Future Interns - Cyber Security Task 1

Web Application Security Testing

Task:

- 1. Set up and explore a test web app (like DVWA or OWASP Juice Shop)
- 2. Use scanning tools like OWASP ZAP, Burp Suite, or Nikto
- 3. Test for common vulnerabilities like SQL injection, XSS, and CSRF
- 4. Map the vulnerabilities to OWASP Top 10 threats
- 5. Document findings with screenshots, impact level, and remediation steps
- 6. Compile a Security Assessment Report (PDF format)

Key Features to Include

- At least 3–5 real vulnerabilities found and documented
- Screenshots of attack vectors and scanner outputs
- Mitigation steps for each vulnerability
- OWASP Top 10 Checklist mapping
- A polished Security Report (PDF) that simulates client work

Final Deliverables

- PDF Security Report with risk rating, screenshots, and suggestions
- OWASP Top 10 Compliance checklist
- Tool logs (ZAP scan reports, Burp Suite issues, etc.)
- (Optional) Video walkthrough of your findings

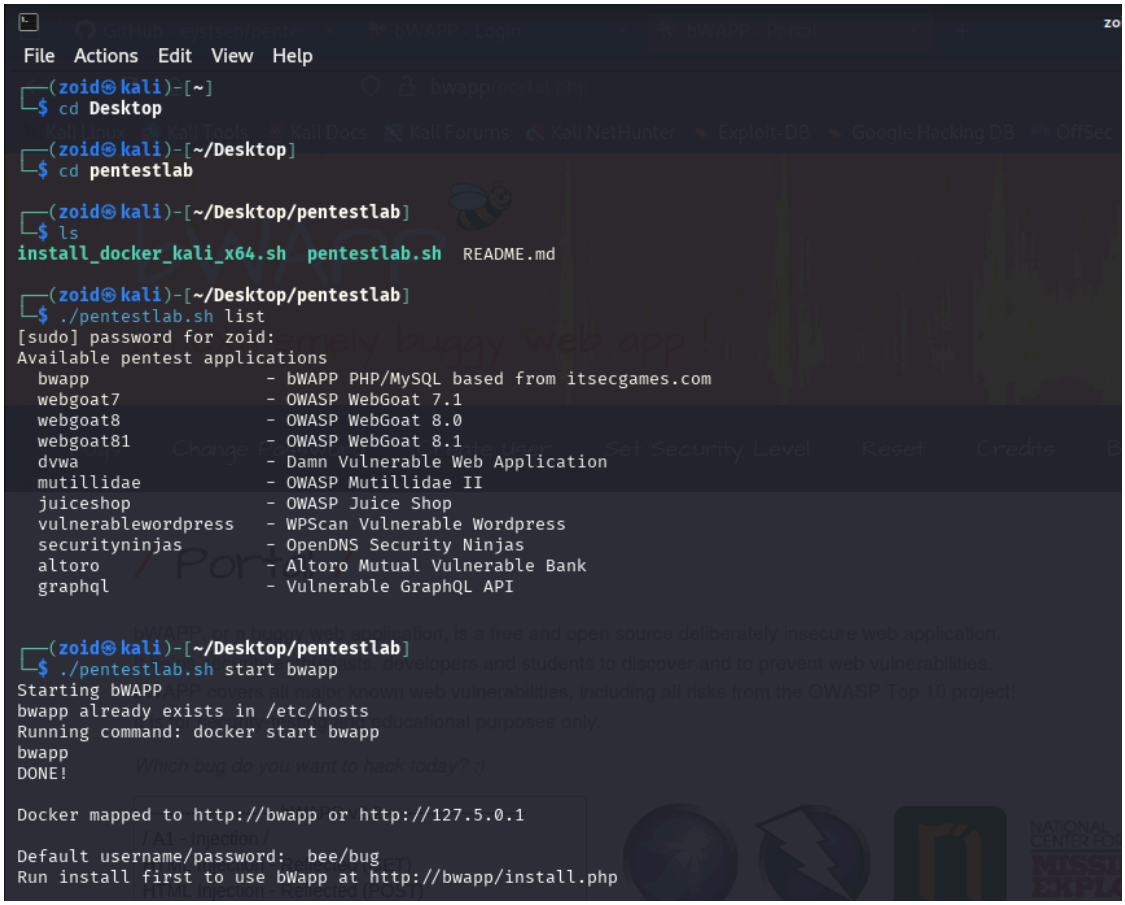
Task 1: Set up and explore a test web app (bWapp)

Target website: bWAPP, Link: http://www.itsecgames.com/

Setting up the target website.

- Copied the https code from github https://github.com/eystsen/pentestlab.git
- Cloned this into my terminal
- Now this belongs to my directory. (/Desktop/pentestlab)
- Before starting anything, I installed docker.io

- Now that everything has been installed, we are ready to start any of the labs.



Exploring - bWAPP

After logging in with the default credentials from the terminal (bee/bug), I began exploring the application before carrying on task 2 (running any automated scans).

- **Login system:** The app requires authentication, including sessions and cookies will be important for testing.

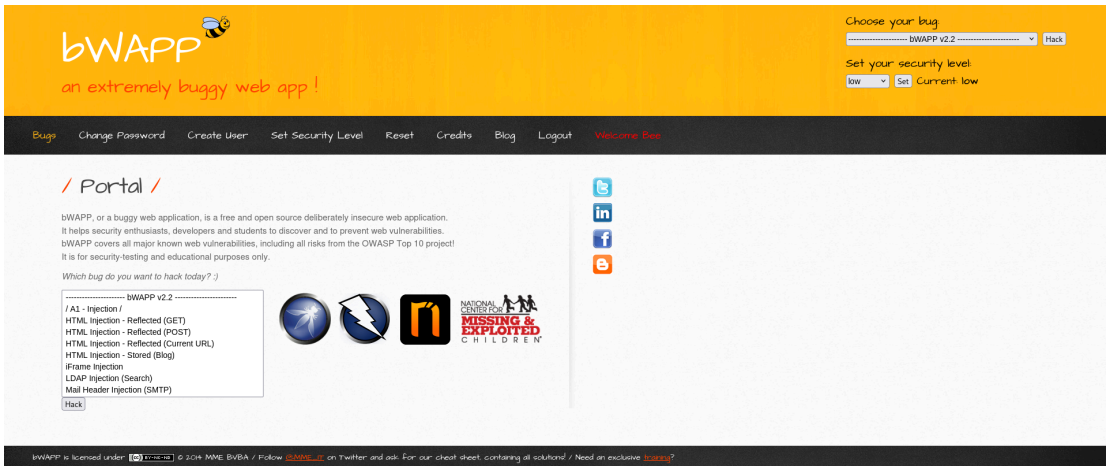
- **Bug Selection:** The dashboard allows choosing different vulnerabilities from the dropdown menu (e.g. SQL injection, XSS, CSRF). It also has a

Security Level setting (low, medium, high) which is designed to control how difficult it is to exploit these flaws.

- **Navigation:** The interface exposes multiple functional pages, such as profile pages, message boards, forms, and file upload features.

- **Technology stack clues:** The app displays PHP errors in some cases, suggesting it runs on a PHP/MySQL backend,

- **Possible ways attackers can get in:**
  - Login form (likely vulnerable to SQL Injection).
  - Input fields in search/messages
  - File upload functionality
  - Cookies/session handling (this may allow manipulation)

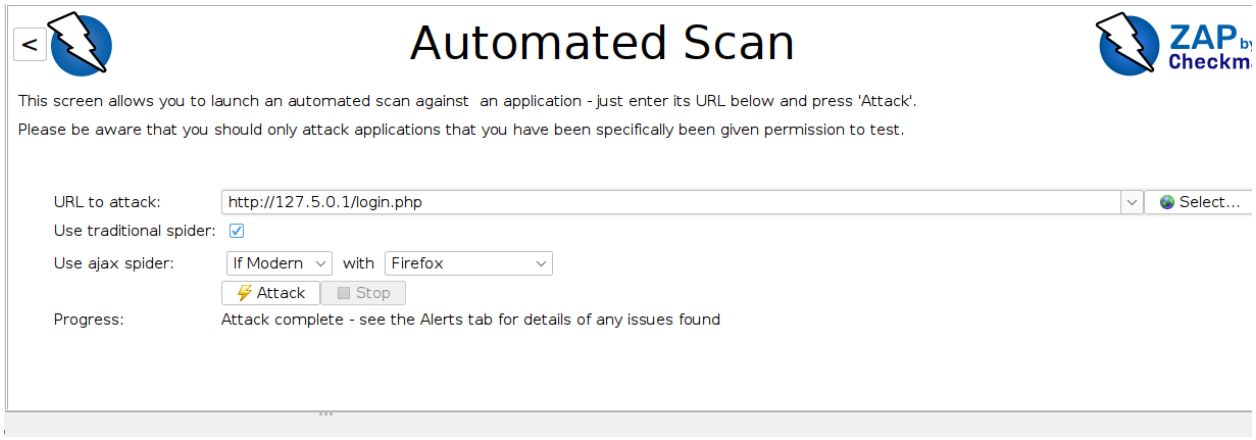


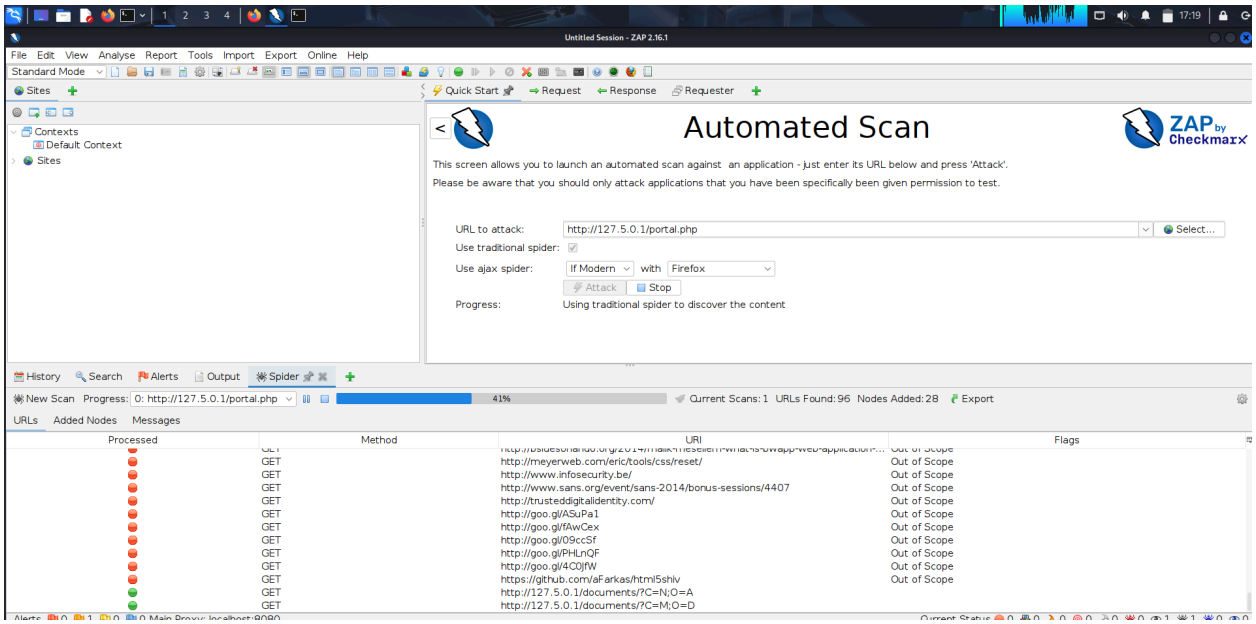
Task 2: Use scanning tools like OWASP ZAP, Burp Suite, Or Nikto.

How I Used OWASP ZAP for Automated Scanning

I used OWASP ZAP (version 2.16.1) on Kalin Linux to scan the bWAPP application by entering its local URL into the “URL to attack” field and running the automated scan by clicking the “attack” button after I entered the target website URL. Zap first crawled the site to discover available pages and then actively tested those pages for vulnerabilities, showing the output in the alert tab with certain ratings. I confirmed the findings by copying the suspicious URLs (such as /phpinfo.php) into my browser and even checking the replaying request in ZAP’s Requester tab to see if the issue could be exploited.

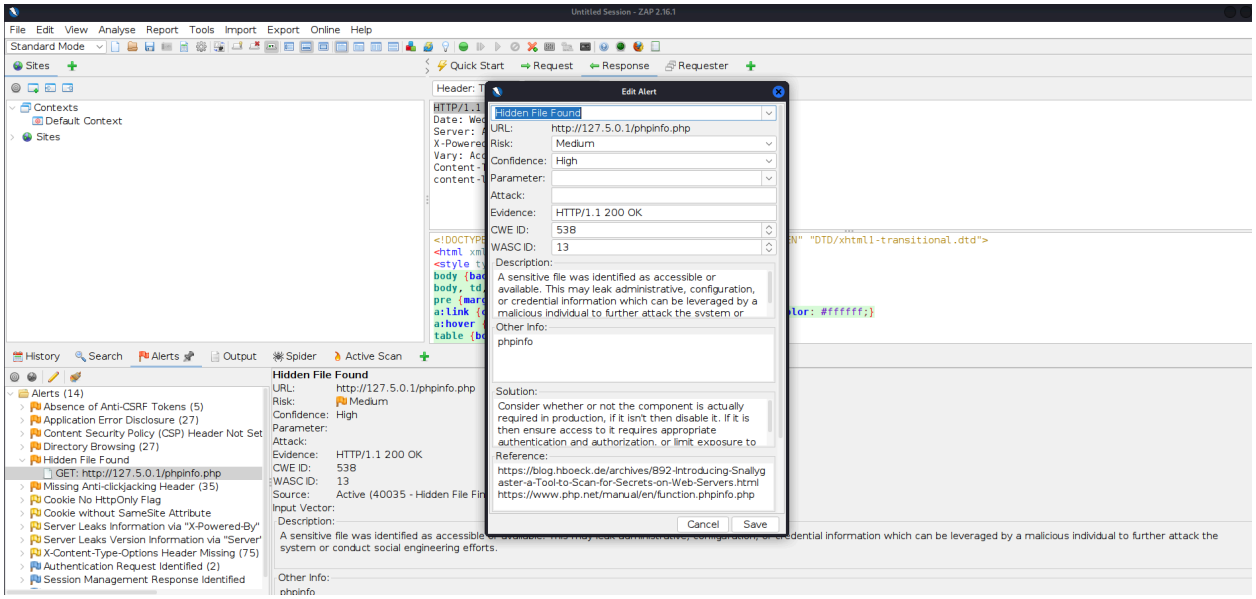
For each confirmed issue, I took screenshots, assessed the harm level and noted key solutions in my report.





As you can see, the zap is currently scanning the bWAPP website to identify any weakness.

The picture below shows the complete scan. Zap managed to identify all the website weaknesses, though not all is true, you must confirm the weakness to see if it's true or false. You can do this by going towards the alert section and double clicking on the vulnerability.



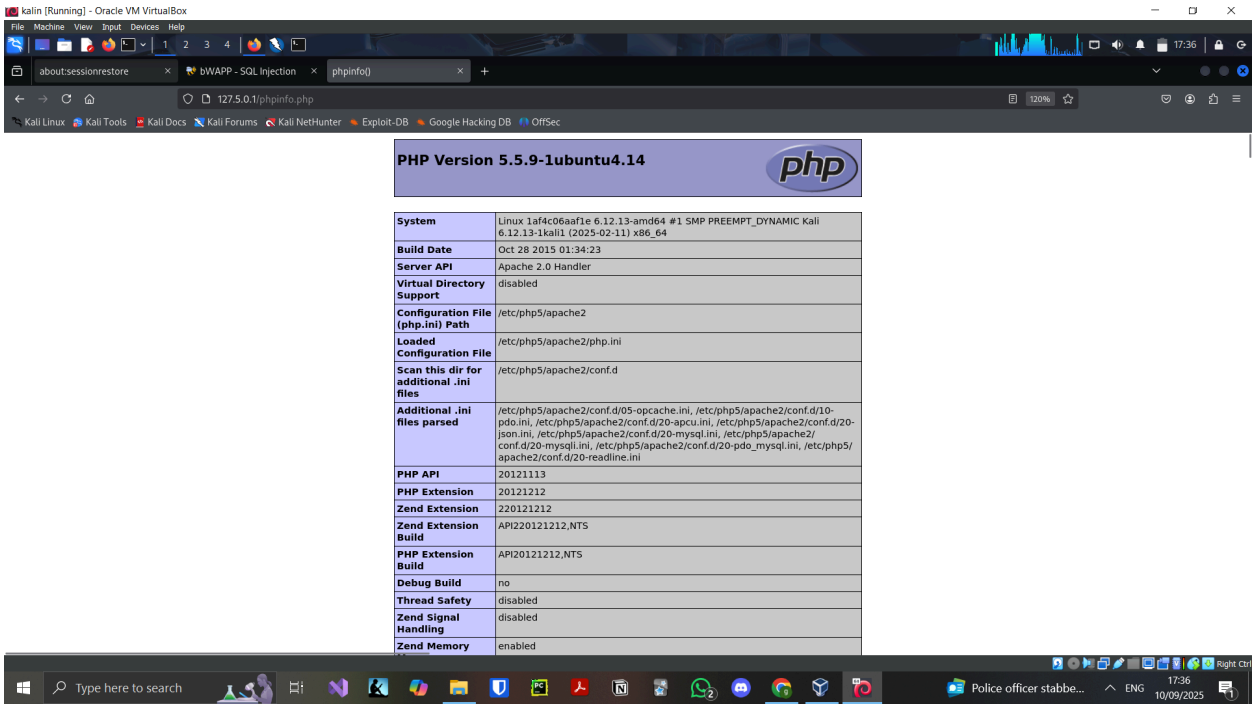
After double clicking any vulnerability that ZAP believes it found, you can see it shows you all the information about that certain vulnerability. It provides you information such as the confidence level (In this case, its high so zap believes its something you must deal with before it's too late.) and the list goes on.

Task 3: Test for common vulnerabilities like SQL injection, XSS, and CSRF

Vulnerability 1.Hidden File Disclosure - /phpinfo.php

Risk rating: Medium
Description: This files exposes a sensitive file "phpinfo.php" that may leak certain factors like administrative, configuration , or credential information which can be leveraged by an attacker that will have the ability to attack the system or conduct an external attack "social engineering"

Evidence: I have confirmed this by visiting http://127.0.0.1/phpinfo.php directly on the firefox browser.



Impact: Attackers can use this piece of information to identify any vulnerabilities in the PHP version system.

Mitigation: Check if the component is actually required in the production, if it is not then make sure to disable it! If it is then make sure access to this php version requires authentication and authorization.

Vulnerability 2.Missing Anti-clickjacking Header

Risk rating: Medium

Description: This application does not contain HTTP security headers (such as X-FRAME-OPTIONS, etc). This response does not defend against “clickjacking” attacks. It should contain a Content-Security-Policy with “frame-ancestors” directive.

Evidence: The zap scanner flagged this issue (“Missing Anti-clickjacking Header”, alert). The url that was affected by this was http://127.5.0.1/admin/

Missing Anti-clickjacking Header

URL:	http://127.5.0.1/admin/
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

```
HTTP/1.1 200 OK
Date: Tue, 09 Sep 2025 20:19:47 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Vary: Accept-Encoding
Content-Length: 3160
Content-Type: text/html
```

Impact: Without these protections, a hacker can load the site inside a hidden frame on another website. This trick can fool users into clicking buttons or filling forms without knowing it. As a result, attackers make users do things they don't want to do. This can be things like money transfer.

Mitigation: Modern web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. Test after deployment using certain scanning tools like OWASP ZAP and make sure to regularly review headers to ensure alignment with security best practices.

<u>Vulnerability 3.Cookie No HttpOnly Flag</u>

Risk rating: Medium

Description: The cookie has been set without the HttpOnlyFlag this means that those cookies can be accessed through javascript. This means if a malicious script is to run on this page then the cookie will be accessible and transmitted through another site. If this is the session cookie then a session hijacking attack is possible.

Evidence: ZAP highlighted cookies accessible by javascript.

```
HTTP/1.1 200 OK
Date: Tue, 09 Sep 2025 20:19:46 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Set-Cookie: PHPSESSID=gvfrldb48v7kf39qq0qapp2ct4; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4013
Content-Type: text/html
```

Impact: If the hacker exploits the XSS, they could steal cookies and hjack sessions.
Mitigation: Make sure the HttpOnly Flag is set for all cookies

Task 4: Map the vulnerabilities to OWASP Top 10 threats

Vulnerability	Evidence (Alert/Screen shot)	Impact Level	Mitigation	OWASP Top 10 Category
Hidden File Discloser	Phpinfo.php page shown. http://127.0.0.1/phpinfo.php	Medium	Remove/Limit access to phpinfo.php	A05:2021-SECURITY MISCONFIGURATIONS
Missing Anti-clickjacking Header	The admin page url affected.	Medium	Ensure one of them is set on all web pages returned by your site/app	A05:2021-SECURITY MISCONFIGURATIONS
Cookie No HttpOnly Flag	Set - Cookie: PHPSESSID	Low	Make sure the HttpOnly Flag is set for all cookies	A02:2021 – Cryptographic Failures