Internship Report On

“CYBER SECURITY”

At

The Red Users

(20 Oct 2024 – 20 Nov 2024)

Submitted By

Name of Student

SHINDE VYANKATESH DATTATRAY

Task 2: Introduction to Web Application Security

Introduction:

What is a Web Application?

A web application is a software application that runs on a web server and can be accessed through a web browser over the internet or an intranet. Unlike traditional desktop applications, web applications require no installation on a user's device, making them widely accessible and versatile. Common examples of web applications include online banking systems, social media platforms, and e-commerce sites.

What is Web Application Security?

Web application security refers to the measures taken to protect web applications from threats and vulnerabilities. It involves ensuring the confidentiality, integrity, and availability of data processed by web applications. Security measures include input validation, authentication, encryption, and regular security assessments to prevent unauthorized access and data breaches. As web applications often handle sensitive user data, robust security practices are essential to protect against various threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

What is OWASP ZAP?

OWASP Zed Attack Proxy (ZAP) is an open-source web application security scanner used to identify vulnerabilities in web applications. Developed by the Open Web Application Security Project (OWASP), ZAP provides automated scanners as well as various tools to assist with manual testing. It is designed to be user-friendly for those new to web security while offering advanced features for experienced professionals. ZAP helps security testers discover common vulnerabilities, analyze web application behavior, and report security issues effectively.

What is WebGoat?

WebGoat is an intentionally vulnerable web application created by OWASP for educational purposes. It serves as a platform for security practitioners and developers to learn about common web application vulnerabilities and how to

exploit and mitigate them. WebGoat provides a hands-on experience, allowing users to explore vulnerabilities in a controlled environment, thereby enhancing their understanding of web application security.

What is CVE?

The Common Vulnerabilities and Exposures (CVE) system is a public database of known cybersecurity vulnerabilities and exposures. Each CVE entry provides a unique identifier, a description of the vulnerability, and references to additional information. The CVE database helps organizations share information about vulnerabilities consistently and enables security professionals to identify and address known threats in their systems.

Setting Up webgoat on Kali Linux

- To run webgoat as a standalone JAR file on Kali Linux, here are the main requirements

System Requirements

- **Operating System**: Kali Linux or any Linux distribution with Java support

Software Requirements

- **Java Development Kit (JDK)**: webgoat requires Java 8 or higher.
- **Web Browser**: A modern web browser (Firefox, Chrome, or any browser on Kali Linux) to access webgoat's web interface.

Network Requirements

- **Localhost Access**: webgoat typically runs on localhost:8080, so ensure that port 8080 is available. If you need a different port, specify it as described previously.
- **Internet Connection**: Required only for downloading webgoat and Java (if not pre-installed); webgoat can run offline once downloaded.

User Permissions

- **Administrative Privileges**: Required to install Java and other dependencies on the system. Running webgoat itself does not require root access.

Downloading and Configuring WebGoat on Kali Linux

Download the WebGoat JAR File

- Open your terminal in Kali Linux.
- Use the following **wget** command to download the WebGoat standalone JAR file directly from the GitHub repository
- Wgethttps://github.com/WebGoat/WebGoat/releases/download/v8.2.1/webgoat-server-8.2.1.jar
- The file will download to your current directory. Confirm the download by listing the directory contents
- You should see **webgoat-server-8.2.1.jar** in the list.

Configure Java Environment

- WebGoat requires Java to run. First, check if Java is installed

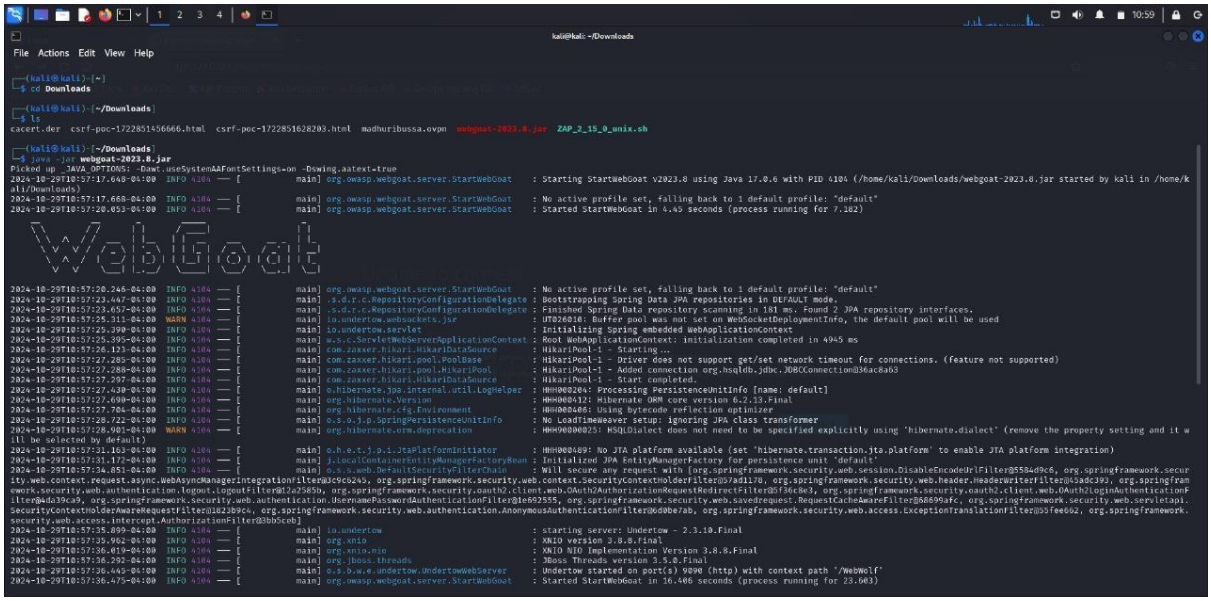
  java –version

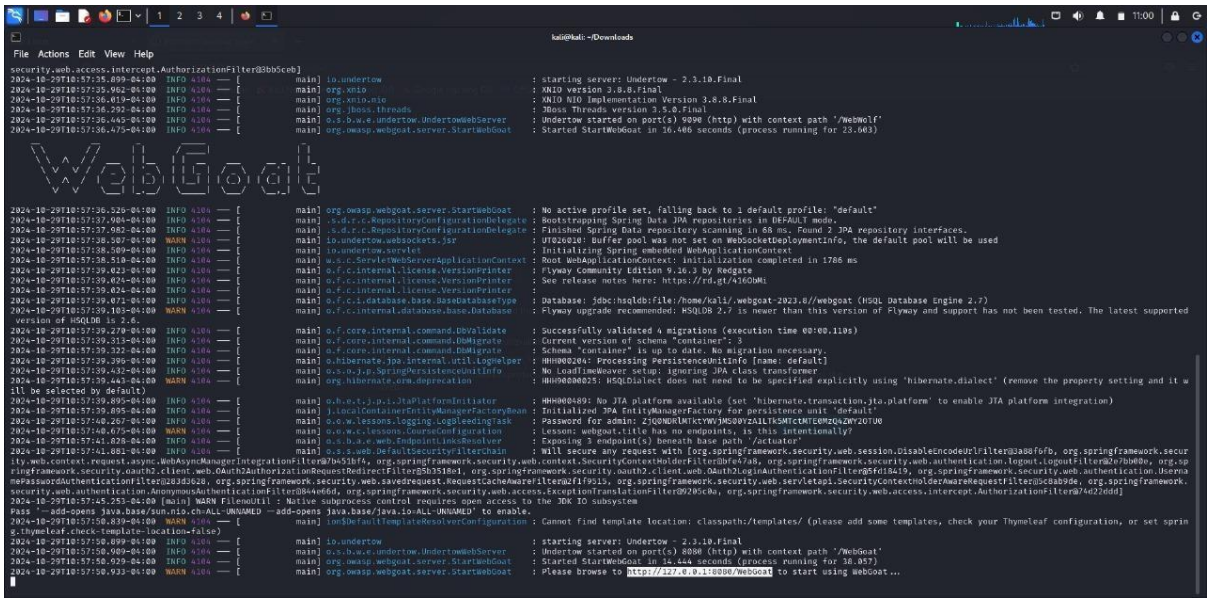
Run the WebGoat Server

- In the terminal, navigate to the directory where the WebGoat JAR file is located. Then, start the server by entering

  java -jar webgoat-server-8.2.1.jar

The terminal will display output as WebGoat starts up. By default, WebGoat runs on port 8080.



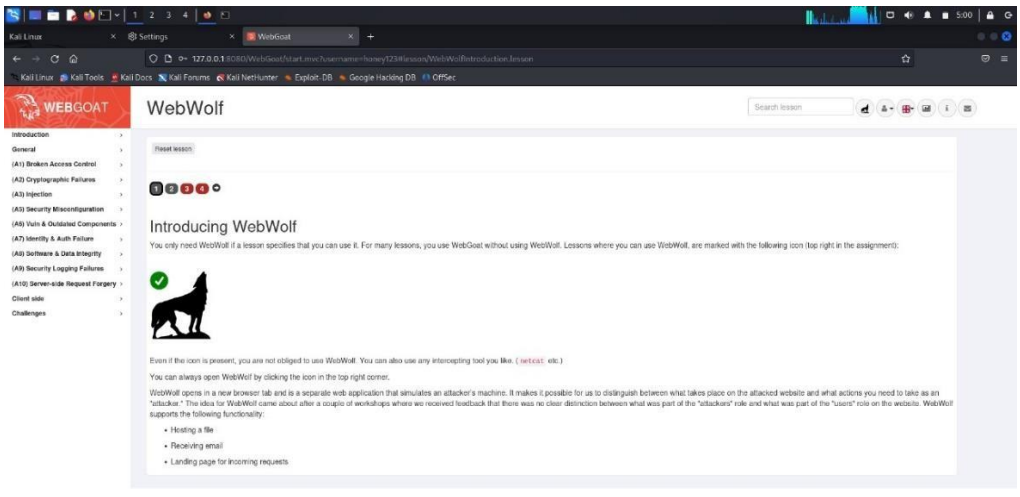


Access WebGoat in the Browser

- Open a web browser in Kali Linux and go to http://localhost:8080/WebGoat
- On your first access, you'll be prompted to create a username and password for future logins.

Initial Configuration and Testing

- Once logged in, explore the modules and confirm WebGoat is fully operational.
- If WebGoat fails to start, ensure port `8080` is free or specify a different port by adding --server.port=<PORT_NUMBER>



Setting Up OWASP ZAP on Kali Linux

1. Update Your System

- Begin by updating your package list to ensure you have the latest information on available packages.

sudo apt update

2. Install OWASP ZAP

- OWASP ZAP is available in the Kali Linux repositories, so you can install it directly with the following command:

sudo apt install zaproxy -y

- This will download and install OWASP ZAP along with its dependencies.

3. Launch OWASP ZAP

- After installation, you can launch OWASP ZAP directly from the terminal by typing:

zaproxy

- Alternatively, you can open it through the applications menu under **Web Application Analysis**.

4. Initial Configuration

- On the first launch, you'll see a setup wizard that guides you through the basic configuration.
- Choose whether you want to persist your ZAP session data (helpful for saving scan data).
- You may see options to configure a local proxy. By default, ZAP runs on localhost:8080, and this will work for most setups.

5. Configuring Browser Proxy for ZAP

- For ZAP to intercept and analyze traffic, you need to configure your browser to use ZAP as a proxy:
    - **Proxy Address**: localhost
    - **Port**: 8080
- In Firefox, you can set this in **Settings > Network Settings > Manual Proxy Configuration**.

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update -y
[sudo] password for kali:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
```

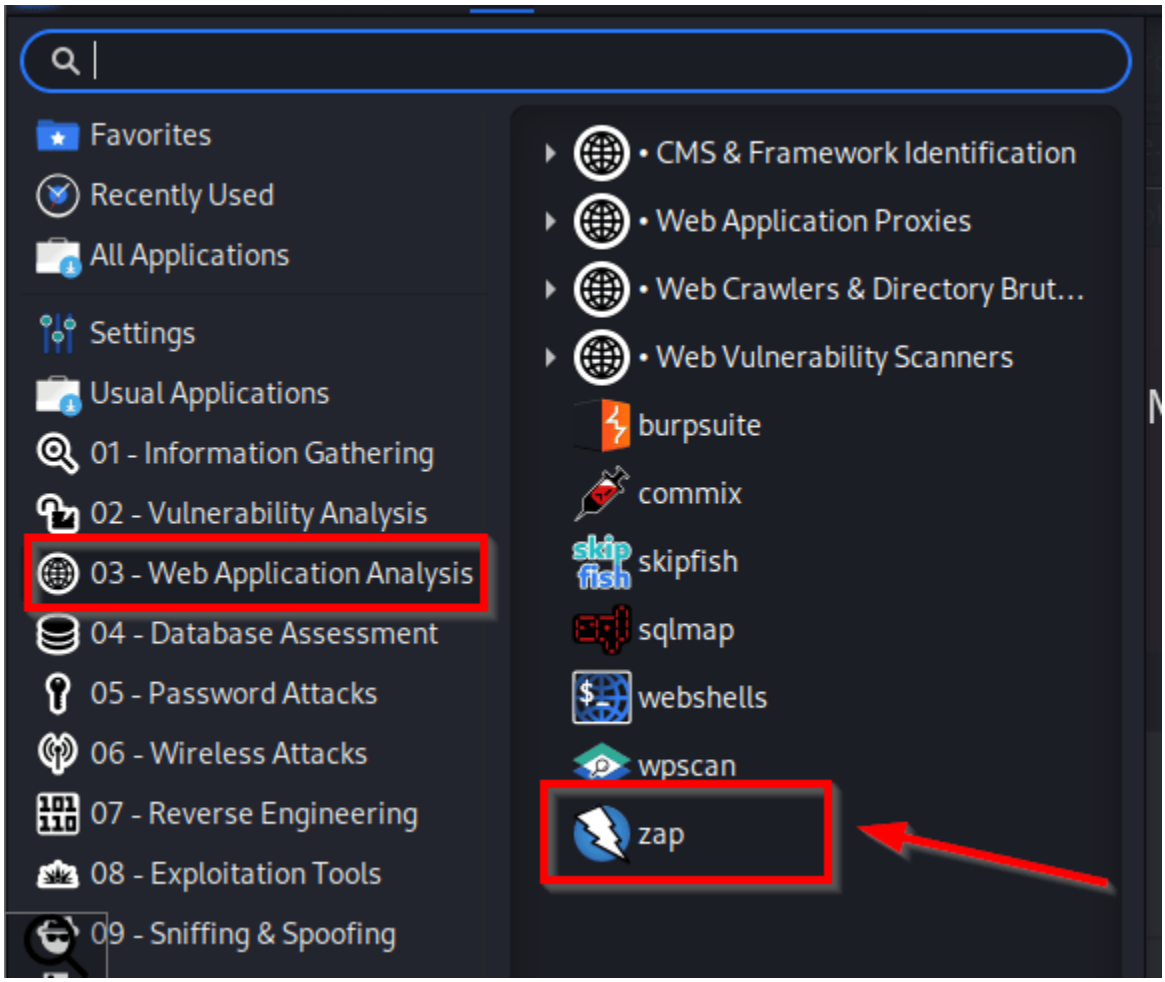
```
┌──(kali㉿kali)-[~]
└─$ sudo apt install zaproxy
[sudo] password for kali:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following packages were automatically installed and are no longer required:
  libboost-dev libboost1.83-dev libopenblas-dev libopenblas-pthread-dev libopenblas0 libpython3-all-dev libpython3.12
  libpython3.12-dev libxsimd-dev python3-all-dev python3-beniget python3-gast python3-pythran python3.12-dev xtl-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  zaproxy
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 197 MB of archives.
After this operation, 248 MB of additional disk space will be used.
Get:1 http://mirror.accuris.ca/kali kali-rolling/main amd64 zaproxy all 2.14.0-0kali1 [197 MB]
Fetched 197 MB in 5s (40.4 MB/s)
Selecting previously unselected package zaproxy.
(Reading database ... 413600 files and directories currently installed.)
Preparing to unpack .../zaproxy_2.14.0-0kali1_all.deb ...
Unpacking zaproxy (2.14.0-0kali1) ...
Setting up zaproxy (2.14.0-0kali1) ...
Processing triggers for kali-menu (2023.4.7) ...
```

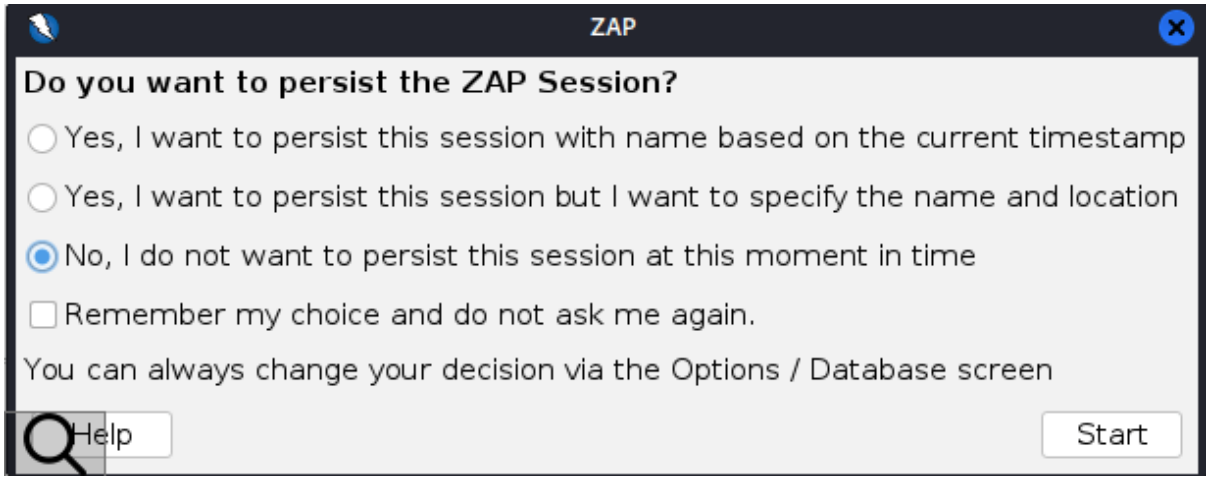
Starting ZAP

You can start ZAP in Kali in one of two ways: by entering zaproxy in the terminal or by opening it from the application menu under “Web Application Analysis.

When you start ZAP, you will see a screen asking, “Do you want to keep this ZAP session?

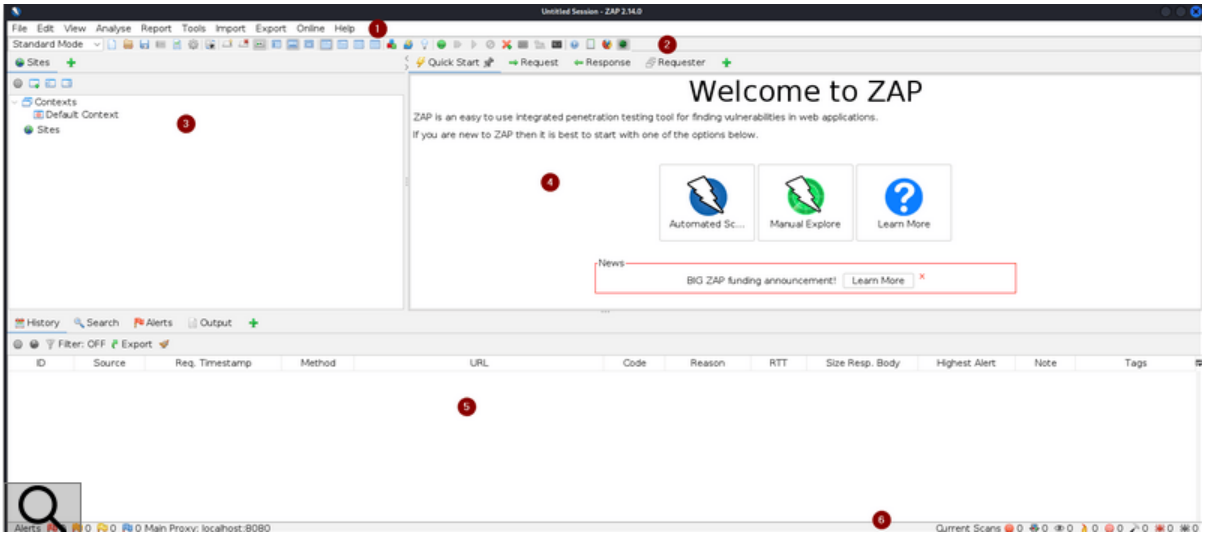
We’ll select “No, I don’t want to keep this session right now.”


Favorites
Recently Used
All Applications
Settings
Usual Applications
01 - Information Gathering
02 - Vulnerability Analysis
03 - Web Application Analysis
04 - Database Assessment
05 - Password Attacks
06 - Wireless Attacks
07 - Reverse Engineering
08 - Exploitation Tools
09 - Sniffing & Spoofing
• CMS & Framework Identification
• Web Application Proxies
• Web Crawlers & Directory Brut...
• Web Vulnerability Scanners
burpsuite
commix
skipfish
sqlmap
webshells
wpscan
zap


ZAP
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
Help
Start

Overview of ZAP

Before we start using ZAP, let's take a look at the main interface and show where some of the main features are located. The interface has a lot of information, but remember, ZAP does a lot of things.

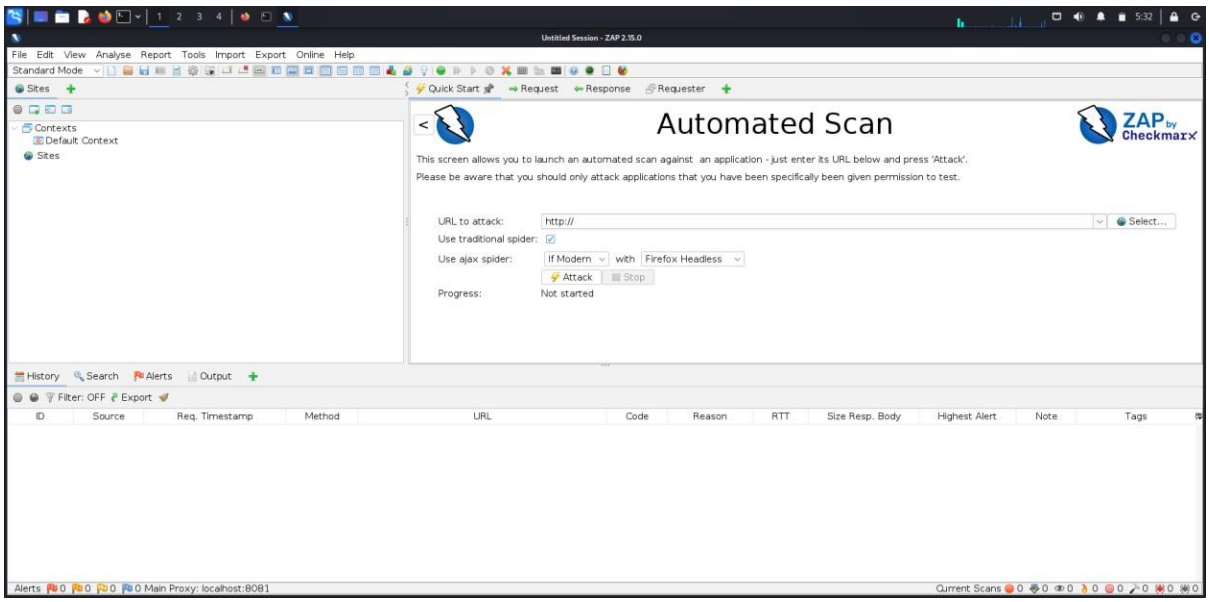


- **Menu:** Here you can create and manage sessions, generate reports, find tools, get help and more.
- **Toolbar:** It includes buttons that provide shortcuts to the most frequently used features.
- **Tree Window:** Displays the hierarchical view of the site you are testing and the script tree.
- **Quick Start/Task Window:** This is a quick and easy way to use ZAP, especially if you are new to it. It also displays requests, responses, and scripts that you can edit.
- **History tab:** Displays a log of all HTTP requests and responses sent and received through ZAP.
- **Search Tab:** Allows you to search through requests and responses.
- **Notifications Tab:** Displays security alerts found during scans.
- **Exit Tab:** Provides detailed output from various scans and processes.
- **Footer:** Displays ZAP status information.

- **Notification Counter:** Displays a summary of the notifications found. It is colored (such as red, yellow, etc.) to represent the severity of alerts found during the scanning process.
- **Main Proxy:** This is the primary proxy setting that ZAP uses, which is set to “localhost:8080.”
- **Current Scans:** This section displays any current scans, with icons indicating their status or progress.

Analyzing Web Application Vulnerabilities in WebGoat with OWASP ZAP

Open OWASP ZAP on your Kali Linux machine by executing zaproxy in the terminal or through your applications menu

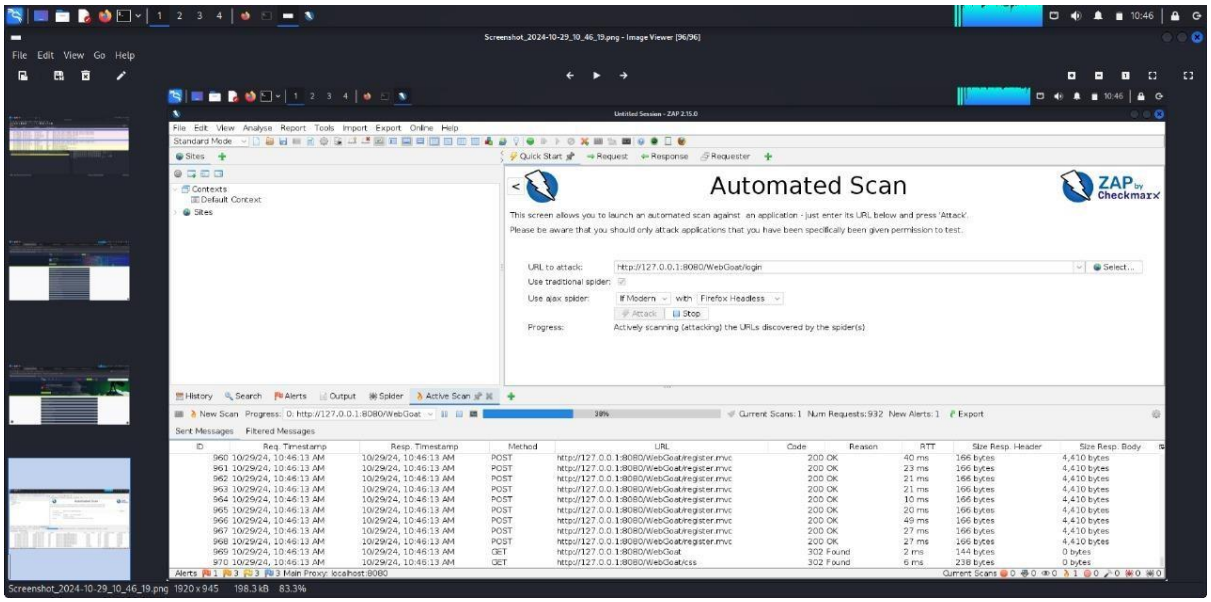


Create a New Session

- If you want to start with a clean slate, go to File > New Session to create a new session. You can choose not to save the previous session if prompted.

Directly Input the WebGoat URL

- In the ZAP interface, look for the **URL to attack** field, typically located at the top of the window.



Start the Active Scan

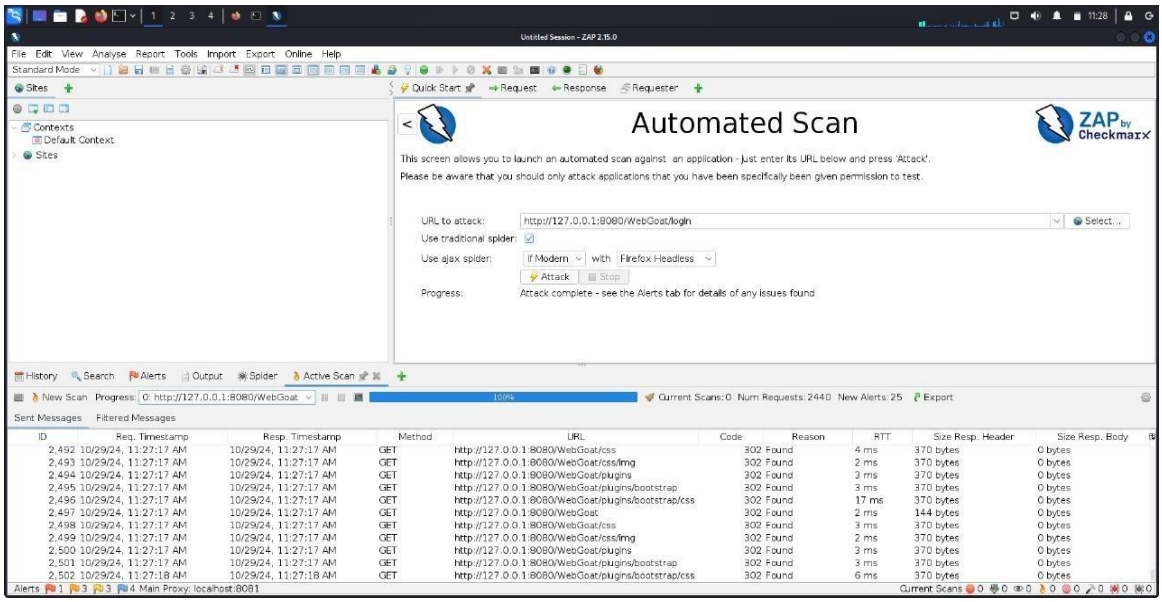
- After entering the WebGoat URL, you will see an option to Attack in the main menu.
- Click on Attack and then select Active Scan from the dropdown menu.

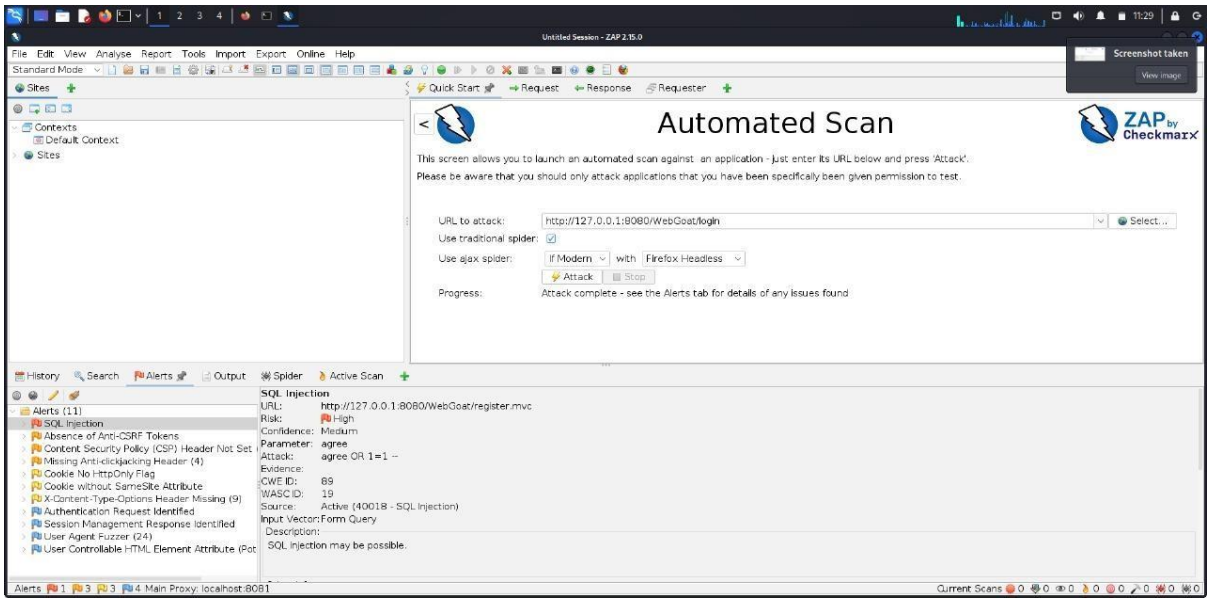
Configure Active Scan Settings

- A dialog box will appear with settings for the active scan.
- Ensure the Context is set to the default (or select the appropriate context if you have configured one).
- Review the options and click OK to initiate the scan.

Monitor the Scan Progress

- As the active scan runs, you can monitor its progress in the Active Scan tab at the bottom of the ZAP interface.
- Vulnerabilities will start populating in the Alerts tab as they are discovered.

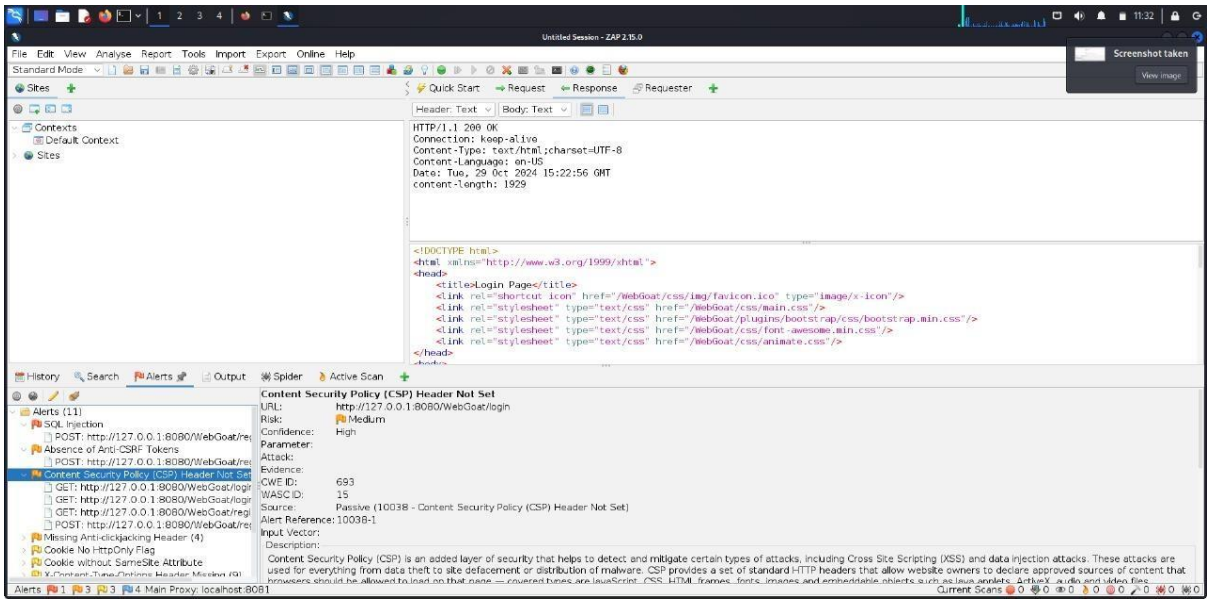




Once the scan is complete, go to the **Alerts** tab to review all the vulnerabilities found.

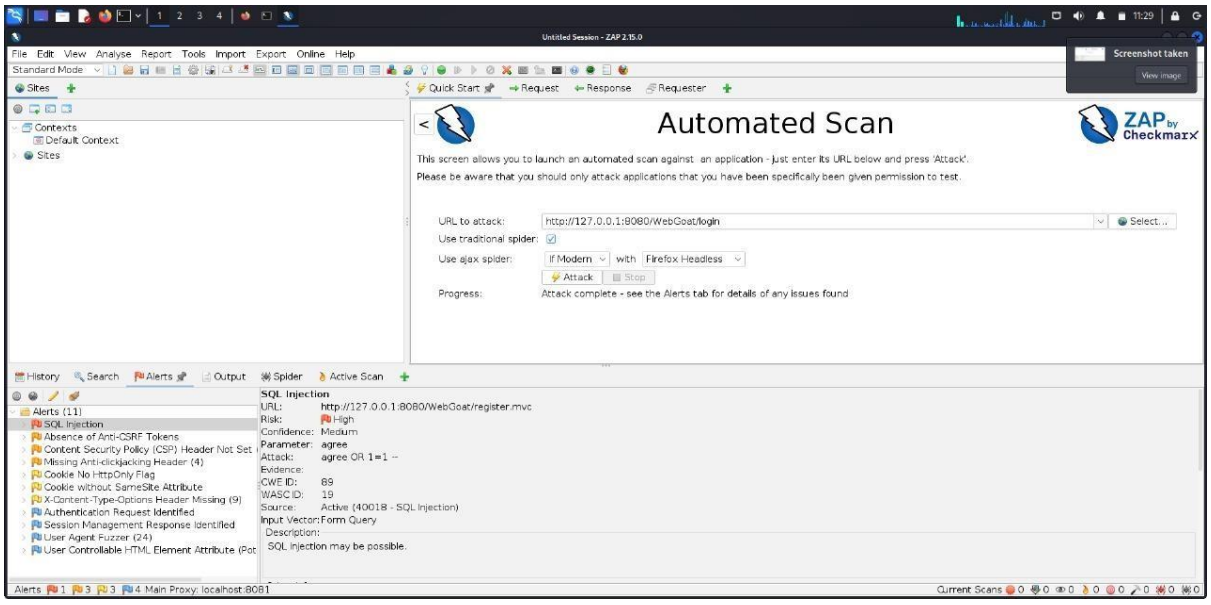
For each vulnerability, ZAP will display:

- **Alert Type**: (e.g., SQL Injection, XSS).
- **Risk Level**: (Low, Medium, High).
- **Description**: Detailed information about the vulnerability.
- **Instances**: Number of occurrences.



successfully conducted a security scan on the WebGoat application using OWASP ZAP. The scan revealed a total of 11 alerts related to various web application vulnerabilities.

Below is a detailed summary of these vulnerabilities, including their descriptions, potential exploit methods, and suggested mitigations.



1. SQL Injection

- **Description**: SQL Injection is a code injection technique that exploits a vulnerability in an application's software by inserting malicious SQL

statements into a query. This can allow attackers to view, manipulate, or delete data stored in the database.

- **Mitigation**: Use parameterized queries or prepared statements to safely handle user inputs. Implement input validation to ensure only expected input is processed.

Exploitation Steps:

1. **Identify Vulnerable Input Fields**:
   Look for input fields that accept user input, such as login forms, search boxes, or URL parameters.
2. **Test for SQL Injection**:
   - Begin by inserting a single quote (') into the input field. This is a common method to test for SQL Injection vulnerabilities.
   - For example, in a login form, you might enter:

admin' OR '1'='1

   - If the application returns an error message indicating a problem with the SQL syntax, it may be vulnerable to SQL Injection.
3. **Extract Data**:
   - If the application is confirmed vulnerable, you can use SQL Injection techniques to extract data. For example, modify the input to:

' UNION SELECT username, password FROM users --

   - This query attempts to combine the results of the original query with a new one that retrieves usernames and passwords from the users table.
4. **Data Manipulation**:
   - You can also manipulate data using SQL Injection.

     For instance:

' DROP TABLE users; --

   - This command would attempt to delete the users table from the database, which could lead to severe consequences.
5. **Use Tools**:

- Consider using automated tools such as **SQLMap,** which can streamline the exploitation process by automating the injection and data extraction methods.

2. Absence of Anti-CSRF Tokens

- **Description**: Cross-Site Request Forgery (CSRF) occurs when an attacker tricks a user into submitting a request that they did not intend to make. The absence of anti-CSRF tokens makes applications vulnerable to these types of attacks.
- **Mitigation**: Implement anti-CSRF tokens in forms and state-changing requests. Validate these tokens on the server-side to ensure that requests are genuine.

3. Content Security Policy (CSP) Not Implemented

- **Description**: A Content Security Policy helps prevent XSS attacks by controlling the sources of content that can be loaded on a web page. Without a CSP, the application is at higher risk of malicious content injection.
- **Mitigation**: Implement a strong Content Security Policy that defines which resources are allowed to load and from where. Regularly review and update the policy based on application needs.

4. Missing Anti-Clickjacking Header

- **Description**: Clickjacking is a malicious technique where a user is tricked into clicking on something different from what the user perceives, potentially leading to unintended actions. This vulnerability can be mitigated by using appropriate headers.
- **Mitigation**: Implement the X-Frame-Options or Content-Security-Policy header to restrict how the application can be embedded in frames.

5. Cookie Without HttpOnly Flag

- **Description**: Cookies without the HttpOnly flag can be accessed via JavaScript, making them vulnerable to theft through XSS attacks. This flag prevents client-side scripts from accessing the cookie data.
- **Mitigation**: Set the HttpOnly flag on cookies to ensure they cannot be accessed by JavaScript, reducing the risk of cookie theft.

6. Cookie Without SameSite Attribute

- **Description**: The SameSite attribute helps prevent CSRF attacks by controlling how cookies are sent with cross-site requests. Cookies without this attribute can be vulnerable to such attacks.
- **Mitigation**: Implement the SameSite attribute on cookies, setting it to Strict or Lax to restrict their usage in cross-origin requests.

7. X-Content-Type-Options Header Missing

- **Description**: The X-Content-Type-Options header prevents browsers from interpreting files as a different MIME type than what is specified by the server. Missing this header increases the risk of content type attacks.
- **Mitigation**: Add the X-Content-Type-Options: nosniff header to the server responses to prevent MIME type confusion.

8. Authentication Request Identified

- **Description**: Identifying authentication requests can reveal potential weaknesses in the login process, making it easier for attackers to exploit vulnerabilities in authentication mechanisms.
- **Mitigation**: Implement rate limiting on authentication endpoints and monitor for unusual login patterns. Consider adding multi-factor authentication to enhance security.

9. User Agent Fuzzer

- **Description**: This vulnerability suggests that the application is allowing unexpected user agent strings, which may be indicative of a security issue. Fuzzing user agents can help identify how the application handles unexpected inputs.
- **Mitigation**: Validate and sanitize user agent inputs to ensure only expected formats are processed. Monitor for unusual patterns that may indicate an attack.

10. User Controllable HTML Element Attribute

- **Description**: When user input is used to set HTML element attributes without proper validation or escaping, it can lead to XSS vulnerabilities. Attackers can manipulate these attributes to inject malicious scripts.

- **Mitigation**: Sanitize and validate all user inputs before using them in HTML attributes. Use libraries that automatically escape HTML to prevent XSS attacks.

Conclusion

The vulnerabilities identified in this assessment highlight critical areas that need addressing to enhance the security posture of the WebGoat application. Implementing the suggested mitigations will significantly reduce the risk of exploitation and improve overall application security.

-THE END-