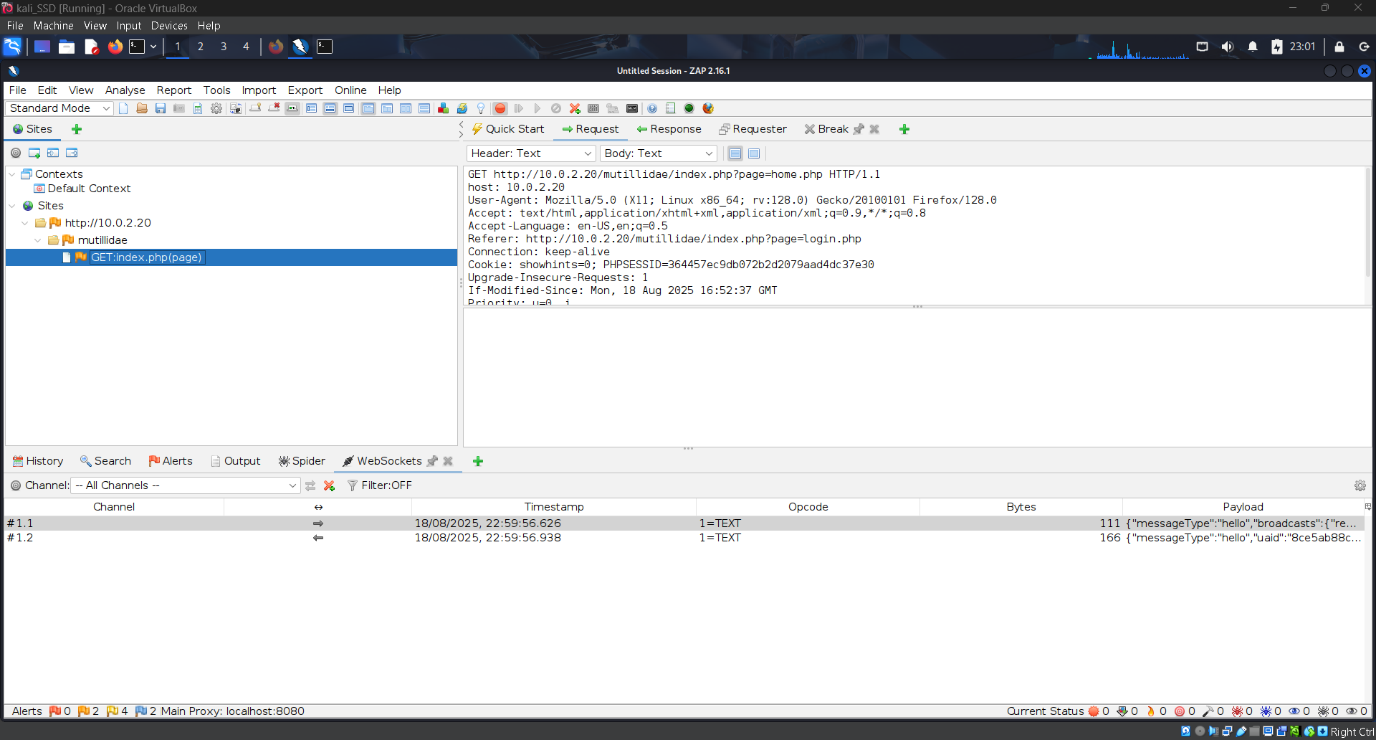
Penetration Testing report for Assessment

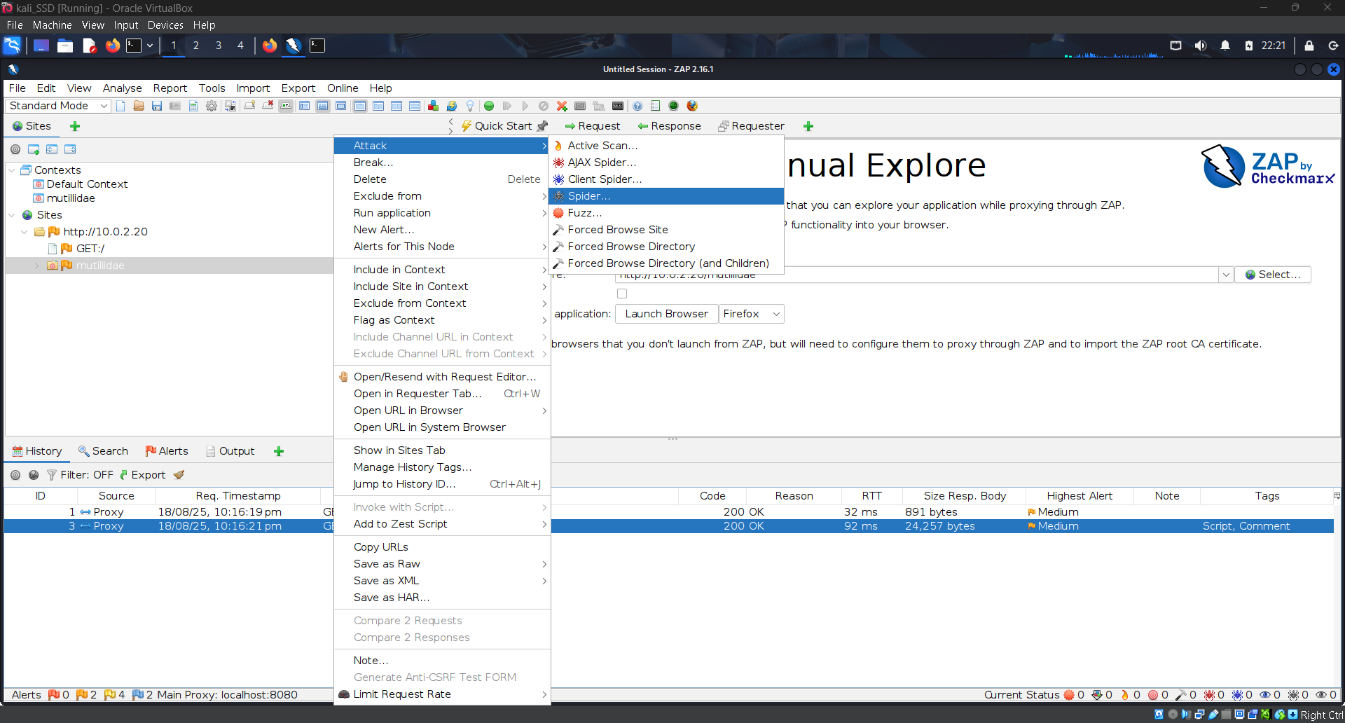
# Answer 1 - OWASP ZAP Uncovering vulnerabilities

The following steps were taken before modifying http request

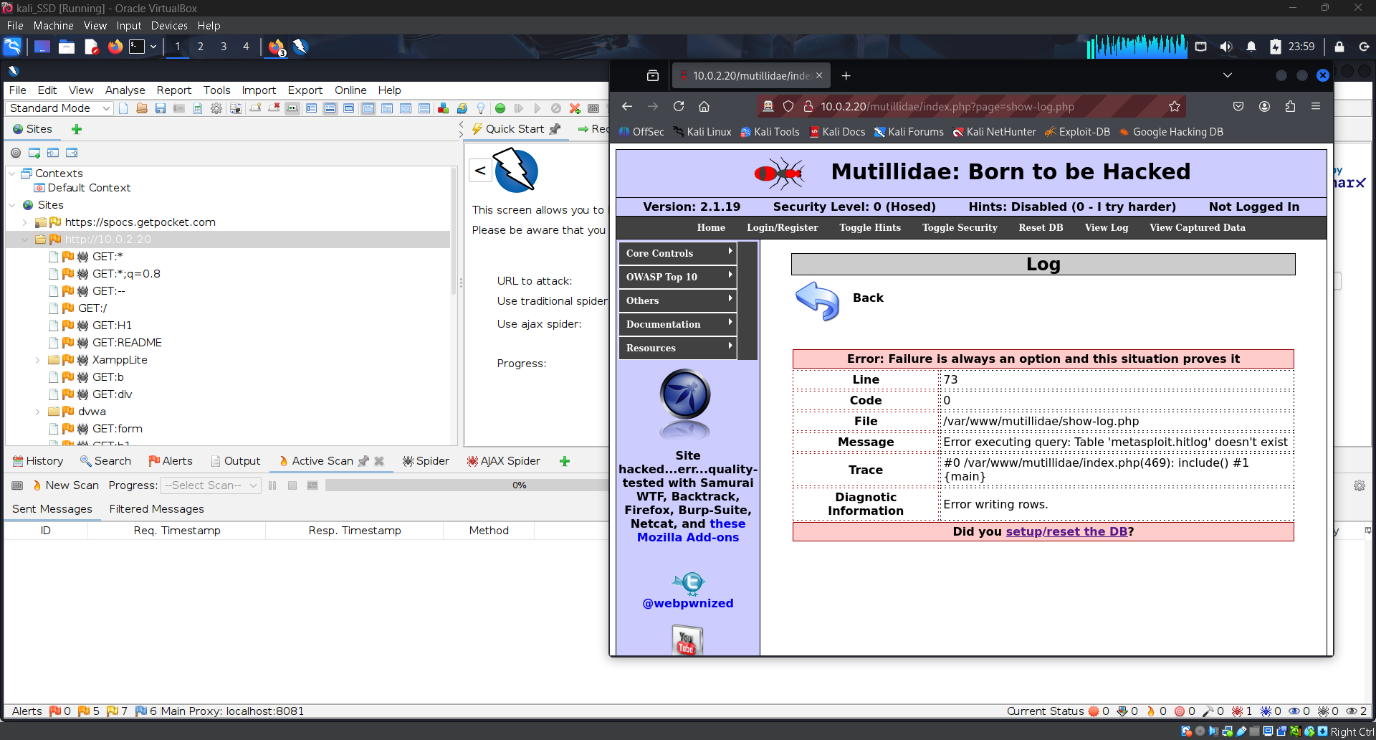
* Configure proxy of zap in Firefox using foxyproxy
* Open the Mutillidae page from the metasploitable 2
* I was able to route the traffic though the zap.
* We can use requester tab for manual request editing of the requests.



Captured the packets using the breaking request



Used spider to map the site



Used the active scan to find out the vulenrabilities of the system

Observation on zap

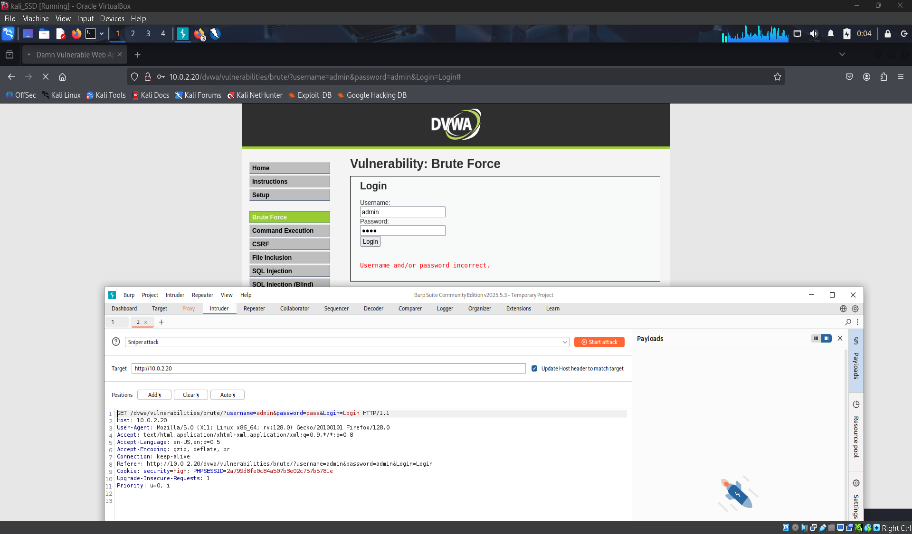
* You can view the full http request and response from History vies
* Breakpoint is used for five interception and editing of parameters

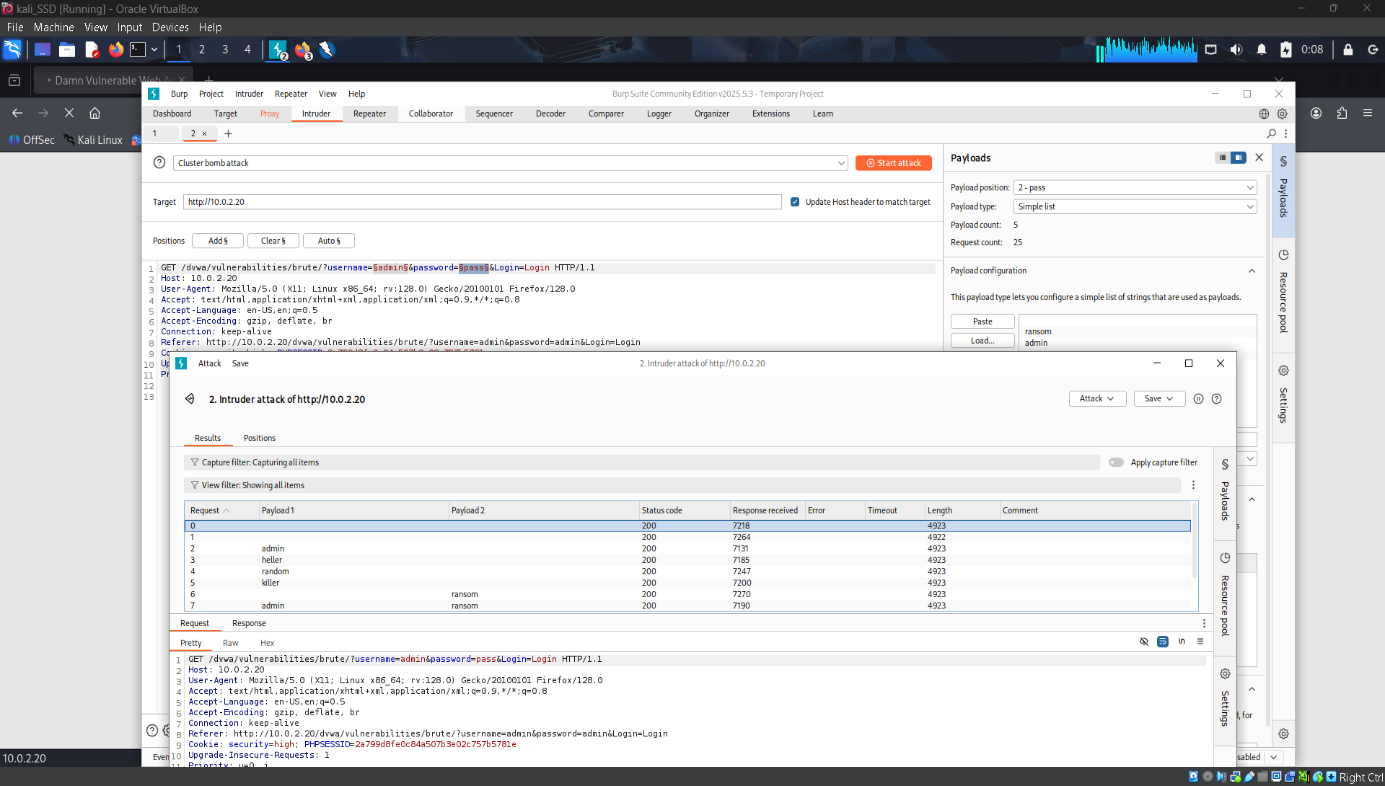
# Answer 2 - Burp Suite

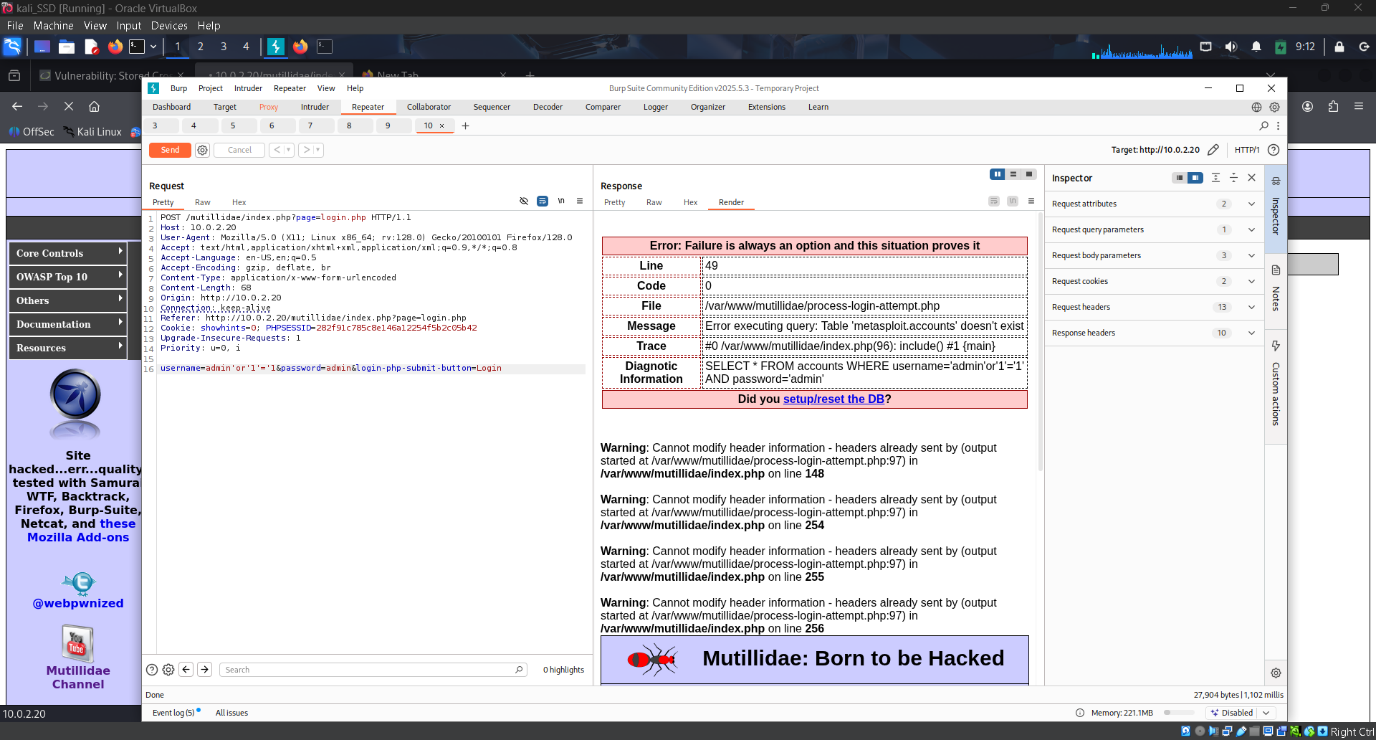
Features

* Proxy – to intercept requests
* Intruder – to automate brute force attacks and for payload injection
* Repeater – to manually configure and edit requests

Here is image of using intruder tab for brute forcing a password



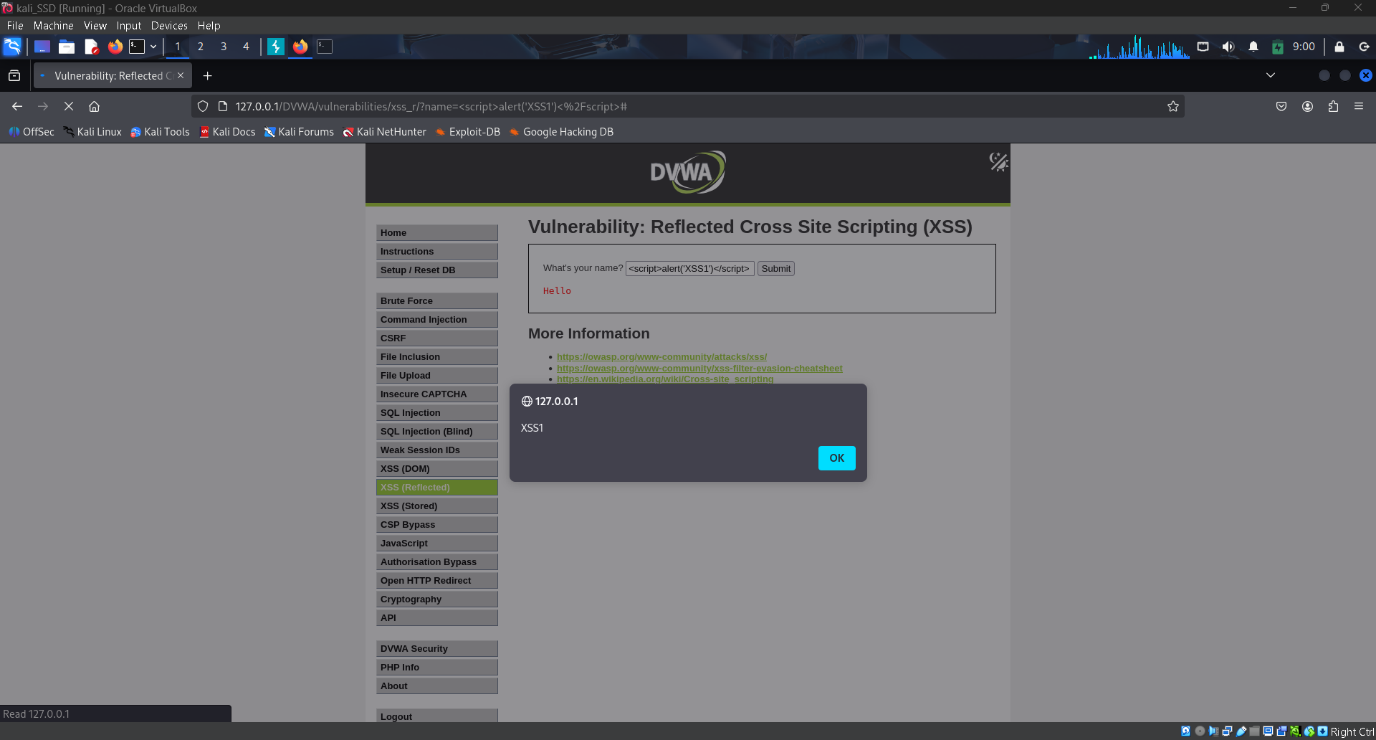


I used the burp repeater to break the mutillidae and the result are show in the above image simple ’ was used to break the site using sqli vulnerability

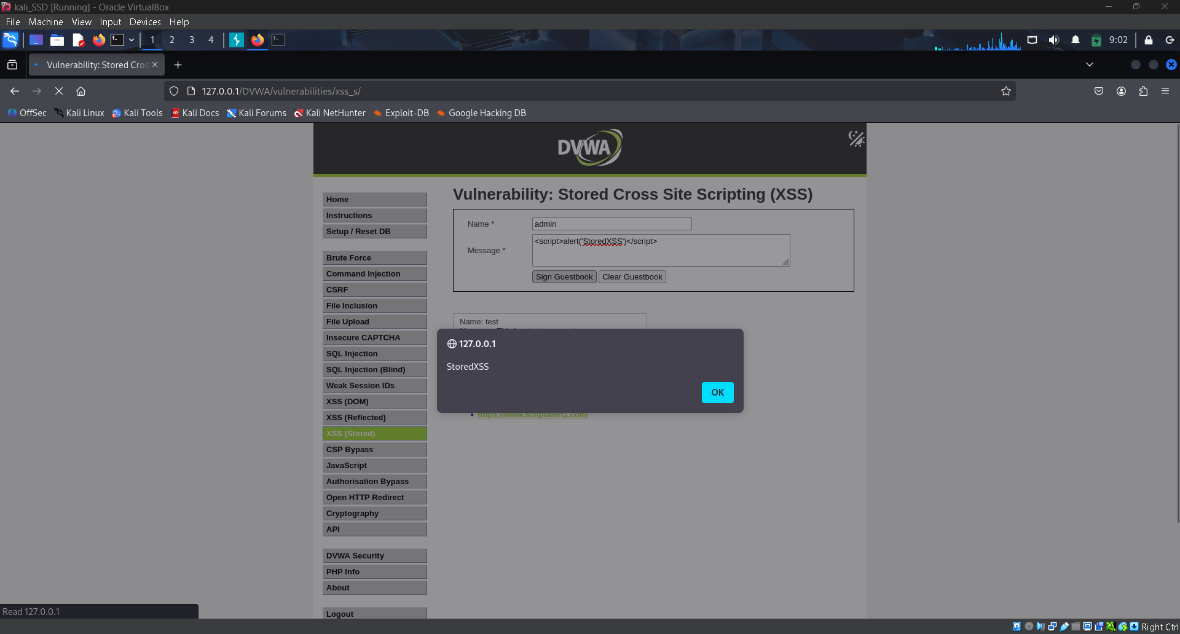
# Answer 3 - Cross-Site Scripting (XSS)

Techniques to detect XSS   
1. Manual input testing

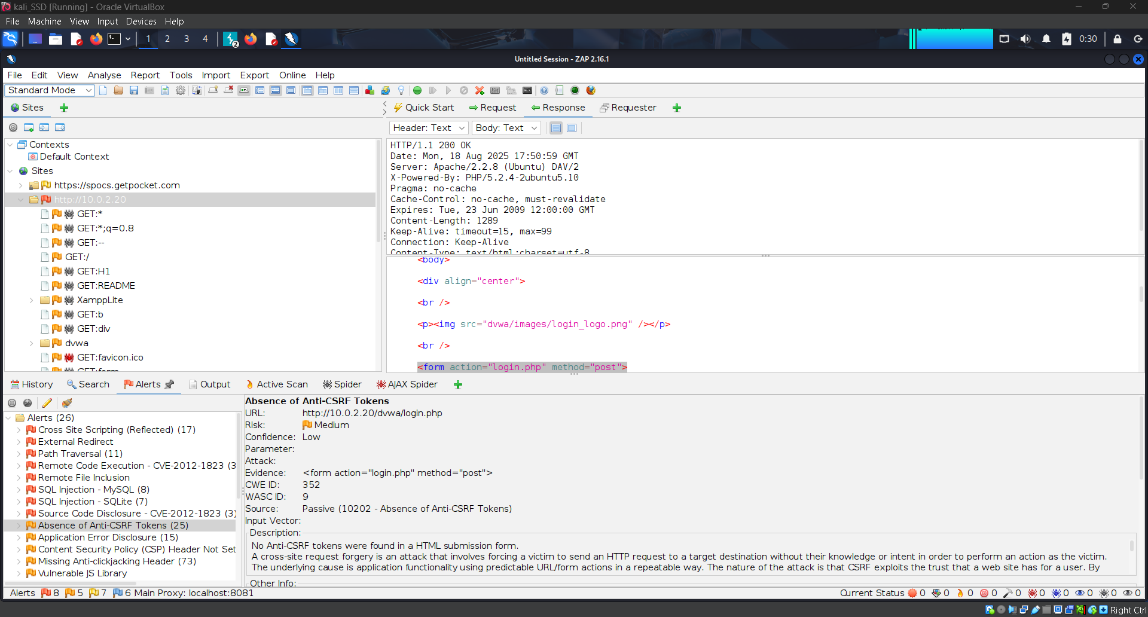
Used this <script>alert("XSS")</script> and alert suggest a xxs vulnerability



2. Stored xxs using the comment section of the site. this vulnerability allow the attacker to run malicious scripts on the other users.



3. Automated scanning using owasp zap

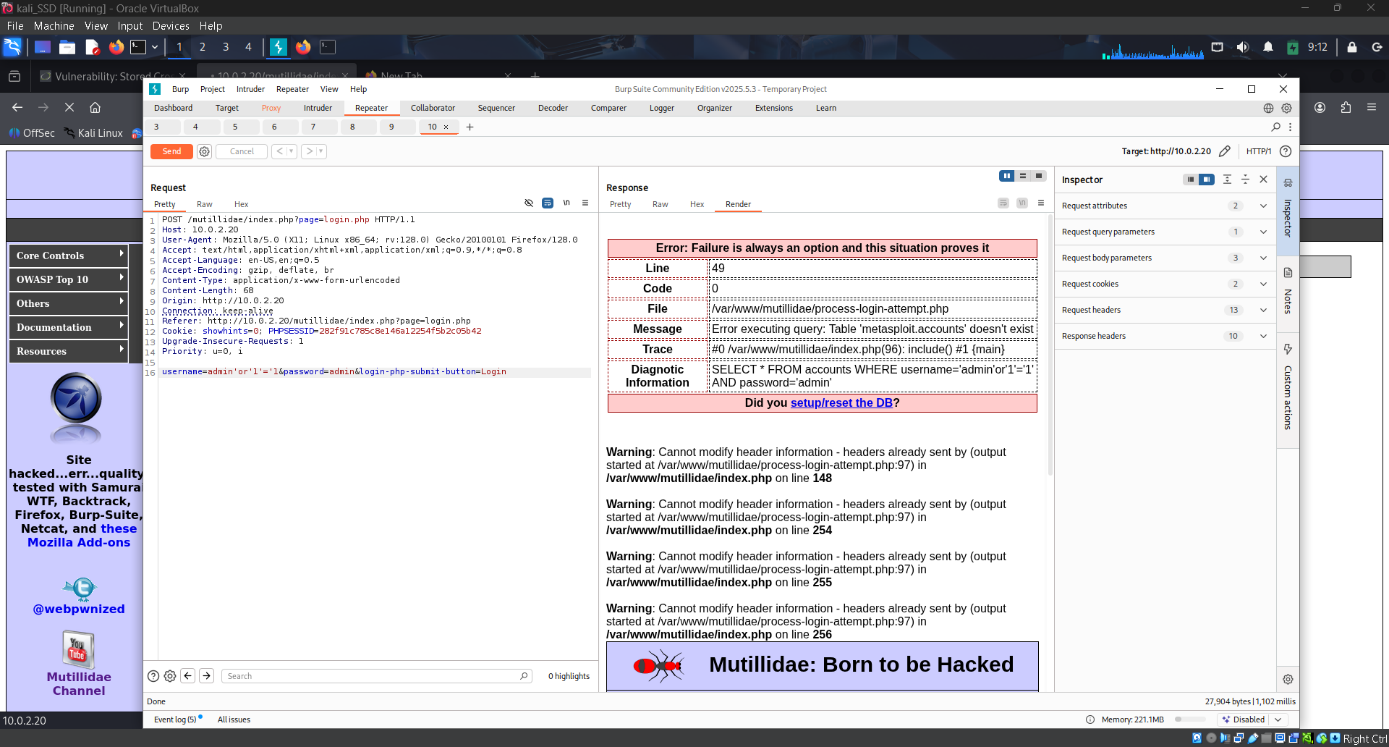


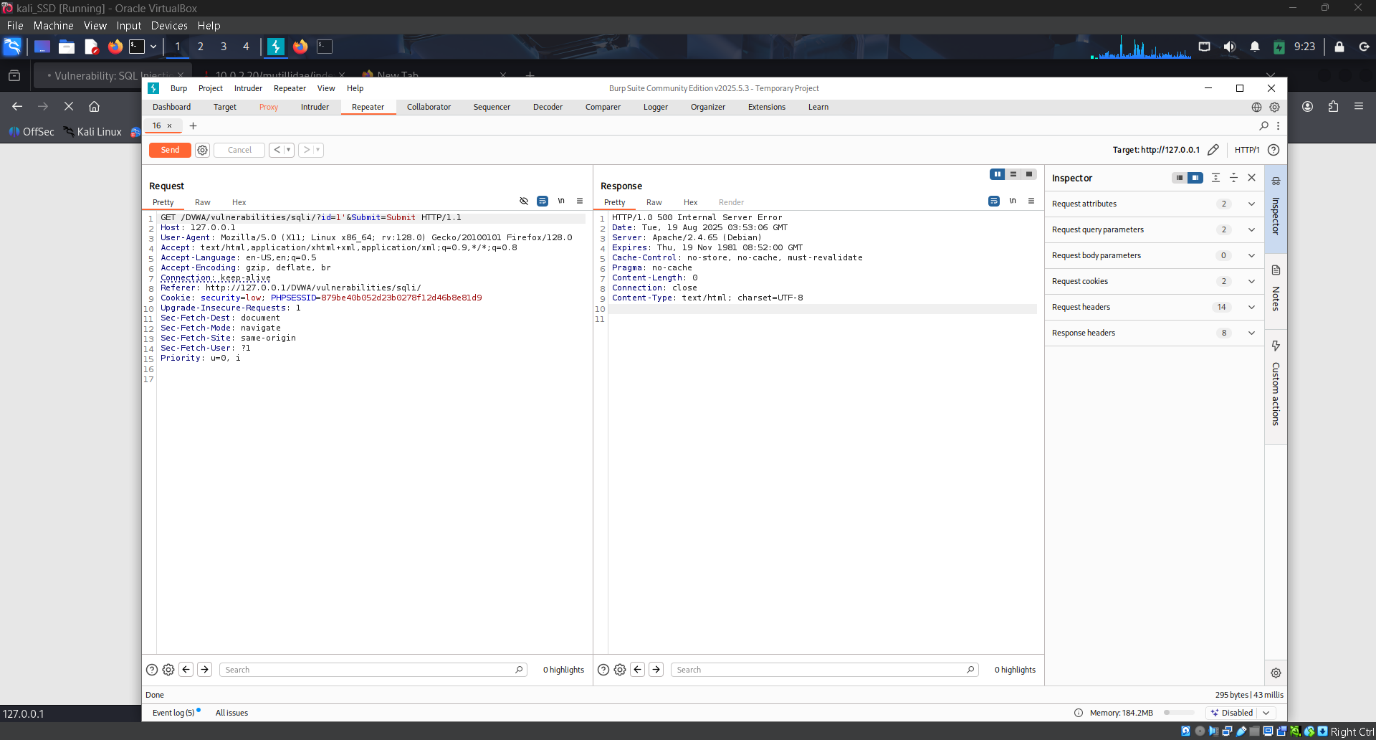
Used the active scanning of the zap for it

# Answer 4 - Error-based SQL injection

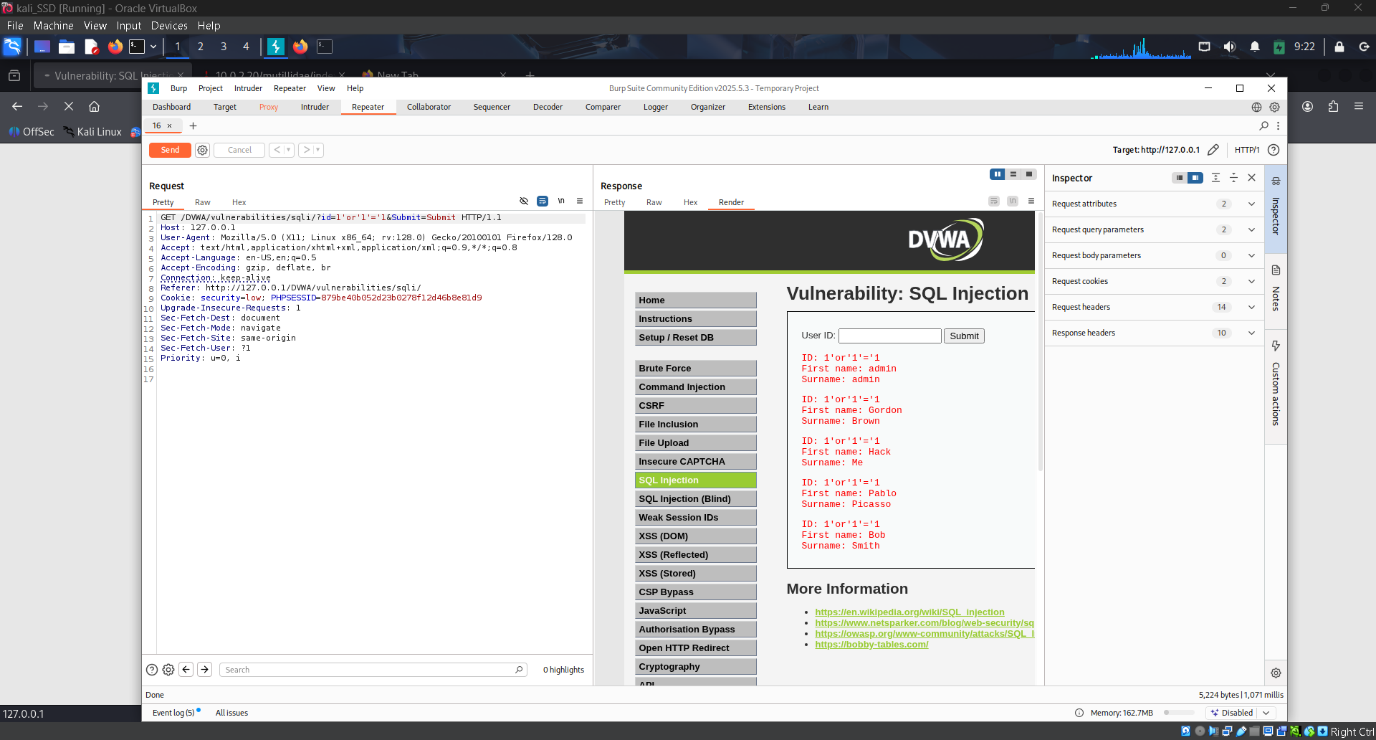
Error-based happen if an attacker injects a malicious sql into the input field which result in an error message being displayed.

I inserted a single quote in the input field and it broke the site and an error was returned as shown in the image below





Captured the packets for the sql injection.



Used the (‘or ‘1’=1’) sql query and the site returned the full table.

## Mitigation

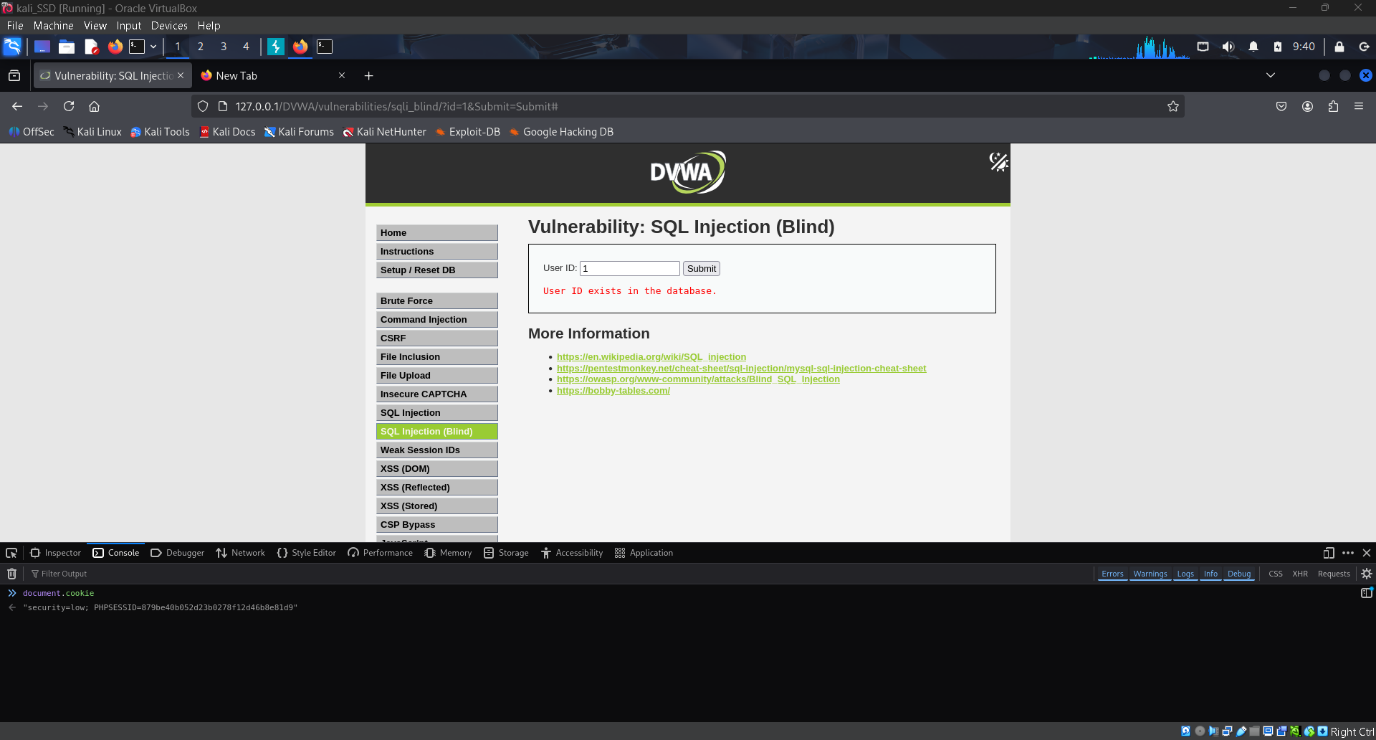
* Prepared Statements - Use parameterized queries instead of string concatenation.
* Input Validation & Sanitization - Reject dangerous characters (', ", ;, --)
* Error Handling - Disable detailed SQL error messages in production.
* Least Privilege Principle - Ensure DB accounts used by applications don’t have admin rights.

# Answer 5 - blind SQL injection vulnerabilities

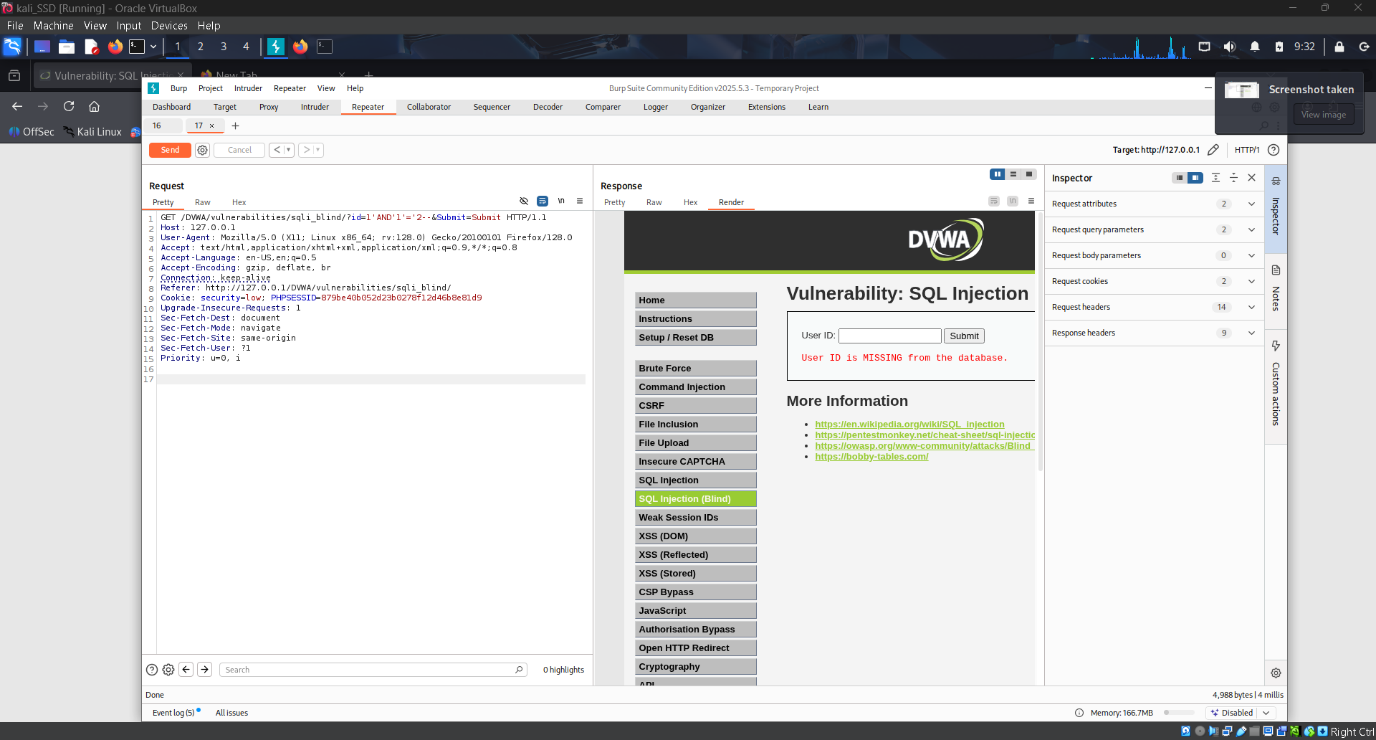
## Detection Strategies

* Boolean-based (Content-based) Detection
* Time-based Detection
* Error-based Testing (in Blind context)
* Automation with Security Tools

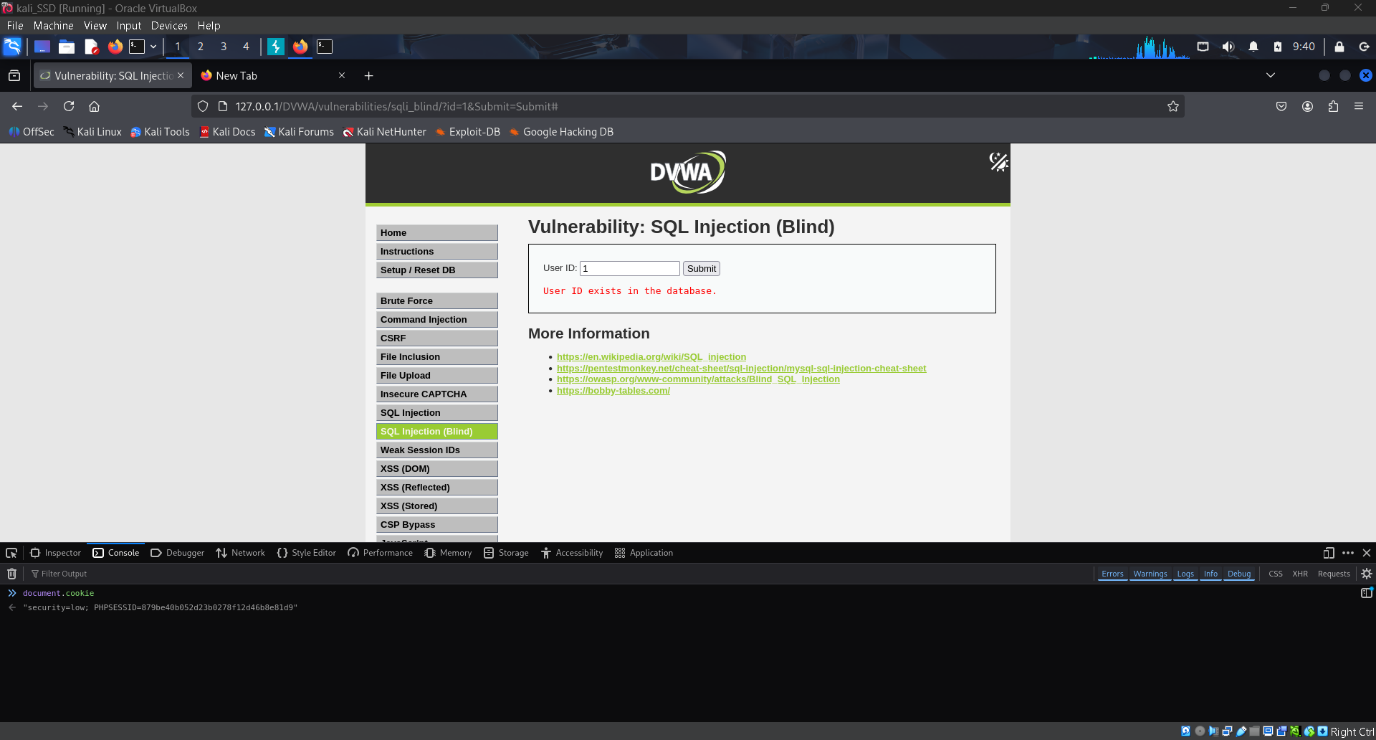
Used Boolean and automation



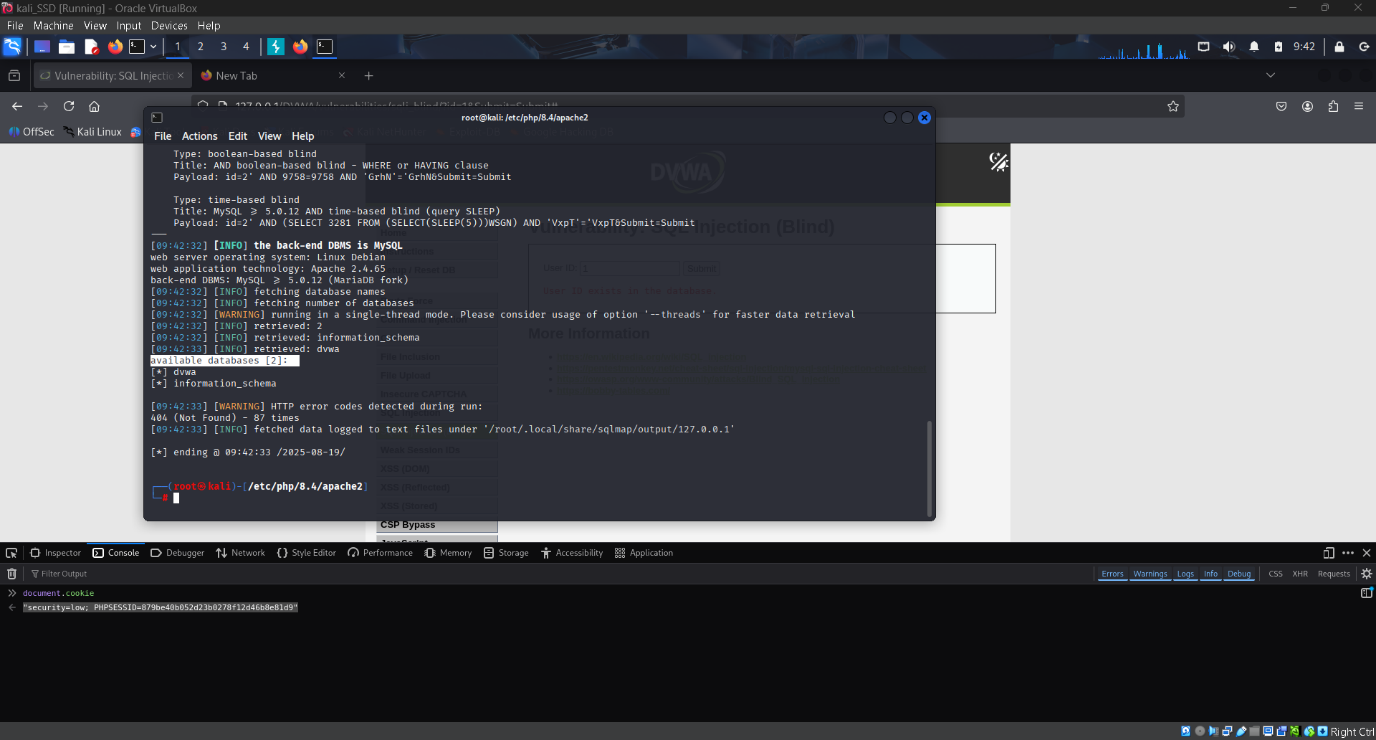
Used Boolean based checking to identify the vulnerability



Used burp to get request



used the console tab to get the cookies.



Used sqlmap to automate the task and get the tables.

## Risks of Blind SQL Injection

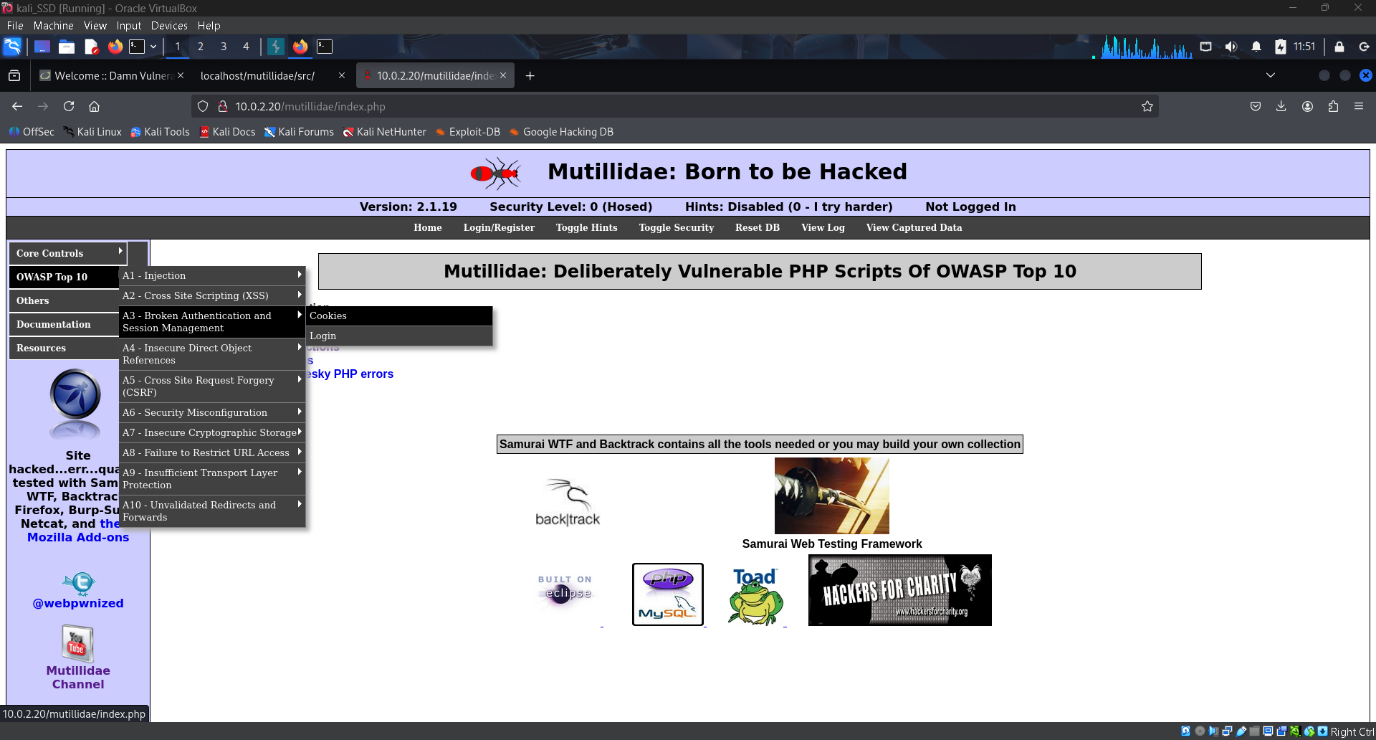
* Data Theft
* Privilege Escalation
* Authentication Bypass
* Database Compromise
* System

# Answer 6 - Cookie-Related Vulnerabilities

## Discovery of Cookie Vulnerabilities

* Check Cookies in Browser/Proxy Tools
* Cookie Poisoning Attack
* Session Hijacking
* Missing Security Flags

I used the Cookie poisoning Attack to gain the administrator privilege



When to the cookie tab of mutillidae



Created the new account and logged into the account



After that I modified the cookie value to gain the admin access

## Mitigation

* Secure Cookie Attributes
* Strong Session Management
* Avoid Storing Sensitive Data in Cookies
* Encrypt Cookies
* Implement HTTPS Everywhere

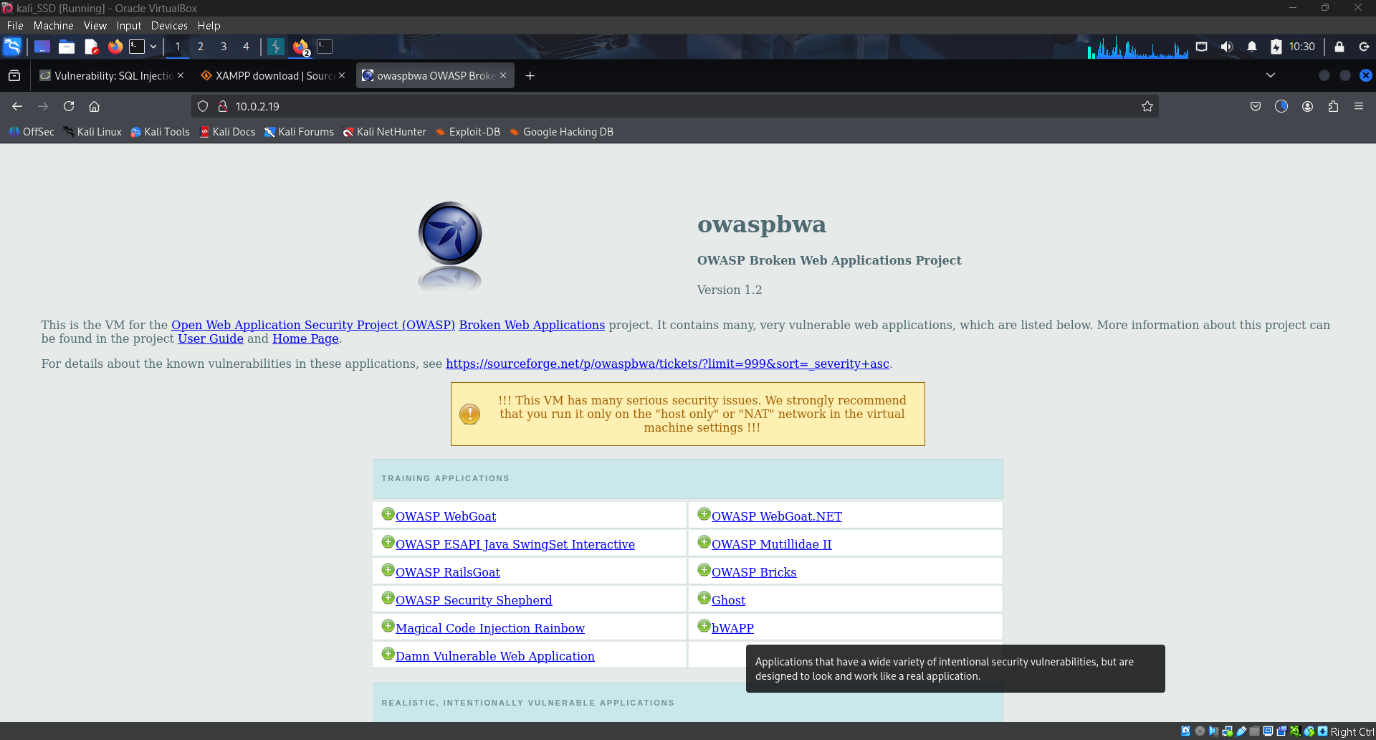
# Answer 7 – SSLScan

SSLScan is a command-line tool that analyzes a server’s SSL/TLS configuration to identify weaknesses.

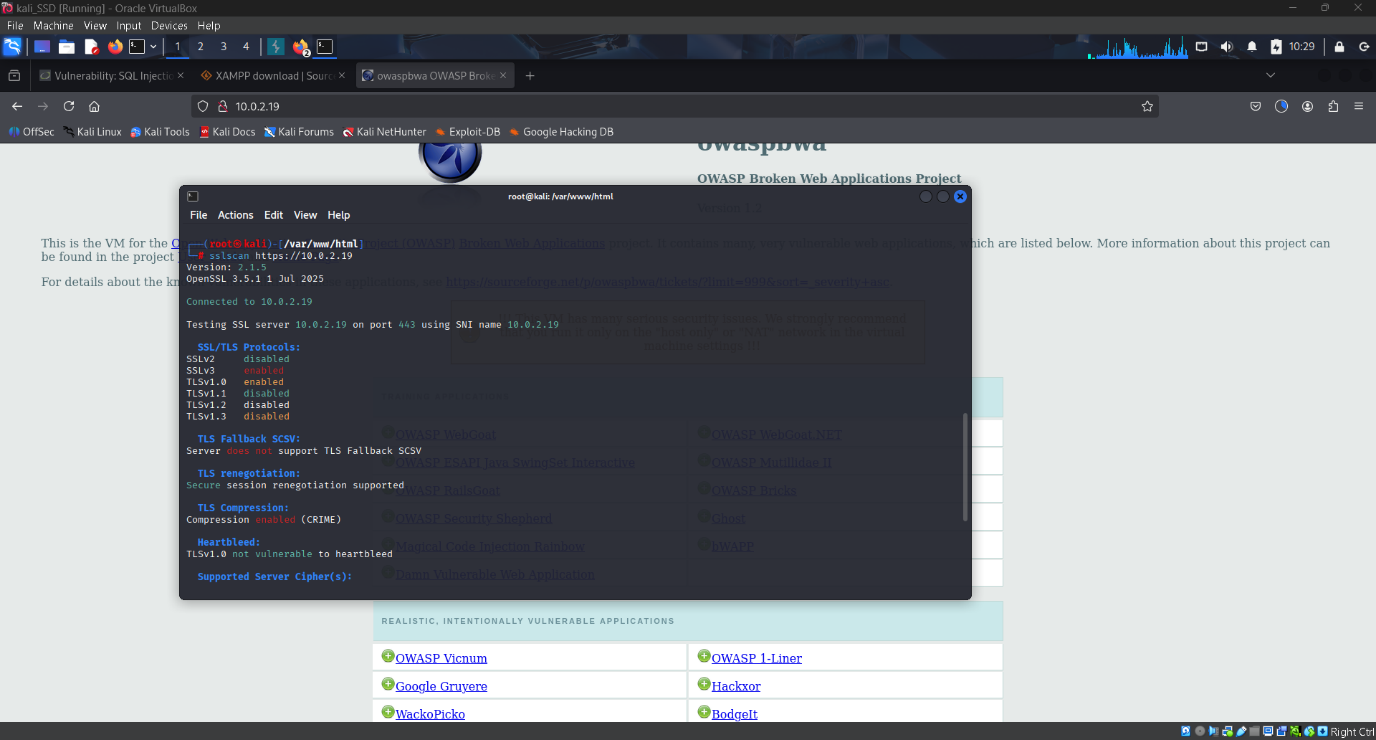
It checks:

* Supported SSL/TLS protocol versions
* Cipher suites enable
* Certificate details (validity, issuer, expiration)
* Vulnerabilities (e.g., SSLv2, SSLv3, weak ciphers, lack of forward secrecy)

Used the command sslscan <http://ip.addr:port/>



Output from the scan



## How SSLScan Contributes to Security Assessments

1.Identifies Outdated Protocols

* Detects SSLv2/SSLv3/TLS 1.0 → must be disabled.
* Ensures TLS 1.2+ or TLS 1.3 are enforced.

2. Finds Weak Ciphers

* Prevents use of RC4, NULL, EXPORT, MD5 ciphers.
* Recommends AES-GCM, ChaCha20, SHA-256/384.

3. Highlights Certificate Problems

* Finds self-signed, expired, mismatched CN.
* Helps ensure certificates are issued by a trusted CA.

4. Assesses Perfect Forward Secrecy (PFS)

* Encourages enabling ECDHE/DHE ciphers.
* Prevents future decryption of past sessions.

5. Supports Compliance (PCI DSS, NIST, ISO 27001)

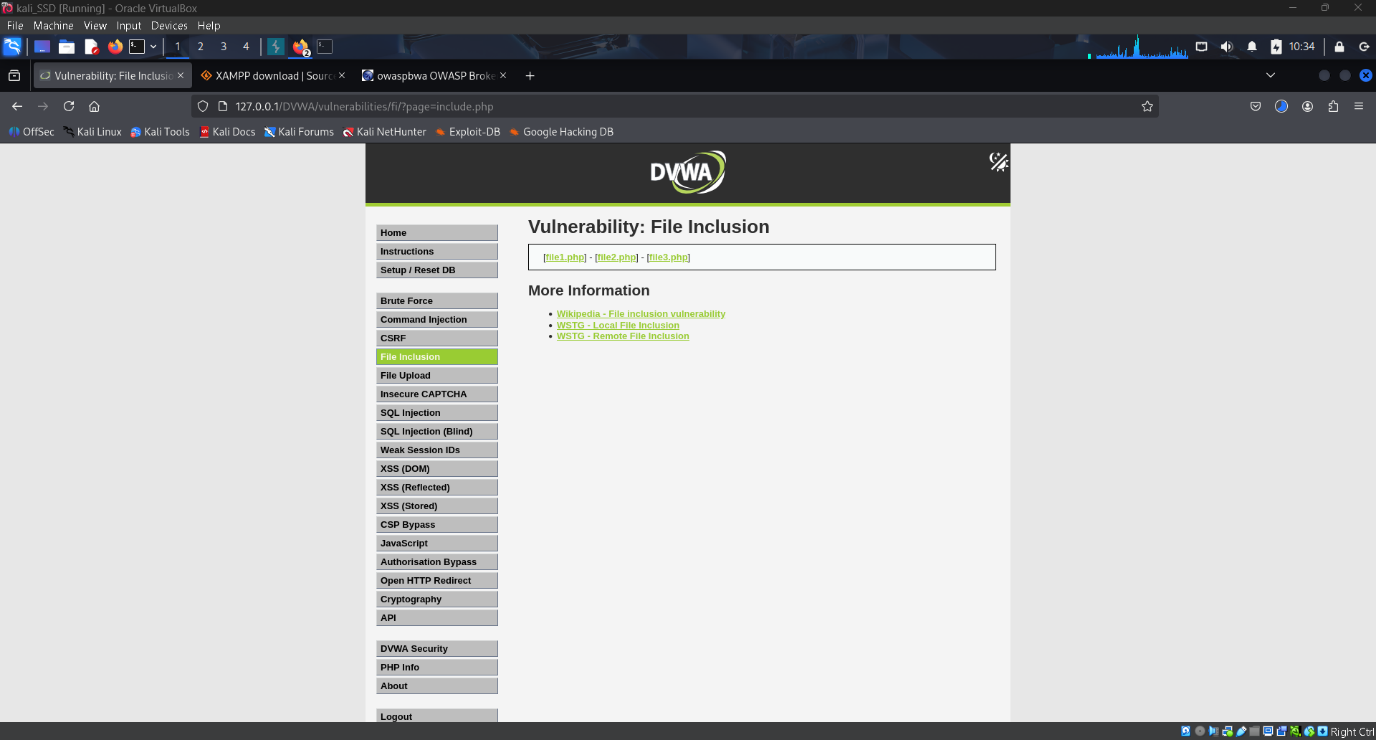
* Ensures SSL/TLS configs meet compliance guidelines.

# Answer 8 - File inclusion vulnerabilities

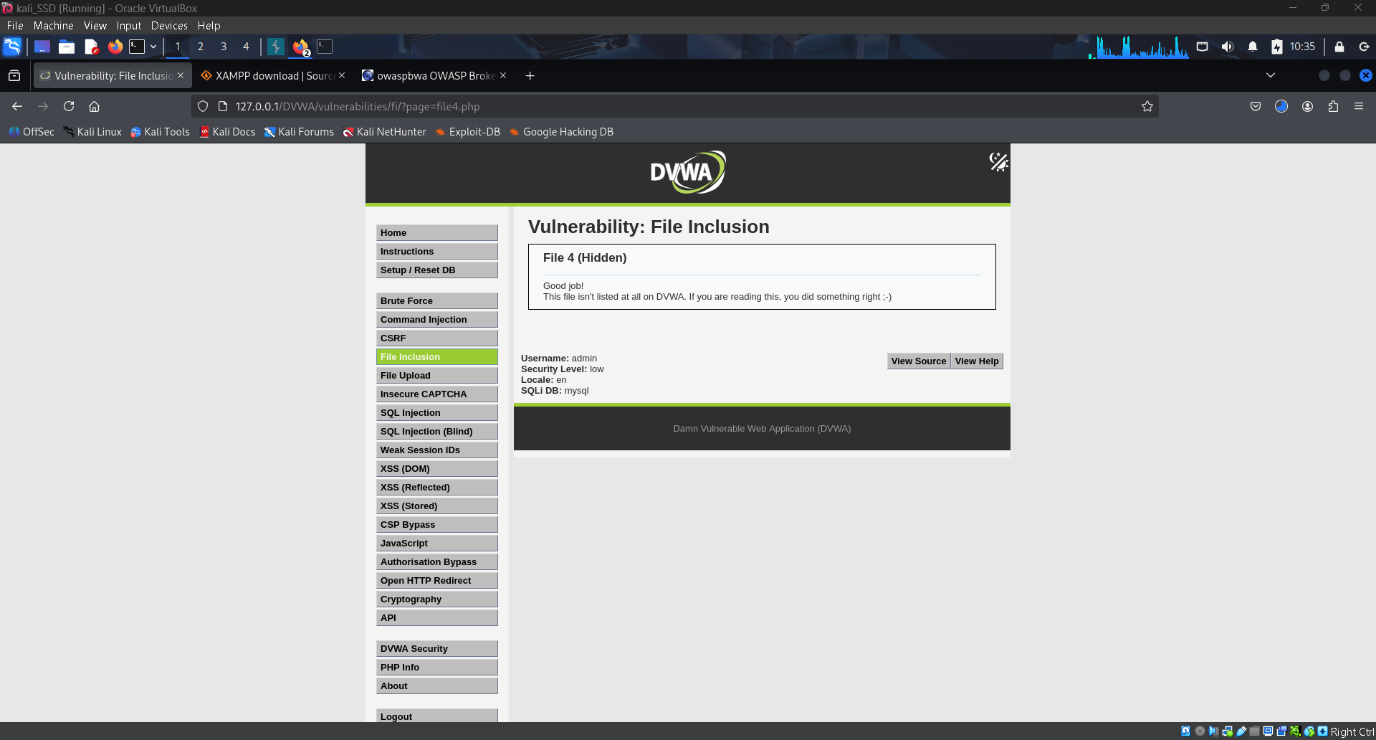
## Methods to Detect File Inclusion Vulnerabilities

* URL Parameter Manipulation
* Traversal Sequences
* Remote File Inclusion (RFI)

Opened the DVWA an went to the file inclusion vulnerability



Used the url parameter manipulation to check the vulnerability



# Answer 9 - mitigation strategies for the vulnerabilities

### 1. OWASP ZAP (Request Tampering / HTTP Manipulation)

Vulnerability: Exposed request parameters and lack of input validation.

Mitigation:

* Enforce strict input validation and sanitization on both client and server sides.
* Implement parameterized queries and whitelisting for accepted inputs.
* Deploy Web Application Firewalls (WAFs) to detect malicious payloads.

### 2. Burp Suite (Request Interception & Manipulation)

Vulnerability: Modifiable HTTP requests (headers, parameters, cookies).

Mitigation:

* Use secure coding practices (never trust user input).
* Apply rate limiting & CSRF tokens to protect sensitive requests.
* Encrypt sensitive data in transit (TLS/HTTPS) and at rest.

3. Cross-Site Scripting (XSS)

Vulnerability: Injection of malicious scripts into web pages (Reflected, Stored, DOM).

Mitigation:

* Encode user input before rendering in HTML (<, >, ', ")
* Use modern Content Security Policy (CSP) to restrict script execution.
* Apply input/output sanitization libraries (e.g., OWASP ESAPI, DOMPurify).
* Adopt frameworks with built-in XSS protection (e.g., React, Angular).

### 4. Error-Based SQL Injection

Vulnerability: Direct injection of SQL commands due to unsanitized inputs.

Mitigation:

* Always use Prepared Statements / Parameterized Queries.
* Remove detailed error messages from production systems.
* Enforce least privilege on database accounts.
* Regularly patch/update DBMS.

### 5. Blind SQL Injection

Vulnerability: Database queries manipulated without visible errors (timing or Boolean-based).

Mitigation:

* Use ORM frameworks (e.g., Hibernate, SQLAlchemy) instead of raw queries.
* Monitor abnormal database query patterns with SIEM/IDS.
* Apply input length restrictions to reduce attack surface.

### 6. Cookie-Related Vulnerabilities

Vulnerability: Session hijacking, weak cookie security, missing flags.

Mitigation:

* Set Secure, HttpOnly, and SameSite flags on cookies.
* Implement session expiration and re-authentication for sensitive actions.
* Store session identifiers in server-side session storage, not cookies.
* Use encryption for session tokens and consider JWT best practices.

### 7. SSL/TLS Misconfigurations (SSLScan Findings)

Vulnerability: Weak ciphers, outdated TLS versions, certificate issues.

Mitigation:

* Enforce TLS 1.2 or TLS 1.3 only.
* Disable weak ciphers & SSLv2/SSLv3/early TLS.
* Use certificates from a trusted Certificate Authority (CA).
* Apply HSTS (HTTP Strict Transport Security) to prevent downgrade attacks.

### 8. File Inclusion Vulnerabilities

Vulnerability: Arbitrary file inclusion (Local File Inclusion - LFI / Remote File Inclusion - RFI).

Mitigation:

* Restrict user input to predefined whitelisted values.
* Disable PHP functions like include(), allow\_url\_include, eval().
* Implement directory traversal protections (realpath(), strict path checks).
* Use least privilege principle on web server accounts.

# Lesson Learned

* From all the vulnerabilities exploited above. One thing is certain that most of the vulnerabilities arises from weak input handling and insecure defaults. Exploits like SQL Injection, XXS and file inclusion are simple product of improper sanitization of the user input. At this point it is safe to say that secure coding practice and defensive programming is essential.
* Having multi-layered security is very important because if one control fail our programme or web application won’t be completely vulnerable. Let’s take XXS as an example even if the site has a cross-site scripting vulnerability but the cookies are protected by other attribute session hijacking can be prevented.
* Attackers usually chain vulnerabilities to get the maximum control of the system. A single flaw allows them to entre but once they are in, they use various techniques to infect as much of the system is possible to increase the overall impact. So, defence in depth is essential.
* Continues monitoring and configuration management is important. Because the outdated system or programmes are more vulnerable to attacks. So continues upgrade and patch management is required of any know vulnerability or application. Also Compliance with standard is crucial.