Web Application Security Testing

Course Work 2: Group Report

Course Title: COMP09109 - Web Application Security Testing

Weeks Covered: 01 to 10

Assignment Type: In-Class Assessment

Weight Toward Final Grade: 40%

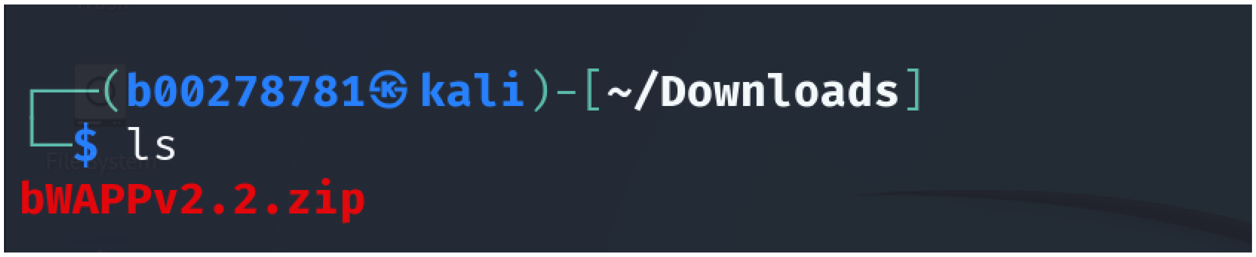
Total Marks Available: 100

My Methodology

Step 1:

Download the bWAPP application zip file from https://sourceforge.net/projects/bwapp/files/bWAPP/

The file is downloaded successfully



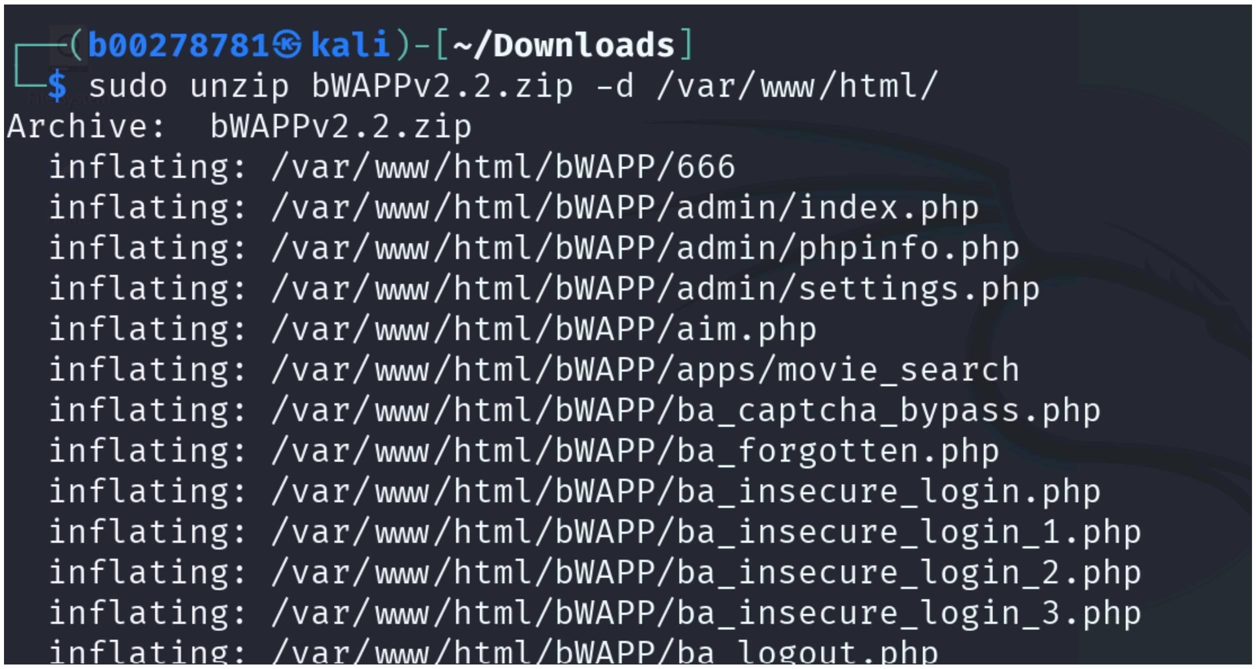
```
┌──(b00278781㉿kali)-[~/Downloads]
└─$ ls
bWAPPv2.2.zip
```

Now make directory a directory in /var directory i.e /var/www/html

- Now make /var/www/html directory

```
mkdir /var/www/html
```

then unzip the bWAPPv2.2.zip into /var/www/html



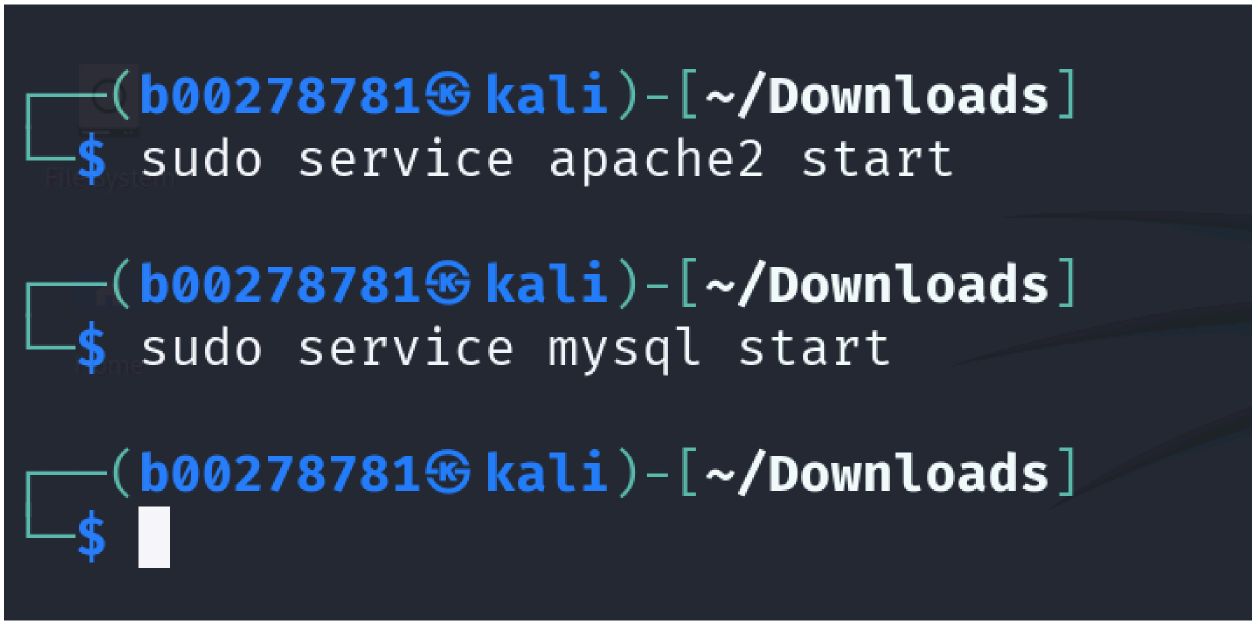
```
┌──(b00278781㉿kali)-[~/Downloads]
└─$ sudo unzip bWAPPv2.2.zip -d /var/www/html/
Archive:  bWAPPv2.2.zip
  inflating: /var/www/html/bWAPP/666
  inflating: /var/www/html/bWAPP/admin/index.php
  inflating: /var/www/html/bWAPP/admin/phpinfo.php
  inflating: /var/www/html/bWAPP/admin/settings.php
  inflating: /var/www/html/bWAPP/aim.php
  inflating: /var/www/html/bWAPP/apps/movie_search
  inflating: /var/www/html/bWAPP/ba_captcha_bypass.php
  inflating: /var/www/html/bWAPP/ba_forgotten.php
  inflating: /var/www/html/bWAPP/ba_insecure_login.php
  inflating: /var/www/html/bWAPP/ba_insecure_login_1.php
  inflating: /var/www/html/bWAPP/ba_insecure_login_2.php
  inflating: /var/www/html/bWAPP/ba_insecure_login_3.php
  inflating: /var/www/html/bWAPP/ba_logout.php
```

After it is successful, run the following commands

```
sudo service apache2 start

sudo service mysql start
```

The above commands will start the Apache ad MySQL server

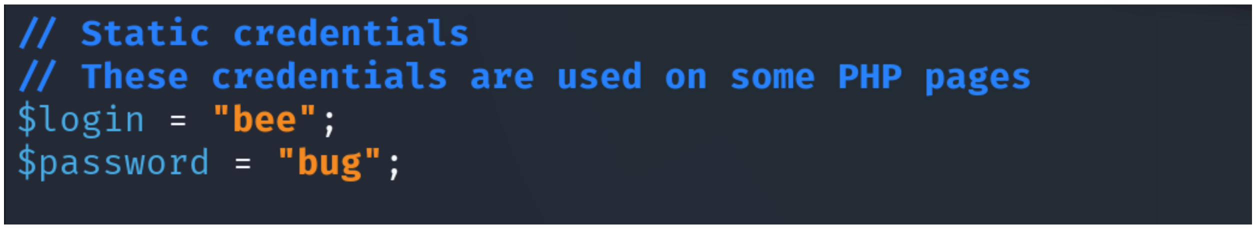


```
┌──(b00278781㉿kali)-[~/Downloads]
└─$ sudo service apache2 start

┌──(b00278781㉿kali)-[~/Downloads]
└─$ sudo service mysql start

┌──(b00278781㉿kali)-[~/Downloads]
└─$ 
```

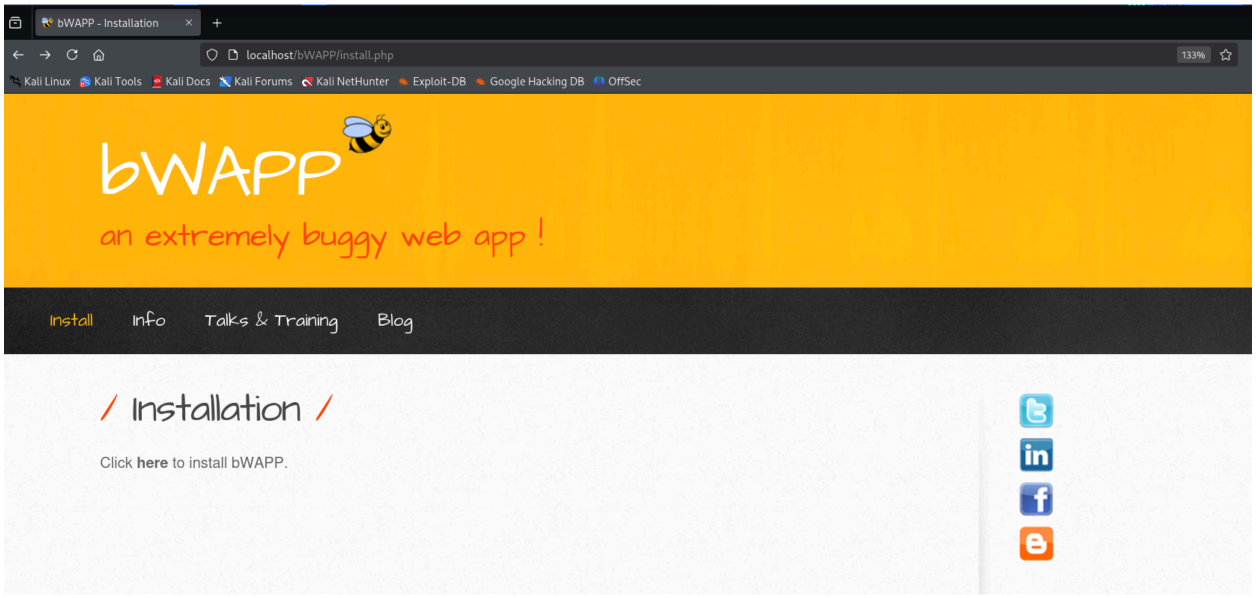
- Navigate to /var/www/html/bWAPP/admin and open settings.php with nano
- The Default credentials of login page are found in settings.php .



```
// Static credentials
// These credentials are used on some PHP pages
$login = "bee";
$password = "bug";
```

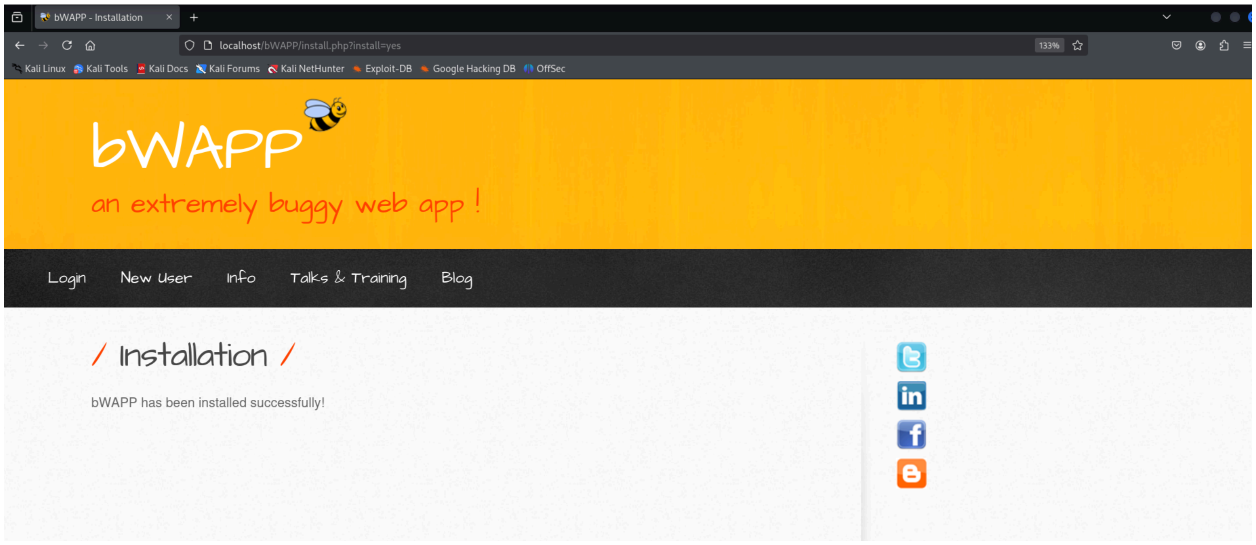
- Open Firefox and go to the following URL

http://localhost/bWAPP/install.php



Click **here** to install bWAPP.

click on install and now it is installed successfully



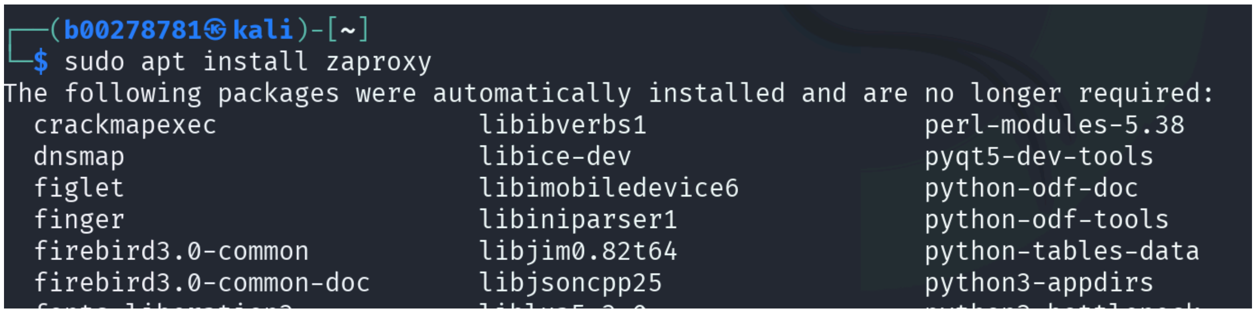
bWAPP has been installed successfully!

Step 2:

Now we will install zaproxy in our kali linux as it is not installed by default on Kali Linux

Run the following command in terminal

```
sudo apt install zaproxy
```

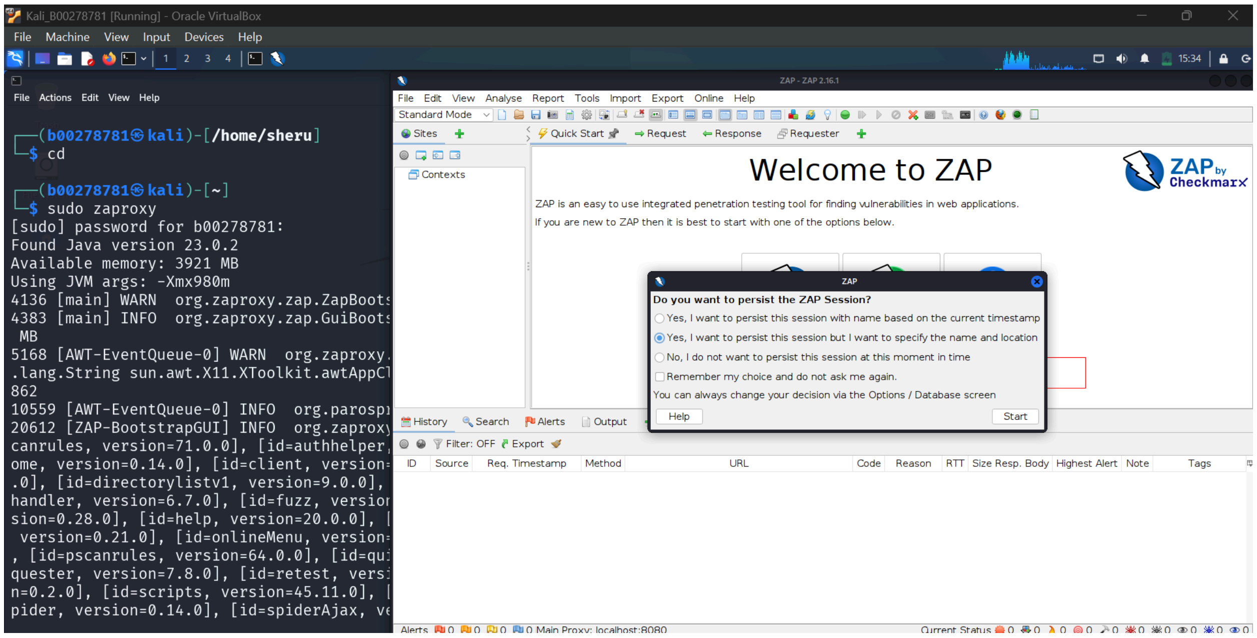


```
┌──(b00278781㉿kali)-[~]
└─$ sudo apt install zaproxy
The following packages were automatically installed and are no longer required:
  crackmapexec                 libibverbs1                  perl-modules-5.38
  dnsmap                       libice-dev                   pyqt5-dev-tools
  figlet                       libimobiledevice6            python-odf-doc
  finger                       libiniparser1                python-odf-tools
  firebird3.0-common           libjim0.82t64                python-tables-data
  firebird3.0-common-doc       libjsoncpp25                 python3-appdirs
```

- Now we will start zaproxy with following command

```
sudo zaproxy
```

- Upon initial launch, one must choose whether to persist the session by saving it to HSQLDB or to opt for a temporary session. Please select "Yes, I want to persist this session, but I want to specify the name and location



Kali_B00278781 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
ZAP - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Welcome to ZAP
ZAP by Checkmarx
History
Search
Alerts
Output
Filter: OFF
Export
ID Source Req. Timestamp Method URL Code Reason RTT Size Resp. Body Highest Alert Note Tags
Main Proxy: localhost:8080
Current Status
ZAP
Help
Start

```
┌──(b00278781㉿kali)-[/home/sheru]
└─$ cd

┌──(b00278781㉿kali)-[~]
└─$ sudo zaproxy
[sudo] password for b00278781:
Found Java version 23.0.2
Available memory: 3921 MB
Using JVM args: -Xmx980m
4136 [main] WARN  org.zaproxy.zap.ZapBoots
4383 [main] INFO  org.zaproxy.zap.GuiBoots
 MB
5168 [AWT-EventQueue-0] WARN  org.zaproxy.
.lang.String sun.awt.X11.XToolkit.awtAppCl
862
10559 [AWT-EventQueue-0] INFO  org.parospr
20612 [ZAP-BootstrapGUI] INFO  org.zaproxy
canrules, version=71.0.0], [id=authhelper,
ome, version=0.14.0], [id=client, version=
.0], [id=directorylistv1, version=9.0.0],
handler, version=6.7.0], [id=fuzz, versior
sion=0.28.0], [id=help, version=20.0.0], [
 version=0.21.0], [id=onlineMenu, version=
, [id=pscanrules, version=64.0.0], [id=qui
quester, version=7.8.0], [id=retest, versi
n=0.2.0], [id=scripts, version=45.11.0], [
pider, version=0.14.0], [id=spiderAjax, ve
```

ZAP is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications.

If you are new to ZAP then it is best to start with one of the options below.

Do you want to persist the ZAP Session?

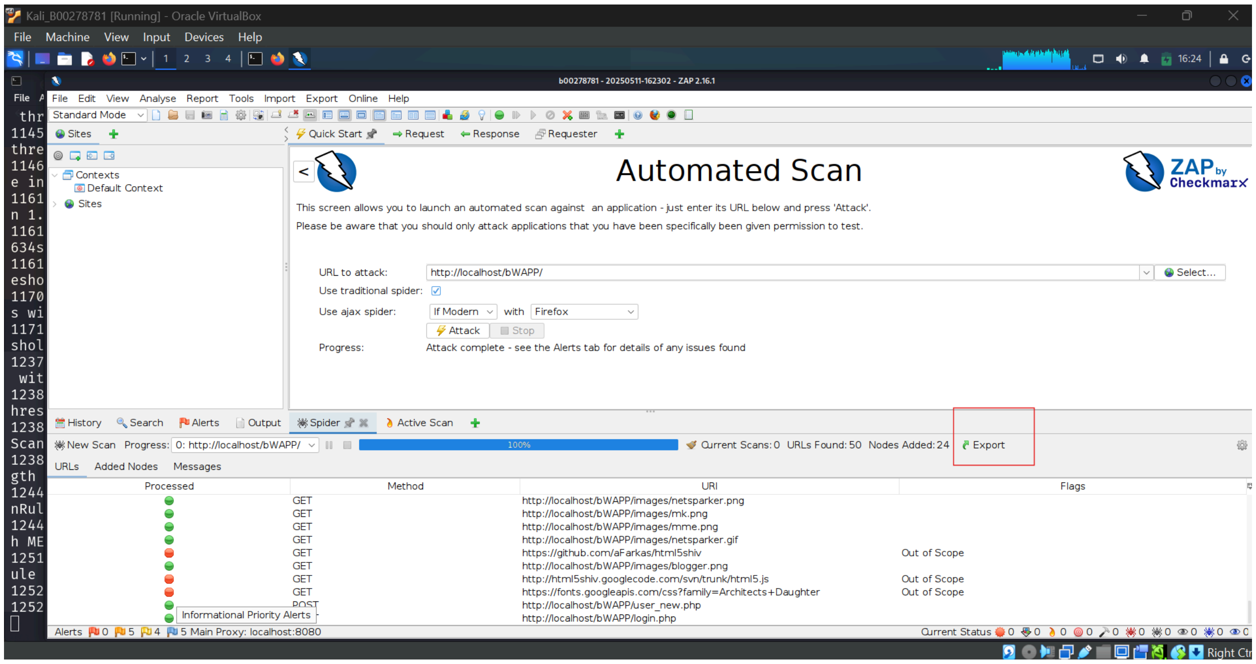
- Yes, I want to persist this session with name based on the current timestamp
- [✓ Yes, I want to persist this session but I want to specify the name and location]
- No, I do not want to persist this session at this moment in time
- ☐ Remember my choice and do not ask me again.

You can always change your decision via the Options / Database screen

Task 1

List down all the URLs/Locations that can be accessed without authentication.

Start quick scan on the target and export the urls from the spider tab into a file



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost/bWAPP/
Use traditional spider: ✓
Use ajax spider: If Modern with Firefox
Progress: Attack complete - see the Alerts tab for details of any issues found

Progress: 0: http://localhost/bWAPP/ 100% Current Scans: 0 URLs Found: 50 Nodes Added: 24

Processed	Method	URI	Flags
🟢	GET	http://localhost/bWAPP/images/netsparker.png	
🟢	GET	http://localhost/bWAPP/images/mk.png	
🟢	GET	http://localhost/bWAPP/images/mme.png	
🟢	GET	http://localhost/bWAPP/images/netsparker.gif	
🔴	GET	https://github.com/aFarkas/html5shiv	Out of Scope
🟢	GET	http://localhost/bWAPP/images/blogger.png	
🔴	GET	http://html5shiv.googlecode.com/svn/trunk/html5.js	Out of Scope
🔴	GET	https://fonts.googleapis.com/css?family=Architects+Daughter	Out of Scope
🟢	POST	http://localhost/bWAPP/user_new.php	
🟢	[illegible]	http://localhost/bWAPP/login.php	

Alerts 0 5 4 5 Main Proxy: localhost:8080

Save them in the file unauthenticated.csv

Here is the list of all the URLS that can be accessed without authentication

http://localhost/robots.txt,Seed

http://localhost/sitemap.xml,Seed

http://localhost/bWAPP/,

http://localhost/bWAPP/portal.php,

http://localhost/bWAPP/login.php,

http://localhost/bWAPP/user_new.php,

http://localhost/bWAPP/info.php,

http://localhost/bWAPP/training.php,

http://localhost/bWAPP/js/html5.js,

http://localhost/bWAPP/stylesheets/stylesheet.css,

http://localhost/bWAPP/images/favicon.ico,

http://localhost/bWAPP/images/twitter.png,

http://localhost/bWAPP/images/linkedin.png,

http://localhost/bWAPP/images/facebook.png,

http://localhost/bWAPP/images/blogger.png,

http://localhost/bWAPP/images/bee_1.png,

http://localhost/bWAPP/images/owasp.png,

http://localhost/bWAPP/images/cc.png,

http://localhost/bWAPP/images/zap.png,

http://localhost/bWAPP/images/netsparker.png,

http://localhost/bWAPP/images/mk.png,

http://localhost/bWAPP/user_new.php,

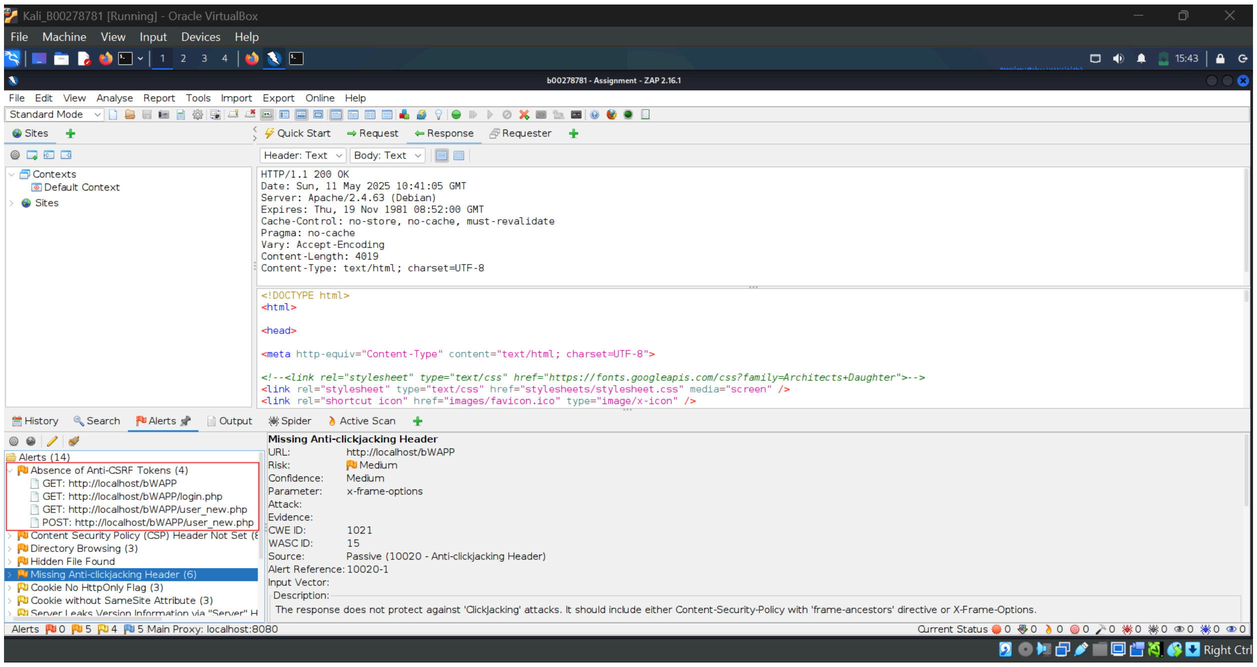
http://localhost/bWAPP/images/mme.png,

http://localhost/bWAPP/images/netsparker.gif, http://localhost/bWAPP/login.php,

Task 2

Investigate all the alerts generated, discuss the vulnerabilities found, and recommend actions to plug in those vulnerabilities. The number of generated alerts depends on the web app, so please provide proof of how many alerts are generated for your case. There may be multiple sub-alerts for an alert; for example, for SQL Injection, you can get multiple examples/URLs for that alert. Please take at least one use case/example for each type of alert generated. Avoid using general statements. For example, instead of stating that the web app is vulnerable to SQL Injection, give appropriate examples/explanations. Rather than saying that the cookie is vulnerable, you can mention that the cookie is vulnerable because it has not been set to HttpOnly and then discuss various settings for cookies and how they impact or how they can be set properly to enhance security.

Absence of Anti-CSRF Token


Kali_B00278781 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
b00278781 - Assignment - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Header: Text
Body: Text
Contexts
Default Context
Sites
History
Search
Alerts
Output
Spider
Active Scan
Right Ctrl

```
HTTP/1.1 200 OK
Date: Sun, 11 May 2025 10:41:05 GMT
Server: Apache/2.4.63 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4019
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<!--<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Architects+Daughter">-->
<link rel="stylesheet" type="text/css" href="stylesheets/stylesheet.css" media="screen" />
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon" />
```

```
Alerts (14)
  Absence of Anti-CSRF Tokens (4)
    GET: http://localhost/bWAPP
    GET: http://localhost/bWAPP/login.php
    GET: http://localhost/bWAPP/user_new.php
    POST: http://localhost/bWAPP/user_new.php
  Content Security Policy (CSP) Header Not Set (
  Directory Browsing (3)
  Hidden File Found
  Missing Anti-clickjacking Header (6)
  Cookie No HttpOnly Flag (3)
  Cookie without SameSite Attribute (3)
  Server Leaks Version Information via "Server" H
```

Missing Anti-clickjacking Header

URL:	http://localhost/bWAPP
Risk:	Medium
Confidence:	Medium
Parameter:	x-frame-options
Attack:	
Evidence:	
CWE ID:	1021
WASC ID:	15
Source:	Passive (10020 - Anti-clickjacking Header)
Alert Reference:	10020-1
Input Vector:	

Description:
The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Alerts 0 5 4 5 Main Proxy: localhost:8080

Current Status 0 0 0 0 0 0 0 0 0 0

An **anti-CSRF token** (also called a **CSRF token** or **synchronizer token**) is a **security measure** used to prevent **Cross-Site Request Forgery (CSRF)** attacks.

What Is a CSRF Attack?

In a **CSRF attack**, a **malicious website** tricks a user (already logged into a secure site) into performing **unwanted actions** (like changing a password or transferring money) without their knowledge.

Example:

- You're logged into `bank.com`.
- A malicious site tricks your browser into sending a request like:

  ```
  http://bank.com/transfer?to=hacker&amount=1000
  ```

- Since you're already logged in, your cookies are sent — and the transfer goes through.

Why CSRF Vulnerabilities Exist

1. **Missing Anti-CSRF Token**

   The application **does not include a CSRF token** in forms or API requests, so it can't tell if the request was made by the real user or by an attacker.

2. **Token Is Not Validated**

   Sometimes the token is present, but the server **does not actually check** if it's correct, or allows reused/expired tokens.

3. **Same-Site Cookie Policy Not Set**

   Browsers send cookies (like session IDs) with cross-site requests unless `SameSite` cookie policy is enforced (`Strict` or `Lax`). Without this, a malicious site can make authenticated requests on behalf of the user.

   ## How to Fix it?

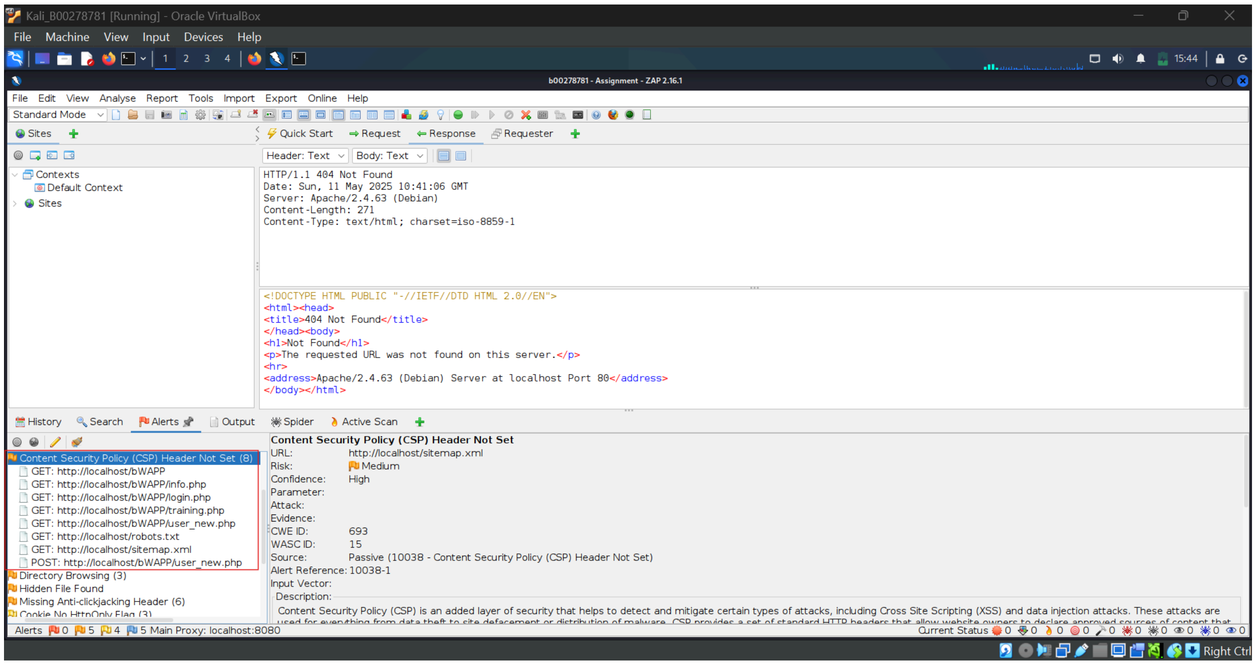
   - Implement CSRF protection using tokens that:
     - Are unique per session or request.

    - Are embedded in HTML forms or request headers.
    - Are validated on the server side.
  - Example implementation:
    - Add hidden input:

```
<input type="hidden" name="csrf_token" value="unique_token_value">
```

- Server verifies the token matches the expected value.
- Frameworks like Laravel, Django, or Express.js with `csurf` middleware have built-in CSRF protection.

Content Security Policy (CSP) Header Not Set


Kali_B00278781 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
b00278781 - Assignment - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Header: Text Body: Text
Contexts
Default Context
Sites
History Search Alerts Output Spider Active Scan

```
HTTP/1.1 404 Not Found
Date: Sun, 11 May 2025 10:41:06 GMT
Server: Apache/2.4.63 (Debian)
Content-Length: 271
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.63 (Debian) Server at localhost Port 80</address>
</body></html>
```

Content Security Policy (CSP) Header Not Set (8)
- GET: http://localhost/bWAPP
- GET: http://localhost/bWAPP/info.php
- GET: http://localhost/bWAPP/login.php
- GET: http://localhost/bWAPP/training.php
- GET: http://localhost/bWAPP/user_new.php
- GET: http://localhost/robots.txt
- GET: http://localhost/sitemap.xml
- POST: http://localhost/bWAPP/user_new.php

Directory Browsing (3)
Hidden File Found
Missing Anti-clickjacking Header (6)
Cookie No HttpOnly Flag (3)

Content Security Policy (CSP) Header Not Set

URL:	http://localhost/sitemap.xml
Risk:	Medium
Confidence:	High
Parameter:	
Attack:	
Evidence:	
CWE ID:	693
WASC ID:	15
Source:	Passive (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference:	10038-1
Input Vector:	

Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that

Alerts 0 5 4 5 Main Proxy: localhost:8080
Current Status 0 0 0 0 0 0 0 0 0 0

What Content Security Policy (CSP)

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

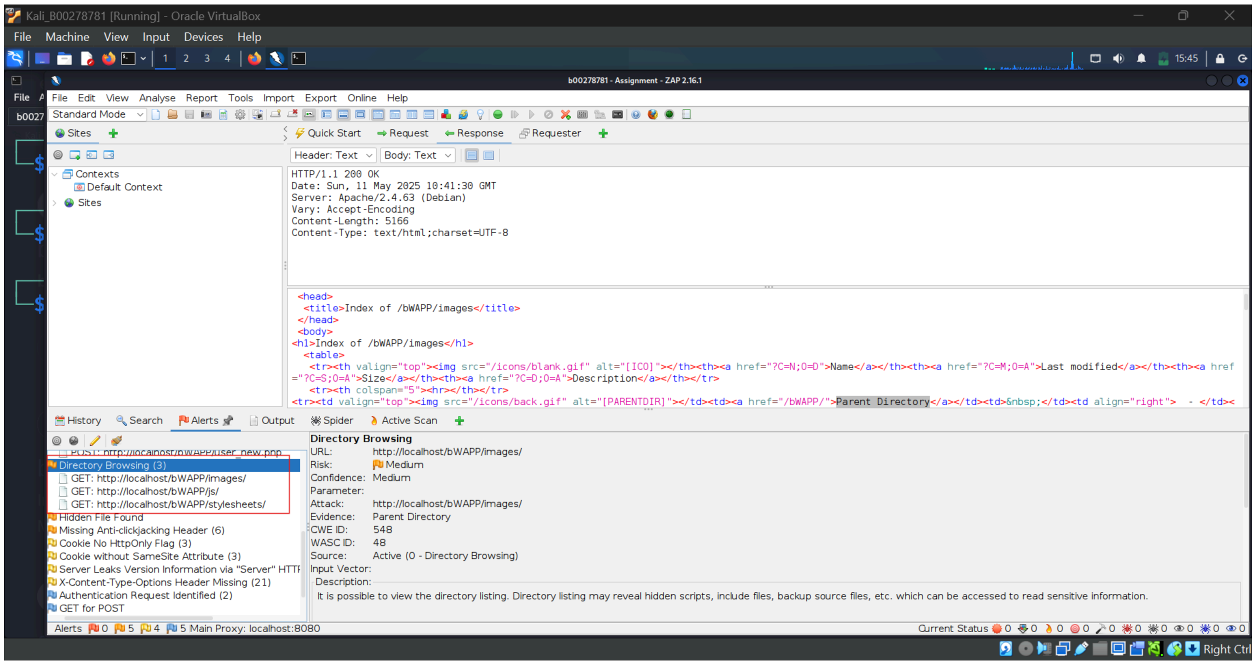
How to Fix this ?

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

- Add a CSP header like:

```
Content-Security-Policy: default-src 'self'; script-src 'self';
```

Directory Traversal



```
HTTP/1.1 200 OK
Date: Sun, 11 May 2025 10:41:30 GMT
Server: Apache/2.4.63 (Debian)
Vary: Accept-Encoding
Content-Length: 5166
Content-Type: text/html;charset=UTF-8
```

```
 <head>
  <title>Index of /bWAPP/images</title>
 </head>
 <body>
<h1>Index of /bWAPP/images</h1>
  <table>
   <tr><th valign="top"><img src="/icons/blank.gif" alt="[ICO]"></th><th><a href="?C=N;O=D">Name</a></th><th><a href="?C=M;O=A">Last modified</a></th><th><a href="?C=S;O=A">Size</a></th><th><a href="?C=D;O=A">Description</a></th></tr>
   <tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"><img src="/icons/back.gif" alt="[PARENTDIR]"></td><td><a href="/bWAPP/">Parent Directory</a></td><td>&nbsp;</td><td align="right">  - </td><
```

```
POST: http://localhost/bWAPP/user_new.php
Directory Browsing (3)
  GET: http://localhost/bWAPP/images/
  GET: http://localhost/bWAPP/js/
  GET: http://localhost/bWAPP/stylesheets/
Hidden File Found
Missing Anti-clickjacking Header (6)
Cookie No HttpOnly Flag (3)
Cookie without SameSite Attribute (3)
Server Leaks Version Information via "Server" HTTP
X-Content-Type-Options Header Missing (21)
Authentication Request Identified (2)
GET for POST
```

```
Directory Browsing
URL:          http://localhost/bWAPP/images/
Risk:         Medium
Confidence:   Medium
Parameter:
Attack:       http://localhost/bWAPP/images/
Evidence:     Parent Directory
CWE ID:       548
WASC ID:      48
Source:       Active (0 - Directory Browsing)
Input Vector:
Description:
It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files, backup source files, etc. which can be accessed to read sensitive information.
```

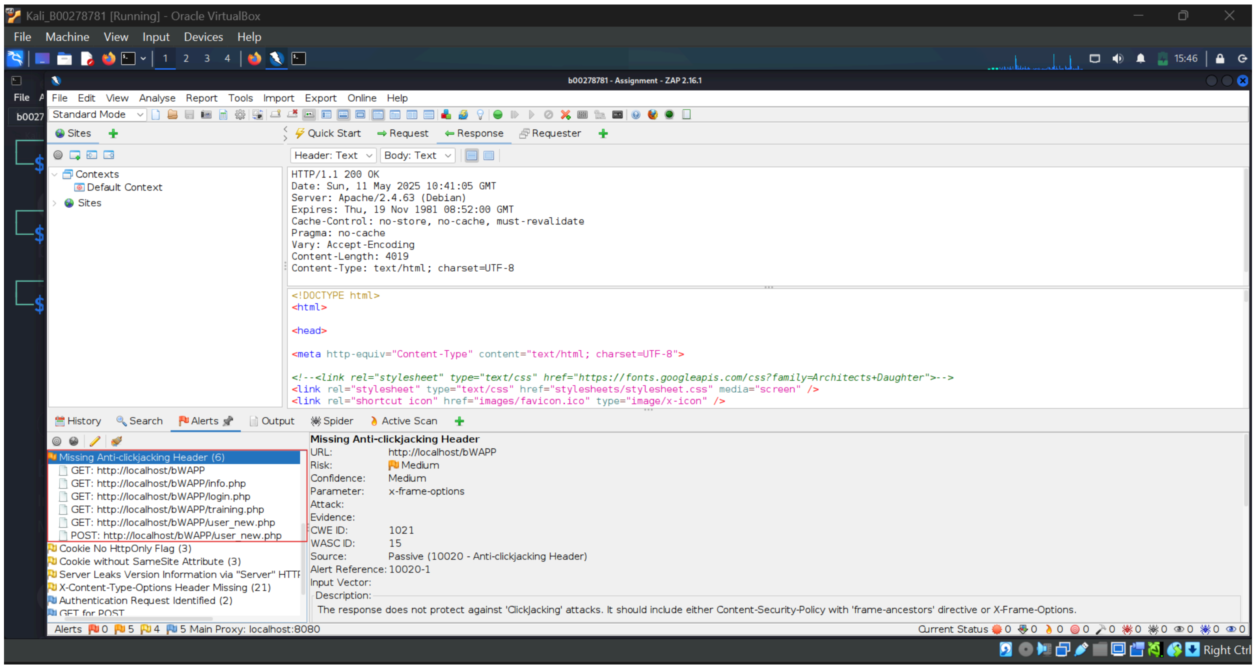
```
Alerts 0 5 4 5 Main Proxy: localhost:8080    Current Status 0 0 0 0 0 0 0 0 0 0
```

Directory listing vulnerability occurs when a web server is **misconfigured to allow users to view the contents of a directory** in the browser — especially when there's no `index.html` or similar default file to load.

How to Fix it?

- Disable directory listing:
    - **Apache:** Use `Options -Indexes`.
    - **Nginx:** Remove `autoindex on;`.

Missing Anti-Clickjacking Token


Kali_B00278781 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
b00278781 - Assignment - ZAP 2.16.1
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start Request Response Requester
Header: Text Body: Text
Contexts
Default Context
Sites
History Search Alerts Output Spider Active Scan

```
HTTP/1.1 200 OK
Date: Sun, 11 May 2025 10:41:05 GMT
Server: Apache/2.4.63 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4019
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<!--<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Architects+Daughter">-->
<link rel="stylesheet" type="text/css" href="stylesheets/stylesheet.css" media="screen" />
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon" />
```

```
Missing Anti-clickjacking Header (6)
  GET: http://localhost/bWAPP
  GET: http://localhost/bWAPP/info.php
  GET: http://localhost/bWAPP/login.php
  GET: http://localhost/bWAPP/training.php
  GET: http://localhost/bWAPP/user_new.php
  POST: http://localhost/bWAPP/user_new.php
Cookie No HttpOnly Flag (3)
Cookie without SameSite Attribute (3)
Server Leaks Version Information via "Server" HTTP
X-Content-Type-Options Header Missing (21)
Authentication Request Identified (2)
GET for POST
```

```
Missing Anti-clickjacking Header
URL:             http://localhost/bWAPP
Risk:            Medium
Confidence:      Medium
Parameter:       x-frame-options
Attack:
Evidence:
CWE ID:          1021
WASC ID:         15
Source:          Passive (10020 - Anti-clickjacking Header)
Alert Reference: 10020-1
Input Vector:
Description:
 The response does not protect against 'ClickJacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.
```

```
Alerts 0 5 4 5 Main Proxy: localhost:8080          Current Status 0 0 0 0 0 0 0 0 0 0
```

A
Missing Anti-Clickjacking Header (or token) vulnerability means that a web application does **not set HTTP response headers that prevent it from being embedded in a frame or iframe** — making it vulnerable to **clickjacking attacks**.

What Is Clickjacking?

Clickjacking is a technique where an attacker tricks a user into clicking something different than what the user perceives — by **embedding the legitimate website into a transparent frame over a malicious page**.

For example, an attacker might:

- Load a bank's website in a hidden iframe
- Ask the victim to "click this button to win a prize"
- But the click actually hits the bank's "Transfer Money" button

Why Missing Headers Are a Problem

If your app **doesn't send proper anti-clickjacking headers**, it can be **framed by another site**, making clickjacking possible.

Example:

No `X-Frame-Options` or `Content-Security-Policy` with `frame-ancestors` directive present in the HTTP response headers.

Explanation:

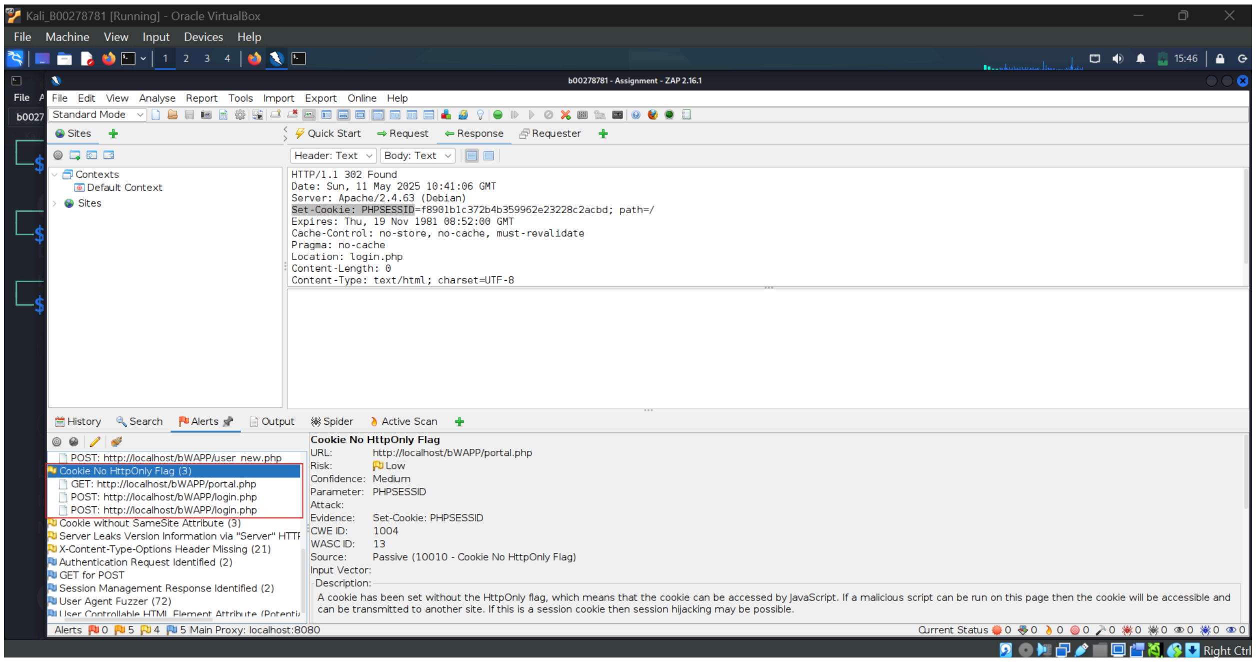
Without these headers, the web page can be embedded into an `<iframe>` on a malicious website, allowing attackers to perform **clickjacking—**tricking users into clicking hidden UI elements, like buttons or links, without their knowledge.

Fix:

- Use the `X-Frame-Options` header to prevent embedding:

  ```
  X-Frame-Options: DENY
  ```

Cookie with NO HttpOnly Flag



```
HTTP/1.1 302 Found
Date: Sun, 11 May 2025 10:41:06 GMT
Server: Apache/2.4.63 (Debian)
Set-Cookie: PHPSESSID=f8901b1c372b4b359962e23228c2acbd; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: login.php
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```

```
POST: http://localhost/bWAPP/user_new.php
Cookie No HttpOnly Flag (3)
  GET: http://localhost/bWAPP/portal.php
  POST: http://localhost/bWAPP/login.php
  POST: http://localhost/bWAPP/login.php
Cookie without SameSite Attribute (3)
Server Leaks Version Information via "Server" HTTI
X-Content-Type-Options Header Missing (21)
Authentication Request Identified (2)
GET for POST
Session Management Response Identified (2)
User Agent Fuzzer (72)
User Controllable HTML Element Attribute (Potenti
```

```
Cookie No HttpOnly Flag
URL:          http://localhost/bWAPP/portal.php
Risk:         Low
Confidence:   Medium
Parameter:    PHPSESSID
Attack:
Evidence:     Set-Cookie: PHPSESSID
CWE ID:       1004
WASC ID:      13
Source:       Passive (10010 - Cookie No HttpOnly Flag)
Input Vector:
Description:
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.
```

```
Alerts 0 5 4 5 Main Proxy: localhost:8080      Current Status 0 0 0 0 0 0 0 0 0 0
```

A **cookie without the `HttpOnly` flag** is vulnerable to **client-side access**, specifically from JavaScript running in the browser. This introduces a **Cross-Site Scripting (XSS)** risk, where an attacker can steal sensitive cookies like session tokens using malicious JavaScript.

What is the `HttpOnly` flag?

- When set, the cookie **cannot** be accessed via `document.cookie` in JavaScript.
- It is meant to **prevent theft of cookies** via XSS attacks.

How to Fix This?

Set the `HttpOnly` Flag on Cookies

This is the **most direct fix**.

When setting cookies that store sensitive data like session tokens, add the `HttpOnly` attribute:

Example in HTTP Header:

```
Set-Cookie: sessionId=abc123; HttpOnly; Secure; SameSite=Strict
```