WEB APPLICATION VULNERABILITY EXPLOITER

A Project Work File

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SECUIRITY

Submitted by:

ABHINANDAN KARMAKAR	**18BCS3555**
AKSHAY MAHALA	**18BCS3556**
DEEPAK JINDAL	**18BCS3572**
DUVVI KALYANRAM	**18BCS3576**
RAMAN SUHAG	**18BCS3586**

Under the Supervision of:

MR. KRISHNENDU RARHI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
APEX INSTITUTE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

APRIL-2021

DECLARATION

I, **Abhinandan Karmakar**, student of **'Bachelor of Engineering in 18AITIS1'**, **session:2022**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled **WAVE** is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 25.04.2021

Place: Home

Abhinandan Karmakar

18BCS3555

Table of Contents

L
ist of Figures

INTRODUCTION

Web Application Vulnerability Exploiter (WAVE) is basically a vulnerability scanner which scans for Security Vulnerabilities in web applications.

1.1.1 Vulnerability scanners are automated tools that scan web applications to look for security vulnerabilities. They test web applications for common security problems such as cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF).

1.2 PROJECT OVERVIEW:

In this we are discussing about the vulnerability scanning which are used for exploiting the web application using vulnerability

1.3 HARDWARE SPECIFICATIONS

A good working laptop

Internet facility

1.4 SOFTWARE SPECIFICATIONS

Python 3.x

2 LITERATURE REVIEW

In this we represents the state of security of web applications and network perimeters. This year's report contains the results and analysis of vulnerabilities detected over the 12-month period between March 2019 and February 2020, based on data from 5,000 scan targets. This analysis mainly applies to high and medium severity vulnerabilities found in web applications, as well as perimeter network vulnerability data.

While people might think that web applications in general are slowly getting more secure, the truth is less optimistic. We have observed that applications that are protected by web vulnerability scanning are the ones that are becoming more secure. We have also noticed that relatively new targets have more vulnerabilities.

This is worrying from a security perspective. It means that new developers do not have the knowledge that is required to avoid vulnerabilities. It also suggests that these developers are working within a development structure that does not promote web security. Old habits, unfortunately, die hard.

We discovered Cross-site Scripting (XSS) vulnerabilities, vulnerable JavaScript libraries, and WordPress-related issues in 25% of the sampled targets – certainly a lot. This means that web applications are still quite vulnerable, but even so, this number is 30% less than for the last year. It seems that experienced website developers and system administrators are making progress. The situation is similar for SQL Injection issues – just like last year, the numbers are decreasing.

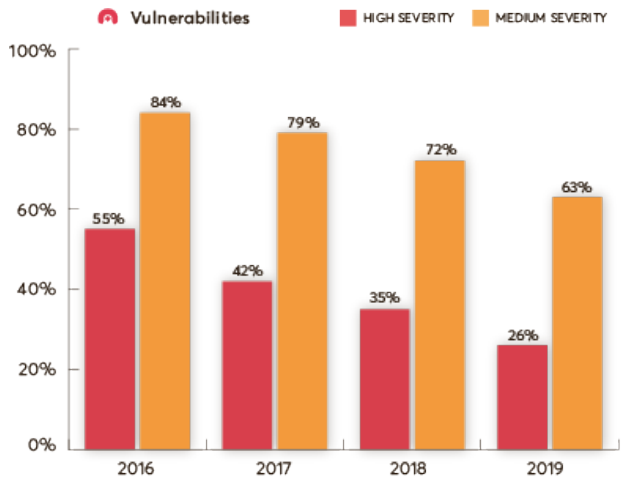


Figure 2.0.1. Vulnerabilities year-by-year.

The demand for interactive web applications is growing. Because of this, web applications use more and more client-side technologies. As a result, the number of JavaScript libraries keeps increasing. Many of these libraries have vulnerabilities. Their authors and users know about these vulnerabilities. And yet, around 25% of web applications use such vulnerable libraries.

It is also interesting when we compare server-side programming languages. We see that PHP remains as popular as before. The second most popular language is ASP.NET, but developers more and more often use other, less popular server-side languages.

When we talk about vulnerabilities, the situation is different.

See the graph below:

- The percentage of PHP vulnerabilities has declined a lot. The percentage of ASP or ASP.NET

vulnerabilities is growing.

- The percentage of vulnerabilities in Apache/nginx has declined a lot. The percentage of IIS vulnerabilities is growing.

```
cvar Cryo = require('cryo');var obj = {
testFunc : function() {return 1111;}
};
var frozen = Cryo.stringify(obj);console.log(frozen)
var hydrated = Cryo.parse(frozen);console.log(hydrated);
```

Figure 2.0.2. Percentage of vulnerabilities detected in various platforms.

Why might this be?

- We assume that most ASP/ASP.NET web applications run on IIS web servers.
- We assume that most PHP web applications run on Apache or nginx web servers.
- We observe that the trend for PHP is similar to the trend for Apache/nginx.
- We also observe that the trend for ASP/ASP.NET is similar to the trend for IIS.

One conclusion comes to mind when we consider this together with general statistics from the previous graph. It seems that the PHP+Apache/nginx platform is becoming more secure, mature, and robust. The market also keeps favoring this platform. On the other hand, the ASP/ASP.NET+IIS platform is slowly losing popularity. At the same time, it is still not as robust and mature as we would hope.

PHP is so popular because a lot of PHP sites are WordPress sites. WordPress sites are often unsafe but rather static. After you select the theme and plugins, you don’t change much. The attack surface changes only when you update WordPress, themes, and plugins. And most of these updates are security updates.

This also suggests that ASP/ASP.NET web applications are more actively developed. The high percentage of vulnerabilities may be caused by active development

PROBLEM FORMULATION

This section lists all the detected vulnerabilities.

Vulnerabilities by Type

The charts list vulnerabilities by type. They are grouped by the vulnerability severity level.

High Severity

This chart illustrates vulnerability types that fall into our *High Severity* category.

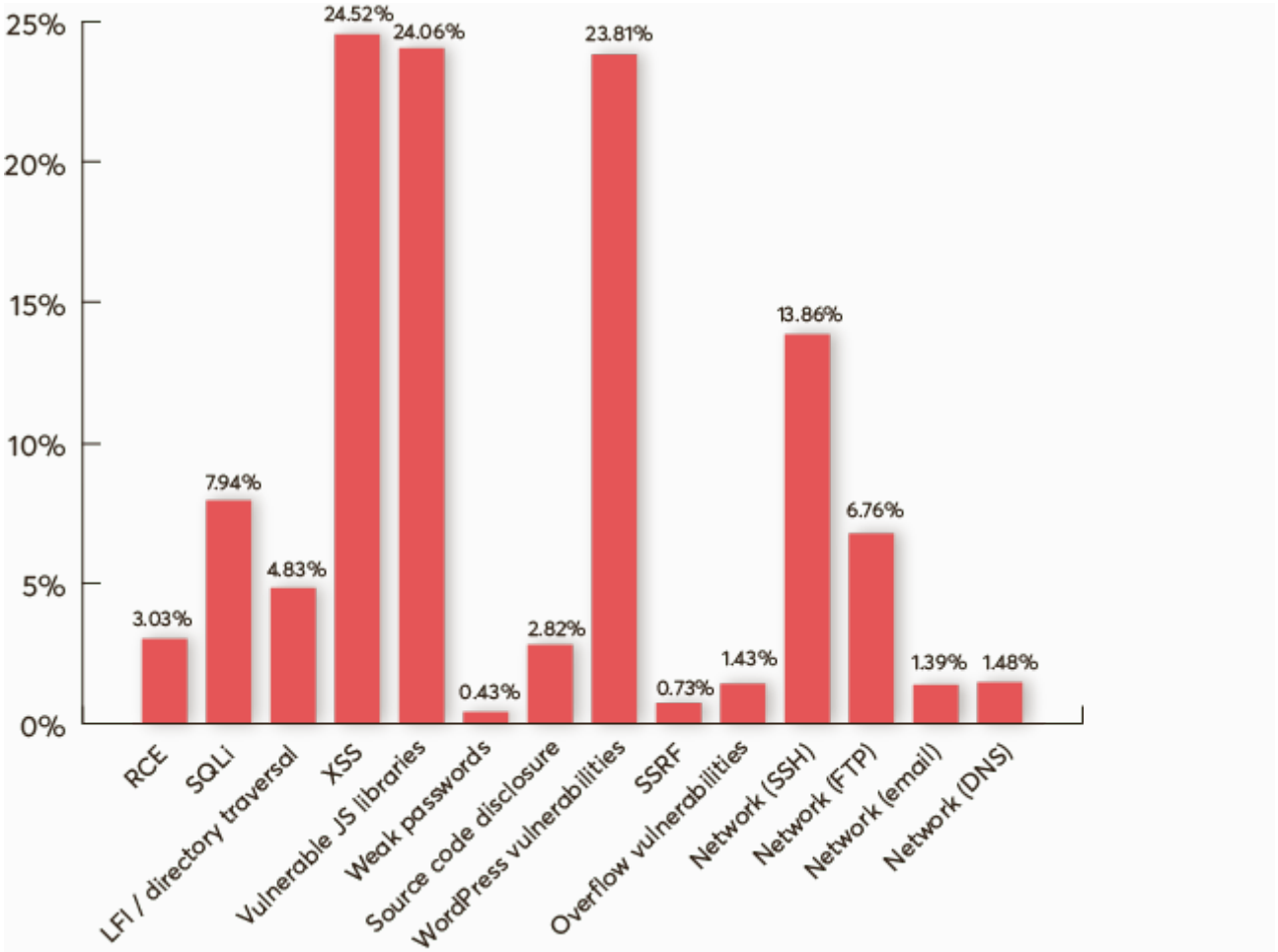


Figure 3.0.1. Analysis of high severity vulnerabilities.

This level indicates that an attacker can fully compromise the confidentiality, integrity, or availability of a system without specialized access, user interaction, or circumstances that are beyond the attacker's control. It is very likely that the attacker may be able to escalate the attack to the operating system and other systems.

Medium Severity

This chart lists vulnerability types that fall into our *Medium Severity* category.

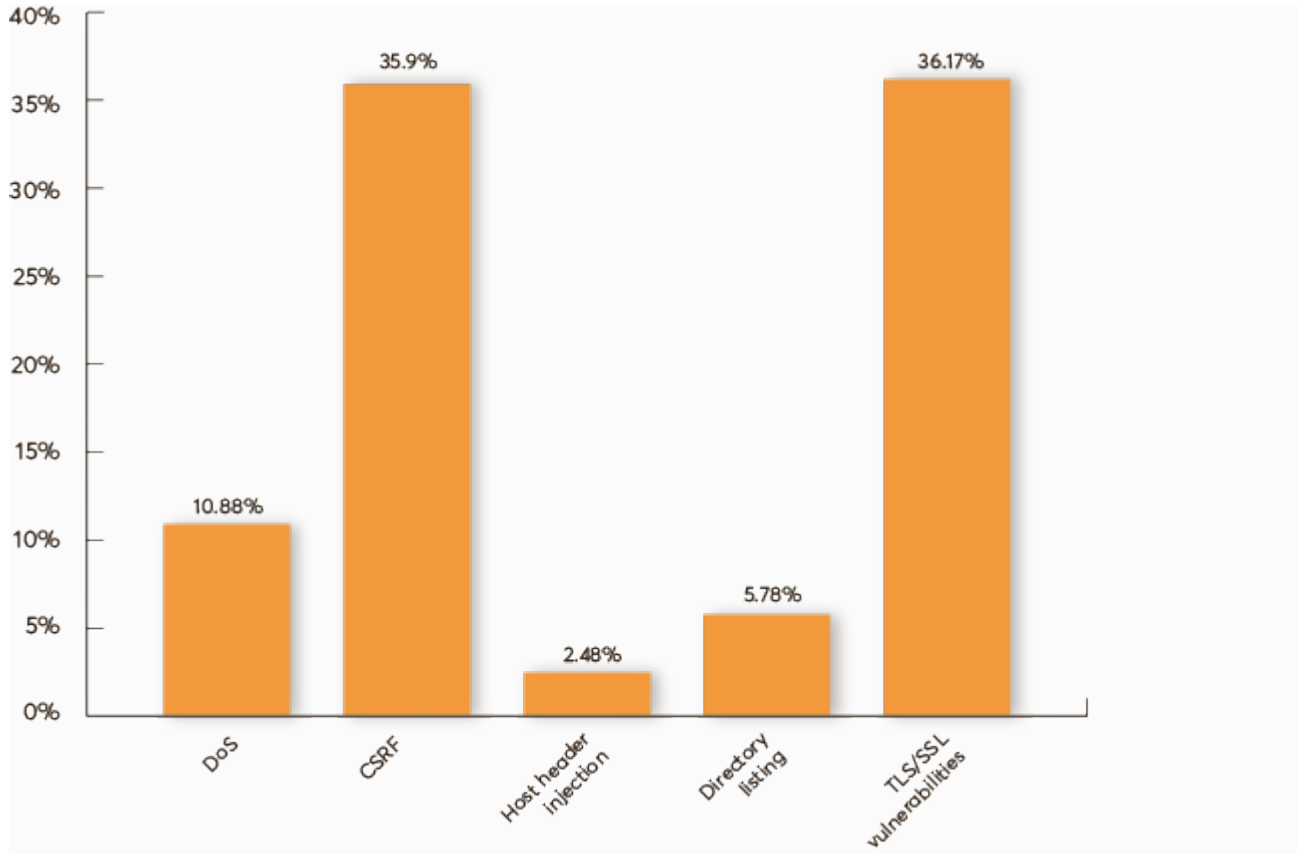


Figure 3.0.2. Analysis of medium severity vulnerabilities.

This level indicates that an attacker can partially compromise the confidentiality, integrity, or availability of a target system. They may need specialized access, user interaction, or circumstances that are beyond the attacker's control. Such vulnerabilities may be used together with other vulnerabilities to escalate an attack.

RESEARCH OBJECTIVES

The proposed research is aimed to carry out work leading to the development of an approach for web application from an vulnerability exploit The proposed aim will be achieved by dividing the work into following
objectives:

requirements

plugins

Vulnerabilities
Src
Python commands
Requirements: in this we have to gather the information required for the web application security .
Plugins:in this we use some plugins in python coding to get the scanning of the web application
Vulnerabilities : as I already informed there are three stages of vulnerabilities we have to gather the vulnerabilities and there severnity.
Src:in this we are going to implement some source files which are inbuilt in python
Python commands:there are some specific commands which we are going to implement while programming in this.

5 METHODOLOGY

We took a random sample of 5,000 scan targets from Acunetix Online from one year back. This sample included web application and network perimeter security scans. We excluded scans for websites that are intentionally vulnerable for educational purposes.

How Automatic Web Scanning Works
Acunetix Online can perform dynamic application security testing (DAST) scans (also called black-box scans), as well as interactive application security testing (IAST) scans (also called gray-box scans).

A DAST scan means that the scanner has no information about the structure of the website or used technologies. An IAST scan means that the scanner has "insider information" about the web application. In Acunetix, this is possible thanks to AcuSensor technology. You install AcuSensor agents on the web server for Java, ASP.NET, and PHP applications. The agents send information from the web server back to the scanner.

When scanning, you typically follow the following four stages and repeat them if necessary:

Crawling – the Acunetix crawler starts from the home or index page. Then it builds a model of the structure of the web application by crawling through all links and inputs. It simulates user+browser behavior to expose all the reachable elements of the website.
Scanning – once the crawler has built the website model, each available page or endpoint is automatically tested to identify all potential vulnerabilities.
Reporting – you can view the progress of a scan in real-time, but the results of a scan are typically summarized in reports. You can use reports for compliance and management purposes. Acunetix offers several report templates for different purposes, for example, OWASP Top 10 and ISO 27001 reports.
Remediation – fixing vulnerabilities:
Patching – first, export Acunetix data to a web application firewall (WAF). This lets you temporarily defend against an attack while you work on a fix.
Issue Management – when you integrate with issue trackers like JIRA, GitHub, and GitLab, you can track vulnerabilities from the moment they are discovered to resolution. You can also integrate with continuous integration solutions such as Jenkins.
Continuous Scanning – Acunetix can perform scheduled scans. You can use them to make sure that vulnerabilities are really fixed.

6 RESULTS AND DISCUSSION

Fig 6.1 OUTLOOK:

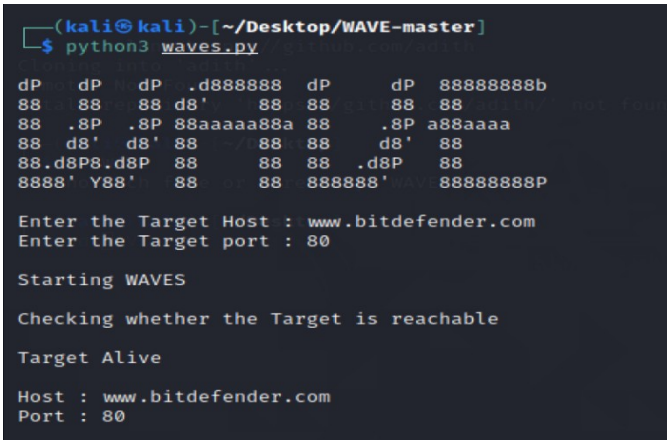


Fig 6.2 INSTALLING PRE-REQUIREMENTS :

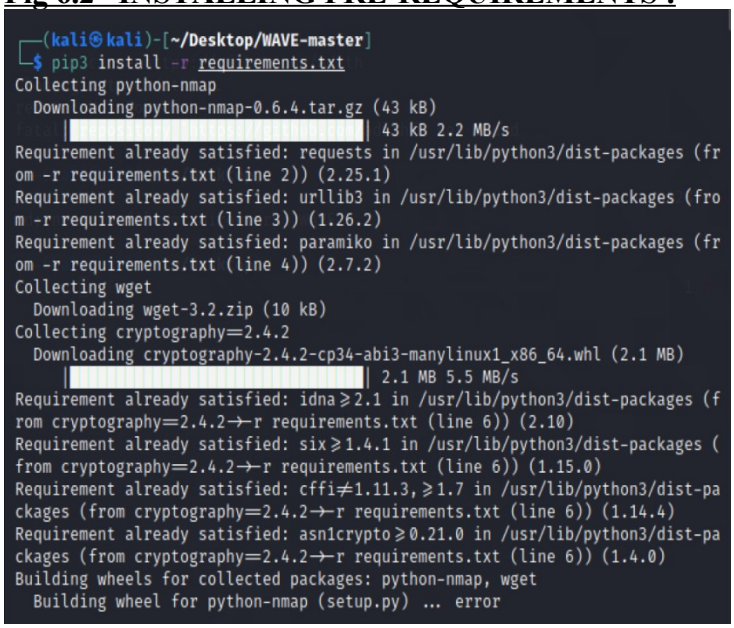


Fig 6.3 PROJECT CONSISTS OF :

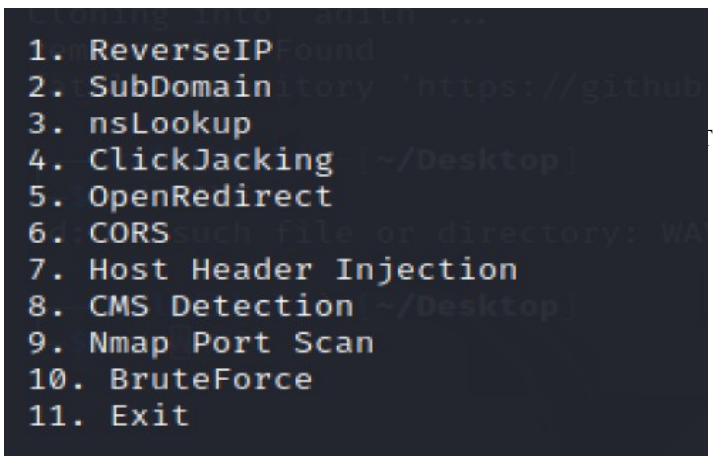


Fig 6.4 REVERSE IP :

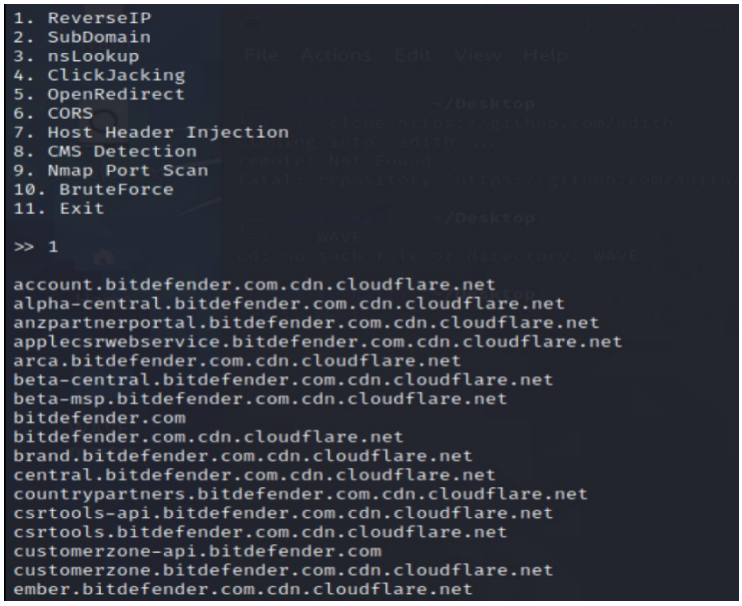
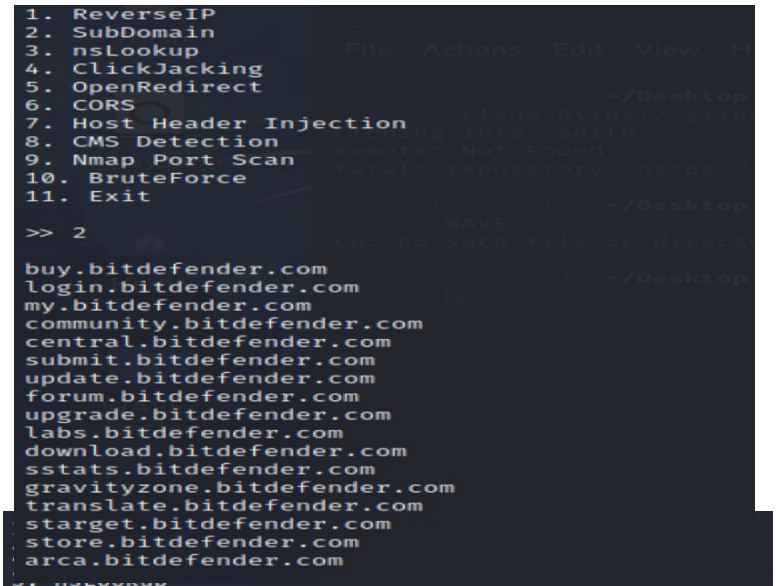


Fig 6.5 SUB DOMAIN :



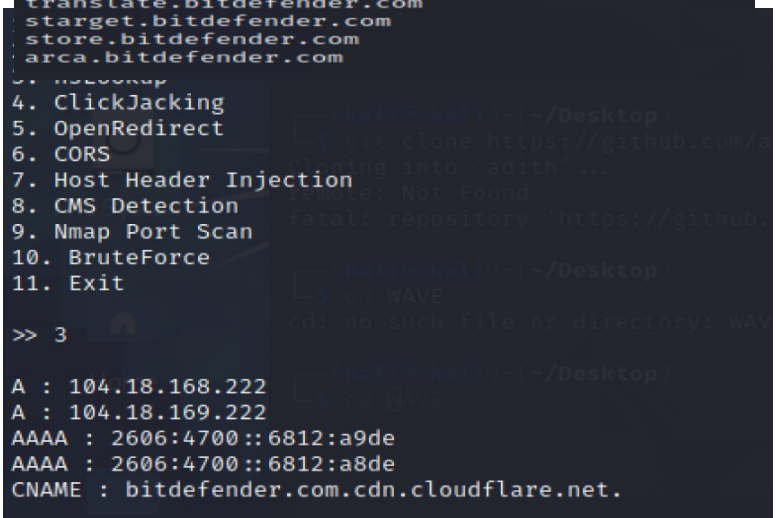


Fig 6.6 NSLOOKUP :

Fig 6.7 CLICKJACKING :

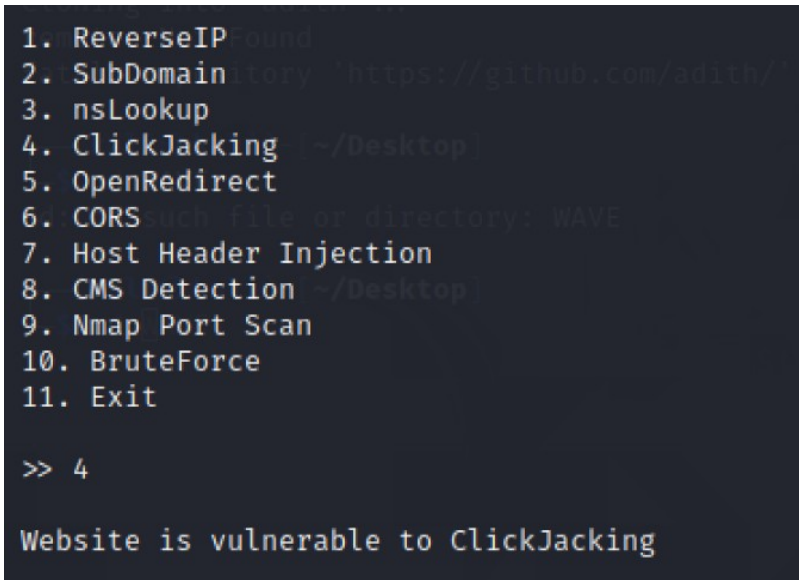
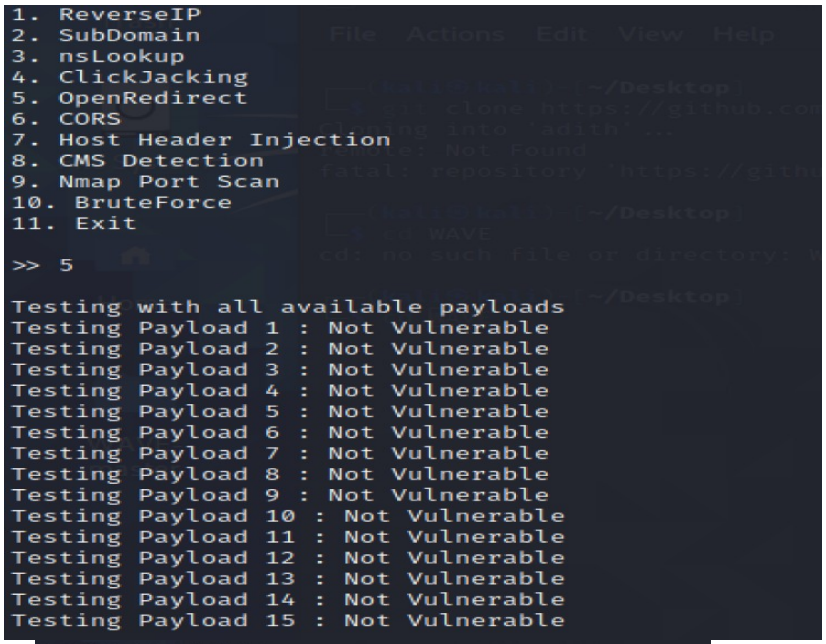


Fig 6.8 OPEN REDIRECT :



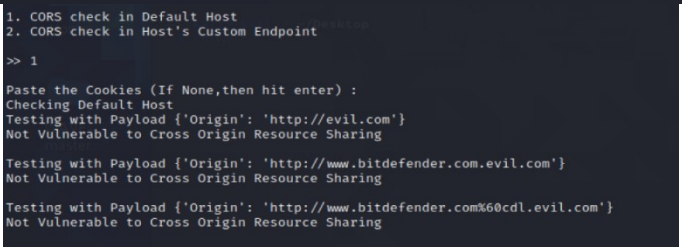


Fig 6.9 CORS(Cross Origin Resource Sharing) :

Fig 6.10 HOST HEADER INJECTION :

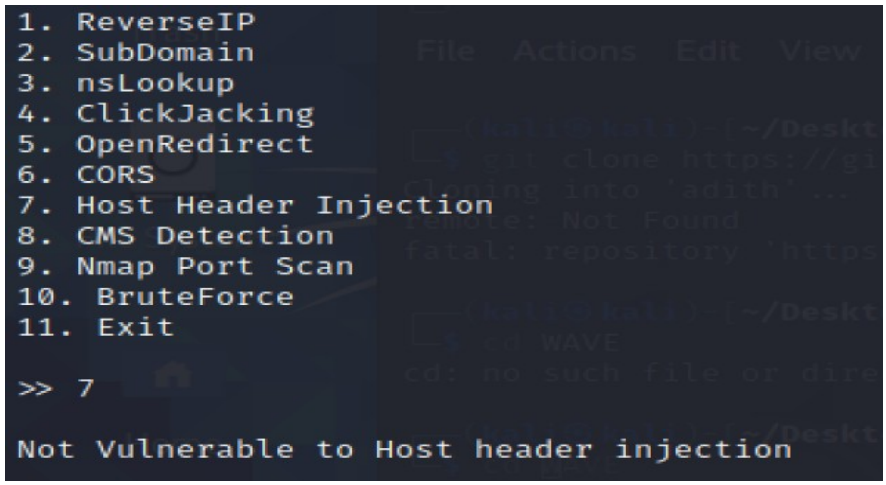


Fig 6.11 NMAP PORT SCAN :

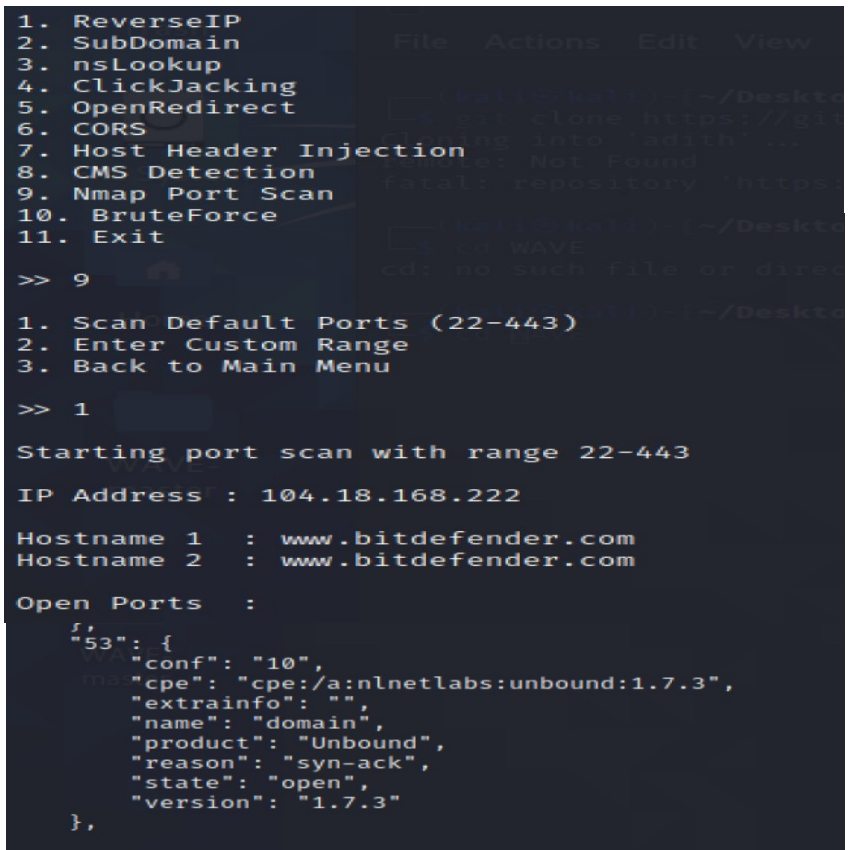
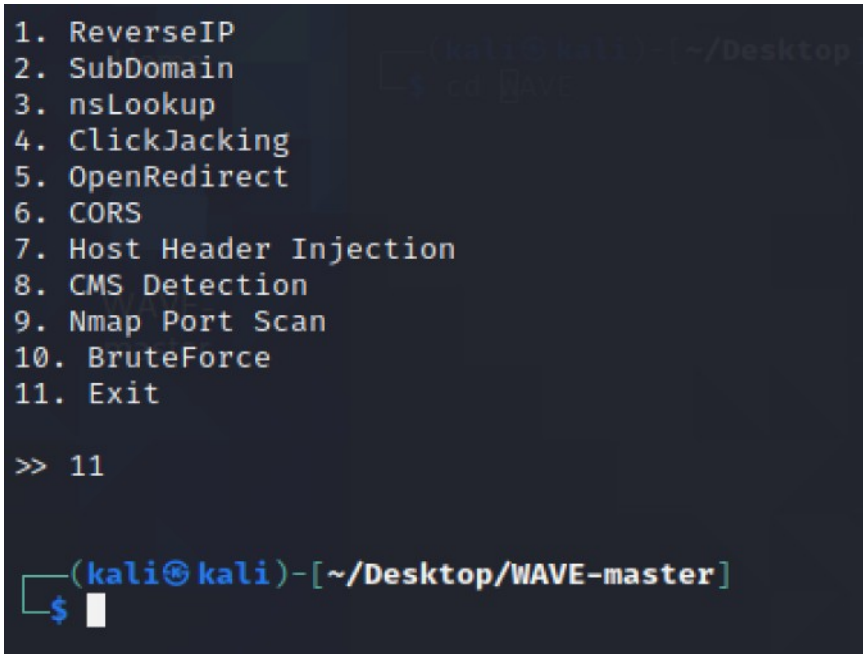


Fig 6.12 OPEN PORTS :

Fig 6.13 EXIT :



Url for video reference of the project:

https://drive.google.com/file/d/1ubyCYCNTCBoqFInQFvOHkFbd3YZIoHE_/view?usp=sharing

Conclusion :

Make Web applications more secured and flexible to use ,detect and report issues to the company.
It makes difficult for Blackhat hackers to exploit the application and make the cyber space more safe.
Web Application Vulnerability Exploiter (WAVE) is basically a vulnerability scanner which scans for Secuirity Vulnerabilities in web applications.

The web applications are tested for common security problems such as Host header injection, SQL injection, and Clickjacking, CORS. And many more.

7 REFERENCES

[1] https://owasp.org/www-community/Vulnerability_Scanning_Tools

[2] https://www.acunetix.com/websitesecurity/the-importance-of-web-application-scanning/

[3] https://portswigger.net/burp/vulnerability-scanner

[4] https://www.acunetix.com/white-papers/acunetix-web-application-vulnerability-report-2020/#vulnerable-javascript-libraries

[5] https://rhaidiz.net/2019/06/10/wafex/