UNIVERSITÀ DEGLI STUDI DI SALERNO

Automatic Software Vulnerability Testing

Report on (Webpage Content Information Leakage)

Course of Study	Software Dependability
Project Supervisor	Professor Fabio Palomba
Project Advisor	Dr.Lannone Dr.Giordano
Academic Years	2020/2021

Group Members

Sarmad Ali Khan
s.khan@studenti.unisa.it

Bilal Haider
b.haider@studenti.unisa.it

Shuja Uddin Qureshi
m.qureshi@studenti.unisa.it

Table of Contents

Abstract **1**

Introduction **2**

OWASP Testing Project 2

Review Webpage content for information leakage 2

Testing **4**

Principles of Testing 4

Testing Objectives 5

Testing Techniques 5

Working on Project **6**

Testing and Findings 6

Phases 7

Testing Results **7**

Juicy data 7

DTD URLs 8

Meta tag 8

Identifying JavaScript code and gathering JavaScript files 9

Identifying Source map files 9

Tools for testing **10**

GitBash 10

Curl 10

JS Beautifier 11

Conclusions and Discussions **11**

References **12**

Abstract

The Web Security Testing Guide (WSTG) Project produces the premier cybersecurity testing resource for web application developers and security professionals.The WSTG is a comprehensive guide to testing the security of web applications and web services. Created by the collaborative efforts of cybersecurity professionals and dedicated volunteers, the WSTG provides a framework of best practices used by penetration testers and organizations all over the world.

For the purposes of this document, testing is a process of comparing the state of a system or application against a set of criteria. In the security industry, people frequently test against a set of mental criteria that are neither well defined nor complete.

There are number of stages in web applications which includes vulnerabilities. More specifically, we have done the testing on information in source code to find vulnerabilities which refers “Webpage information content leakage (WSTG-INFO-05)”.

Similar to the comments and metadata in HTML code, many programmers also hardcode sensitive information in JavaScript variables on the front-end. Sensitive information can include (but is not limited to): Private API Keys (*e.g.* an unrestricted Google Map API Key), internal IP addresses, sensitive routes (*e.g.* route to hidden admin pages or functionality), or even credentials. This sensitive information can be leaked from such front-end JavaScript code.

Different tools are used to find these vulnerabilities in webpage content. There are several tools on web for this purpose. The specific tools used in the mentioned projects for testing are “Curl, Jsbeautifier”.

The vulnerabilities we found are like juicy data, keywords, DTD URL’s, API, hidden links in javascript. These are things intrusions can be easy for the hackers to attack application.

Introduction

OWASP Testing Project

The OWASP Testing Project has been in development for many years. The aim of the project is to help people understand the *what*, *why*, *when*, *where*, and *how* of testing web applications. The project has delivered a complete testing framework, not merely a simple checklist or prescription of issues that should be addressed. Readers can use this framework as a template to build their own testing programs or to qualify other people's processes.

One aspect that should be emphasized is that security measurements are about both the specific technical issues (e.g., how prevalent a certain vulnerability is) and how these issues affect the economics of software. Most technical people will at least understand the basic issues, or they may have a deeper understanding of the vulnerabilities.

Review Webpage Content for Information Leakage

It is very common, and even recommended, for programmers to include detailed comments and metadata on their source code. However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers. Comments and metadata review should be done in order to determine if any information is being leaked.

For large web applications, performance issues are a big concern to programmers. Programmers have used different methods to optimize front-end performance, including Syntactically Awesome Style Sheets (SASS), Sassy CSS (SCSS), webpack, etc. Using these technologies, front-end code will sometimes become harder to understand and difficult to debug, and because of it, programmers often deploy source map files for debugging purposes.

A "source map" is a special file that connects a minified/uglified version of an asset (CSS or JavaScript) to the original authored version. However, it is undeniable that source map files or files for debugging if released to the production environment will make their source more human-readable. It can make it easier for attackers to find vulnerabilities from the front-end or collect sensitive information from it. Depending on the context and sensitivity of the project, a security expert should decide whether the files should exist in the production environment or not.

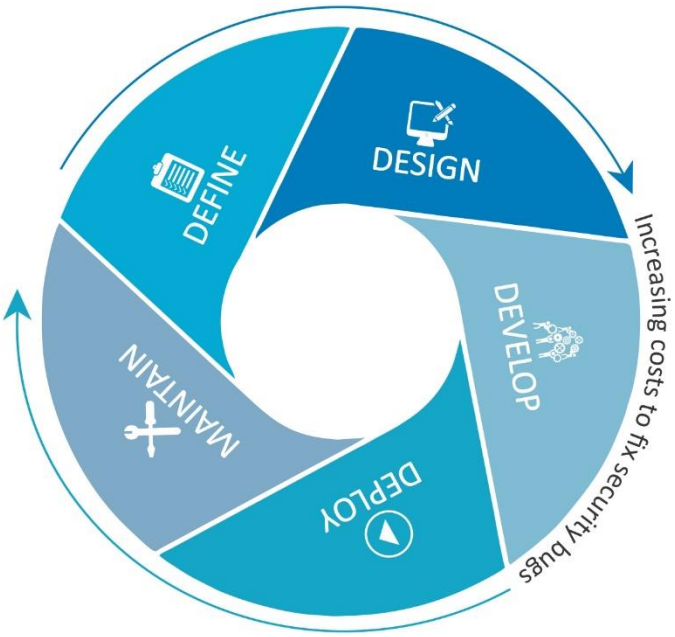
Testing

Testing is a process of comparing the state of a system or application against a set of criteria. In the security industry, people frequently test against a set of mental criteria that are neither well defined nor complete.

This document is designed to help organizations understand what comprises a testing program, and to help them identify the steps that need to be undertaken to build and operate a modern web application testing program. The guide gives a broad view of the elements required to make a comprehensive web application security program.

One of the best methods to prevent security bugs from appearing in production applications is to improve the Software Development Life Cycle (SDLC) by including security in each of its phases.

An SDLC is a structure imposed on the development of software artifacts.



Principles of Testing

There are some common misconceptions when developing a testing methodology to find security bugs in software. This report covers some of the basic principles that

professionals should take into account when performing security tests on webpage content of web application.

Testing Objectives

WEB TESTING, or website testing is checking your web application or website for potential bugs before it made live and is accessible to general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

Testing Techniques

This report presents a high-level overview of various testing techniques that can be employed when building a testing program. It does not present specific methodologies for these techniques. This report is included to provide context for the framework presented and to highlight the some of the techniques that should be considered. In particular, we will cover:

- Manual Inspections & Reviews

  Manual inspections are human reviews that typically test the security implications of people, policies, and processes. Manual inspections can also include inspection of technology decisions such as architectural designs. They are usually conducted by analyzing documentation or performing interviews with the designers or system owners.

- Source Code Review

  Source code review is the process of manually checking the source code of a web application for security issues. Many serious security vulnerabilities cannot be detected with any other form of analysis or testing. All the information for identifying security problems is there in the code, somewhere. Unlike testing closed software such as operating systems, when testing web applications (especially if they have been developed in-house) the source code should be made available for testing purposes.

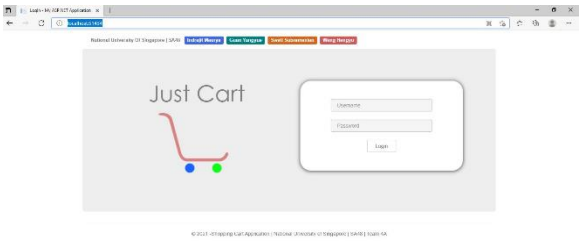
Working on Projects

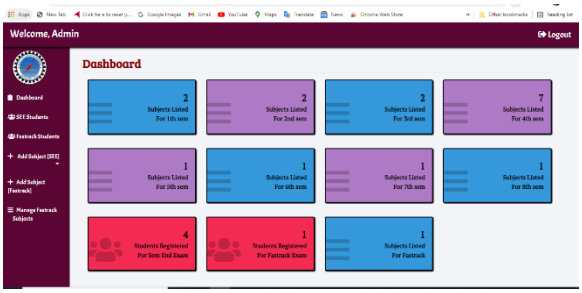
Finding and Testing

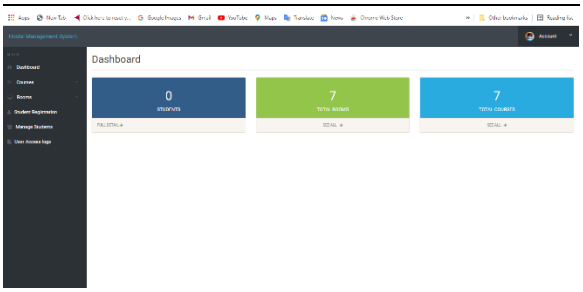
Testing is performed on random open source projects, deployed on localhost and live websites. These projects were developed on C#, HTML, PHP and JavaScript.

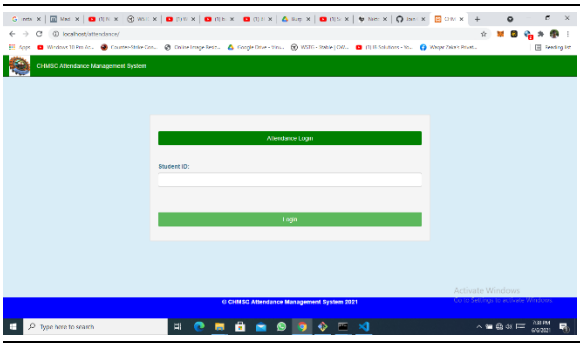
Some of web applications are:

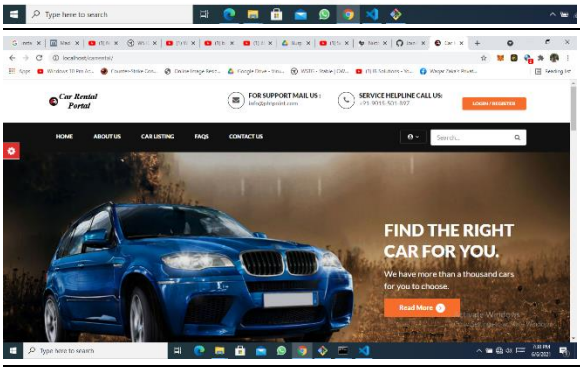
- Car Rental
- Blood Bank
- Student attendance
- E-commerce
- Tourism Management System
- Hotel Management System
- Student Exams Management











Phase

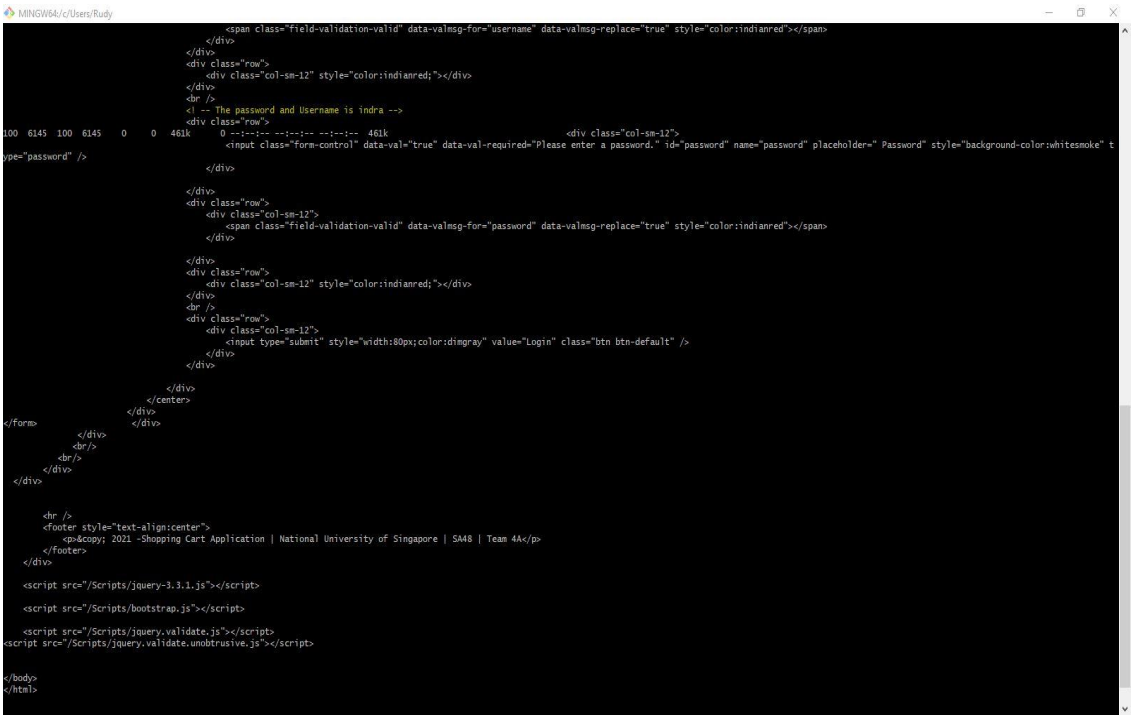
There are different phases which includes to find the projects form Github and other free websites. After finding the projects developed on required development environment. We started to analyze the projects to find the tools for their testing. For this purpose, It was necessary to check each and every file to find the corresponding tool for testing.

Testing Results

While testing, we found the many vulnerabilities which commonly includes

- **Juicy Data**

  In this Website of Ecommerce we find in the Html Comment the VULNERABILTY which is very sensitive data . The password and User Name to login/Access the Website.



it is clearly Seen in the Html comments

```
<!—The Password and Username is indra -->
```

DTD URLs

DTD stands for Document Type Definition. A DTD defines the structure and the legal elements and attributes of an XML document.

- `strict.dtd` – default strict DTD
- `loose.dtd` – loose DTD
- `frameset.dtd` – DTD for frameset documents

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

The DTD goes after the XML declaration or is in a separate file:

<?xml version="1.0"?>
<!DOCTYPE exam SYSTEM "exam.dtd">

Remember:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

- PUBLIC indicates a publicly available standard
- SYSTEM indicates a local URL
- The DOCTYPE statement is the Document Type Declaration
- This and the other declarations make up the DTD
- *exam* is the root element

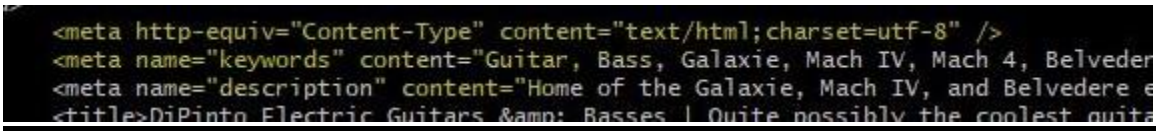
META Tags

Some `META` tags do not provide active attack vectors but instead allow an attacker to profile an application

```
<META name="Author" content="Andrew Muller">
```

A common (but not WCAG compliant) `META` tag is Refresh.

```
<META http-equiv="Refresh"    content="15;URL=https://www.owasp.org/index.html">
```



A common use for META tag is to specify keywords that a search engine may use to improve the quality of search results.

```
<META name="keywords" lang="en-us" content="OWASP, security, sunshine, lollipops">
```

Although most web servers manage search engine indexing via the robots.txt file, it can also be managed by META tags. The tag below will advise robots to not index and not follow links on the HTML page containing the tag.

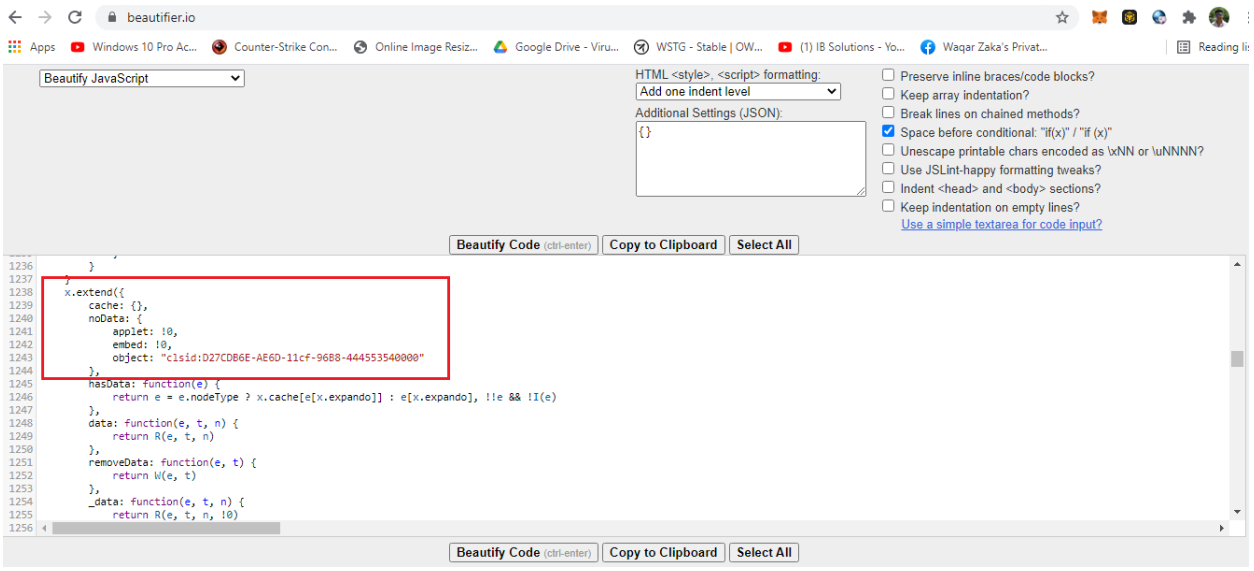
```
<META name="robots" content="none">
```

JavaScript Code and Gathering JavaScript Files

Programmers often hardcode sensitive information with JavaScript variables on the front-end. Testers should check HTML source code and look for JavaScript code between <script> and </script> tags. Testers should also identify external JavaScript files to review the code (JavaScript files have the file extension .js and name of the JavaScript file usually put in the src (source) attribute of a <script> tag).

Check JavaScript code for any sensitive information leaks which could be used by attackers to further abuse or manipulate the system. Look for values such as: API keys, internal IP addresses, sensitive routes, or credentials. For example:

```
const myS3Credentials = {
  accessKeyId: config('AWSS3AccessKeyID'),
  secretAcccessKey: config('AWSS3SecretAccessKey'),
};
```



When an API Key is found, testers can check if the API Key restrictions are set per service or by IP, HTTP referrer, application, SDK, etc.

For example, if testers found a Google Map API Key, they can check if this API Key is restricted by IP or restricted only per the Google Map APIs. If the Google API Key is restricted only per the Google Map APIs, attackers can still use that API Key to query unrestricted Google Map APIs and the application owner must to pay for that.

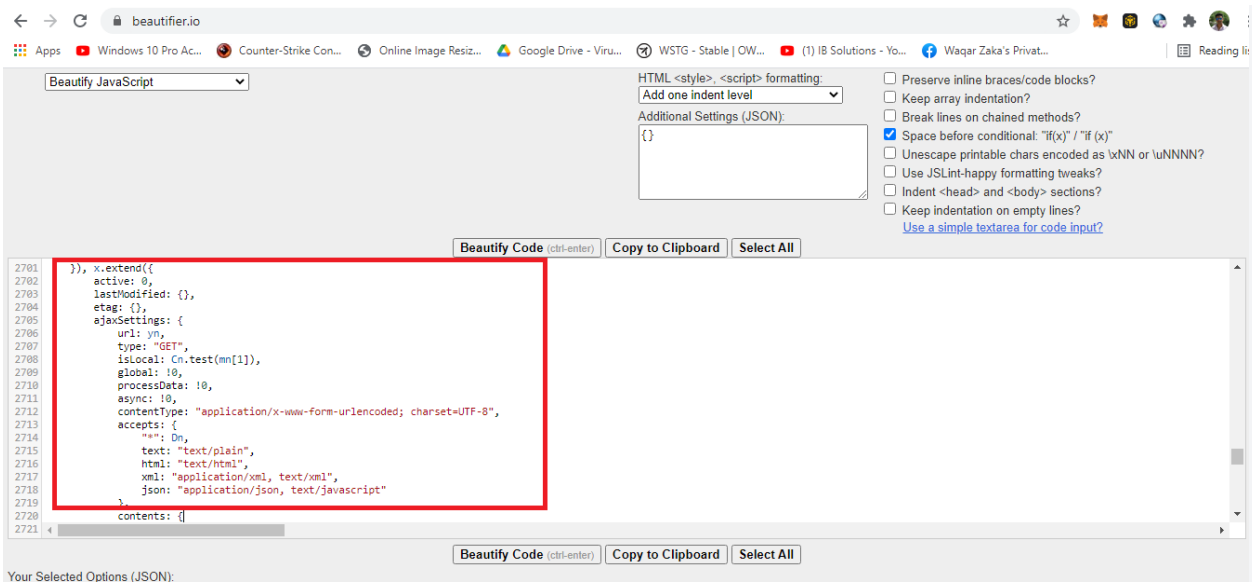
```
<script type="application/json">
...
{"GOOGLE_MAP_API_KEY":"AIzaSyDUEBnKgwiqMNpDplT6ozE4Z0XxuAbqDi4",
"RECAPTCHA_KEY":"6LcPscEUiAAAAHOwwM3fGvIx9rsPYUq62uRhGjJ0"}
...
</script>
```

In some cases, testers may find sensitive routes from JavaScript code, such as links to internal or hidden admin pages.

```
<script type="application/json">
...
"runtimeConfig":{"BASE_URL_VOUCHER_API":"https://staging-
voucher.victim.net/api", "BASE_BACKOFFICE_API":"https://10.10.10.2/api",
"ADMIN_PAGE":"/hidden_administrator"}
...
</script>
```

Identifying Source Map Files

Source map files will usually be loaded when DevTools open. Testers can also find source map files by adding the ".map" extension after the extension of each external JavaScript file. For example, if a tester sees a `/static/js/main.chunk.js` file, they can then check for its source map file by visiting `/static/js/main.chunk.js.map`.



Black-Box Testing

Check source map files for any sensitive information that can help the attacker gain more insight about the application. For example:

```
"version": 3,
"file": "static/js/main.chunk.js",
"sources": [
  "/home/sysadmin/cashsystem/src/actions/index.js",
  "/home/sysadmin/cashsystem/src/actions/reportAction.js",
  "/home/sysadmin/cashsystem/src/actions/cashoutAction.js",
  "/home/sysadmin/cashsystem/src/actions/userAction.js",
  "..."
],
"..."
```

When websites load source map files, the front-end source code will become readable and easier to debug.

Tools for Testing

GitBash

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience. Bash is an acronym for Bourne Again Shell. A shell is a terminal application used to interface with an operating system through written commands. Bash is a popular default shell on Linux and macOS. Git Bash is a package that installs Bash, some common bash utilities, and Git on a Windows operating system

Curl

Curl is used in command lines or scripts to transfer data. curl is also used in cars, television sets, routers, printers, audio equipment, mobile phones, tablets, settop boxes, media players and is the Internet transfer engine for thousands of software applications in over *ten billion installations*.

JS Beautifier

JavaScript is a web based programming language that is used extensively in many websites. ... This site give you a quick and easy way to format (beautifier) the JavaScript so you can easily read it.

Conclusions and Discussions

While testing Webpage Information Leakage ID WSTG-INFO-05 it goes through a different Phases, we check Juicy Data , the hidden formation in Html Tags, we check Meta Tags , DTD URLs, Black Box Testing, Java Script Code and Java Scripts Files . We came into conclusion that There are many Vulnerabilities hidden in Web Page Content through which attacker can get many Information and by the help of that data we are able to attack the website.

References

- OWASP Web Application Security Project
  By using OWASP website, the content was taken to get knowledge about "Review webpage content leakage", methods and tools to deploy testing on random projects.
  https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/05-Review_Webpage_Content_for_Information_Leakage.html

- Random web application projects for testing
  These are free websites and the Github links from where the projects where taken for testing.
  https://www.phptpoint.com/projects/car-rental-system/
  http://github.com/Chandana047/Blood-Bank-Management-System
  https://itsourcecode.com/wp-content/uploads/2021/02/attendance.zip

- Tools used for Testing and finding vulnerabilities
  Various tools are used for testing and finding vulnerabilities. These tool are
  Gitbash (https://git-scm.com/downloads)
  Curl (https://curl.se/)
  Javascript beautifier (https://beautifier.io/)

- Xampp localhost server to run and deploy Projects
  The projects were written in PHP language. So, we had to need xampp to run the projects on localhost servers and find vulnerabilitites.
  Xampp (https://www.apachefriends.org/download.html)