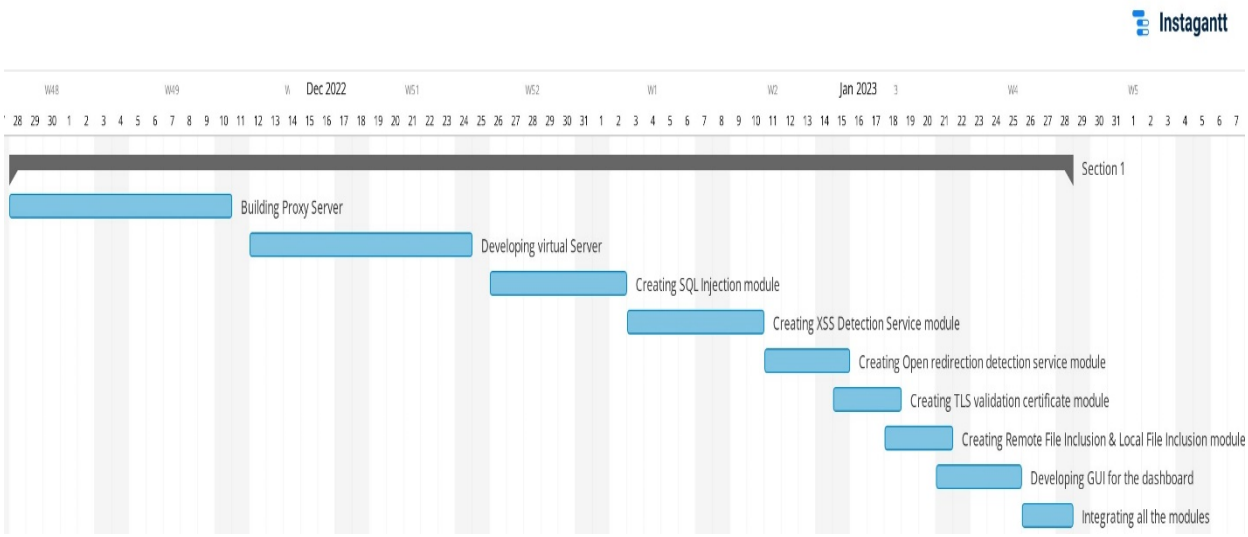
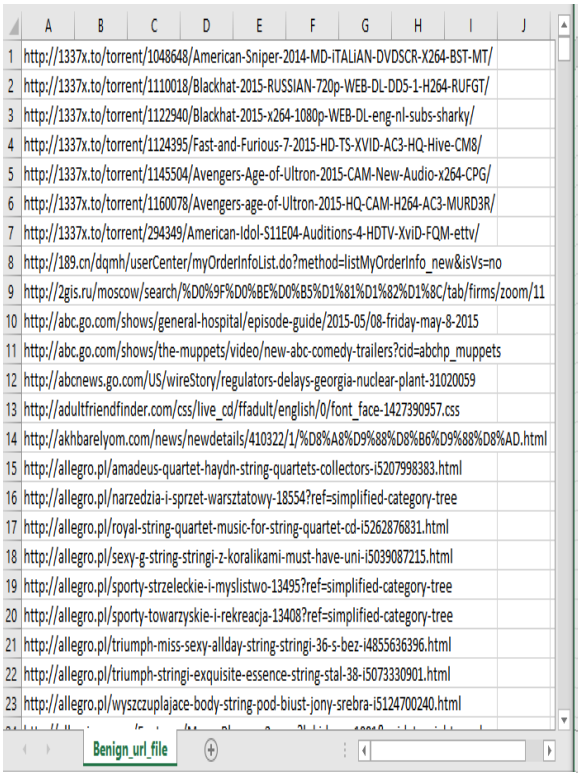


Fig 4.1 Gantt chart

Chapter 5

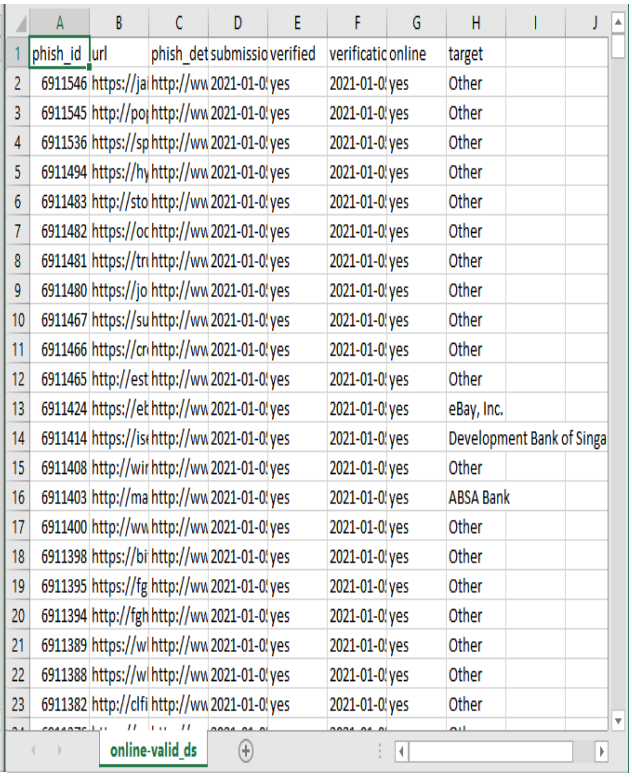
Partial Implementation and Results

This chapter provides the partial implementation and screenshots of the result. Currently, the basic flow design and implementation for the phishing module is completed. Here, some classification algorithm is used to check which algorithm gives the best accuracy on the basis of legitimate and phishing datasets. In this, we implemented algorithm by extracting features from the given URL which will be fetch later on through the proxy server. After comparing the algorithm, best algorithm is being chosen for the further process of detecting the potential phishing attack.



	A
1	http://1337x.to/torrent/1048648/American-Sniper-2014-MD-iTALiAN-DVDSCR-X264-BST-MT/
2	http://1337x.to/torrent/1110018/Blackhat-2015-RUSSIAN-720p-WEB-DL-DD5-1-H264-RUFGT/
3	http://1337x.to/torrent/1122940/Blackhat-2015-x264-1080p-WEB-DL-eng-nl-subs-sharky/
4	http://1337x.to/torrent/1124395/Fast-and-Furious-7-2015-HD-TS-XVID-AC3-HQ-Hive-CM8/
5	http://1337x.to/torrent/1145504/Avengers-Age-of-Ultron-2015-CAM-New-Audio-x264-CPG/
6	http://1337x.to/torrent/1160078/Avengers-age-of-Ultron-2015-HQ-CAM-H264-AC3-MURD3R/
7	http://1337x.to/torrent/294349/American-Idol-S11E04-Auditions-4-HDTV-XviD-FQM-ettv/
8	http://189.cn/dqmh/userCenter/myOrderInfoList.do?method=listMyOrderInfo_new&isVs=no
9	http://2gis.ru/moscow/search/%D0%9F%D0%BE%D0%B5%D1%81%D1%82%D1%8C/tab/firms/zoom/11
10	http://abc.go.com/shows/general-hospital/episode-guide/2015-05/08-friday-may-8-2015
11	http://abc.go.com/shows/the-muppets/video/new-abc-comedy-trailers?cid=abchp_muppets
12	http://abcnews.go.com/US/wireStory/regulators-delays-georgia-nuclear-plant-31020059
13	http://adultfriendfinder.com/css/live_cd/ffadult/english/0/font_face-1427390957.css
14	http://akhbarelyom.com/news/newdetails/410322/1/%D8%A8%D9%88%D8%B6%D9%88%D8%AD.html
15	http://allegro.pl/amadeus-quartet-haydn-string-quartets-collectors-i5207998383.html
16	http://allegro.pl/narzedzia-i-sprzet-warsztatowy-18554?ref=simplified-category-tree
17	http://allegro.pl/royal-string-quartet-music-for-string-quartet-cd-i5262876831.html
18	http://allegro.pl/sexy-g-string-stringi-z-koralikami-must-have-uni-i5039087215.html
19	http://allegro.pl/sporty-strzeleckie-i-myslistwo-13495?ref=simplified-category-tree
20	http://allegro.pl/sporty-towarzyskie-i-rekreacja-13408?ref=simplified-category-tree
21	http://allegro.pl/triumph-miss-sexy-allday-string-stringi-36-s-bez-i4855636396.html
22	http://allegro.pl/triumph-stringi-exquisite-essence-string-stal-38-i5073330901.html
23	http://allegro.pl/wyszczuplajace-body-string-pod-biust-jony-srebra-i5124700240.html

Benign_url_file



	A	B	C	D	E	F	G	H
1	phish_id	url	phish_det	submissio	verified	verificatic	online	target
2	6911546	https://ja	http://ww	2021-01-0	yes	2021-01-0	yes	Other
3	6911545	http://po	http://ww	2021-01-0	yes	2021-01-0	yes	Other
4	6911536	https://sp	http://ww	2021-01-0	yes	2021-01-0	yes	Other
5	6911494	https://hy	http://ww	2021-01-0	yes	2021-01-0	yes	Other
6	6911483	http://sto	http://ww	2021-01-0	yes	2021-01-0	yes	Other
7	6911482	https://oc	http://ww	2021-01-0	yes	2021-01-0	yes	Other
8	6911481	https://tr	http://ww	2021-01-0	yes	2021-01-0	yes	Other
9	6911480	https://jo	http://ww	2021-01-0	yes	2021-01-0	yes	Other
10	6911467	https://su	http://ww	2021-01-0	yes	2021-01-0	yes	Other
11	6911466	https://cr	http://ww	2021-01-0	yes	2021-01-0	yes	Other
12	6911465	http://est	http://ww	2021-01-0	yes	2021-01-0	yes	Other
13	6911424	https://el	http://ww	2021-01-0	yes	2021-01-0	yes	eBay, Inc.
14	6911414	https://is	http://ww	2021-01-0	yes	2021-01-0	yes	Development Bank of Singa
15	6911408	http://wir	http://ww	2021-01-0	yes	2021-01-0	yes	Other
16	6911403	http://ma	http://ww	2021-01-0	yes	2021-01-0	yes	ABSA Bank
17	6911400	http://ww	http://ww	2021-01-0	yes	2021-01-0	yes	Other
18	6911398	https://bi	http://ww	2021-01-0	yes	2021-01-0	yes	Other
19	6911395	https://fg	http://ww	2021-01-0	yes	2021-01-0	yes	Other
20	6911394	http://fgh	http://ww	2021-01-0	yes	2021-01-0	yes	Other
21	6911389	https://wi	http://ww	2021-01-0	yes	2021-01-0	yes	Other
22	6911388	https://wi	http://ww	2021-01-0	yes	2021-01-0	yes	Other
23	6911382	http://clfi	http://ww	2021-01-0	yes	2021-01-0	yes	Other

online-valid_ds

Fig 5.1 Dataset of Phishing URLs

Fig 5.2 Dataset of Legitimate URLs

Figure 5.1 In this figure, the dataset of phishing URLs as been shown and Figure 5.2 shows the dataset of legitimate URLs which will be further depicted for the training and testing purposes.

```
In [27]:
# 13.Survival  time of domain: The difference between termination time and creation time (Domain_Age)
def domainAge(domain_name):
  creation_date = domain_name.creation_date
  expiration_date = domain_name.expiration_date
  if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
    try:
      creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if ((expiration_date is None) or (creation_date is None)):
      return 1
  elif ((type(expiration_date) is list) or (type(creation_date) is list)):
      return 1
  else:
      oiration_date - creation_date).days)

      age =
    else:
      age = 0
  return age
```

Fig 5.3 Age of Domain

Figure 5.3 depicts the Age of Domain, in this feature can be extracted from WHOIS database. Most phishing websites lives for short periods of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time. If age of a domain > 12 months, the value of this feature is 1 (phishing) else 0 (legitimate).

```
In [30]:
# 15. IFrame Redirection (iFrame)
def iframe(response):
  if response == "":
      return 1
  else:
      if re.findall(r"[<iframe>|<frameBorder>]", response.text):
          return 0
      else:
          return 1
```

Fig 5.4 Frame Redirection

Figure 5.4 shows the Frame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without “frameBorder”. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

```
In [12]:
# importing required packages for this section
from urllib.parse import urlparse,urlencode
import ipaddress
import re
```

Fig 5.5 Machine Learning Models & Training

Figure 5.5 represent many features can be extracted that can be considered as address bar based features like domain of URL, IP address inn URL, “@” symbol in address, “//” redirection in URL, “http/https” in Domain name, Using URL shortening Services “TinyURL”.

.

```
In [13]:
# 1.Domain of the URL (Domain)
def getDomain(url):
  domain = urlparse(url).netloc
  if re.match(r"^www.",domain):
        domain = domain.replace("www.","")
  return domain
```

```
In [26]:
# 12.Web traffic (Web_Traffic)
def web_traffic(url):
  try:
    #Filling the whitespaces in the URL if any
    url = urllib.parse.quote(url)
    rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(), "xml").find(
        "REACH")['RANK']
    rank = int(rank)
  except TypeError:
        return 1
  if rank <100000:
    return 1
  else:
    return 0
```

Fig 5.6 Domain of the URL

Figure 5.6 Here, we are just extracting the domain present in the URL. This feature doesn’t have much significance in the training. May even be dropped while training the mode

In [40]:
```
#Function to extract features
# There are 17 features extracted from the dataset
def featureExtractions(url):

  features = []
  #Address bar based features (9)
  features.append(getDomain(url))
  features.append(havingIP(url))
  features.append(haveAtSign(url))
  features.append(getLength(url))
  features.append(getDepth(url))
  features.append(redirection(url))
  features.append(httpDomain(url))
  features.append(prefixSuffix(url))
  features.append(tinyURL(url))


  #Domain based features (4)
  dns = 0
  try:
    domain_name = whois.whois(urlparse(url).netloc)
  except:
    dns = 1

  features.append(dns)
  features.append(web_traffic(url))
  features.append(1 if dns == 1 else domainAge(domain_name))
  features.append(1 if dns == 1 else domainEnd(domain_name))

  # HTML & Javascript based features (4)
```

Fig 5.7 Computing URL features

Figure 5.7 Here, created a list and a function that calls the other functions and stores all the features of the URL in the list. We will extract the features of each URL and append to this list.

.

In [118]:
```
#importing packages
from sklearn.metrics import accuracy_score
```

In [119]:
```
# Creating holders to store the model performance results
ML_Model = []
acc_train = []
acc_test = []

#function to call for storing the results
def storeResults(model, a,b):
  ML_Model.append(model)
  acc_train.append(round(a, 3))
  acc_test.append(round(b, 3))
```

In [149]:
```
#Sorting the dataframe on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

Out[149]:

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.866	0.864
2	Multilayer Perceptrons	0.865	0.864
0	Decision Tree	0.814	0.812
1	Random Forest	0.818	0.811
5	SVM	0.804	0.794
4	AutoEncoder	0.161	0.177

Fig 5.8 Best Algorithm

Figure 5.8 illustrates that XGBoost algorithm gives the best accuracy and this classification module is going to use for the detecting the phishing attack.

Chapter 6

Conclusion

The proposed system that will crawl the entire web application, scan different types of vulnerabilities, and generate a report specifying an overview of the detected vulnerabilities. The proposed system will provide vulnerabilities of the requested URL where the user will get the results through proxy server which will be provided by virtual server where the vulnerabilities are scanned and detected.

References:

[1] Haibo Chen, Junzuo Chen, Jinfu Chen, Shang Yin Yiming Wu, Japing Xu,"An Automatic Vulnerability Scanner for Web Applications" IEEE, 2020.

[2] R. Brenn Mr.Kalyan D Bamane, Vaibhav Gaikwad, Nikhil Ahire, Kunal Sambhe, Chetan Jagtap, "Web Vulnerability Scanner" International Research Journal of Engineering and Technology (IRJET)Vol 7, Issue,03 March 2020.

[3] Pranav Gadekar,Samruddhi Kulkarni, Shalaka Kulkarni "Shruti More, "Automated Web Application Vulnerability Scanner"Vol 8, Issue 8, August 2021.

[4] Bin Wang, Lu Liu, Feng Li Jianye Zhang, Tao Chen Zhenwan Zou, "Research on Web Application Security Vulnerability Scanning Technology" IEEE,2019.

[5] Binny George', Jenu Maria Scaria, Jobin B', Praseetha VM, "Web Application Security Scanner for Prevention and Protection against Vulnerabilities" International Research Journal of Engineering and Technology (IRJET) ,Vol 8, Issue, 05 March 2020

[6] Mohammad Izadi, Vahid Shahrivar, Mohammad Mahdi Darabi,"Phishing Website Detection using Machine Learning Algorithms" International Journal of Computer Applications,20 September 2020.

[7] Rishikesh Mahajan, Irfan siddavatam, "Phishing Website Detection using Machine Learning Algorithms" International Journal of Computer Applications Volume 181-25, October 2018.

[8] P.S.Sadaphule, Priyanka Kamble,Sanika Mehre,Utkarsha Dhande,Rashmi Savant" Prevention of Website Attack Based on Remote File Inclusion-A survey" International Journal of Advance Engineering and Research Development", 2017.

[9] C. Anley, "Advanced SQL Injection in SQL Server Applications", White paper, Next Generation Security Software Ltd., 2020.

[10] W. G. Halfond, J. Viegas and A. Orso, "A Classification of SQL Injection Attacks and

Countermeasures," College of Computing Georgia Institute of Technology IEEE, 2020.

Acknowledgment

We would like to express a deep sense of gratitude towards our guide
Prof. Akshaya Prabhu, Computer Engineering Department for her constant encouragement and valuable suggestions. The work that we are able to present is possible because of her timely guidance.
We would like to pay gratitude to the panel of examiners for time, and effort they put into evaluating our work and their valuable suggestions from time to time.
We would like to thank the Project Head of the Computer Engineering Department, Prof. Janhavi Sangoi for her support and coordination.
We would like to thank Head of the Computer Engineering Department, Prof. Ashwini Save for her support and coordination.
We are also grateful to the teaching and non-teaching staff of the Computer Engineering Department who lend their helping hands in providing continuous support.

Rohan Sharma

Shubham Yadav

Dhiraj Mishra