

## **Task 2: Introduction to Web Application Security**

**Submitted by:-Ahatesham Mopagar**

### **Objective:**

**Learn about common web application vulnerabilities by analyzing a simple web application.**

### **Skills:**

**Basic Web Security,  
Vulnerability Identification**

### **Tools:**

**OWASP ZAP, Webgoat**

### **Web Application Vulnerability:**

**Web application vulnerabilities are weaknesses in software applications that can be exploited by attackers to compromise the integrity, confidentiality, or availability of the application.**

**Understanding these vulnerabilities is crucial for developers and organizations to protect their systems from potential threats.**

## **Common Web Application Vulnerabilities:**

### **1. SQL Injection:**

SQL injection occurs when an attacker inserts malicious SQL commands into a query, allowing them to manipulate database operations.

### **2. Cross-Site Scripting (XSS):**

XSS vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users. This can lead to session hijacking, defacement of websites, or redirecting users to malicious sites.

### **3. Cross-Site Request Forgery (CSRF):**

CSRF tricks a user into executing unwanted actions on a web application in which they are authenticated.

### **4. Security Miss configuration:**

This vulnerability arises from improper configuration of security settings in applications and servers.

### **5. Broken Access Control:**

Broken access control occurs when users can act outside their intended permissions, allowing unauthorized access to sensitive data or functionality.

## **6. Insecure Cartographic Storage**

Sensitive data must be stored securely; however, many applications fail to encrypt sensitive information adequately, making it accessible to attackers if they gain access to the storage.

## **7. Remote Code Execution (RCE):**

RCE vulnerabilities allow attackers to execute arbitrary code on a server by exploiting flaws in the application's code or configuration. This can lead to complete control over the affected server.

## **8. XML External Entities (XXE)**

XXE vulnerabilities occur when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This can lead to data exposure or server-side request forgery.

## **9. Unrestricted File Upload**

When an application allows users to upload files without proper validation, attackers can upload malicious files that may compromise the server or other users.

## **10. Session Management Issues**

Weaknesses in session management can lead to session hijacking or fixation attacks, where an attacker takes over a user's session by exploiting poor session ID management practices.

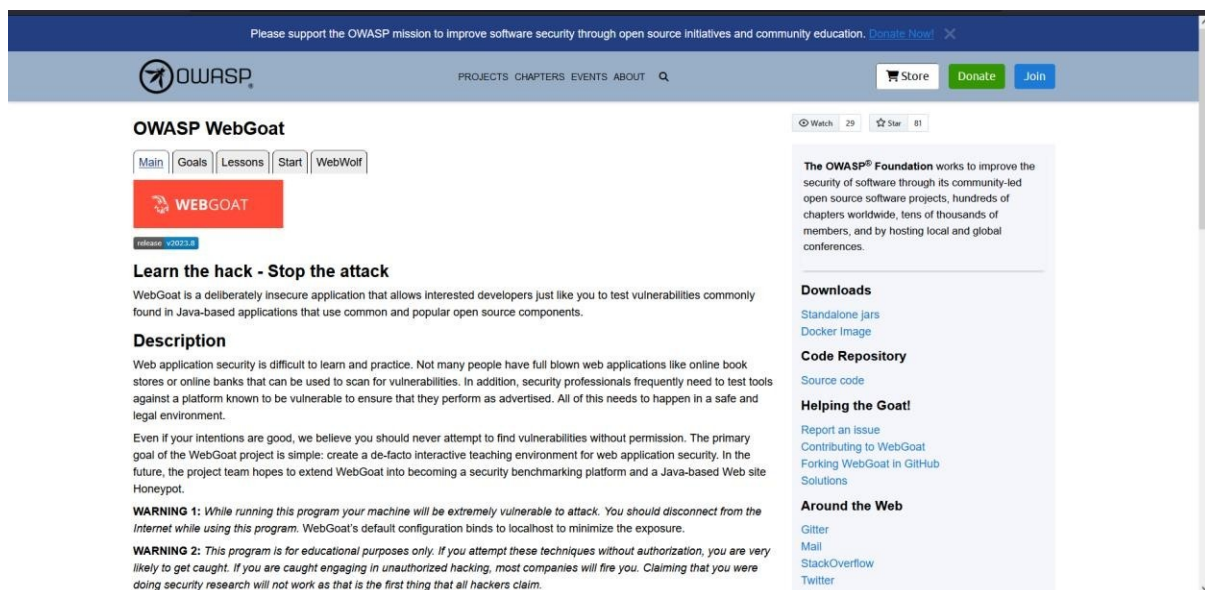
## WEBGOAT:



WebGoat is an intentionally insecure web application developed by the Open Web Application Security Project (OWASP) designed to teach web application security concepts through hands-on experience. It provides a safe environment for users to learn about and exploit common vulnerabilities found in Java-based applications.

## SETTING UP WEBGOAT:

Visit the url(<https://owasp.org/www-project-webgoat/>)



Please support the OWASP mission to improve software security through open source initiatives and community education. [Donate Now!](#)

OWASP WebGoat

[Main](#) [Goals](#) [Lessons](#) [Start](#) [WebWolf](#)

**WEBGOAT**

Release: v2023.8

### Learn the hack - Stop the attack

WebGoat is a deliberately insecure application that allows interested developers just like you to test vulnerabilities commonly found in Java-based applications that use common and popular open source components.

### Description

Web application security is difficult to learn and practice. Not many people have full blown web applications like online book stores or online banks that can be used to scan for vulnerabilities. In addition, security professionals frequently need to test tools against a platform known to be vulnerable to ensure that they perform as advertised. All of this needs to happen in a safe and legal environment.

Even if your intentions are good, we believe you should never attempt to find vulnerabilities without permission. The primary goal of the WebGoat project is simple: create a de-facto interactive teaching environment for web application security. In the future, the project team hopes to extend WebGoat into becoming a security benchmarking platform and a Java-based Web site Honeypot.

**WARNING 1:** While running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program. WebGoat's default configuration binds to localhost to minimize the exposure.

**WARNING 2:** This program is for educational purposes only. If you attempt these techniques without authorization, you are very likely to get caught. If you are caught engaging in unauthorized hacking, most companies will fire you. Claiming that you were doing security research will not work as that is the first thing that all hackers claim.

**Downloads**

- [Standalone jars](#)
- [Docker Image](#)

**Code Repository**

- [Source code](#)

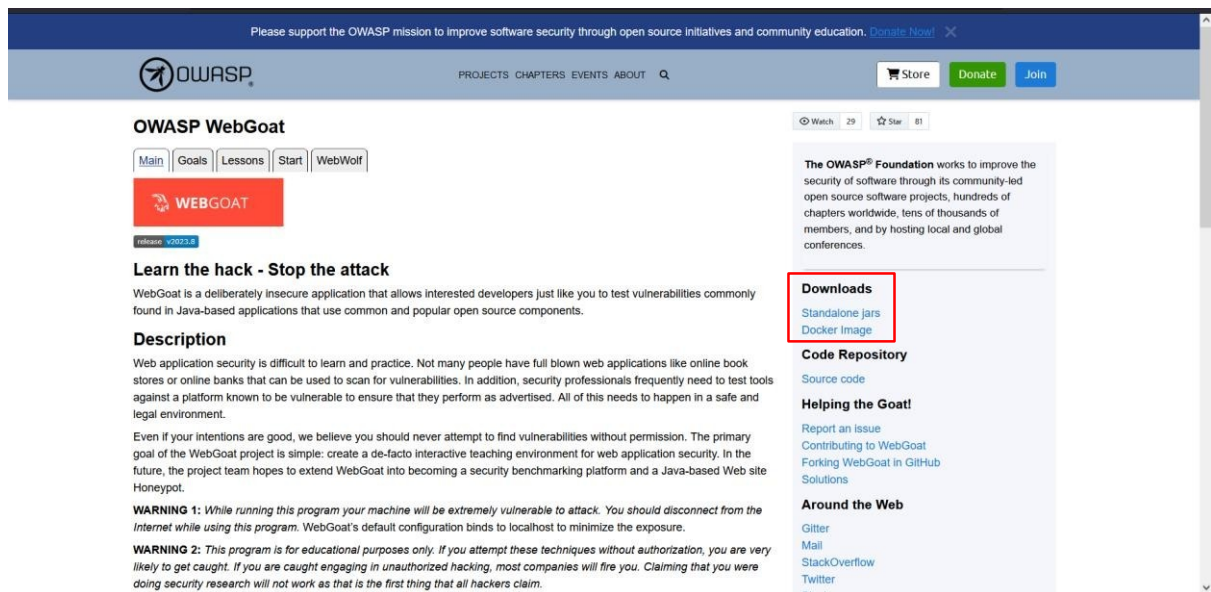
**Helping the Goat!**

- [Report an issue](#)
- [Contributing to WebGoat](#)
- [Forking WebGoat in GitHub](#)
- [Solutions](#)

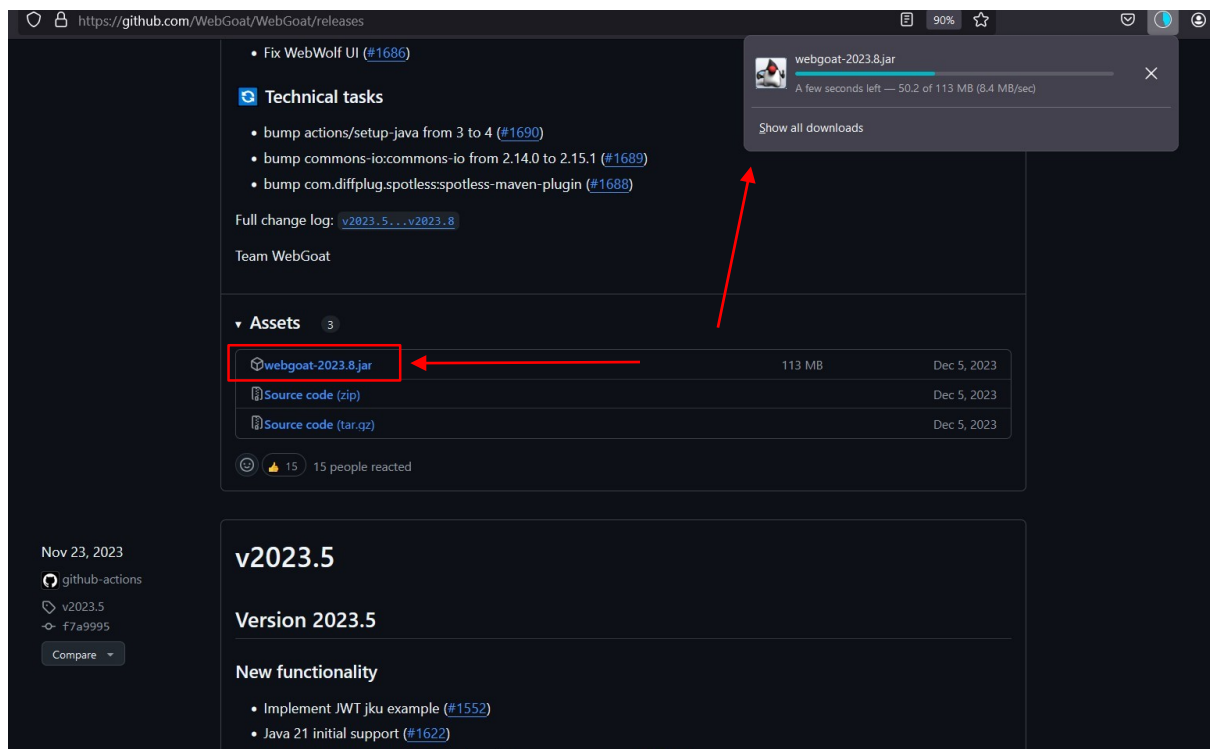
**Around the Web**

- [Gitter](#)
- [Mail](#)
- [StackOverflow](#)
- [Twitter](#)

Under the Downloads => click "Standalone jars".



You will be redirected github website and scroll down for this section.



Click the **webgoat-2023.8.jar** and the download will begin

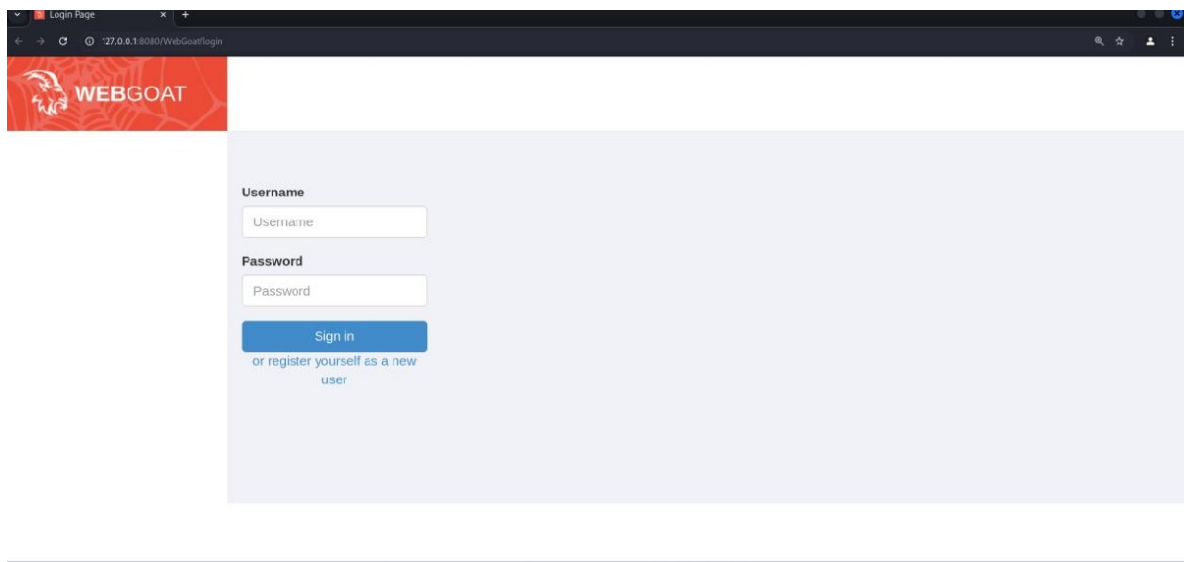
## Run the webgoat using the java -jar webgoat-<version>.jar

```
y.web.access.intercept.AuthorizationFilter@cd26422)
2024-11-01T02:12:42.189-04:00 INFO 13781 [main] io.undertow : Starting server: Undertow - 2.3.18.Final
2024-11-01T02:12:42.227-04:00 INFO 13781 [main] org.xnio : XNIO version 3.8.8.Final
2024-11-01T02:12:42.274-04:00 INFO 13781 [main] org.xnio.nio : XNIO NIO Implementation Version 3.8.8.Final
2024-11-01T02:12:42.413-04:00 INFO 13781 [main] org.jboss.threads : JBoss Threads version 3.5.0.Final
2024-11-01T02:12:42.476-04:00 INFO 13781 [main] o.a.s.h.w.s.undertow.UndertowWebServer : Undertow started on port(s) 8080 (http) with context path '/WebGoat'
2024-11-01T02:12:42.492-04:00 INFO 13781 [main] org.omasp.webgoat.server.StartWebGoat : Started StartWebGoat in 8.584 seconds (process running for 8.468)

WebGoat

2024-11-01T02:12:42.535-04:00 INFO 13781 [main] org.omasp.webgoat.server.StartWebGoat : No active profile set, falling back to 1 default profile: "default"
2024-11-01T02:12:42.591-04:00 INFO 13781 [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-11-01T02:12:43.194-04:00 WARN 13781 [main] io.undertow.websockets.jsr : UT026018: Buffer pool was not set on WebsocketDeploymentInfo, the default pool will be used
2024-11-01T02:12:43.196-04:00 INFO 13781 [main] io.undertow.servlet : Initializing Spring embedded WebApplicationContext
2024-11-01T02:12:43.197-04:00 INFO 13781 [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 628 ms
2024-11-01T02:12:43.297-04:00 INFO 13781 [main] o.f.c.internal.license.VersionPrinter : Flyway Community Edition 9.16.3 by Redgate
2024-11-01T02:12:43.398-04:00 INFO 13781 [main] o.f.c.internal.license.VersionPrinter : See release notes here: https://red-gt.com/
2024-11-01T02:12:43.413-04:00 INFO 13781 [main] o.f.c.i.database.base.BaseDatabaseType : Database: jdbc:hqldb:file:/home/kali/.webgoat-2022.8/webgoat (HSQL Database Engine 2.7)
2024-11-01T02:12:43.423-04:00 WARN 13781 [main] o.f.c.i.internal.database.base.Database : Flyway upgrade recommended: HSQLDB 2.7 is newer than this version of Flyway and support has not been tested. The latest supported version of HSQLDB is 2.6.
2024-11-01T02:12:43.449-04:00 INFO 13781 [main] o.f.c.core.internal.command.DBMigrate : Successfully validated 4 migrations (execution time 00:00.832s)
2024-11-01T02:12:43.491-04:00 INFO 13781 [main] o.f.c.core.internal.command.DBMigrate : Current version of schema "container": 3
2024-11-01T02:12:43.492-04:00 INFO 13781 [main] o.f.c.core.internal.command.DBMigrate : Schema "container" is up to date. No migration necessary.
2024-11-01T02:12:43.539-04:00 INFO 13781 [main] o.hibernate.jpa.internal.util.LogHelper : HHN000048: processing PersistenceUnitInfo [name: default]
2024-11-01T02:12:43.548-04:00 INFO 13781 [main] o.s.a.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup; ignoring JPA class transformer
2024-11-01T02:12:43.551-04:00 WARN 13781 [main] org.hibernate.orm.deprecation : HHN000002: HSQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it w
11 be selected by default)
2024-11-01T02:12:43.784-04:00 INFO 13781 [main] o.h.w.t.j.p.i.JtaPlatformInitiator : HHH000049: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2024-11-01T02:12:43.795-04:00 INFO 13781 [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-11-01T02:12:43.925-04:00 INFO 13781 [main] o.o.w.lessons.logging.LogEventListener : Password for admin: ZW64Z2WwYmYtZTQwYmQzUjU1LnJ2dktW6400VZNDc2MkZz
2024-11-01T02:12:44.043-04:00 WARN 13781 [main] o.o.w.c.lessons.CourseConfiguration : Lesson: webgoat.title has no endpoints, is this intentional?
2024-11-01T02:12:44.079-04:00 INFO 13781 [main] o.s.s.s.s.web.EndpointLinksResolver : Exposing 3 endpoint(s) beneath base path '/actuator'
2024-11-01T02:12:44.306-04:00 INFO 13781 [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.session.DisableEncodeUrlFilter@9d5b399, org.springframework.securi
ty.web.context.request.async.WebAsyncManagerIntegrationFilter@7aaf8d6, org.springframework.security.web.context.SecurityContextHolderFilter@32ab6d4, org.springframework.security.web.authentication.logout.LogoutFilter@7c692ba, org.sp
ringframework.security.oauth2.client.web.OAuth2AuthorizationRequestRedirectFilter@bbcb21b6, org.springframework.security.oauth2.client.web.OAuth2LoginAuthenticationFilter@14e3d439, org.springframework.security.web.authentication.Username
PasswordAuthenticationFilter@8ac51de, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@733ab495, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@4ebfb45, org.springframework.secu
rity.web.authentication.AnonymousAuthenticationFilter@4d6df2b6, org.springframework.security.web.access.ExceptionTranslationFilter@2d6b35fa, org.springframework.security.web.access.intercept.AuthorizationFilter@6caeb36]
2024-11-01T02:12:45.934-04:00 [main] WARN FileUtil : Native subprocess control requires open access to the /dev/pts subsystem
Pass - and opens java.base/sun.nio.ch-ALL-UNNAMED - and opens java.base/java.io-ALL-UNNAMED to enable.
2024-11-01T02:12:47.566-04:00 WARN 13781 [main] io.thymeleaf.check-template-location : Cannot find template location: classpath:/templates/ (please add some templates, check your Thymeleaf configuration, or set sprin
g.thymeleaf.check-template-location=false)
2024-11-01T02:12:47.599-04:00 INFO 13781 [main] io.undertow : Starting server: Undertow - 2.3.18.Final
2024-11-01T02:12:48.004-04:00 INFO 13781 [main] o.s.s.w.s.undertow.UndertowWebServer : Undertow started on port(s) 8080 (http) with context path '/WebGoat'
2024-11-01T02:12:48.080-04:00 INFO 13781 [main] org.omasp.webgoat.server.StartWebGoat : Started StartWebGoat in 5.584 seconds (process running for 13.983)
2024-11-01T02:12:48.010-04:00 WARN 13781 [main] org.omasp.webgoat.server.StartWebGoat : Please browse to http://127.0.0.1:8080/WebGoat to start using WebGoat ...
```

Now open the browser and paste the link give in the screenshot (http://localhost:8080/WebGoat)



## SETTING UP OWASP ZAP:



---

OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner developed by the Open Web Application Security Project (OWASP).

It is designed to help security professionals and developers identify vulnerabilities in web applications effectively.

### FEATURES:

OWASP ZAP functions as a Dynamic Application Security Testing (DAST) tool, meaning it analyzes running applications without needing access to the source code. It can detect a wide range of vulnerabilities, including:

1. SQL Injection
2. Cross-Site Scripting (XSS)
3. Compromised Authentication
4. Exposure of Sensitive Data
5. Security Misconfigurations

## SETTING UP ZAPROXY / OWASP ZAP:

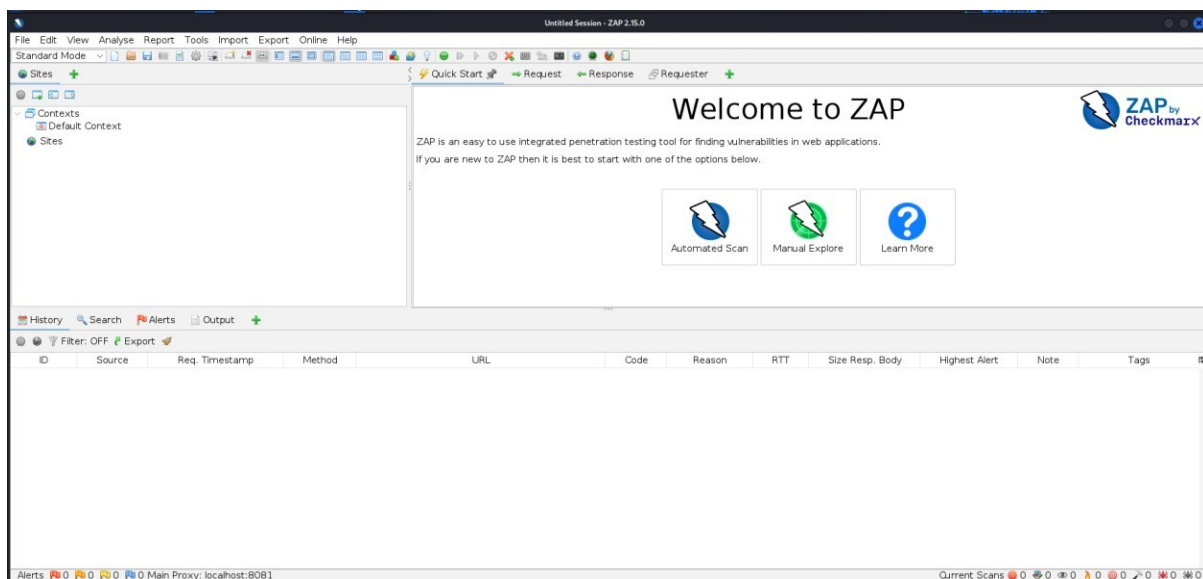
### STEP 1: Run “sudo apt install zaproxy”

```
(kali㉿kali)-[~]
└─$ sudo apt install zaproxy
Installing:
  zaproxy

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 307
  Download size: 213 MB
  Space needed: 266 MB / 54.7 GB available

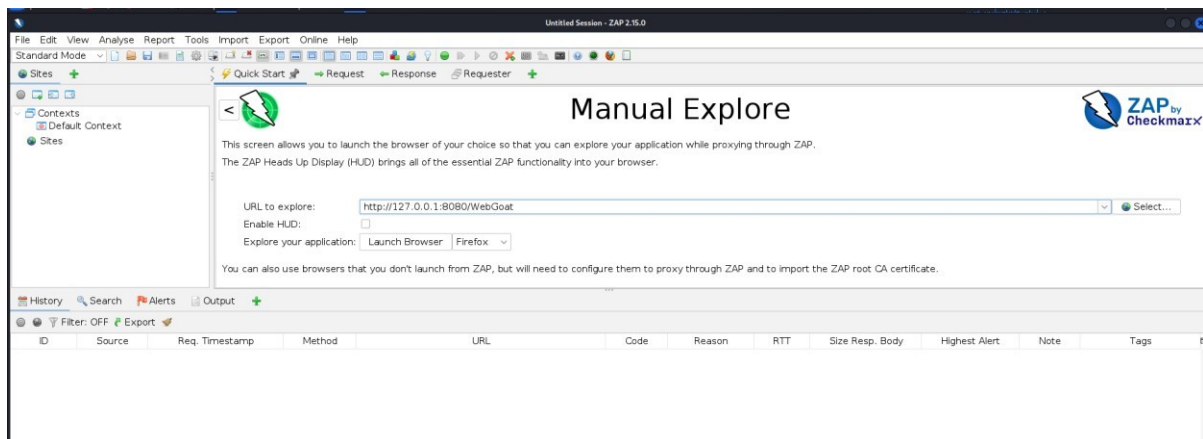
Get:1 http://kali.download/kali kali-rolling/main amd64 zaproxy all 2.15.0-0kali1 [213 MB]
Ign:1 http://kali.download/kali kali-rolling/main amd64 zaproxy all 2.15.0-0kali1
Get:1 http://kali.download/kali kali-rolling/main amd64 zaproxy all 2.15.0-0kali1 [213 MB]
5% [1 zaproxy 14.0 MB/213 MB 7%]
```

### STEP 2: Open the zaproxy and displays like it

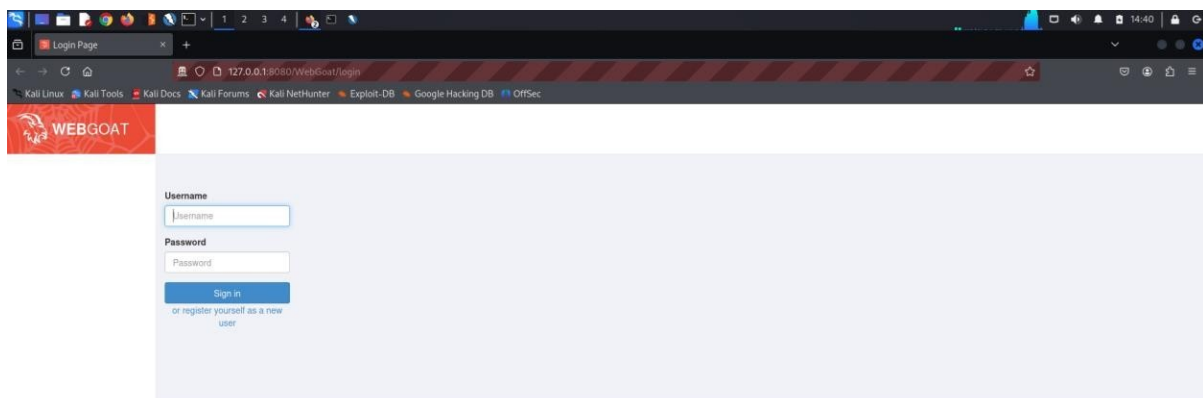




## Scanning the localhost website using the Vulnerability scanner (OWASP ZAP):

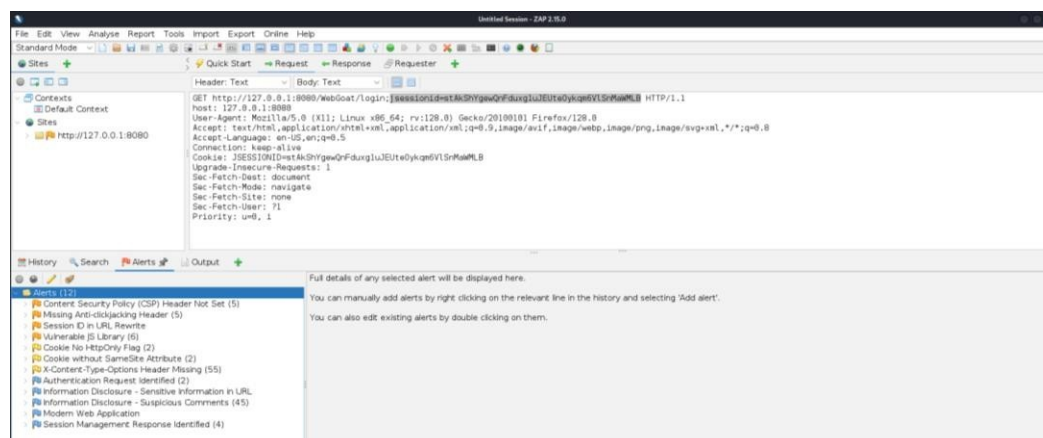


In Manual Explore launch the browser in Firefox / chrome:



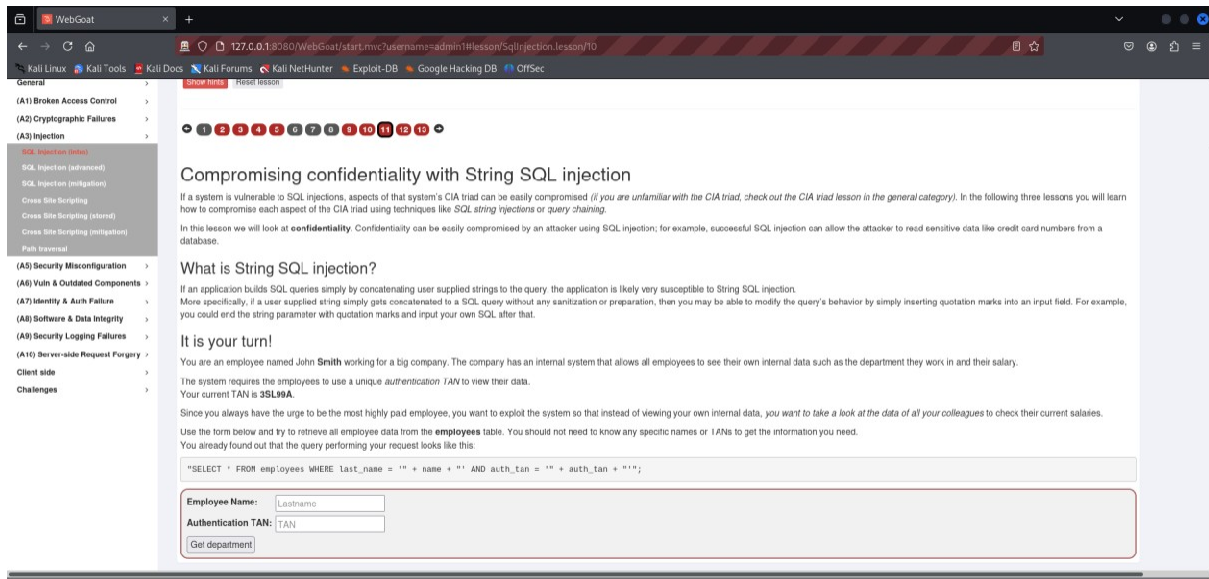
The browser is totally automated by Vulnerability scanner (dark red lines) for capturing the request and response

In “alerts” section we found the vulnerabilities present in the login page.(Demo purpose)

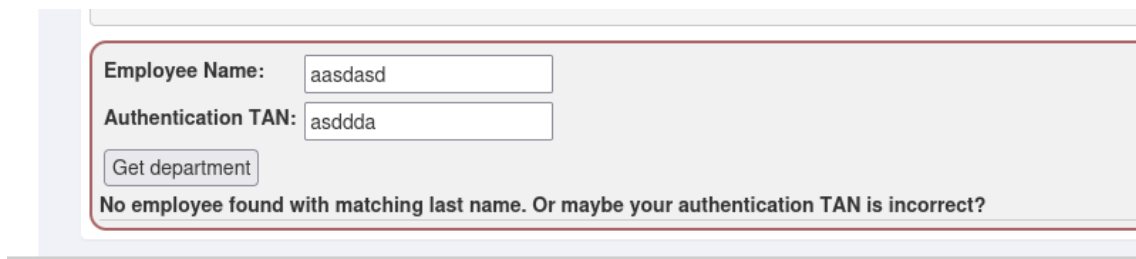


# Finding the SQLi vulnerability in WebGoat:

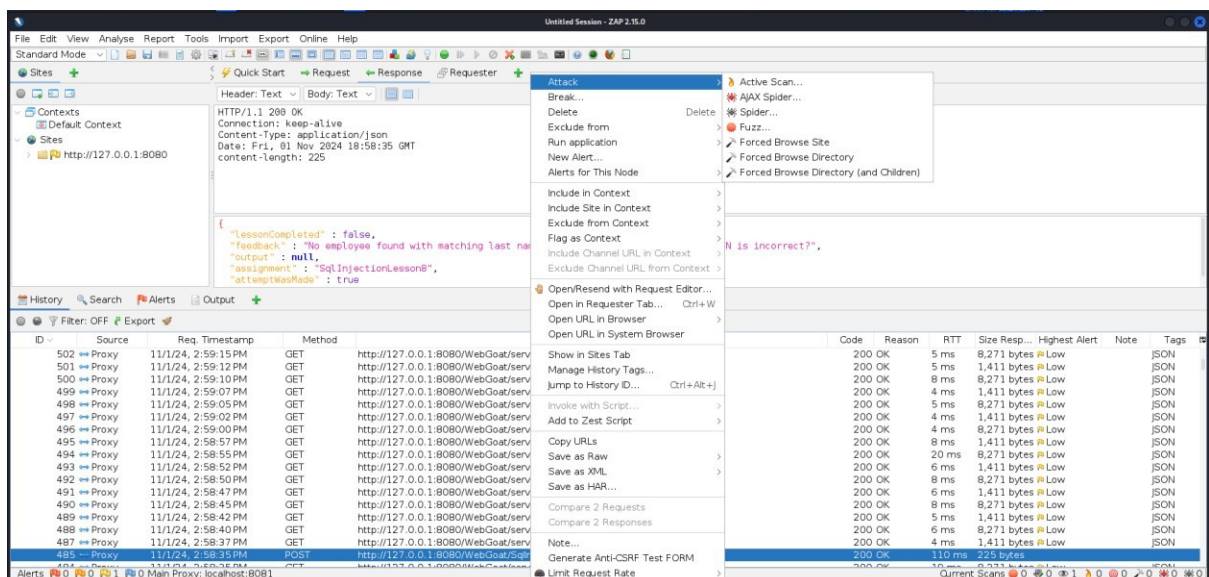
## Step 1: Copy the URL and paste the URL in browser.



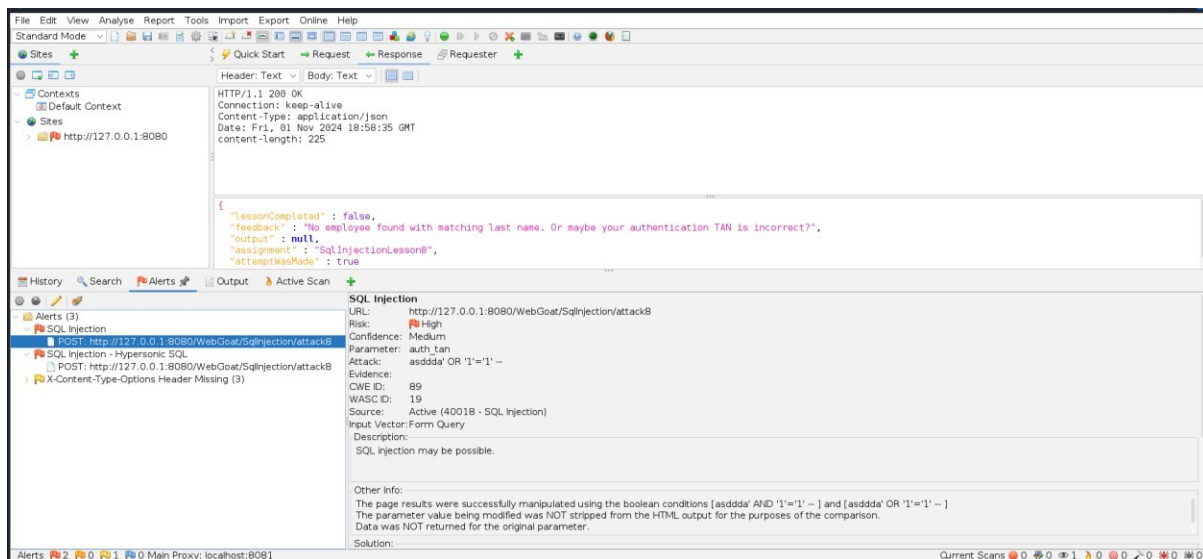
## Step 2: Enter the random words to capture the request and response.



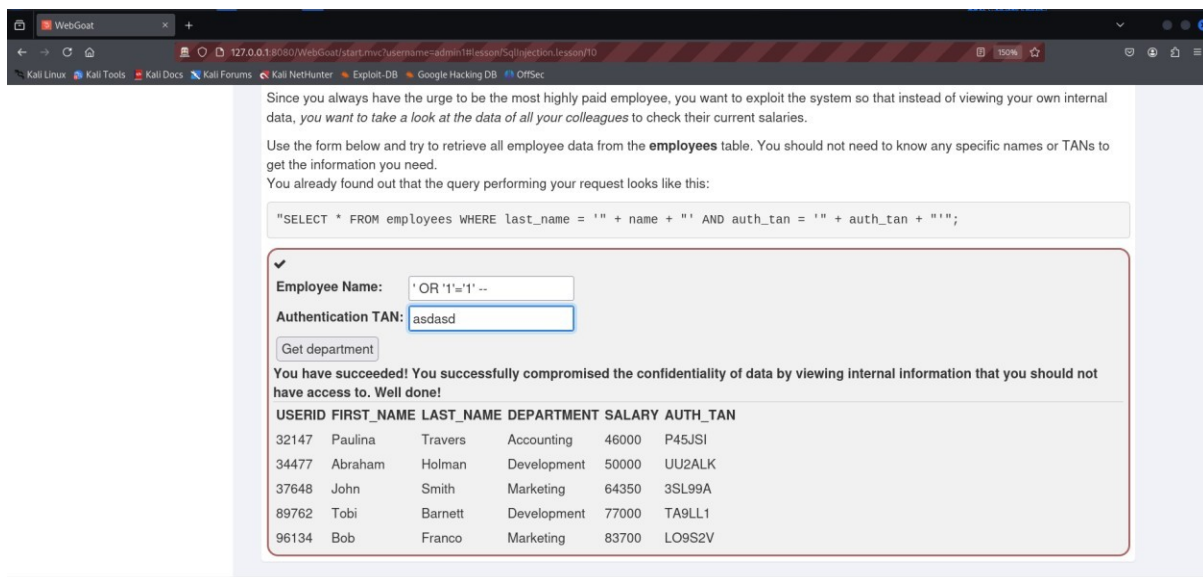
## Step 3: Clear the "Alert" section. On the Request of URL Right click => Attack => Active Scan => Start Scan.



**Step 4: After the 100% scan navigate to the “Alert” section for result of alerts. We found the “SQLi” vulnerabilities.**



**Step 5: Implement in the WebGoat.**



## **Consequences of SQL Injection:**

### **Unauthorized Data Access:**

Attackers can bypass authentication mechanisms, gaining unauthorized access to sensitive data, including personal information, financial records, and trade secrets. This can lead to significant data breaches and identity theft.

### **Data Manipulation:**

SQLi allows attackers to modify or delete records in a database. For instance, they can alter account balances in financial applications or delete critical data, compromising data integrity and operational functionality.

### **Complete Database Control:**

A successful SQL injection can grant attackers full control over the database server. This includes executing administrative operations such as creating new user accounts with elevated privileges or even dropping entire tables<sup>2</sup>

### **Escalation of Attacks:**

If the database server is compromised, attackers may leverage this access to infiltrate other systems behind firewalls.

### **Reputation Damage:**

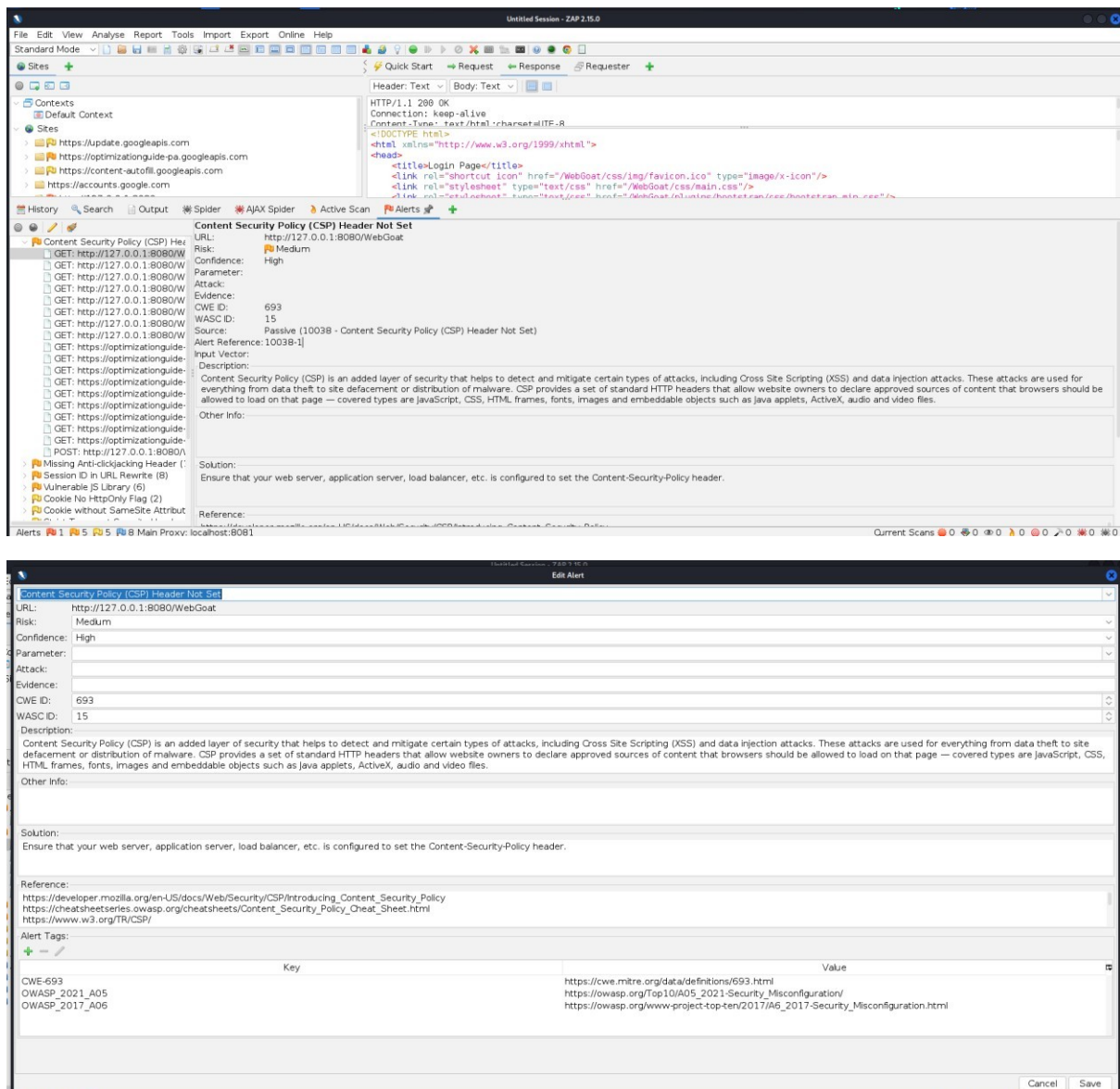
The fallout from an SQL injection attack can severely damage a business's reputation. Customers may lose trust in the organization, leading to loss of business and long-term financial repercussions.

### **Prevention from the SQLi:**

1. Implement Input Validation and Sanitization
2. Use Prepared Statements (Parameterized Queries)
3. Employ Stored Procedures
4. Restrict Database User Privileges
5. Use Web Application Firewalls (WAF)
6. Conduct Regular Security Audits
7. Keep Software Updated

## XSS: (Cross Site Scripting)

Open the WebGoat and copy the URL of the WebGoat login page and paste in the Automated Scan URL header option and start the scan.



## Content Security Policy (CSP) Header Not Set:

Content Security Policy (CSP) is missing, which limits sources from which content (scripts, images, etc.) can be loaded. Without CSP, the application is more vulnerable to attacks like Cross-Site Scripting (XSS), where attackers can inject malicious scripts that get executed in the user's browser.

## **Consequence of XSS:**

### **Session Hijacking**

Attackers steal session cookies to impersonate users, gaining unauthorized access to accounts.

### **Credential Theft**

XSS can be used to trick users into entering login details on fake pages.

### **Data Theft**

Sensitive information on the page can be accessed and stolen by attackers.

### **Malware Distribution**

Malicious code injected through XSS can redirect users to phishing sites or download malware.

### **Phishing and Social Engineering**

XSS can display fake interfaces, tricking users into revealing personal information.

### **Defacement of Websites**

Attackers can alter the appearance of web pages, damaging credibility and reputation.

### **Denial of Service**

XSS may overload the server, making the site inaccessible to legitimate users.

### **Impacts on SEO and Reputation**

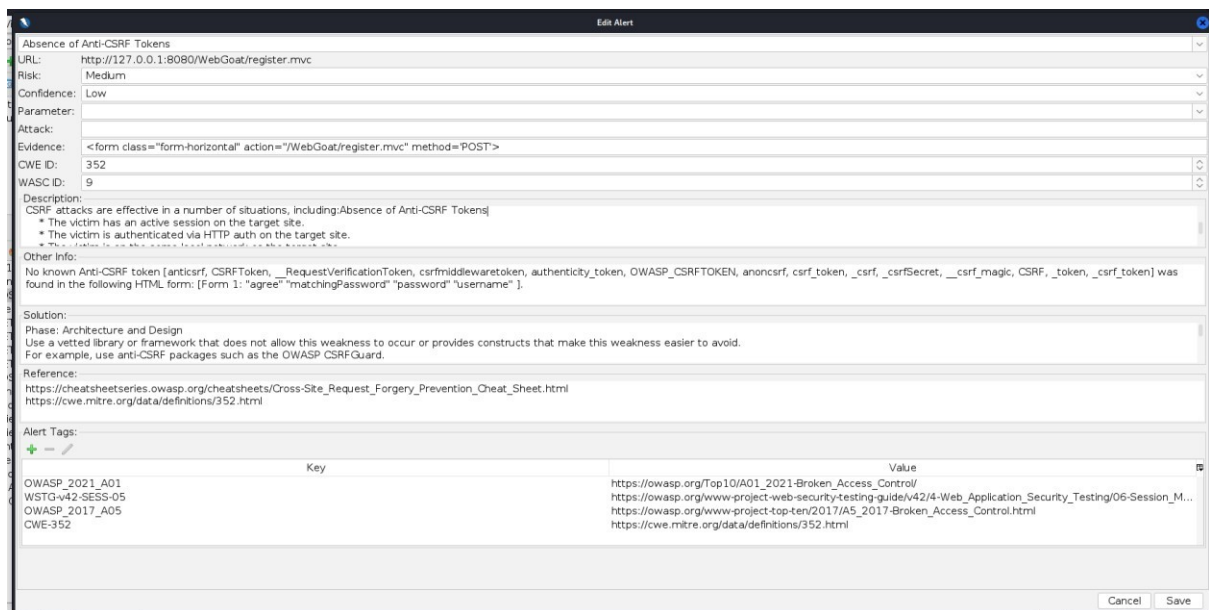
Hidden spam links or malware can harm a site's SEO and get it blacklisted.

## **Prevention:**

1. Sanitize and validate user input on both the client and server sides.
2. Use Content Security Policy (CSP) to limit where scripts can be loaded from.
3. Encode output (especially when rendering user-generated content in HTML).
4. Use secure headers such as X-Content-Type-Options, X-Frame-Options, and X-XSS-Protection.



**Open the WebGoat and copy the URL of the WebGoat login page and paste in the Automated Scan URL header option and start the scan**



The application does not use Anti-CSRF tokens, which protect against Cross-Site Request Forgery (CSRF) attacks. A hacker could trick an authenticated user into performing unintended actions on the application (like transferring funds or changing settings) by exploiting the user's authenticated session.

## **Consequence of CSRF:**

### **Unauthorized Actions on Behalf of Users:**

Attackers can modify settings, change passwords, or initiate transactions without user consent.

### **Privilege Escalation and Account Takeover:**

Attackers may gain unauthorized access or escalate privileges to control accounts.

### **Data Leakage or Manipulation:**

Sensitive data can be exposed or modified, leading to data integrity issues.

### **Financial Loss:**

Unauthorized transactions can cause direct financial harm to victims.

### **Reputational Damage:**

Breaches due to CSRF attacks can damage the organization's reputation.

### **Loss of User Trust:**

Users may lose confidence in the platform after unauthorized actions occur.

### **Legal and Compliance Issues:**

CSRF vulnerabilities can result in regulatory violations and fines.



## **Prevention:**

### **Anti-CSRF Tokens:**

Generate unique tokens for each session and validate them on the server to confirm request authenticity.

### **SameSite Cookies:**

Set cookies with the SameSite attribute to restrict cross-site requests.

### **User Authentication Validation:**

Ensure that critical actions require reauthentication or confirmation from the user.

### **Implement Strong CORS Policies:**

Restrict cross-origin requests to trusted sources.

### **Use Secure Headers:**

Enforce X-Frame-Options and Content-Security-Policy headers to limit content injection and clickjacking.

### **Educate Users:**

Inform users to avoid suspicious links and to log out when not actively using the application.