Dissertation 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. Sunita Naik Dr. Arun Kumar

In Charge HOD Principal

# Project Dissertation 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**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.**  **No.** | **Topics** | | | **Page No.** |
|  |  |  | **Abstract** | **i** |
|  |  |  | **List of Figures** | **iii** |
|  |  |  | **List of Tables** | **iv** |
| **1.** |  |  | **Introduction** | 1 |
| **2.** |  |  | **Literature Survey** | 3 |
|  | 2.1 |  | Survey Existing System | 3 |
|  | 2.2 |  | Research gap | 9 |
|  | 2.3 |  | Problem Statement and Objective | 9 |
|  | 2.4 |  | Scope | 10 |
| **3.** |  |  | **Proposed System & Implementation** | 11 |
|  | 3.1 |  | Analysis/Framework/Algorithm | 11 |
|  | 3.2 |  | Details of Hardware and Software | 11 |
|  |  | 3.2.1 | Software Requirements | 11 |
|  |  | 3.2.2 | Hardware Requirements | 11 |
|  | 3.3 |  | Design Details | 13 |
|  | 3.4 |  | Methodology | 16 |
|  | 3.5 |  | UML Diagram | 17 |
|  |  | 3.5.1 | Use case Diagram | 17 |
| **4.** |  |  | **Results and Analysis** | 19 |
| **5.** |  |  | **Conclusion** | 25 |
|  |  |  | **References** | 26 |
|  |  |  | **Acknowledgment** |  |

**List of Figures**

| **Figure No.** | **Name of figure** | **Page.  No.** |
| --- | --- | --- |
| 3.1 | WAVD flow diagram | 13 |
| 3.2 | WAVD Block Diagram | 14 |
| 3.3 | System Architecture | 15 |
| 3.4 | Use Case Diagram | 17 |
| 4.1 | Dataset of Phishing URLs | 21 |
| 4.2 | Dataset of Legitimate URLs | 21 |
| 4.3 | Age of Domain | 22 |
| 4.4 | Frame Redirection | 22 |
| 4.5 | Machine Learning Models & Training | 23 |
| 4.6 | Domain of the URL | 23 |
| 4.7 | Computing URL features | 24 |
| 4.8 | Best Algorithm | 24 |

## List of Tables

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of Table** | **Page. No.** |
| 2.1 | Analysis Table | 6 |
| 4.1 | Planned Activity | 18 |

## 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 developed 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 which gives 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 scanners, 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 realization. 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 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 Scanners 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 hunts 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 a Web crawler module which can crawl from a URL to any URL associated with it. Core vulnerability detection module enhances 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 analyzing 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 features. 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 attacks but can harm the whole system so this project focuses on RFI and LFI vulnerabilities detection.

Mita Patil,et. al [9] proposed an approach that allows to automatically identify whether above vulnerabilities present in Web applications or not. System uses black box approach for the analysis of the targeted application. First to find out all the notions 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, 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 affected, payload used, and generalized remediation. | Supports automated and reliable crawling. Optimized use of the number of threads to control the load on the target application. | Python, JavaScript |
| 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. | The 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 scanners is that it identifies known security exposures before attackers find them.. | Burp Suite, fuzzy testing, Python. |
| 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 attacks but can harm the whole system so this project focuses on RFI and LFI vulnerability 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, 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. | Automatically identify whether above vulnerabilities is present in web applications of no.  . | Python |

**2.2 Research** **gap**

There are tools that detect vulnerabilities, but they are not open source and also expensive. 10 Most of the scanners detect few vulnerabilities, generally the most common SQL Injection and XSS injection. One of the most common attacks is phishing which 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 affect 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 the whole system if you have unauthorized access. SQLi, XSS, File Upload Vulnerability and TLS Certificate Validation vulnerabilities will be added in the system.

**Chapter 3**

This chapter gives an overview of the Proposed system.

## Algorithm

The algorithm for the proposed system is as follows:

Step 1: Start

Step 2: Configure Browser

Step 3: Start Web Browser

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

## Details of System

* + 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
    2. **Hardware Requirements**

1.Processor: intel i5 (7th gen).

2. Ram: 8GB+.

3. Graphic Card: 2GB.

4. Storage: 120GB

## Design Details

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

* + 1. **Flow Chart Diagram**

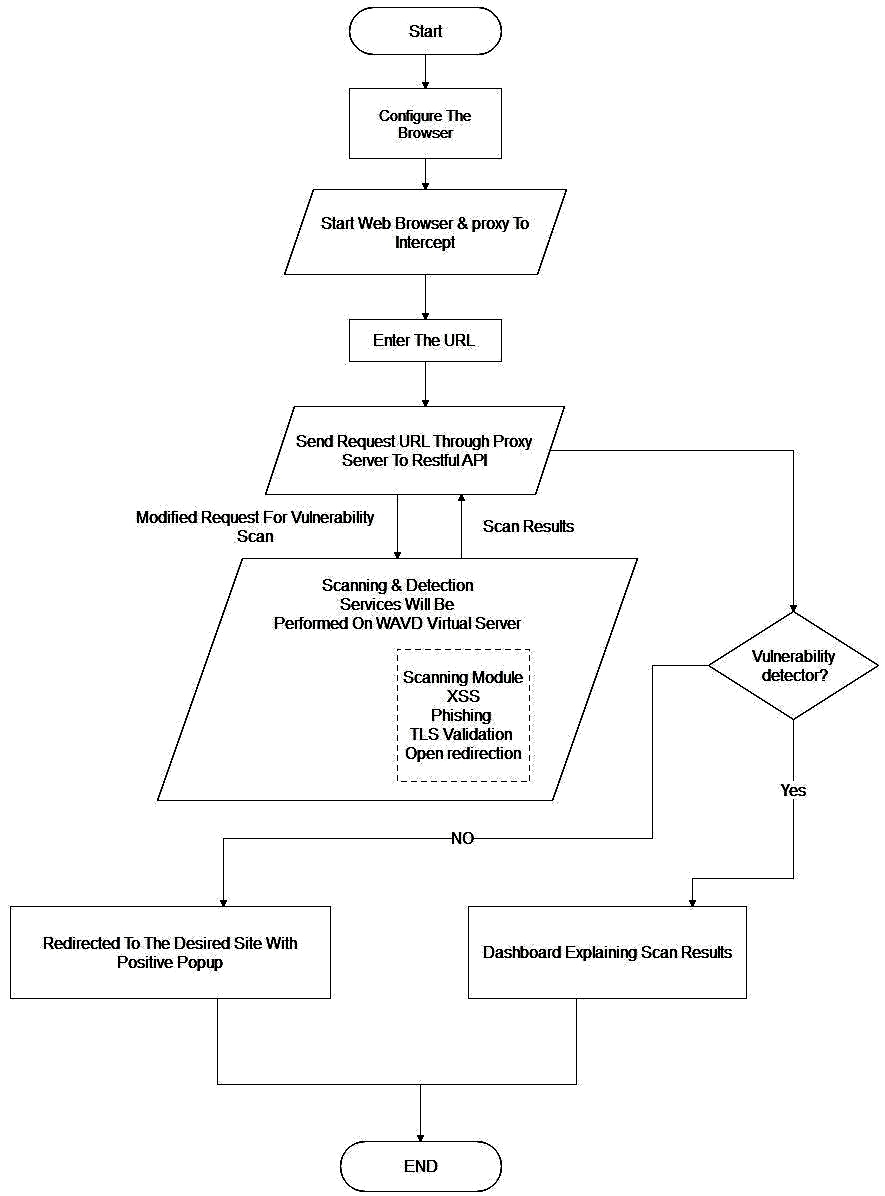
****

Figure 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**

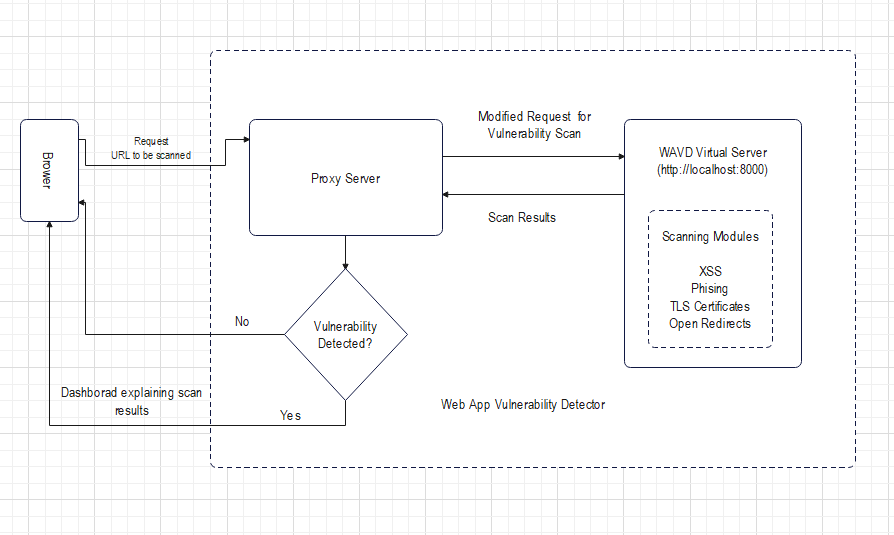


Figure 3.3.2 WAVD block diagram

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

**3.3 System Architecture**

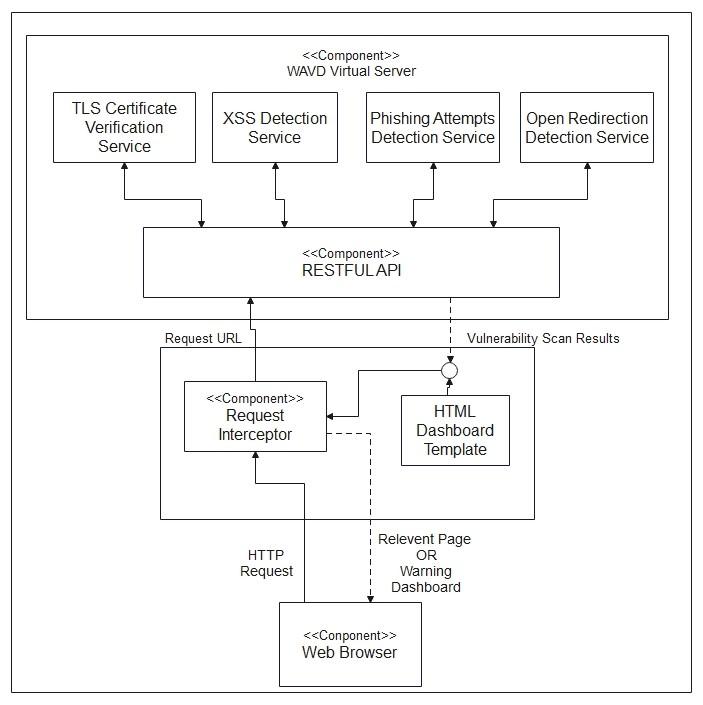
****

Figure 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 the user can enter the URL then it is connected to Proxy Server. The proxy passes the URL to the virtual server where the models are shown each model is 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, the URL is being sent to the virtual server where all the modules are present and the URL gets served to all different modules simultaneously and the module performs assessment on the given URL. Each module performs different tasks based on their detection service. There are a total of six modules in the virtual server and to develop the 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 services & inject them into DOM elements that are susceptible for the particular attacks. After checking all these 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.

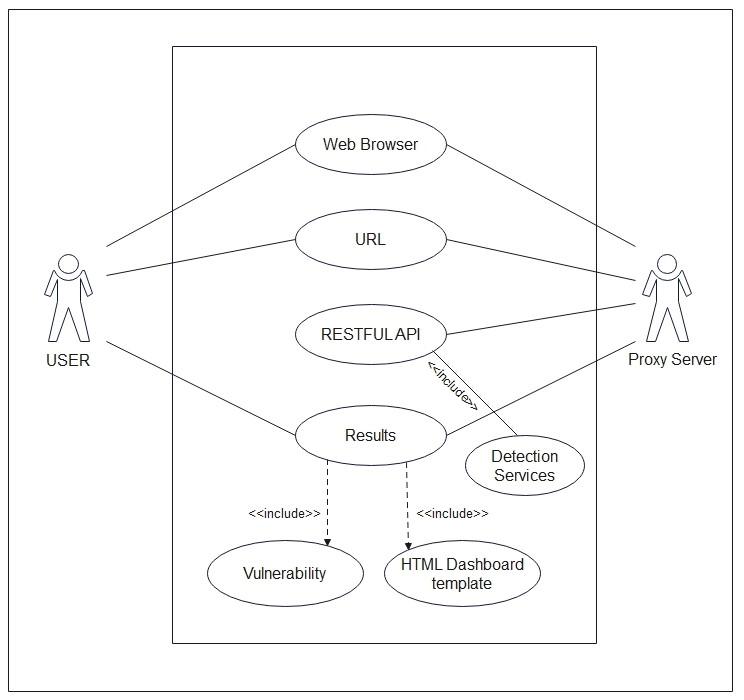


Fig 3.5.1 Use case diagram

**Chapter 4**

**Results and analysis**

This chapter covers the results and analysis of the system. The main focus is to create the Proxy Server & Virtual Server, as it will be the one to handle 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.

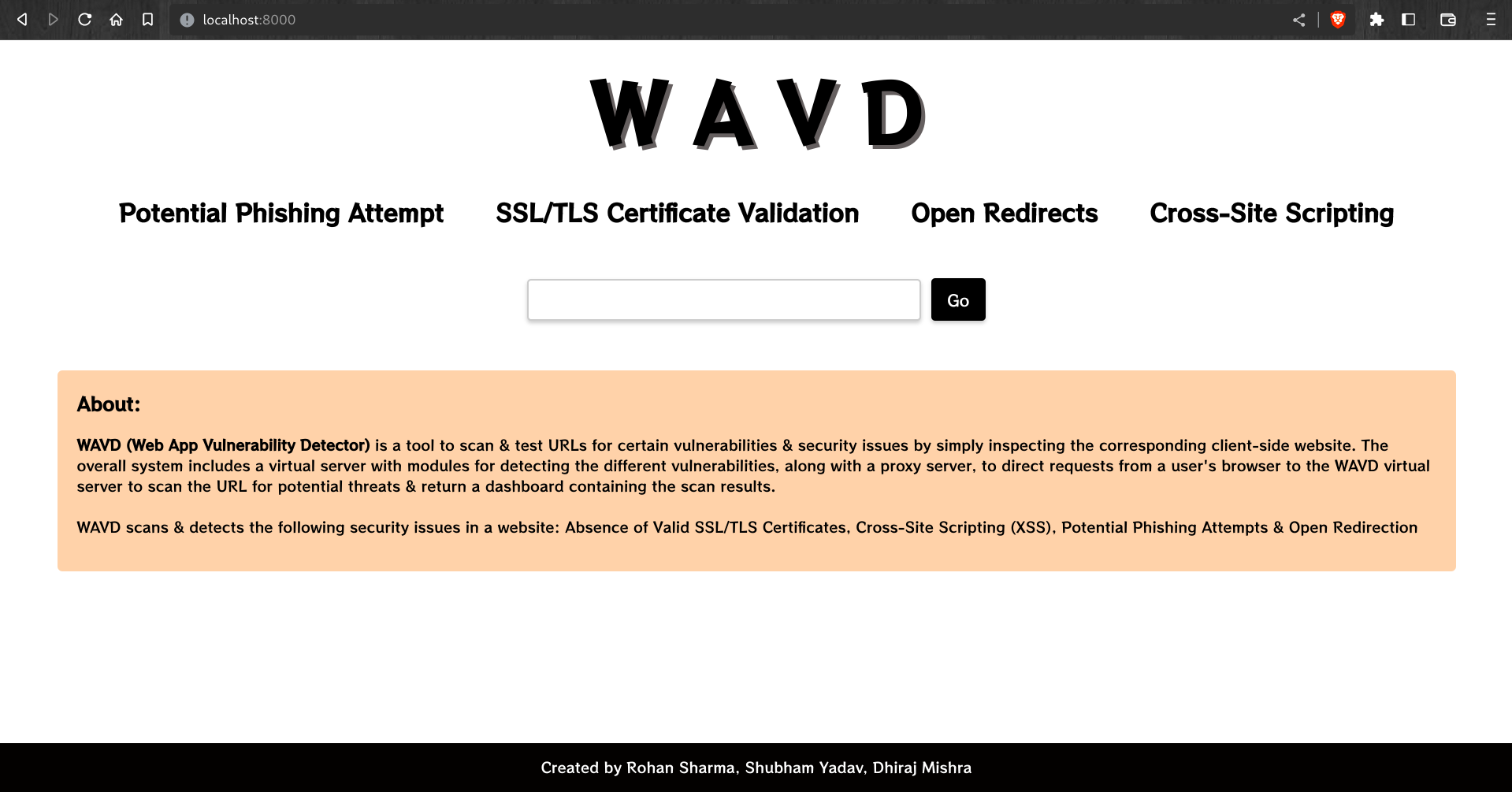


Figure 4.1 Dashboard

This is the homepage dashboard that consists of a central input field where the user can enter a URL to scan for vulnerabilities. Once the URL is submitted, the dashboard initiates a scan of the website and displays the results in several panels.

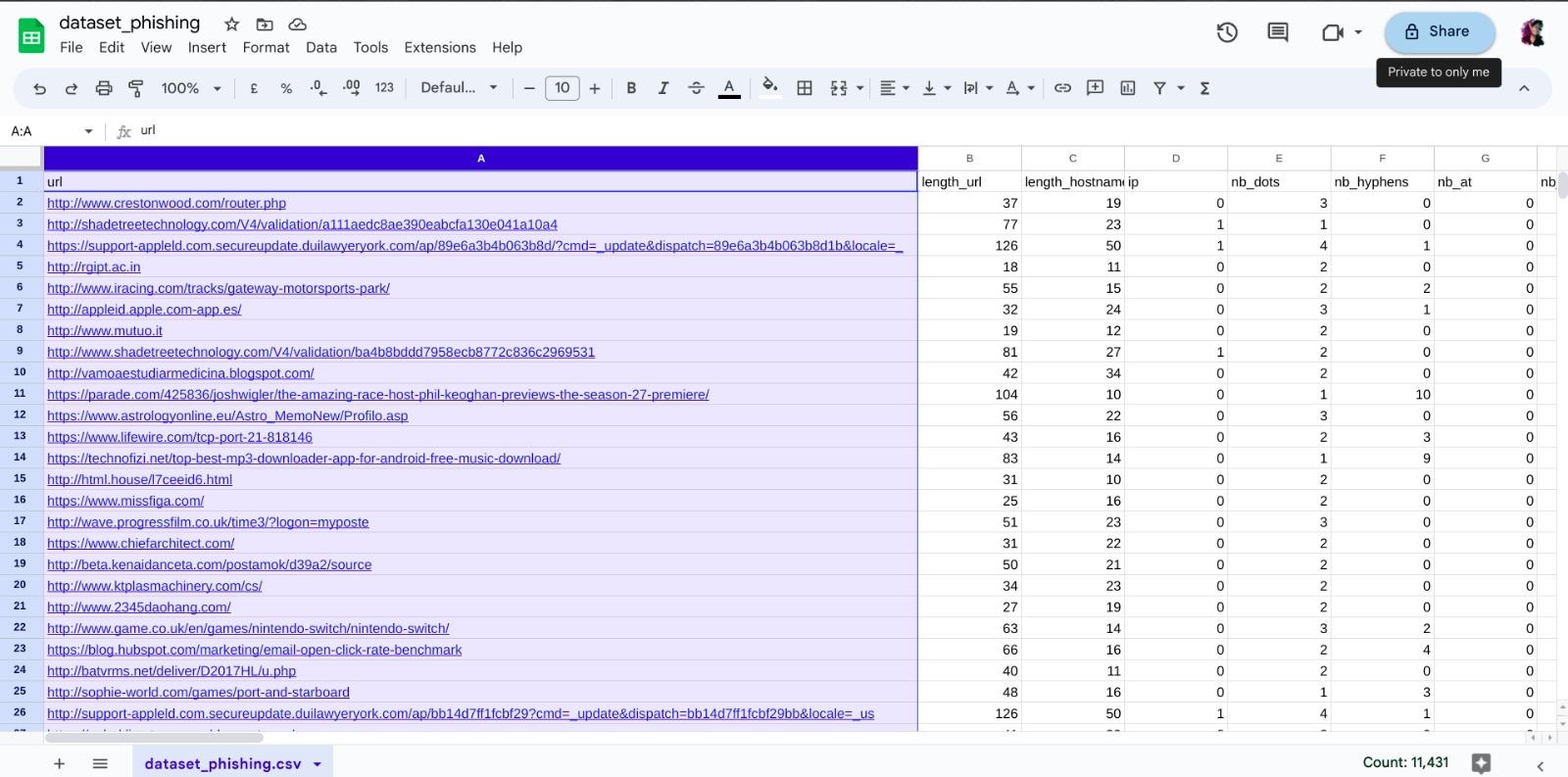


Figure 4.2 Phishing Dataset

In this figure, the dataset of phishing URLs & legitimate URLs has been shown which is depicted for the training and testing purposes for the phishing detection module

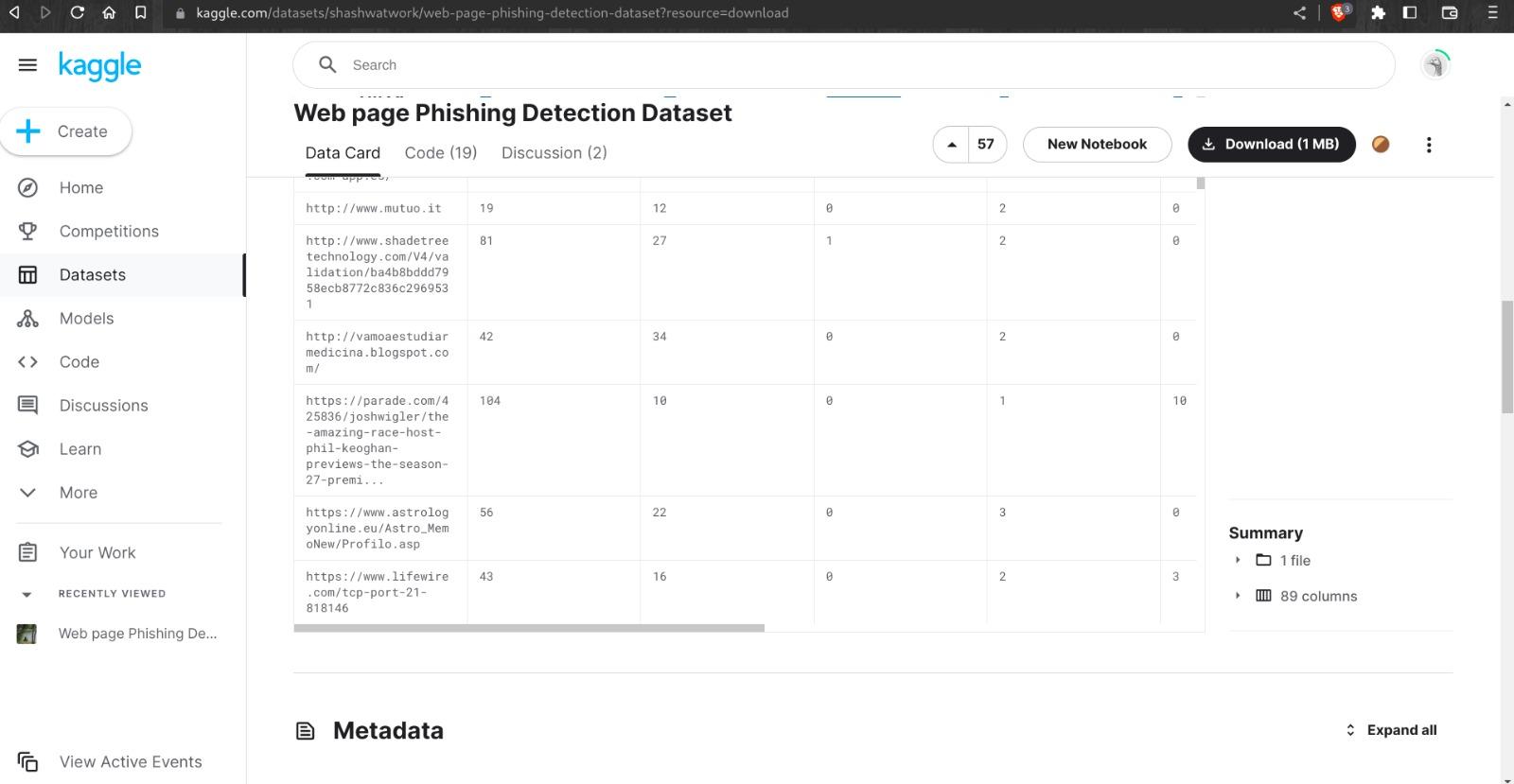


Figure 4.3 Phishing Dataset and its Attributes

This figure depicts the dataset of the phishing URLs which is further used for scanning purposes of the phishing detection module

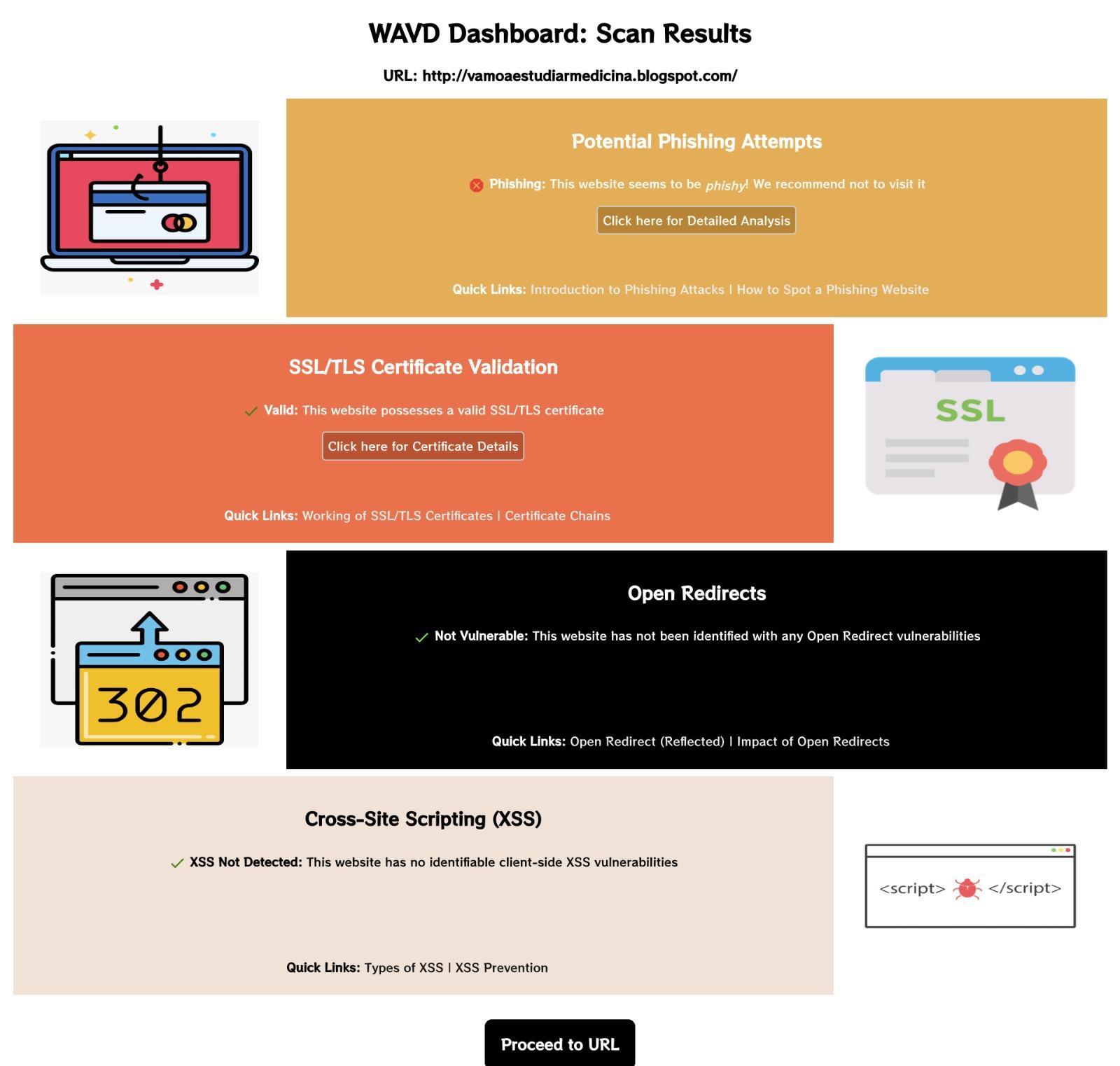


Figure 4.4 Potential Phishing Detected

The first panel shows an overview of the scan results, including the number of vulnerabilities detected, their severity level, and a summary of the affected pages and components of the phishing URL. Here, the panel shows that the provided URL could be the potential phishing URL.



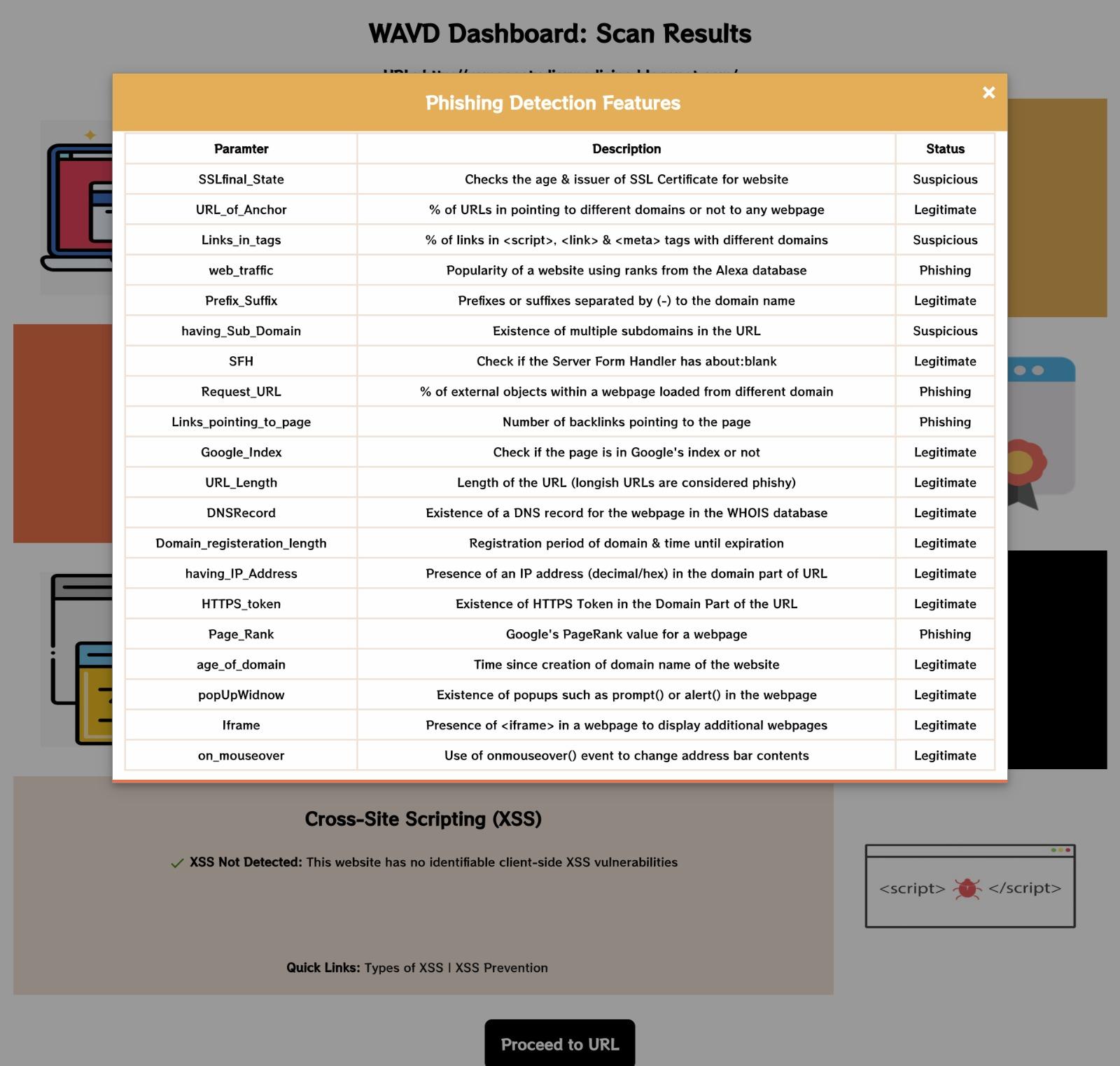


Figure 4.5 Phishing Detection Features

This panel provides a detailed report of phishing vulnerability including its parameter, description and status. It shows which parameter of URL is suspicious and which parameter is legitimate.

.

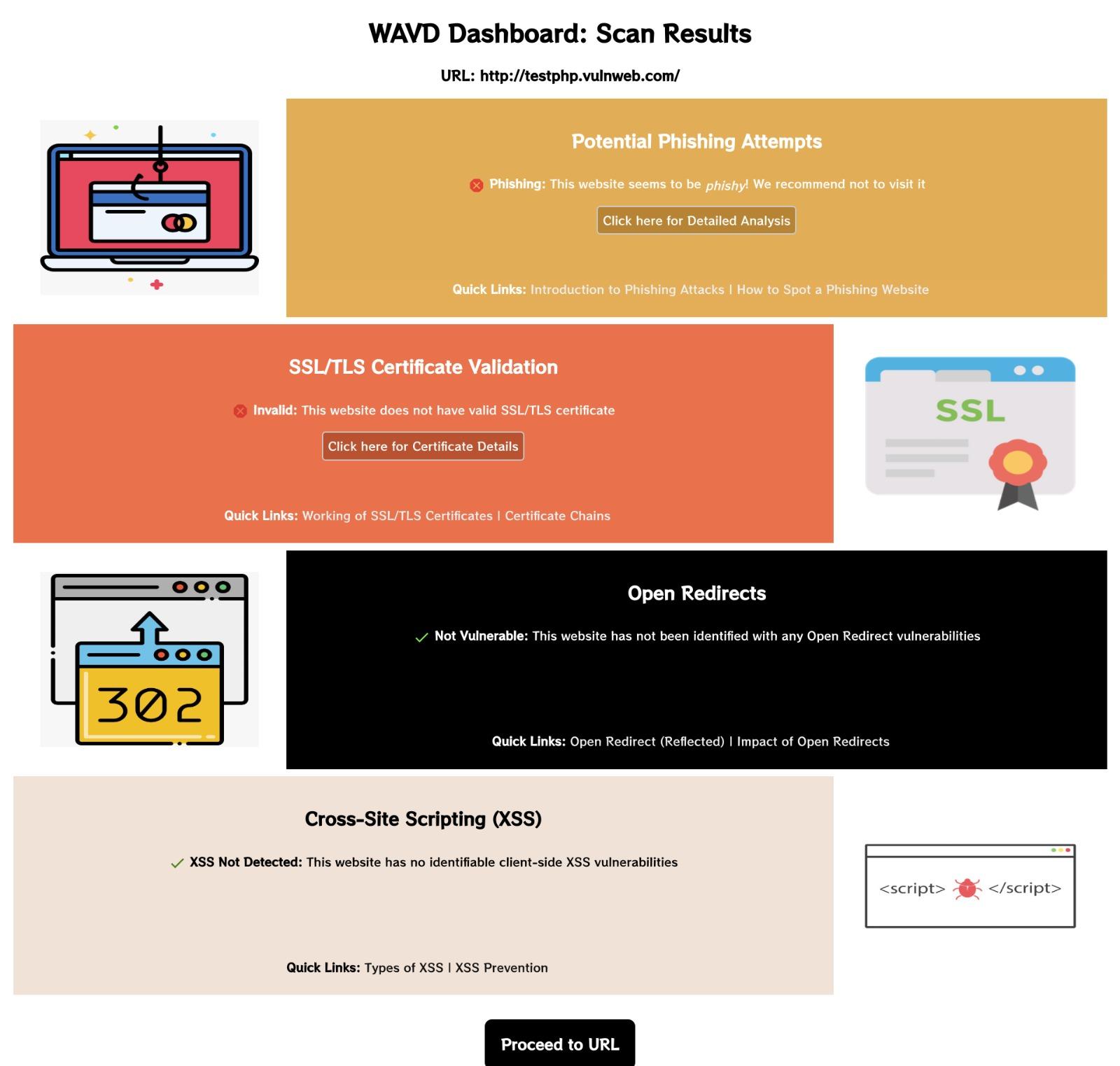


Figure 4.6 SSL/TLS Certificate Scanning

This panel shows an overview of the scan results, including phishing, xss, tls certification validation and open-redirect vulnerabilities. The picture depicts the scan result of potential vulnerable site which is vulnerable of potentail phishing attempts and also it does not contain SSL/TLS validation

# 

Figure 4.7 SSL/TLS Certificate Details

This panel shows detail description of SSL/TLS certificate validation of potential vulnerable web application

# 

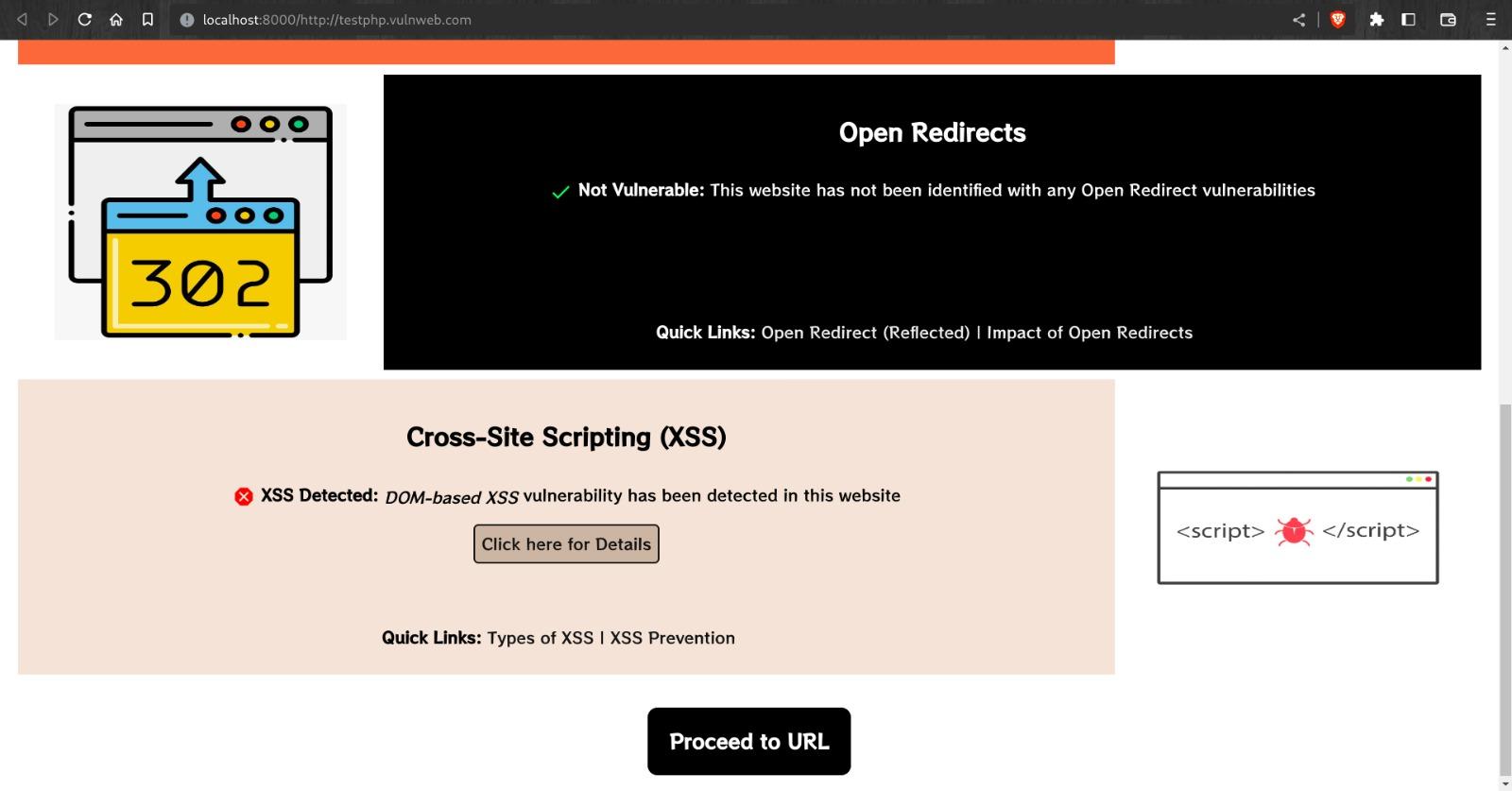


Figure 4.8 Cross-Site-Scripting Scanning

This panel shows the XSS detection of the potential vulnerable web application which is vulnerable for cross-site scripting

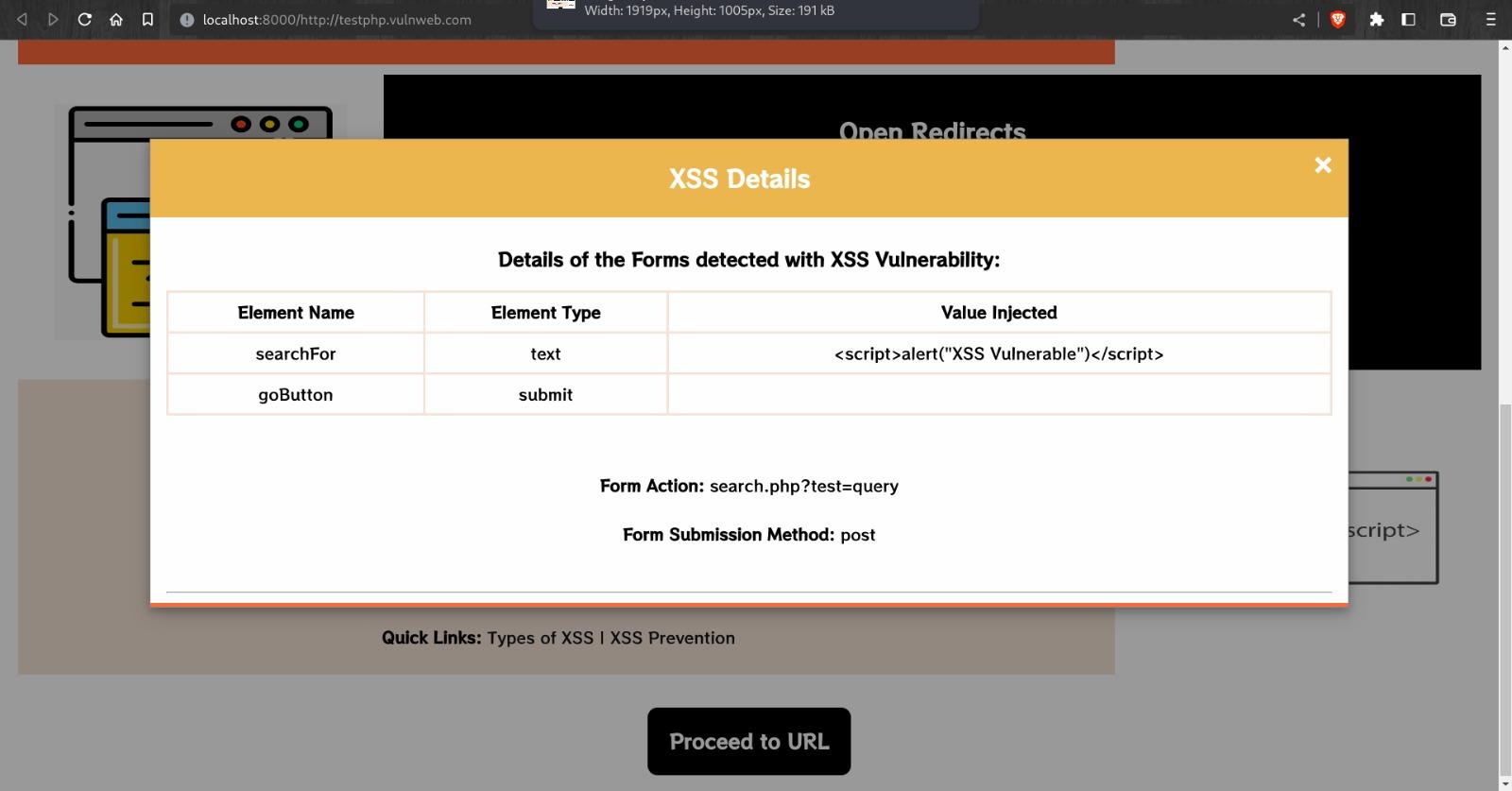


Figure 4.9 Cross-Site-Scripting Details

This panel shows the detailed report of cross site scripting vulnerability of potential web application where the malicious payload is injected in the search box section and it got executed using the post method and hence it can lead to cookie stealing

# 

Figure 4.10 Results with no vulnerabilities

# This panel shows that there are no given vulnerabilities present on github.com. As it shows it is legitimate, valid, not vulnerable and xss not detected, so we can proceed to URL further using "Proceed to URL" button.

# 

Figure 4.11 Phishing Detection Details

# 

Figure 4.12 SSL/TLS Certificate Details

# 

Figure 4.13

# Chapter 5

**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 a proxy server which will be provided by a 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.

[11] “https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset”, last accessed on 2/10/2022.

[12] “https://portswigger.net/web-security/cross-site-scripting/cheat-sheet”, last accessed on 29/12/2022.

**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 the 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 and In charge HOD Prof. Sunita Naik, 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**