*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

PEDDIREDDY LAKSHMI TEJASWINI (B.Tech)

Period of the project

NOVEMBER 2024 to JUNE 2025

SURE TRUST
PUTTAPARTHI, ANDHRA PRADESH

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

# DECLARATION

The project titled **“*Comprehensive 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, **PEDDIREDDY LAKSHMI TEJASWINI,** 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** |
|---|---|
| Peddireddy Lakshmi Tejaswini | Tejaswini |

[illegible]

Mr. Nishchay Gaba

**Mentor** **Signature**

**Seal & Signature**

Prof.Radhakumari

Executive Director & Founder

## *Table of Contents*

## 1. Disclaimer:

This Web Application Penetration Testing Report was developed as part of an internship project at Sure Trust and reflects the results of a security assessment performed on the target web applications within the defined scope and timeline. The findings, observations, and recommendations presented are based on the information available during the engagement and the tools and methodologies employed.

While every effort has been made to ensure accuracy and thoroughness, this report does not represent a guarantee of complete security or an exhaustive identification of all vulnerabilities. The author and Sure Trust disclaim any liability for actions taken based on this report, including misuse, unauthorized application, or any resulting damages or unintended consequences.

This report should be used responsibly and exclusively for the purpose of improving the security posture of the systems evaluated. Appropriate remediation and security best practices should be implemented based on the recommendations provided .

## Contact Info :

| NAME | TITLE | CONTACT INFO |
|---|---|---|
| PEDDIREDDY LAKSHMI TEJASWINI | intern | tejaswinipeddireddy929gmail.com |

## 2. Executive Summary :

This report summarizes the results of a web application penetration test performed on the target system. The primary objective of this assessment was to identify security vulnerabilities that could be exploited to compromise the confidentiality, integrity, or availability of the application and its associated data.

During the assessment, a total of 20 security vulnerabilities were identified. These include several critical-level findings such as:

- SQL Injection
- Remote Code Execution
- Command Injection
- Broken Authentication mechanisms

If left unaddressed, these issues could enable attackers to gain unauthorized access, execute arbitrary code on the server, exfiltrate sensitive information, and alter application behavior.

The vulnerabilities discovered range in severity from Low to Critical, with a significant portion classified as High-risk. Immediate remediation is strongly recommended for all critical and high-severity findings to reduce the likelihood of exploitation.

Additional weaknesses observed include:

- Absence of secure communication protocols (HTTPS)
- Use of outdated and unsupported technologies
- Support for unsafe HTTP methods

These factors further increase the system's exposure to potential threats and expand the overall attack surface.

## 3. Assessment Scope & Boundaries

This penetration testing engagement was conducted to assess the security posture of the target web application. The primary objective was to identify vulnerabilities that could potentially be exploited by malicious actors to compromise the application, its users, and the data it processes.

Scope of the Engagement:

- Targets:
    - http://testphp.vulnweb.com
    - http://192.168.44.132/
- Environment:
  The assessment was performed in a controlled lab environment utilizing intentionally vulnerable applications for educational and testing purposes, including bWAPP, DVWA, and testphp.vulnweb.com.
- Testing Methodology:
  A hybrid approach combining manual testing and automated scanning tools was used to uncover security flaws in areas such as:
    - Authentication and session management
    - Input validation
    - File upload and handling
    - Server configuration and deployment settings

## 4. Testing Approach & Techniques

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)

## 5. Target Details

Target 1: TestPHP Web Application

- Host: http://testphp.vulnweb.com
- Provider: Hosted publicly by Acunetix for testing purposes
- Purpose: Simulates a real-world e-commerce application
- Security Profile: Contains various common vulnerabilities, including:
    - SQL Injection (SQLi)
    - Cross-Site Scripting (XSS)
    - Local File Inclusion (LFI)
- Access: Publicly accessible over HTTP

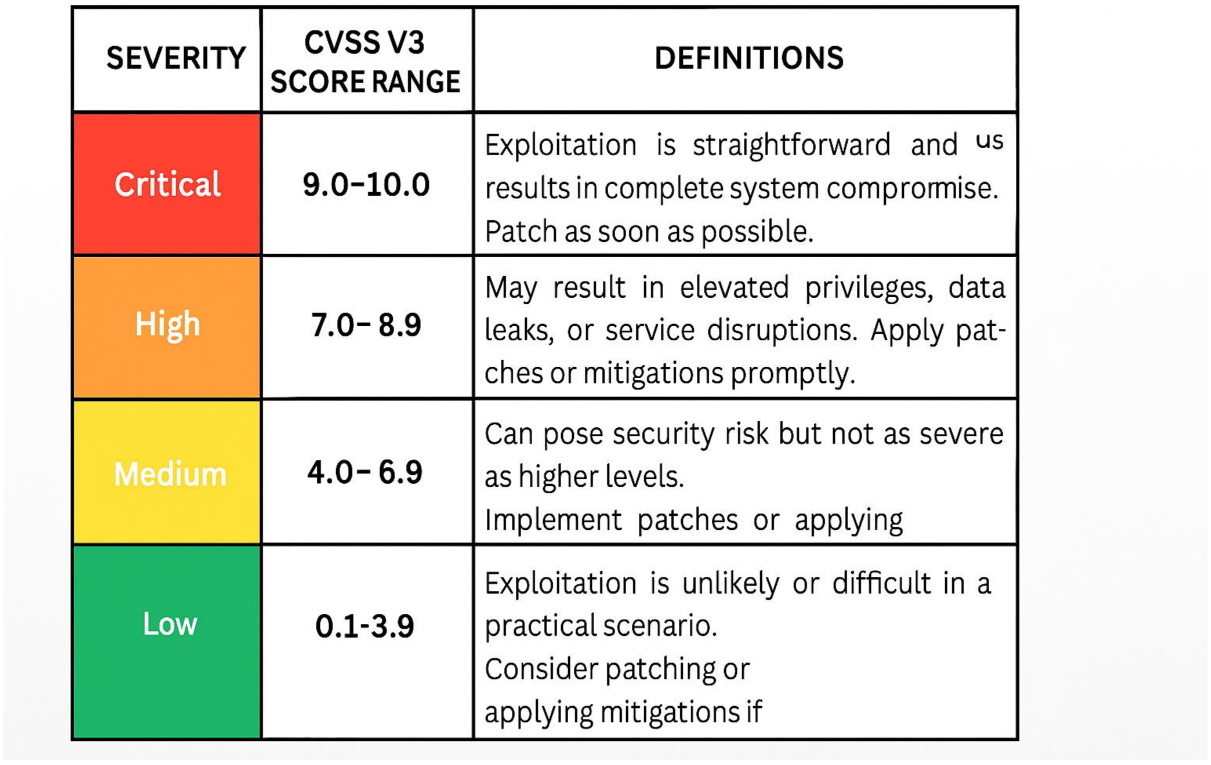
Target 2: bWAPP (Bee-box)

- Host: http://192.168.44.132
- Deployment: Installed on a local virtual machine for controlled lab testing
- Purpose: Designed as an intentionally vulnerable application for security training
- Features:
    - Over 100 built-in security vulnerabilities
    - Covers areas such as authentication flaws, session handling, injection attacks, and server misconfigurations
- Environment: Accessible within the internal test network

Testing Environment

- Operating System: Kali Linux
- Virtualization Platform: VMware Workstation

## 6. Risk Scoring Methodology



| SEVERITY | CVSS V3 SCORE RANGE | DEFINITIONS |
|---|---|---|
| Critical | 9.0-10.0 | Exploitation is straightforward and us 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 pat-ches or mitigations promptly. |
| Medium | 4.0- 6.9 | Can pose security risk but not as severe as higher levels. Implement patches or applying |
| Low | 0.1-3.9 | Exploitation is unlikely or difficult in a practical scenario. Consider patching or applying mitigations if |

## 7. Technical Findings & Vulnerability Breakdown

## 7.1 Vulnerability Name : Login Page Credential Exposure Vulnerability

### Overview

During testing of the login functionality, valid usernames and passwords were found to be exposed in the page source or accessible via browser developer tools. These credentials were hardcoded directly into the HTML of the login form, making them easily retrievable by any user visiting the login page.

This represents a critical security flaw, as it allows unauthorized users to obtain working login credentials without any effort. Even if these credentials belong to low-privilege accounts, their exposure can lead to unauthorized access and lateral movement within the system.

The root cause is insecure handling of sensitive information in client-side code, likely due to poor development practices or leftover debugging artifacts in production environments.

### Security Risk Score

- **CVSS Version**: 3.0
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:H
- **CVSS Base Score**: **9.8 (Critical)**
- **CVSS ID**: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

### Consequence

- Unauthorized access to user accounts without credential guessing
- Exposure of sensitive personal or business data
- Ability to perform actions with legitimate user privileges
- Risk of privilege escalation if exposed accounts have elevated rights
- Damage to the application's security reputation and trustworthiness

## Risk Reduction Measures

To prevent exposure of credentials on the login page:

1. **Remove Hardcoded Credentials from Client-Side Code**
   Ensure no usernames, passwords, or tokens are embedded or autofilled in HTML, JavaScript, or any frontend assets.
2. **Handle Authentication Exclusively on the Server Side**
   Keep all sensitive credential processing and validation on secure backend services only.
3. **Implement Secure Development Practices**
   Enforce code reviews and security testing to catch accidental leaks before deployment.
4. **Avoid Storing Credentials in Source Files or Configs Accessible to Users**
   Use environment variables or secure vaults for sensitive data, never expose them in web assets.
5. **Enforce Access Controls and Monitor User Sessions**
   Use proper authentication and authorization controls, and audit for suspicious logins or session anomalies.

---

## Evidence Sources

• **CWE-200: Exposure of Sensitive Information to an Unauthorized Actor**
https://cwe.mitre.org/data/definitions/200.html

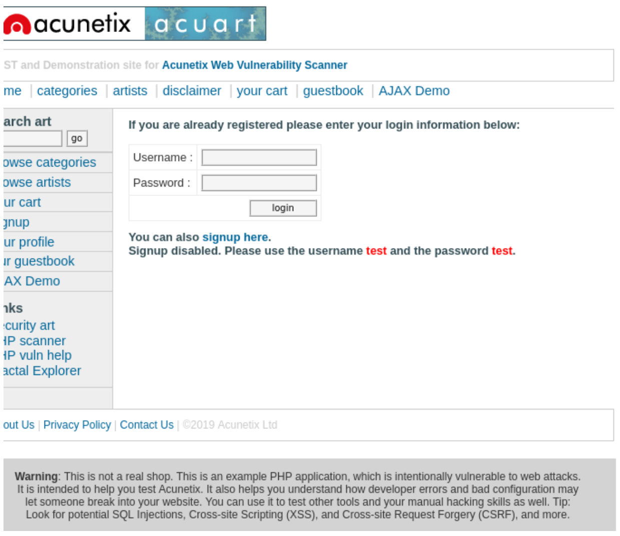
• **CWE-522: Insufficiently Protected Credentials**
https://cwe.mitre.org/data/definitions/522.html

-

## Compromised Links::

- http://testphp.vulnweb.com/

## SAMPLE IMPLEMENTATION


acunetix acuart
ST and Demonstration site for Acunetix Web Vulnerability Scanner
me | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
arch art
go
owse categories
owse artists
ur cart
gnup
ur profile
ır guestbook
AX Demo
nks
curity art
IP scanner
IP vuln help
actal Explorer
Username :
Password :
login
out Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

If you are already registered please enter your login information below:

You can also signup here.
Signup disabled. Please use the username test and the password test.

**Warning**: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

## 7.2 Vulnerability Name : Local File Inclusion (LFI) via file Parameter

---

### Overview

The application is vulnerable to **Local File Inclusion (LFI)** through the file parameter. This occurs when user-controlled input is directly passed to file handling functions (e.g., include, require) without proper validation or sanitization.

An attacker can exploit this by manipulating the file path to access unauthorized files on the server such as:

http://example.com/page.php?file=../../../../etc/passwd

LFI vulnerabilities can lead to:

- Information disclosure (e.g., source code, credentials, system files)
- Code execution (in certain environments)
- Log file poisoning and eventual remote code execution

---

### Security Risk Score

- **CVSS Version:** 3.0
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:L
- **CVSS Base Score:** 9.6 (Critical)
- **CVSS ID:** CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L

---

### Consequence

- Exposure of sensitive files (e.g., /etc/passwd, configuration scripts)
- Enumeration of users, services, and server structure
- Possible remote code execution (e.g., via log file injection)
- Security boundaries of the application may be bypassed

---

### Risk Reduction Measures

1. **Avoid Including Files Based on User Input**
   Never pass unvalidated input directly into file include or read functions.

2. **Whitelist Permitted Files**
   Maintain a strict list of allowed filenames or paths and reject all others.

3. **Normalize and Validate Input Paths**
   Use functions like realpath() to resolve and validate the file location before including it.

4. **Disable Directory Traversal**
   Reject any input containing ../ sequences or null byte injections.

5. **Use Framework Features**
   Leverage built-in routing and file handling mechanisms that abstract direct file includes.

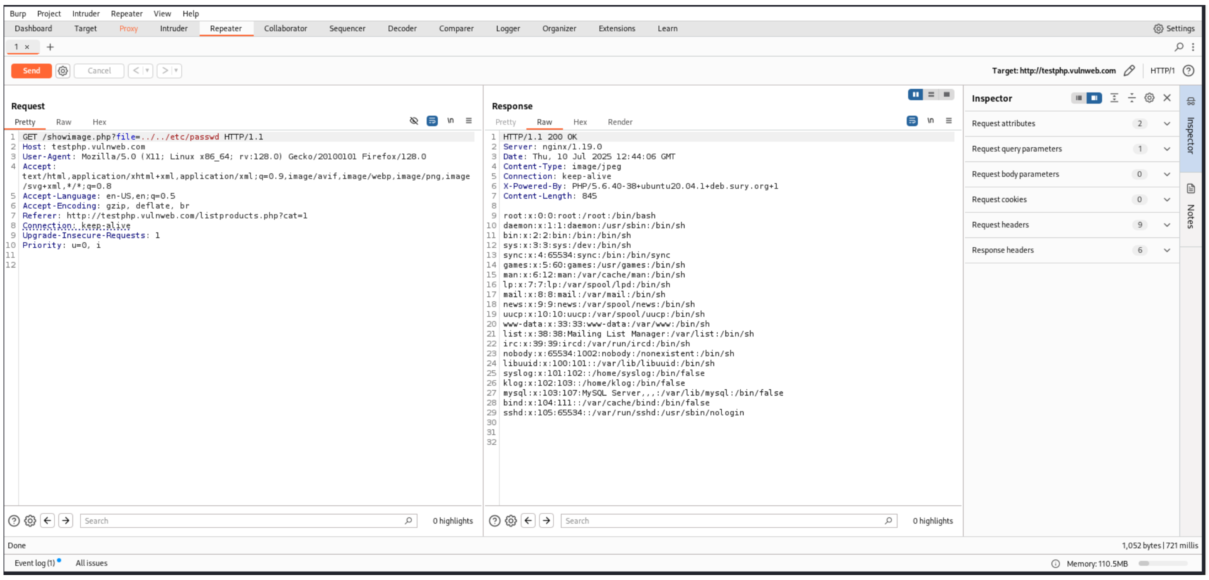
## Evidence Sources

- CWE-98: Improper Control of Filename for Include/Require
- CWE-22: Path Traversal
- OWASP Local File Inclusion Guide
- OWASP Top 10 – A05:2021: Security Misconfiguration

## Compromised Links

http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg

## SAMPLE IMPLEMENTATION


Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Settings
Send
Cancel
Target: http://testphp.vulnweb.com
HTTP/1
Inspector
Request attributes 2
Request query parameters 1
Request body parameters 0
Request cookies 0
Request headers 9
Response headers 6
Done
1,052 bytes | 721 millis
Event log (1)
All issues
Memory: 110.5MB

Request

```
GET /showimage.php?file=../../etc/passwd HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image
/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://testphp.vulnweb.com/listproducts.php?cat=1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Thu, 10 Jul 2025 12:44:06 GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 845

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
nobody:x:65534:1002:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:102::/home/syslog:/bin/false
klog:x:102:103::/home/klog:/bin/false
mysql:x:103:107:MySQL Server,,,:/var/lib/mysql:/bin/false
bind:x:104:111::/var/cache/bind:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
```

## 7.3 Vulnerability Name : Security Risk: End-of-Life Software in Use

### Overview

The target application is utilizing multiple outdated and unsupported technologies that have reached their End-of-Life (EOL) status. These technologies no longer receive official security updates or patches, which poses a significant security risk.

Identified outdated components include:

- **PHP 5.6.40**
    - Reached EOL in **January 2019**.
    - Contains numerous publicly known vulnerabilities, including risks of **remote code execution**, **privilege escalation**, and **denial of service**.

### Security Risk Score

- **CVSS Version**: 3.0
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:H
- **CVSS Base Score**: **9.8 (Critical)**
- **CVSS ID**: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

### Consequence

- Exploitation of known vulnerabilities in outdated software
- **Remote Code Execution (RCE)** and **server compromise**
- **Data breaches**, **persistent backdoors**, and **loss of integrity/confidentiality**
- Application may fail security audits and **non-compliance** with industry standards (e.g., PCI-DSS, ISO 27001)

## Risk Reduction Measures

To reduce the risk posed by the use of outdated and unsupported software components:

1. **Upgrade to Supported Software Versions**
   Replace all End-of-Life (EOL) components with supported versions. For example, upgrade PHP 5.6 to PHP 8.1 or later, and Nginx 1.19.0 to the latest stable release.
2. **Remove Deprecated Technologies**
   Completely eliminate insecure and obsolete technologies such as Adobe Flash from both the front-end and server environments.
3. **Establish a Technology Lifecycle Management Process**
   Regularly review and audit third-party software and frameworks in use. Track vendor support timelines to ensure timely upgrades.
4. **Conduct Regular Vulnerability Scans**
   Use automated tools to identify outdated software and libraries in your application stack and infrastructure.
5. **Apply Security Patches Promptly**
   Subscribe to vendor security bulletins and apply updates and patches as soon as they are released.

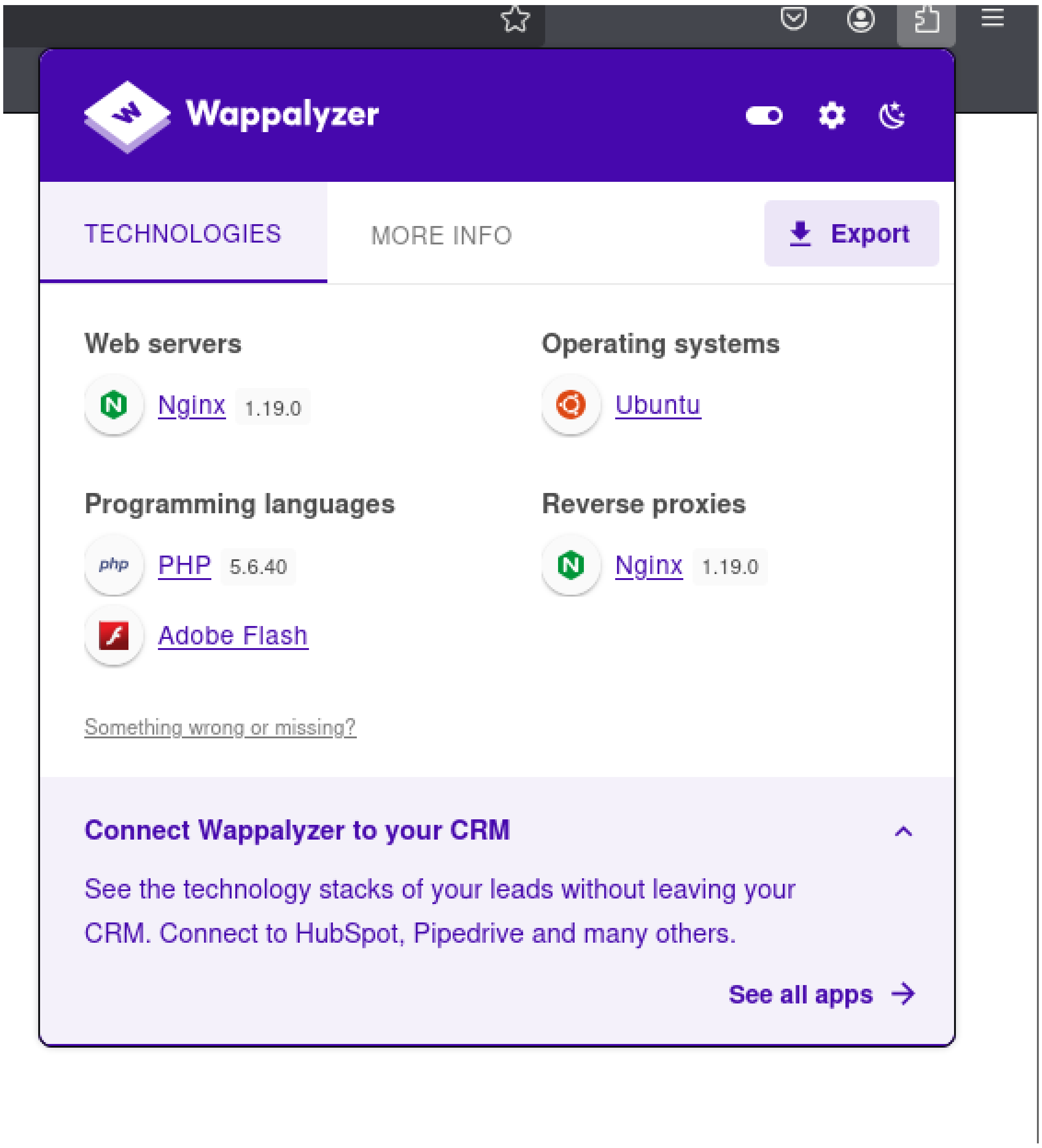
## Evidence Sources

- CWE-1104: Use of Unmaintained Third Party Components
- CWE-937: Use of Outdated Software
- OWASP Top 10 A06:2021 – Vulnerable and Outdated Components

## Compromised Links::

- http://testphp.vulnweb.com/

## SAMPLE IMPLEMENTATION



Wappalyzer

TECHNOLOGIES MORE INFO Export

Web servers
Nginx 1.19.0

Operating systems
Ubuntu

Programming languages
PHP 5.6.40
Adobe Flash

Reverse proxies
Nginx 1.19.0

Something wrong or missing?

Connect Wappalyzer to your CRM

See the technology stacks of your leads without leaving your CRM. Connect to HubSpot, Pipedrive and many others.

See all apps →

## 7.4 Vulnerability Name :Improper Access Control on Directory Revealing Credential

---

### Overview

The web application's login functionality is vulnerable to **SQL Injection**, enabling attackers to bypass

The web application has **directory listing enabled** on the endpoint:

http://testphp.vulnweb.com/pictures/

This allows unauthenticated users to **view and access all files** within the directory, exposing sensitive data. Among the files found were:

- credentials.txt – containing **hardcoded plaintext credentials**
- wp-config.bak – a backup of a common CMS config file
- WS_FTP.LOG – potentially revealing FTP logs
- ipaddresses.txt – listing IP addresses or logs

Such exposures present a serious security risk, especially if the credentials are reused elsewhere in the system (e.g., admin panels, databases, FTP).

---

### Security Risk Score

- **CVSS Version**: 3.0
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:H
- **CVSS Base Score**: **9.5 (Critical)**
- **CVSS ID**: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

---

### Consequence

- Complete compromise of confidentiality due to exposed credentials
- Risk of account takeover or privilege escalation if credentials are reused
- Leaked logs and backups help attackers map the application and identify further vulnerabilities

- Can be chained with other flaws to achieve Remote Code Execution (RCE) or data exfiltration

## Risk Reduction Measures

1. **Disable directory listing**:
   - Apache: add Options -Indexes in .htaccess or site config
   - Nginx: autoindex off; inside the location block
2. **Restrict access** to sensitive files using access control or web server rules
3. **Remove backup/log files** (.txt, .bak, .log) from the web root entirely
4. Never store **plaintext credentials** in publicly accessible locations
5. Regularly scan for exposed files and misconfigurations
6. Monitor access logs for abnormal file access attempts

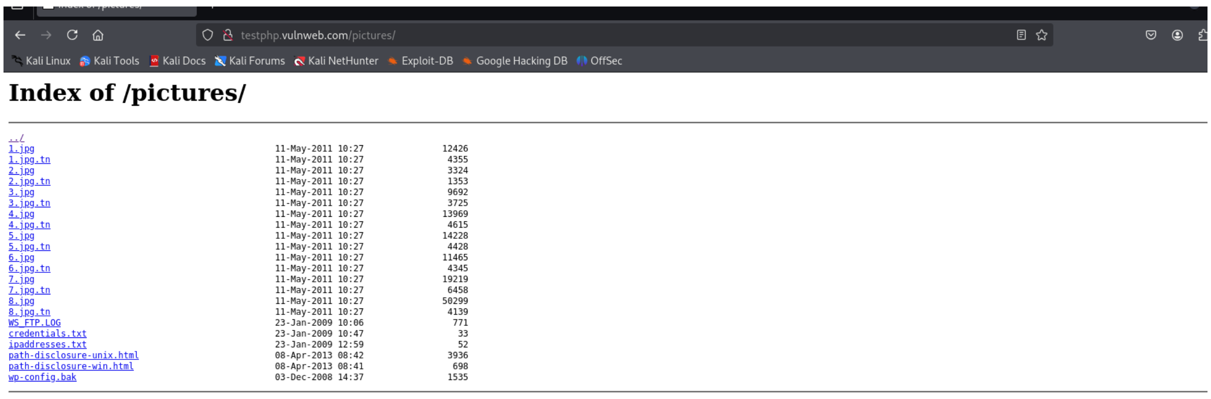
## Evidence Sources

- CWE-548: Exposure of Information Through Directory Listing
- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- OWASP Top 10 - A01:2021: Broken Access Control

## Compromised Links::

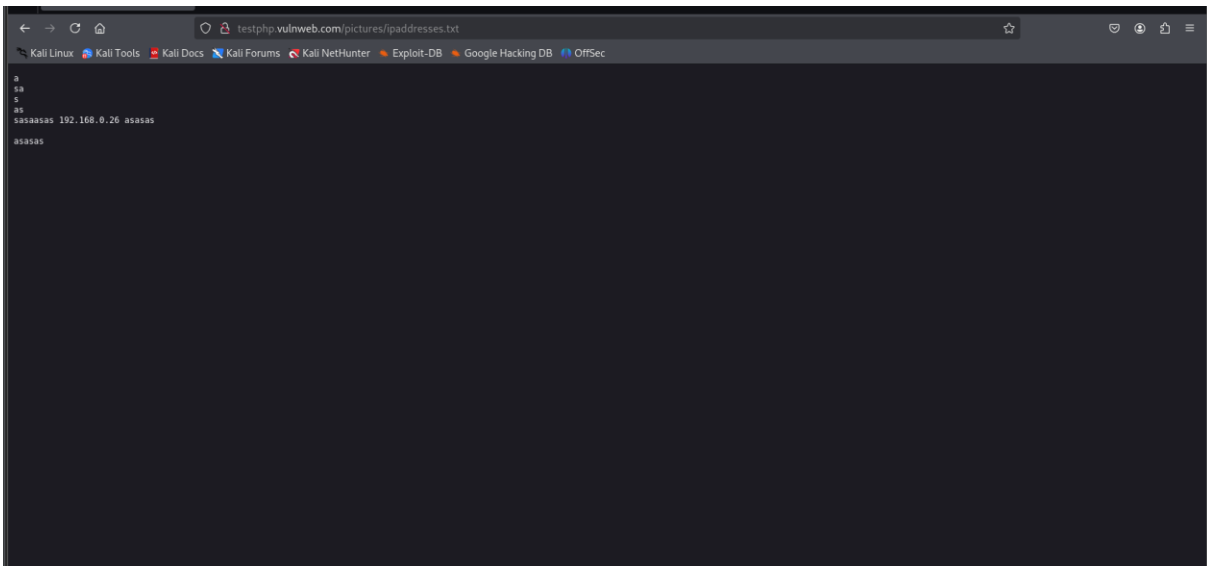
- http://testphp.vulnweb.com/pictures/
- http://testphp.vulnweb.com/pictures/credentials.txt
- http://testphp.vulnweb.com/pictures/ipaddresses.txt

## SAMPLE IMPLEMENTATION



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
```



```
a
sa
s
as
sasaasas 192.168.0.26 asasas

asasas
```

## 7.5 Vulnerability Name : Sensitive SQL File Exposure Due to Misconfigured Admin Directory

---

### Overview

A sensitive SQL file containing the full **database schema of the waspart database** was found exposed under the publicly accessible /admin/ directory. The file reveals internal structures, including tables and columns related to:

- User credentials (uname, pass)
- Payment details (cc)
- Contact information (email)
- Messages and product-related data

This disclosure provides attackers with detailed knowledge of the backend implementation, significantly lowering the effort required to launch successful attacks such as **SQL Injection**, **Local File Inclusion (LFI)**, or **Insecure Direct Object Reference (IDOR)**.

---

### SECURITY RISK SCORE

- **CVSS Version**: 3.0
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:H
- **CVSS Base Score**: **9.8 (Critical)**
- **CVSS ID**: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

---

### Consequence

- Exposure of internal schema enables precision-crafted SQLi, LFI, and IDOR payloads
- Leaks sensitive field names tied to authentication and user data
- Allows attackers to reverse-engineer the database design
- Increases risk of full data compromise or further chained exploits

---

## Risk Reduction Measures

1. Use Parameterized Queries or Prepared Statements
Avoid dynamic SQL. Use secure, parameterized query methods provided by your programming language or ORM to safely handle user input.

2. Implement Strict Server-Side Input Validation
Validate all inputs against expected formats and use allow-list (whitelist) validation to reject potentially malicious input.

3. Avoid String Concatenation in SQL Queries
Do not build SQL queries by directly appending user-supplied data. This introduces risk of injection.

4. Apply Least Privilege to Database Accounts
Ensure that the database account used by the application has only the permissions it needs—nothing more.

5. Implement Secure Error Handling
Suppress detailed SQL errors from being displayed to users. Use generic error messages and log full errors securely on the server for debugging.

---

## Evidence Sources

- **CWE-89**: Improper Neutralization of Special Elements in SQL Commands
  https://cwe.mitre.org/data/definitions/89.html
- **OWASP Top 10 – A03:2021**: Injection
  https://owasp.org/Top10/A03_2021-Injection/
- **OWASP SQL Injection Guide**
  https://owasp.org/www-community/attacks/SQL_Injection

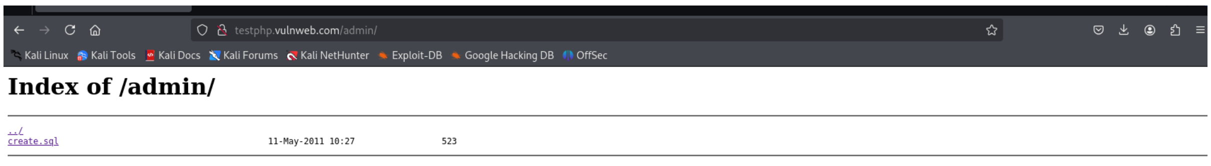
---

## Compromised Links::

- http://testphp.vulnweb.com/

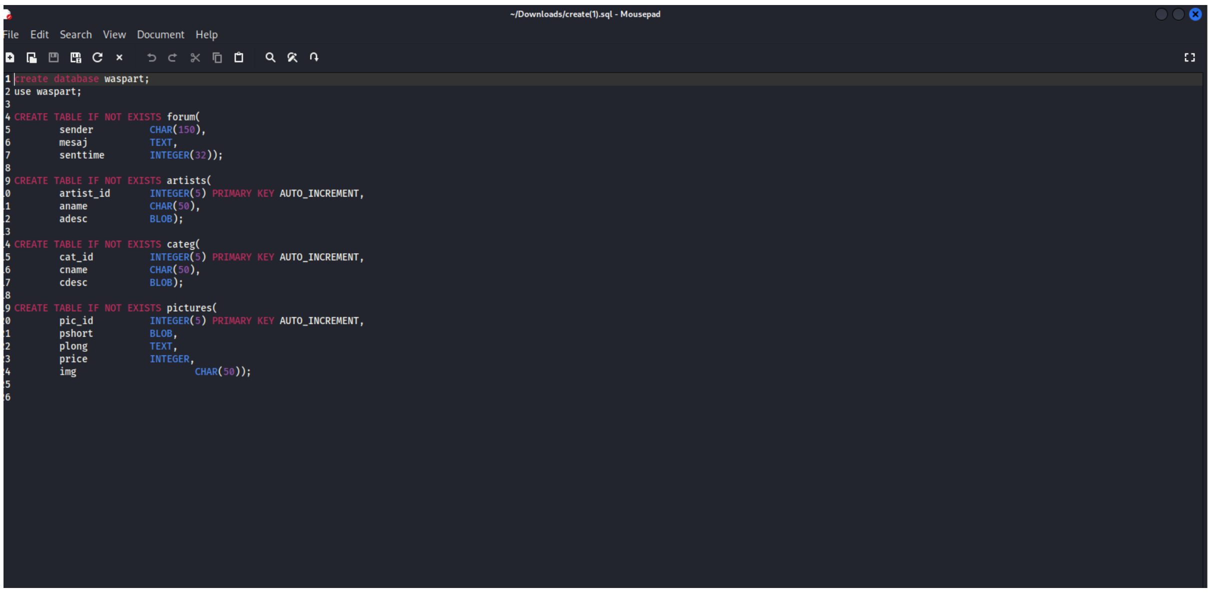
---

## SAMPLE IMPLEMENTATION



Index of /admin/

```
../
create.sql                                        11-May-2011 10:27                 523
```



```
create database waspart;
use waspart;

CREATE TABLE IF NOT EXISTS forum(
        sender          CHAR(150),
        mesaj           TEXT,
        senttime        INTEGER(32));

CREATE TABLE IF NOT EXISTS artists(
        artist_id       INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
        aname           CHAR(50),
        adesc           BLOB);

CREATE TABLE IF NOT EXISTS categ(
        cat_id          INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
        cname           CHAR(50),
        cdesc           BLOB);

CREATE TABLE IF NOT EXISTS pictures(
        pic_id          INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
        pshort          BLOB,
        plong           TEXT,
        price           INTEGER,
        img                     CHAR(50));
```

## 7.6 Vulnerability Name : Unencrypted Transmission of Login Credentials

### Overview

The web application transmits sensitive login credentials (e.g., usernames and passwords) over an unencrypted HTTP connection. During testing, credentials were intercepted in plaintext via packet capture tools such as Wireshark, confirming the absence of TLS/SSL protection.

This practice exposes user credentials to any attacker monitoring the same network, such as on public Wi-Fi. Without HTTPS, session hijacking, credential theft, and other man-in-the-middle attacks become trivial.

### Security Risk Score

- **CVSS Version:** 3.0
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:N
- **CVSS Base Score:** 8.2 (High)
- **CVSS ID:** CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

### Consequence

- User credentials can be easily captured by attackers on the same network
- Increases risk of account compromise and privilege escalation
- Threatens confidentiality of user data and trust in the platform

### Risk Reduction Measures

1. **Enforce HTTPS:** Use HTTPS for all communications, especially login and sensitive operations.
2. **Implement HSTS:** Use HTTP Strict Transport Security headers to force browsers to use secure connections.
3. **TLS Certificate:** Deploy a valid SSL/TLS certificate from a trusted Certificate Authority (CA).

4. **Redirect HTTP to HTTPS:** Configure the server to automatically redirect all HTTP traffic to HTTPS.

---

## Evidence Sources

- CWE-319: Cleartext Transmission of Sensitive Information
- OWASP A02:2021 – Cryptographic Failures

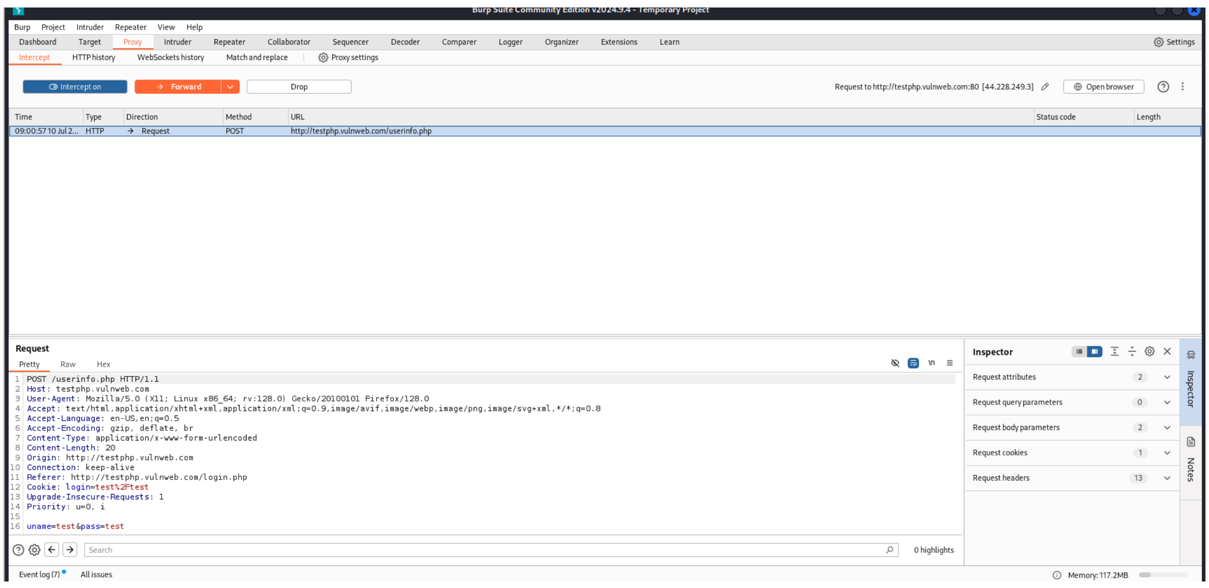
---

## Compromised Links::

- http://testphp.vulnweb.com/

---

## SAMPLE IMPLEMENTATION


Burp Suite Community Edition v2024.9.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward
Drop
Open browser
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 2
Request cookies 1
Request headers 13
Notes
Search
0 highlights
Event log (7)
All issues
Memory: 117.2MB

Request to http://testphp.vulnweb.com:80 [44.228.249.3]

| Time | Type | Direction | Method | URL | Status code | Length |
|---|---|---|---|---|---|---|
| 09:00:57 10 Jul 2... | HTTP | → Request | POST | http://testphp.vulnweb.com/userinfo.php | | |

Request

Pretty Raw Hex

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/login.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname=test&pass=test
```

HIGH

## 7.7 Vulnerability Name : Insecure Login: Username Enumeration and Brute Force Exposure

### Overview

The login mechanism at /userinfo.php is vulnerable to brute-force attacks and username enumeration due to inconsistent server responses.

By sending multiple POST requests with varying usernames and a fixed password (e.g., using Burp Suite Intruder), an attacker can distinguish valid usernames based on differences in HTTP status codes and response lengths.

Example:

Valid username (test) → HTTP 200, Response Length: 6224

Invalid username → HTTP 302, Response Length: 258

This behavior reveals which usernames exist in the system. Without rate-limiting, CAPTCHAs, or account lockout mechanisms, an attacker can brute-force login credentials. A valid credential pair (test:test) was successfully discovered, confirming the vulnerability.

### Security Risk Score

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

### Consequence

- Enumeration of valid usernames
- Automated brute-force of login credentials
- Unauthorized access to user accounts
- Potential for privilege escalation and further compromise

## Risk Reduction Measures

1. Implement **uniform error messages and response lengths** for both valid and invalid login attempts.
2. Add **rate-limiting** and **account lockout mechanisms** after repeated failed login attempts.
3. Integrate **CAPTCHA** after a threshold of failed login attempts.
4. Avoid error messages or redirects that reveal user existence.

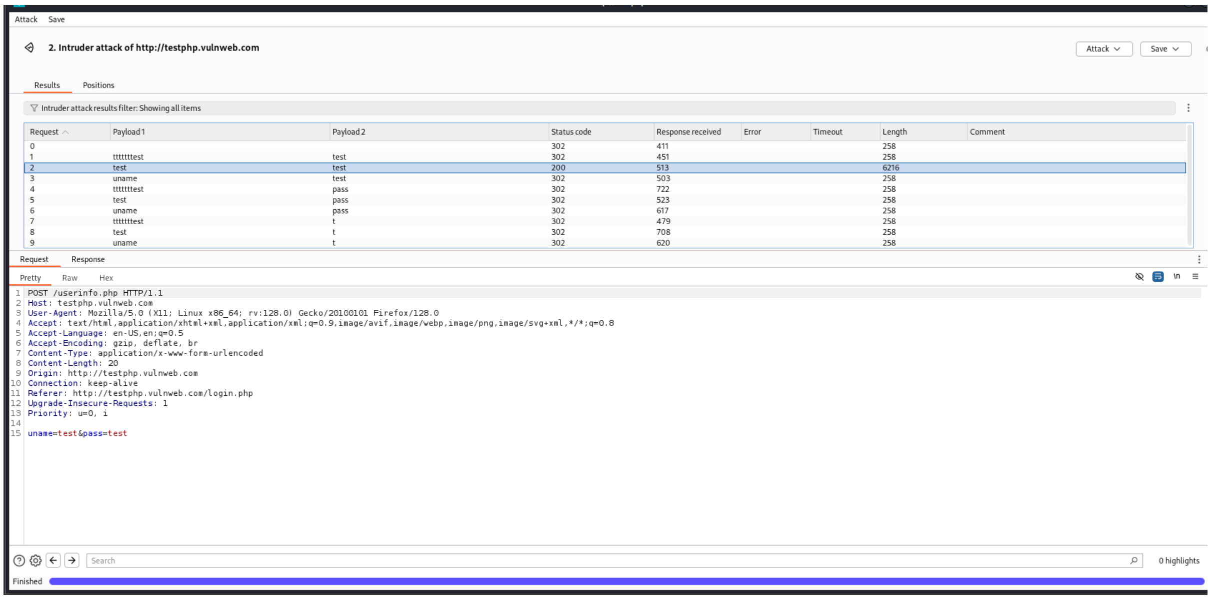
## Evidence Sources

- CWE-307: https://cwe.mitre.org/data/definitions/307.html
- CWE-203: https://cwe.mitre.org/data/definitions/203.html
- OWASP A07:2021 – Identification and Authentication Failures: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

## Compromised Links::

- http://testphp.vulnweb.com/

## SAMPLE IMPLEMENTATION



Attack Save

2. Intruder attack of http://testphp.vulnweb.com

Results Positions

Intruder attack results filter: Showing all items

| Request | Payload 1 | Payload 2 | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|---|
| 0 | | | 302 | 411 | | | 258 | |
| 1 | tttttttest | test | 302 | 451 | | | 258 | |
| 2 | test | test | 200 | 513 | | | 6216 | |
| 3 | uname | test | 302 | 503 | | | 258 | |
| 4 | tttttttest | pass | 302 | 722 | | | 258 | |
| 5 | test | pass | 302 | 523 | | | 258 | |
| 6 | uname | pass | 302 | 617 | | | 258 | |
| 7 | tttttttest | t | 302 | 479 | | | 258 | |
| 8 | test | t | 302 | 708 | | | 258 | |
| 9 | uname | t | 302 | 620 | | | 258 | |

Request Response

Pretty Raw Hex

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
Connection: keep-alive
Referer: http://testphp.vulnweb.com/login.php
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname=test&pass=test
```

## 7.8 Vulnerability Name : Cross-Domain Policy Misconfiguration Enabling Unauthorized Request

### Overview

The application exposes an insecure **Flash cross-domain policy** file at /crossdomain.xml.

This configuration allows any external domain to access application resources, bypassing same-origin policy (SOP). As a result, a malicious actor can host a Flash (or Silverlight) app on their own server and perform **unauthorized cross-domain requests** using a victim's authenticated session.

The attacker can exploit this behavior to steal sensitive data, hijack sessions, or perform actions on behalf of a logged-in user.

### Security Risk Score

- **CVSS Version:** 3.0
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:R / S:C / C:H / I:H / A:N
- **CVSS Base Score:** 8.1 (High)
- **CVSS ID:** CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:N

### Consequence

1. **Remove crossdomain.xml** if Flash/Silverlight support is no longer needed.
2. **Avoid using wildcards (*)** in domain and to-ports attributes.
3. **Whitelist only trusted domains** explicitly.
4. Set secure="true" to enforce HTTPS-only communication.
5. Implement logging and validation for all cross-origin requests on the server

### Risk Reduction Measures

1. **Remove crossdomain.xml** if Flash/Silverlight support is no longer needed.
2. **Avoid using wildcards (*)** in domain and to-ports attributes.
3. **Whitelist only trusted domains** explicitly.

4. Set secure="true" to enforce HTTPS-only communication.
5. Implement logging and validation for all cross-origin requests on the server.

---

## Evidence Sources

- CWE-942: https://cwe.mitre.org/data/definitions/942.html
- CWE-264: https://cwe.mitre.org/data/definitions/264.html
- OWASP Flash Cross-Domain Policy Risk: https://owasp.org/www-community/attacks/Flash_Cross-Domain_Policy
- OWASP A05:2021 – Security Misconfiguration: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

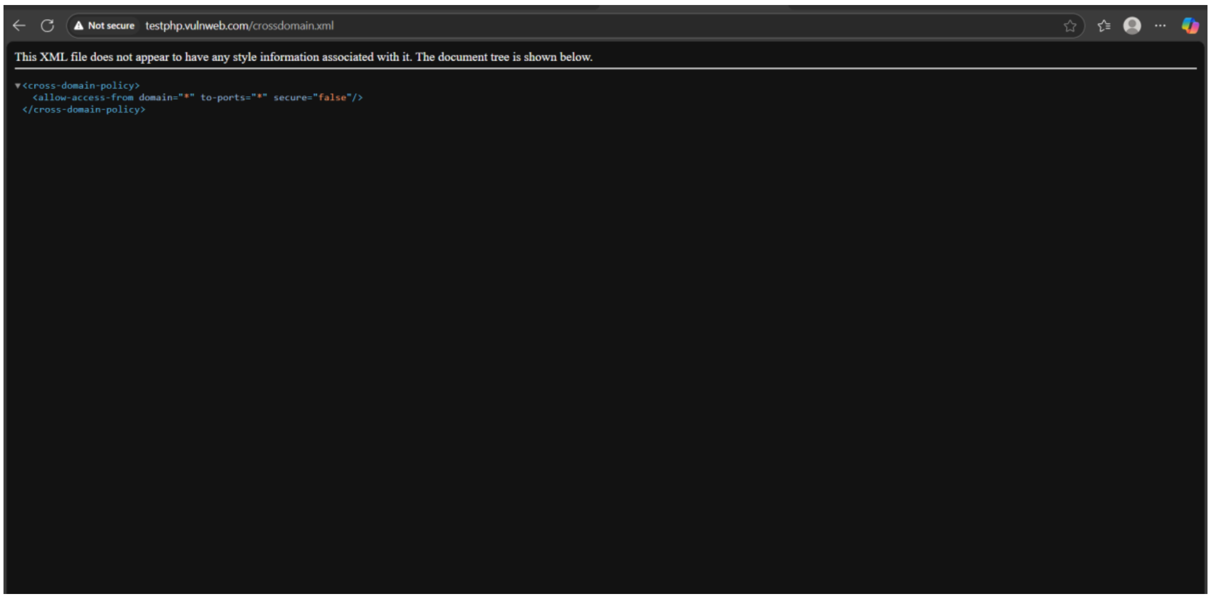
---

## Compromised Links::

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

---

## SAMPLE IMPLEMENTATION



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<cross-domain-policy>
   <allow-access-from domain="*" to-ports="*" secure="false"/>
 </cross-domain-policy>
```

## 7.9 Vulnerability Name : Combined Reflected and Stored XSS Vulnerabilities

---

### Overview

The application is vulnerable to both **Reflected** and **Stored Cross-Site Scripting (XSS)**, enabling attackers to inject and execute arbitrary JavaScript in users' browsers.

- **Reflected XSS** occurs in the cat parameter of listproducts.php, where user-supplied input is echoed into the response without sanitization.
  **Example:**
  http://testphp.vulnweb.com/listproducts.php?cat=<script>alert(1)</script>
- **Stored XSS** is found in the guestbook.php feature, where input submitted via comment fields is stored in the backend and executed whenever a user views the guestbook page.

Both forms of XSS can be used to steal session tokens, hijack accounts, or launch phishing attacks under the guise of a trusted site.

---

### Security Risk Score

- **CVSS Version:** 3.1
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:R / S:C / C:L / I:L / A:N
- **CVSS Base Score:** 8.0 (High)
- **CVSS ID:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

---

### Consequence

- Theft of session cookies or authentication tokens
- Browser-based exploitation (e.g., keylogging, redirecting)
- Account hijacking or UI defacement
- Phishing attacks using the trusted domain
- Execution of unauthorized actions in a user's context

---

## Risk Reduction Measures

1. Escape all user input using **HTML entity encoding** before rendering.
2. Use **secure templating engines** that perform auto-escaping.
3. Apply strict **server-side input validation** and **client-side sanitization**.
4. Implement a strong **Content Security Policy (CSP)** to limit script execution

---

## Evidence Sources

- CWE-79: https://cwe.mitre.org/data/definitions/79.html
- OWASP XSS Guide: https://owasp.org/www-community/attacks/xss/
- OWASP A03:2021 – Injection: https://owasp.org/Top10/A03_2021-Injection/

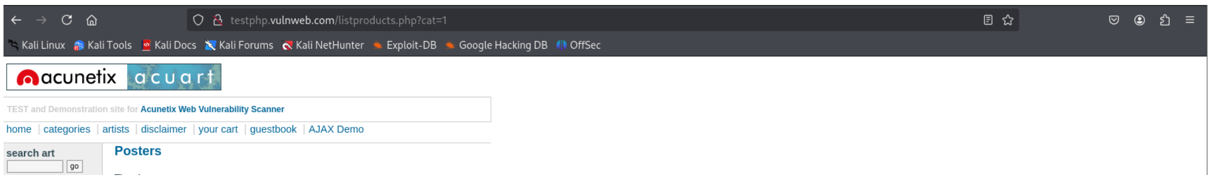
---

## Compromised Links::

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

---

## SAMPLE IMPLEMENTATION


testphp.vulnweb.com/listproducts.php?cat=1

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

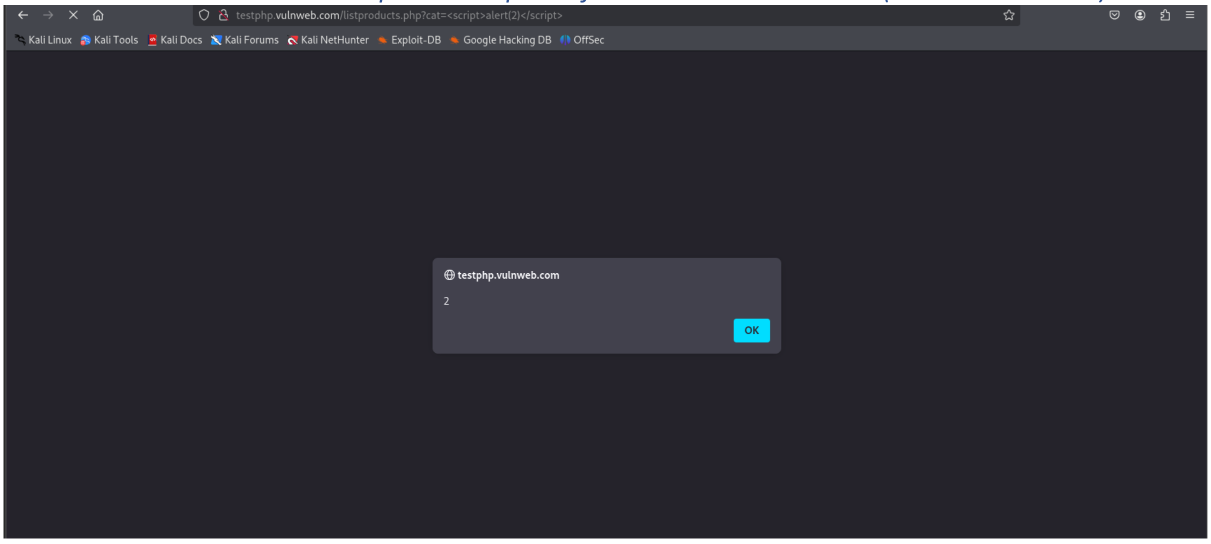
acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

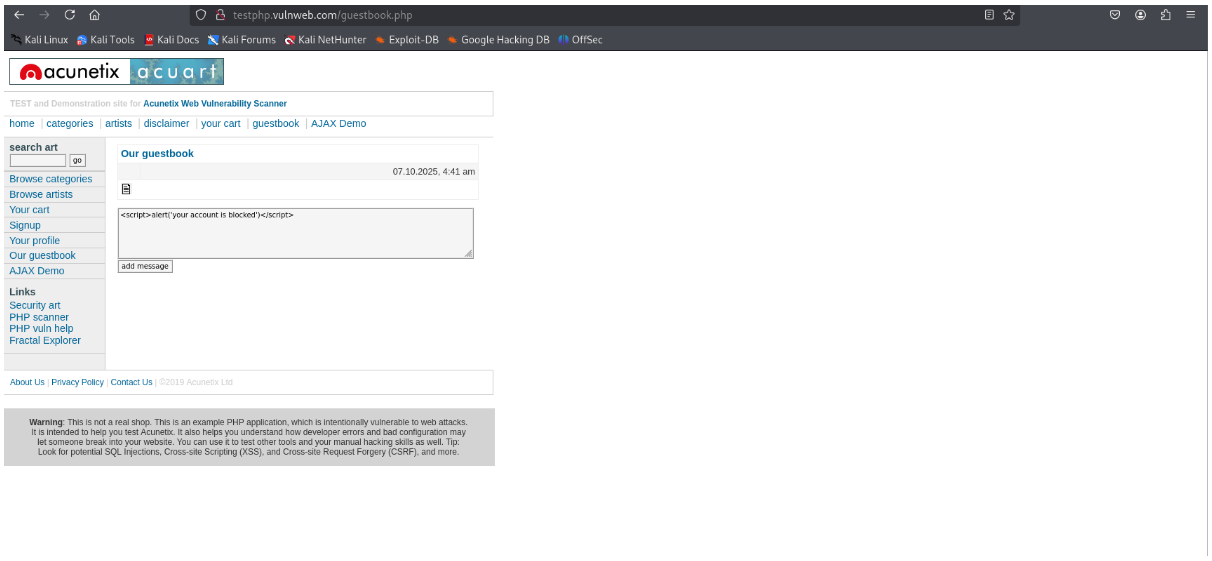
search art go

Posters


testphp.vulnweb.com/listproducts.php?cat=<script>alert(2)</script>

testphp.vulnweb.com

2


testphp.vulnweb.com/guestbook.php

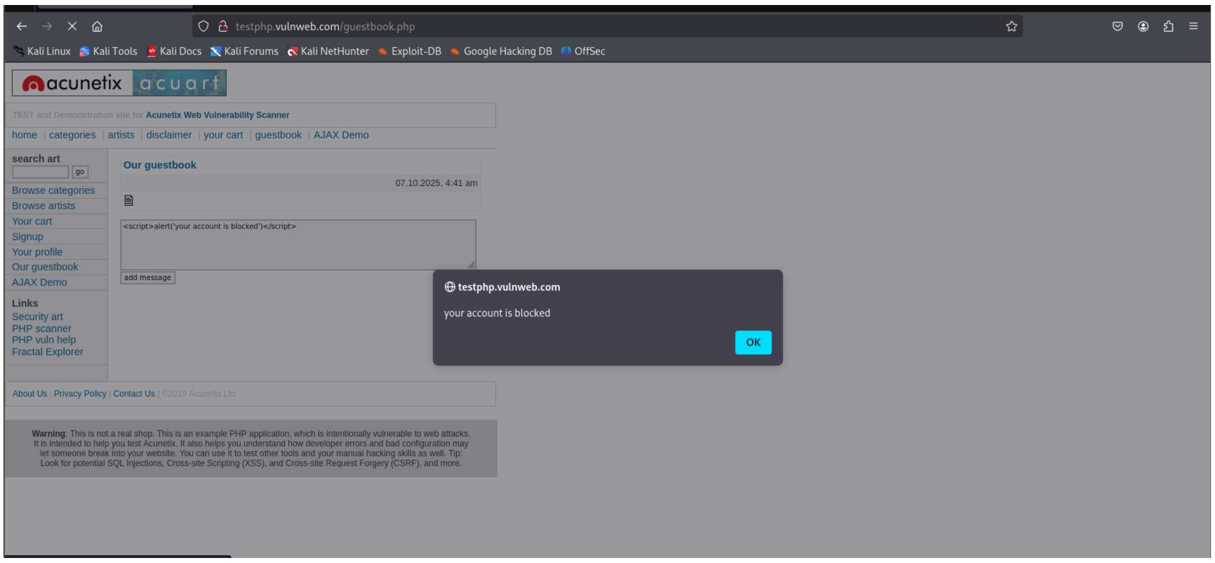
TEST and Demonstration site for Acunetix Web Vulnerability Scanner

Our guestbook

07.10.2025, 4:41 am

```
<script>alert('your account is blocked')</script>
```

**Warning**: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.


testphp.vulnweb.com/guestbook.php

testphp.vulnweb.com

your account is blocked

## 7.10 Vulnerability Name : Improper Server Validation of Cart Parameters

### Overview

The web application fails to enforce server-side validation of cart and pricing details, making it vulnerable to **price manipulation attacks**. An attacker can intercept and modify cart-related parameters (e.g., price, quantity, product ID) using tools like **Burp Suite**, **browser dev tools**, or custom scripts.

For instance, modifying the price field in a POST or GET request to a lower value allows unauthorized reduction of total payable amount. This vulnerability is a **business logic flaw** that undermines transactional integrity and can result in **unauthorized purchases, fraud, and direct financial loss**.

### Security Risk Score

- CVSS Version: 3.1
- CVSS Base Vector: AV:N / AC:L / PR:N / UI:N / S:U / C:N / I:H / A:N
- CVSS Base Score: 8.2 (High)
- CVSS ID: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N

### Consequence

- Unauthorized price reduction (e.g., zero or negative pricing)
- Loss of revenue and potential legal issues
- Damaged trust and integrity of the platform
- Abuse of promotional logic or cashback scenarios

### Risk Reduction Measures

1. **Never Trust Client-Side Data**
   Always treat client-side price or quantity data as untrusted.

2. **Enforce Server-Side Price Validation**
   Fetch original pricing from a secure server-side database using trusted product IDs.
3. **Use Integrity Checks (e.g., HMAC or Tokenization)**
   Digitally sign cart values to detect tampering.
4. **Implement Business Logic Validation**
   Apply consistency and sanity checks to prevent unrealistic pricing or quantities.
5. **Log and Alert on Suspicious Cart Modifications**
   Detect and flag anomalies like frequent price changes or zero-cost transactions.

---

## Evidence Sources

- **CWE-302: Authentication Bypass by Assumed-Immutable Data**
- **CWE-451: UI Misrepresentation of Critical Information**
- **OWASP A04:2021 – Insecure Design**

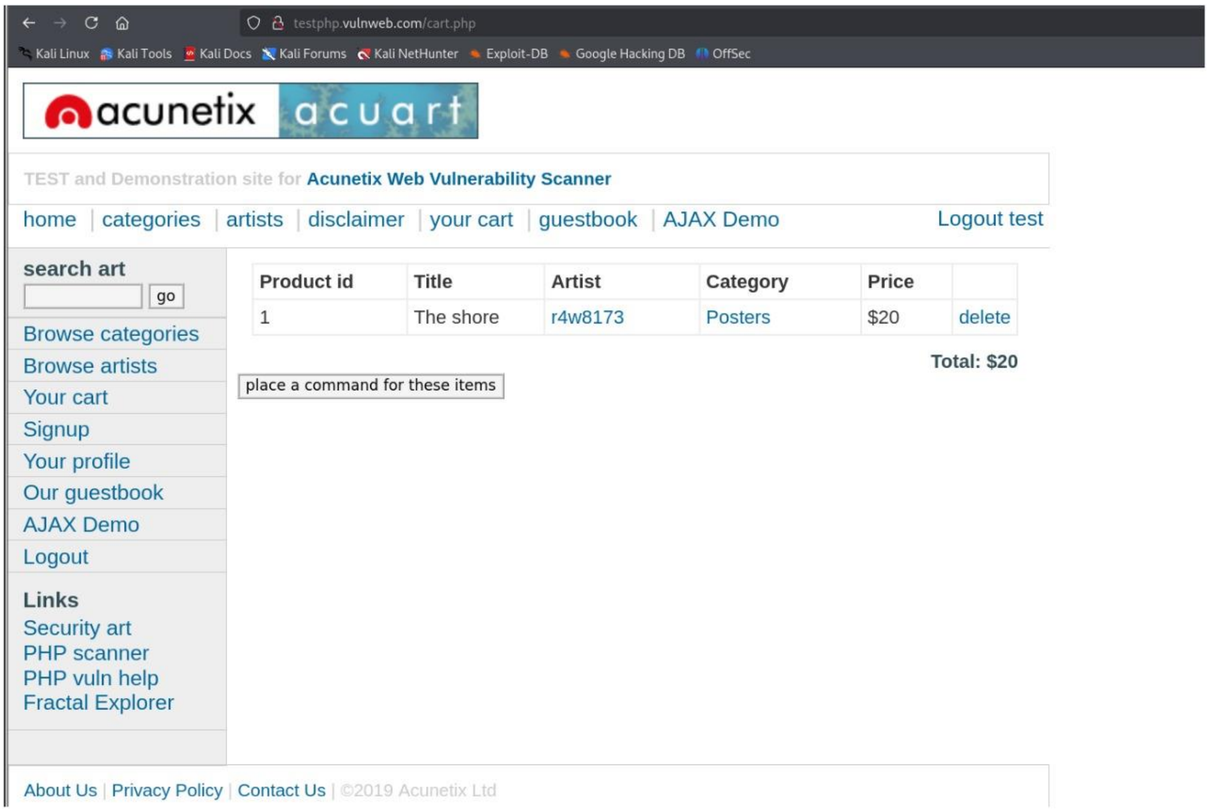
- **A**

## Compromised Links::

- http://testphp.vulnweb.com/cart.phpw

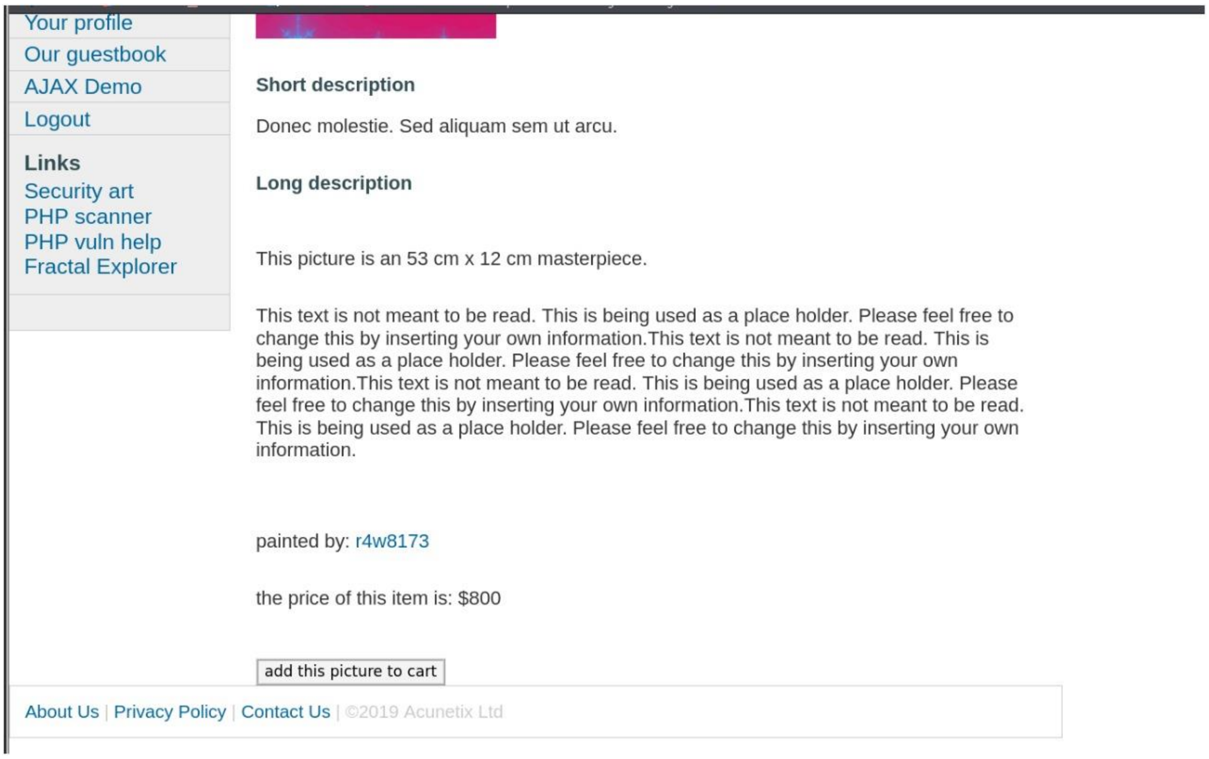
---

## SAMPLE IMPLEMENTATION


testphp.vulnweb.com/cart.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
place a command for these items
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

| Product id | Title | Artist | Category | Price | |
|---|---|---|---|---|---|
| 1 | The shore | r4w8173 | Posters | $20 | delete |

Total: $20



**Short description**

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: $800

## 7.11 Vulnerability Name : Insecure HTTP Method Exposure via PUT on /artists.php

### Overview

The application exposes the HTTP PUT method on the /artists.php endpoint, even though the endpoint is not designed to handle such request types. Accepting unsupported or unused HTTP methods can lead to unintended behaviors and expand the application's attack surface.

While this specific exposure does not currently enable file upload or remote code execution, it **violates the principle of least privilege** in HTTP method handling and can be abused in method override attacks, parameter smuggling, or chained with injection vulnerabilities.

### Security Risk Score

- CVSS Version: 3.1
- CVSS Base Vector: AV:N / AC:L / PR:N / UI:N / S:C / C:L / I:H / A:L
- CVSS Base Score: 8.2 (High)
- CVSS ID: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:H/A:L

### Consequence

- Unintended access to internal logic via PUT requests
- Potential chaining with injection flaws (e.g., SQLi, XSS)
- Parameter confusion or logic abuse from unsupported methods
- Increased risk during automated fuzzing or scanning
- Expanded attack surface and weakened input validation posture

### Risk Reduction Measures

1. **Restrict Unsupported HTTP Methods**
   Configure your web server (e.g., Apache, NGINX, IIS) to disallow unused methods like PUT, DELETE, TRACE, etc.

2. **Return Proper Response Codes**
   Ensure the server returns 405 Method Not Allowed for disallowed HTTP methods.

3. **Apply Input Validation on All Methods**
   Validate and sanitize all inputs regardless of request method or endpoint.

4. **Monitor for Unusual HTTP Methods**
   Use WAF rules and server logs to detect and alert on unexpected HTTP method usage.

5. **Follow Principle of Least Exposure**
   Only expose application logic through necessary and secure endpoints/methods.

---

## Evidence Sources

- CWE-749: Exposed Dangerous Method or Function
- CWE-668: Exposure of Resource to Wrong Sphere
- OWASP A05:2021 – Security Misconfiguration
- OWASP HTTP Methods Best Practices

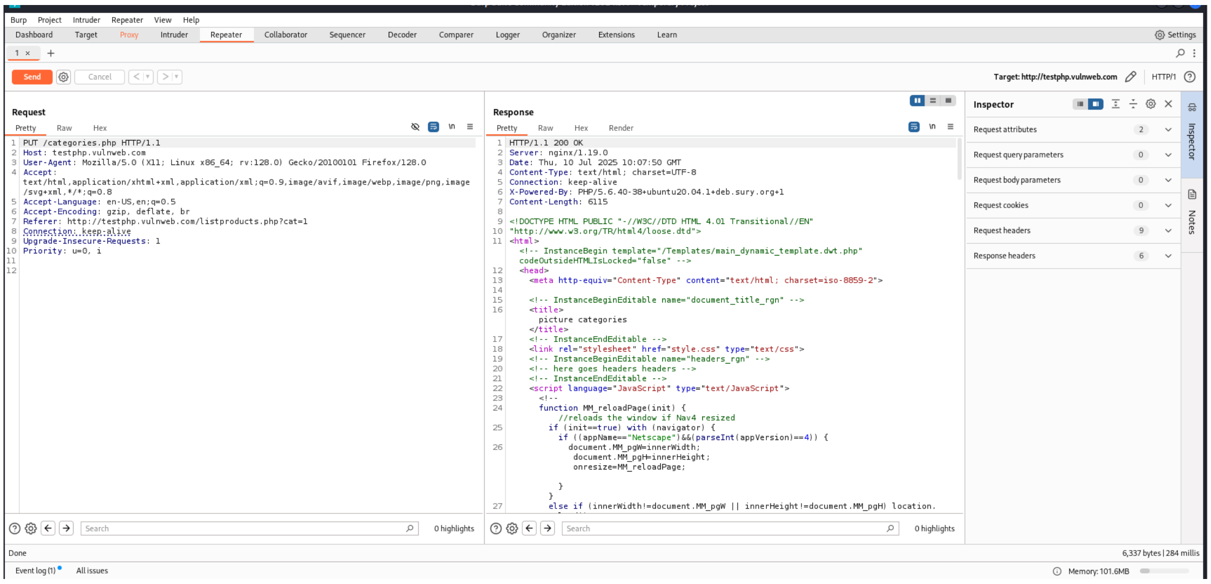
---

## Compromised Links::

http://testphp.vulnweb.com/artists.php

---

## SAMPLE IMPLEMENTATION



```
PUT /categories.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image
/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://testphp.vulnweb.com/listproducts.php?cat=1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Thu, 10 Jul 2025 10:07:50 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 6115

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <!-- InstanceBegin template="/Templates/main_dynamic_template.dwt.php"
  codeOutsideHTMLIsLocked="false" -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

    <!-- InstanceBeginEditable name="document_title_rgn" -->
    <title>
      picture categories
    </title>
    <!-- InstanceEndEditable -->
    <link rel="stylesheet" href="style.css" type="text/css">
    <!-- InstanceBeginEditable name="headers_rgn" -->
    <!-- here goes headers headers -->
    <!-- InstanceEndEditable -->
    <script language="JavaScript" type="text/JavaScript">
      <!--
      function MM_reloadPage(init) {
          //reloads the window if Nav4 resized
        if (init==true) with (navigator) {
          if ((appName=="Netscape")&&(parseInt(appVersion)==4)) {
            document.MM_pgW=innerWidth;
             document.MM_pgH=innerHeight;
             onresize=MM_reloadPage;

          }
        }
        else if (innerWidth!=document.MM_pgW || innerHeight!=document.MM_pgH) location.
```

## 7.12 Vulnerability Name : Session ID Exposed in URL (Session Management Misconfiguration)

### Overview

The application transmits the session identifier (e.g., PHPSESSID) via the URL query string, such as

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

This practice is insecure and exposes the session ID to multiple vectors including:

- Browser history
- HTTP referer headers (sent to third-party domains)
- Web server logs and analytics tools

Manual testing confirmed that by modifying the PHPSESSID parameter in the URL, it was possible to hijack another user's session — indicating a **Session Fixation** or **Session Prediction** vulnerability. This behavior compromises session confidentiality and application integrity.

### Security Risk Score

- **CVSS Version:** 3.1
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:N / S:U / C:H / I:H / A:N
- **CVSS Base Score:** 8.1 (High)
- **CVSS ID:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

### Consequence

- Exposure of session IDs via URLs
- Possibility of session hijacking through shared or logged URLs
- Risk of unauthorized access or privilege escalation
- Compromised user confidentiality and application trust

- Enables session prediction or fixation attacks in certain scenarios

---

## Risk Reduction Measures

1. **Do Not Pass Session IDs in URLs**
   Avoid using session IDs in query parameters. Use secure, cookie-based session handling instead.
2. **Use Secure Cookie Attributes**
   Set the following on session cookies:
   - HttpOnly to prevent access via JavaScript
   - Secure to enforce HTTPS-only transmission
   - SameSite=Strict or Lax to reduce CSRF risk
3. **Enforce HTTPS Site-wide**
   Use TLS to encrypt all session data during transmission.
4. **Regenerate Session IDs**
   On login or privilege change, regenerate session IDs to prevent fixation.
5. **Implement Short Session Timeouts**
   Invalidate idle sessions after a defined timeout period.

---

## Evidence Sources

- CWE-598: Information Exposure Through Query Strings in GET Request
- OWASP Session Management Cheat Sheet
- OWASP A07:2021 – Identification and Authentication Failures

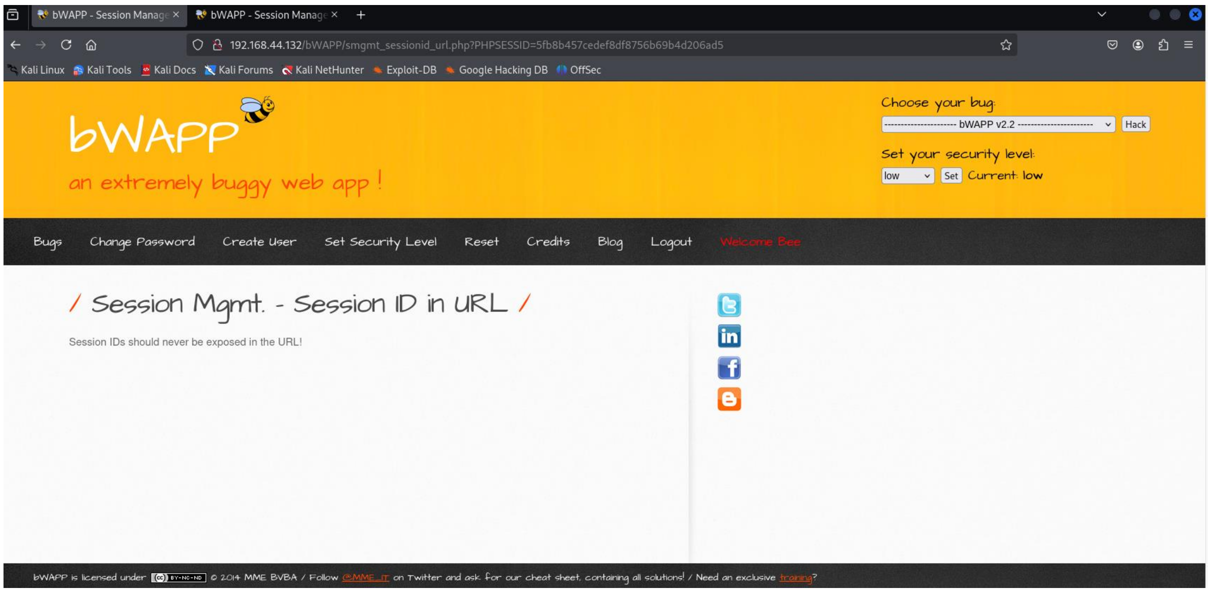
---

## Compromised Links::

- http://192.168.204.137/bWAPP/smgmt_sessionid_url.php?PHPSESSID=0781bbee1c5272cecdd952168c85604

---

## SAMPLE IMPLEMENTATION



```
192.168.44.132/bWAPP/smgmt_sessionid_url.php?PHPSESSID=5fb8b457cedef8df8756b69b4d206ad5
```

bWAPP – an extremely buggy web app !

Choose your bug: bWAPP v2.2 — Hack

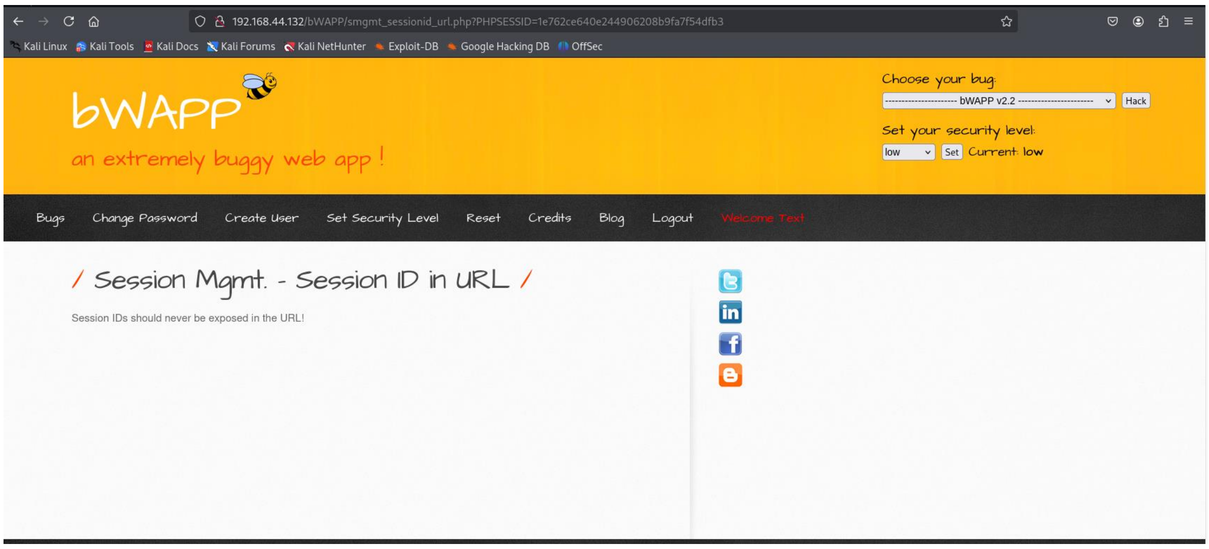
Set your security level: low — Set — Current: low

Bugs | Change Password | Create User | Set Security Level | Reset | Credits | Blog | Logout | Welcome Bee

/ Session Mgmt. - Session ID in URL /

Session IDs should never be exposed in the URL!

bWAPP is licensed under (cc) BY-NC-ND © 2014 MME BVBA / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need an exclusive training?



```
192.168.44.132/bWAPP/smgmt_sessionid_url.php?PHPSESSID=1e762ce640e244906208b9fa7f54dfb3
```

bWAPP – an extremely buggy web app !

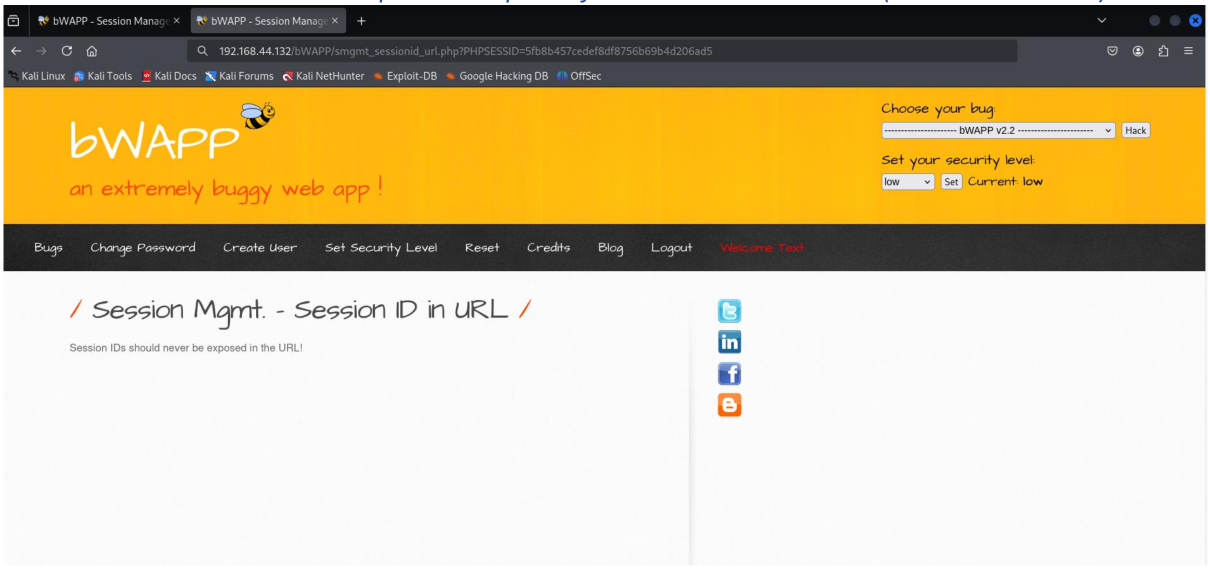
Choose your bug: bWAPP v2.2 — Hack

Set your security level: low — Set — Current: low

Bugs | Change Password | Create User | Set Security Level | Reset | Credits | Blog | Logout | Welcome Test

/ Session Mgmt. - Session ID in URL /

Session IDs should never be exposed in the URL!



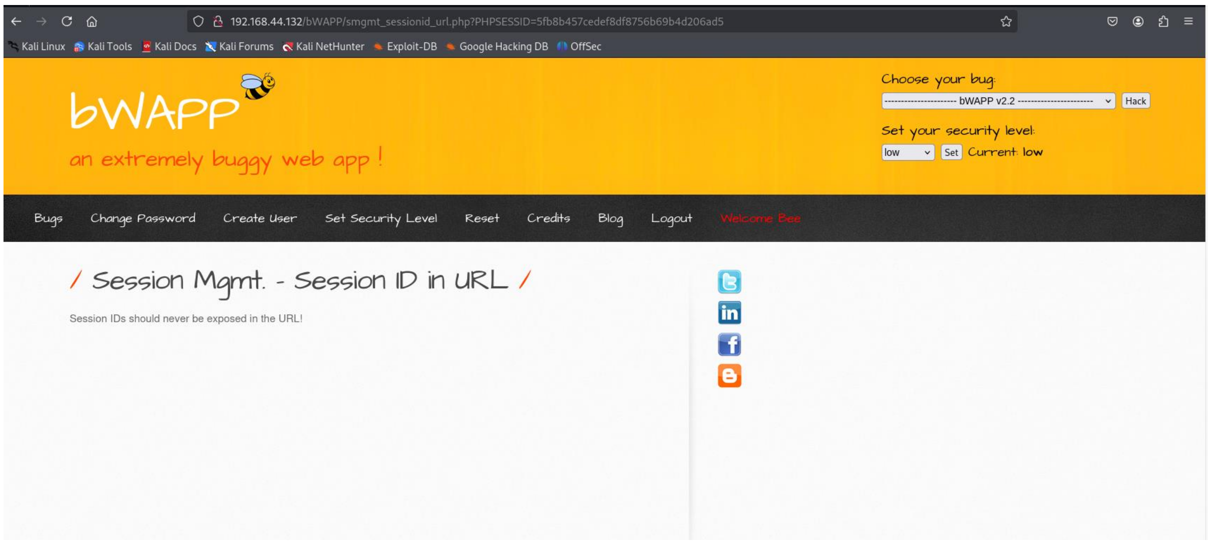
192.168.44.132/bWAPP/smgmt_sessionid_url.php?PHPSESSID=5fb8b457cedef8df8756b69b4d206ad5

Choose your bug: bWAPP v2.2 — Hack

Set your security level: low — Set — Current: low

/ Session Mgmt. - Session ID in URL /

Session IDs should never be exposed in the URL!



192.168.44.132/bWAPP/smgmt_sessionid_url.php?PHPSESSID=5fb8b457cedef8df8756b69b4d206ad5

Choose your bug: bWAPP v2.2 — Hack

Set your security level: low — Set — Current: low

/ Session Mgmt. - Session ID in URL /

Session IDs should never be exposed in the URL!

## 7.13 Vulnerability Name : Insecure Session Handling During Logout

### Overview

The application does not properly terminate user sessions on logout. After a user clicks the "Logout" button or link, the session token (e.g., PHPSESSID) remains valid. This token can still be reused to access the application without re-authentication, indicating a broken logout mechanism where the session is not invalidated on the server side.

An attacker who obtains the session token (via XSS, MITM, or shoulder surfing) can impersonate the user even after logout, unless the browser is completely closed or the token is manually deleted

### Security Risk Score

• CVSS Version: 3.1
• CVSS Base Vector: AV:N / AC:L / PR:L / UI:N / S:U / C:H / I:H / A:N
• CVSS Base Score: 8.8 (High)
• CVSS ID: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N

### Consequence

- **Session Hijacking Risk:** Attackers with access to a valid session token can bypass authentication.
- **Persistent Unauthorized Access:** Session remains active even after logout, allowing prolonged exploitation.
- **False User Assurance:** Users believe they've logged out securely while their session is still valid.
- **MFA Bypass Risk:** If multi-factor authentication is only enforced during login, attackers can avoid re-verification.
- **Risk on Shared Devices:** Insecure logout can allow the next user on a shared device to reuse the active session.

### Risk Reduction Measures

1. **Invalidate Session Server-Side:** Use session_destroy() or equivalent to terminate sessions upon logout.
2. **Regenerate Session Tokens:** On both login and logout to prevent session fixation.
3. **Set Expiration Timers:** Use short session lifetimes and idle timeouts.
4. **Clear Cookies:** Explicitly expire session cookies using HTTP headers.
5. **User Feedback:** Provide proper redirection and confirmation after logout.

---

## Evidence Sources

- CWE-613: Insufficient Session Expiration
- OWASP Session Management Cheat Sheet

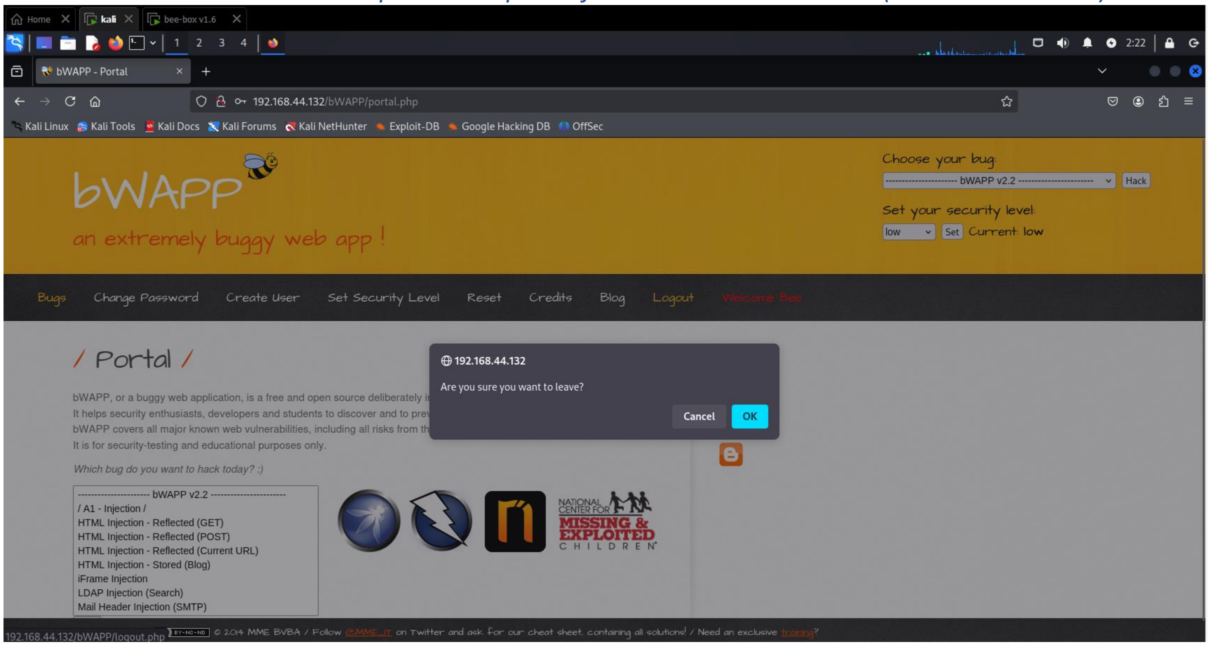
---

## Compromised Links::

- http://192.168.204.137/bWAPP/ba_logout.php

---

## SAMPLE IMPLEMENTATION


bWAPP - Portal
192.168.44.132/bWAPP/portal.php
bWAPP
an extremely buggy web app !
Choose your bug:
Set your security level:
Current: low
Bugs Change Password Create User Set Security Level Reset Credits Blog Logout Welcome Bee
/ Portal /
Cancel
OK

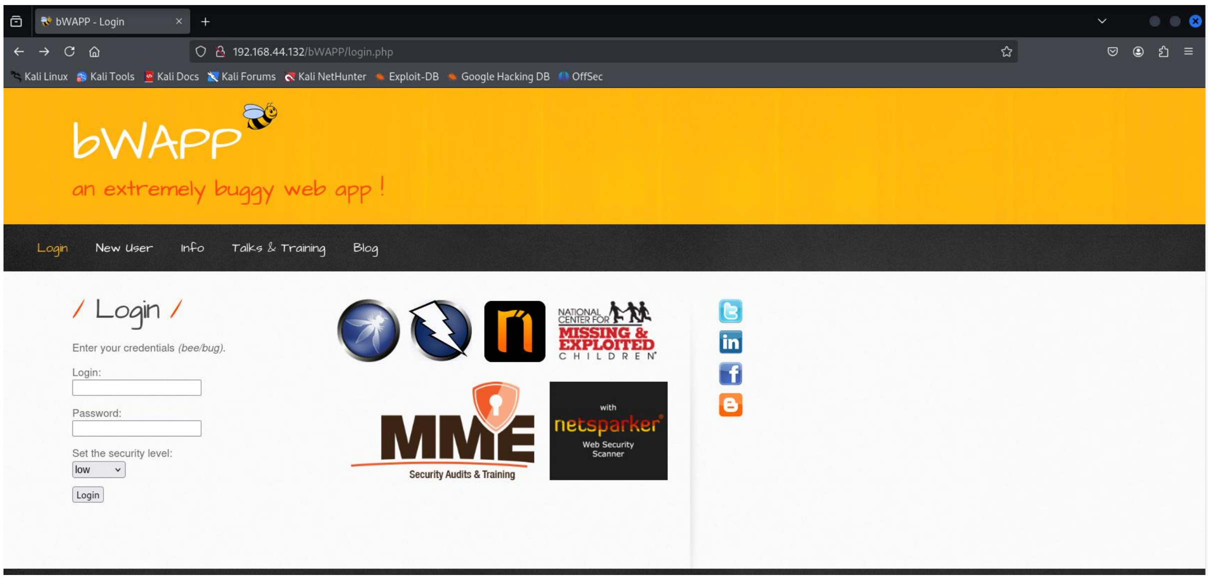
192.168.44.132

Are you sure you want to leave?

bWAPP, or a buggy web application, is a free and open source deliberately i... It helps security enthusiasts, developers and students to discover and to pre... bWAPP covers all major known web vulnerabilities, including all risks from th... It is for security-testing and educational purposes only.

Which bug do you want to hack today? :)

```
--------------------- bWAPP v2.2 ---------------------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
```


bWAPP - Login
192.168.44.132/bWAPP/login.php
bWAPP
an extremely buggy web app !
Login New User Info Talks & Training Blog
/ Login /
Login
MME Security Audits & Training
with netsparker Web Security Scanner

Enter your credentials (bee/bug).

Login:

Password:

Set the security level: low

## 7.14 Vulnerability Name : Cross-Site Request Forgery (CSRF)

### Overview

Cross-Site Request Forgery (CSRF) is a security vulnerability that allows an attacker to trick a victim into submitting malicious requests on their behalf — without the victim's knowledge or consent. This exploits the user's authenticated session with a trusted web application, effectively bypassing the same-origin policy.

### Security Risk Score

- **CVSS Version:** 3.1
- **CVSS Base Vector:** AV:N / AC:L / PR:L / UI:R / S:U / C:H / I:H / A:N
- **CVSS Base Score:** 8.8 (High)
- **CVSS ID:** CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:N

### Consequence

A successful CSRF attack can lead to unauthorized actions such as:

- Changing user account details (e.g., email, password)
- Performing transactions (e.g., funds transfer)
- Escalating privileges or taking control of other accounts, especially if the victim has administrative access

If the victim is a privileged user (e.g., admin), the attacker may gain full access to the application's data and functionality.

### Risk Reduction Measures

- Use **anti-CSRF tokens** in all state-changing operations (e.g., synchronizer token pattern)
- Validate the **Referer** and **Origin** headers for sensitive requests
- Set **SameSite** attribute on cookies (SameSite=Lax or Strict)

- Avoid using **GET** requests for state-changing operations (use POST instead)

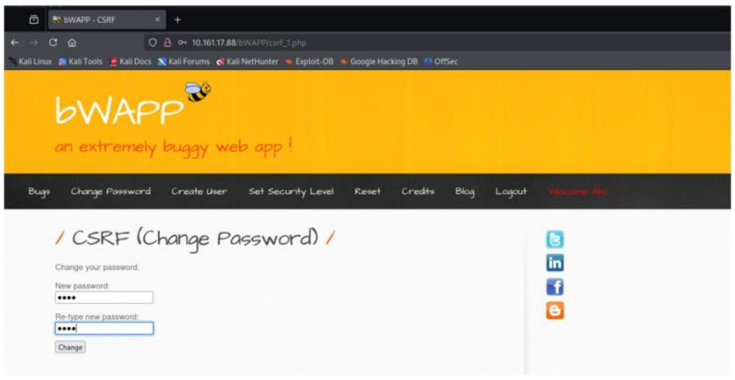
## Evidence Sources

- OWASP – Cross-Site Request Forgery (CSRF)
  Comprehensive overview of CSRF attacks, Consequence, and prevention methods.
- OWASP Top 10 – A01:2021 Broken Access Control
  Highlights how CSRF is linked to broken access controls in insecure web apps.

## Compromised Links::

- http://testphp.vulnweb.com/

## SAMPLE IMPLEMENTATION:

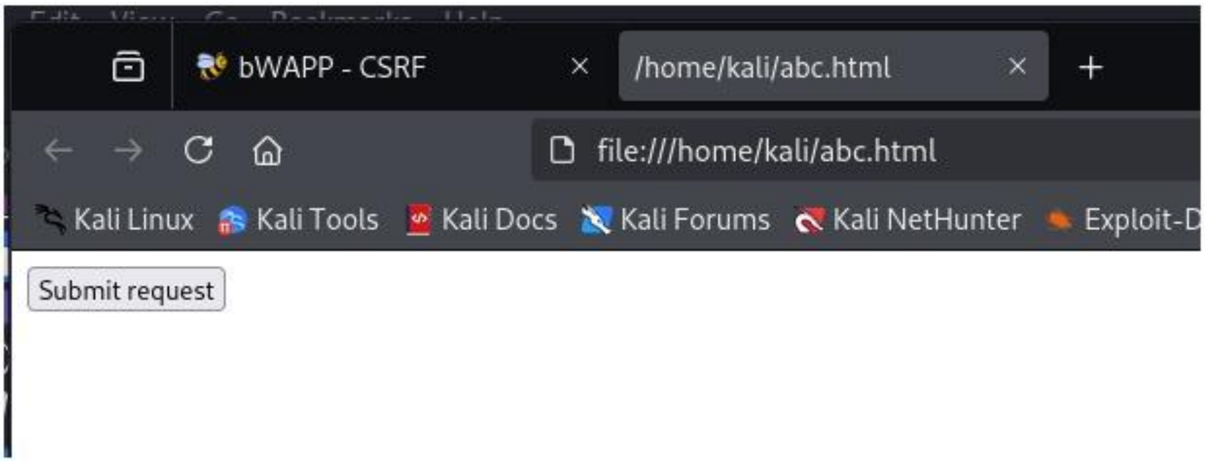


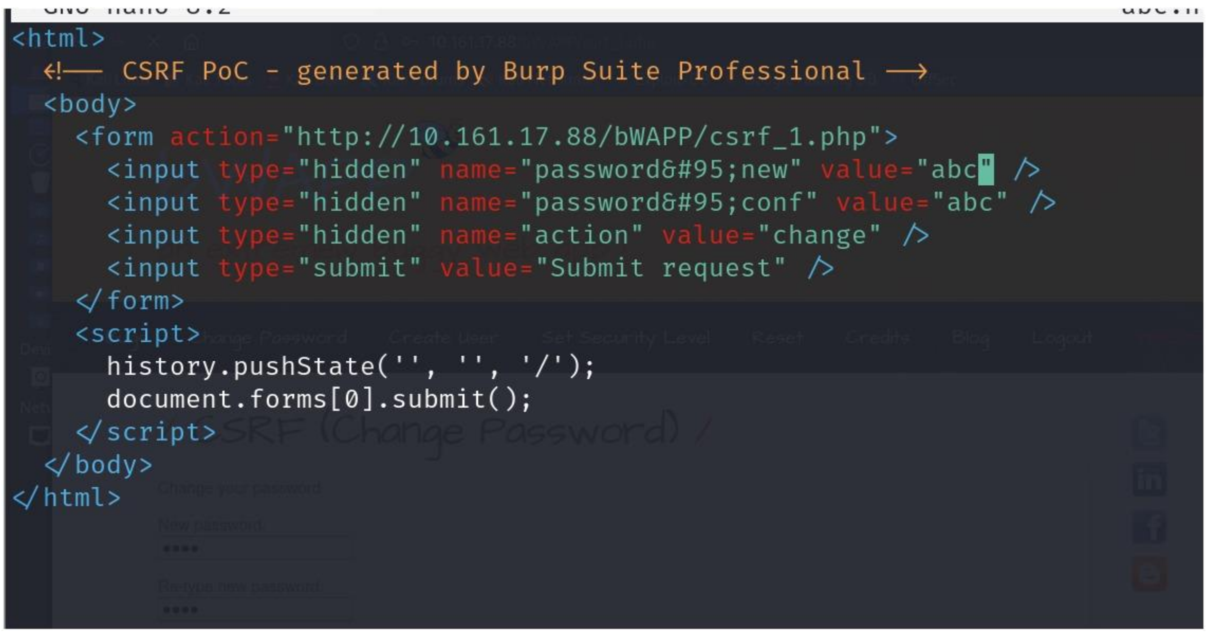
/ CSRF (Change Password) /

Change your password.

New password: ••••

Re-type new password: ••••





```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
    <form action="http://10.161.17.88/bWAPP/csrf_1.php">
      <input type="hidden" name="password&#95;new" value="abc" />
      <input type="hidden" name="password&#95;conf" value="abc" />
      <input type="hidden" name="action" value="change" />
      <input type="submit" value="Submit request" />
    </form>
    <script>
      history.pushState('', '', '/');
      document.forms[0].submit();
    </script>
  </body>
</html>
```

## 7.15 Vulnerability Name : Server-Side Request Forgery (SSRF) via file Parameter

---

### Overview

**Server-Side Request Forgery (SSRF)** is a vulnerability that allows an attacker to manipulate a server into making unintended HTTP requests. These requests can target:

- **Internal resources** (e.g., localhost, 127.0.0.1, internal APIs)
- **External systems** (potentially for pivoting or data exfiltration)

The attacker abuses server-side functionality to send crafted requests, potentially exposing sensitive data (e.g., metadata, cloud keys, internal APIs) or causing damage to internal infrastructure.

---

### Security Risk Score

- **CVSS Version:** 3.1
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:N / S:C / C:H / I:H / A:N
- **CVSS Base Score:** 8.6 (High)
- **CVSS ID:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N

---

### Consequence

A successful SSRF attack can lead to:

- Unauthorized access to **internal systems and services**
- Disclosure of **sensitive information** (e.g., AWS metadata, internal admin panels)
- **Command execution**, depending on how the request is processed
- Abuse of the server to launch **further attacks** on external or internal systems

---

### Risk Reduction Measures

- **Input Validation:** Rigorously validate and sanitize all user-supplied URLs or file paths
- **Allowlisting:** Only allow trusted, pre-defined URLs or internal endpoints
- **Network Segmentation:** Restrict backend services from making outbound requests when unnecessary
- **Disable Risky Functions:** Disable features like allow_url_fopen (in PHP) or other remote include mechanisms
- **DNS Pinning & IP Filtering:** Prevent DNS rebinding attacks and restrict IP ranges (e.g., block internal IPs like 169.254.*.*, 127.0.0.1)

---

## Evidence Sources

CWE-918: Server-Side Request Forgery (SSRF)

OWASP Top 10: A10 – Server-Side Request Forgery

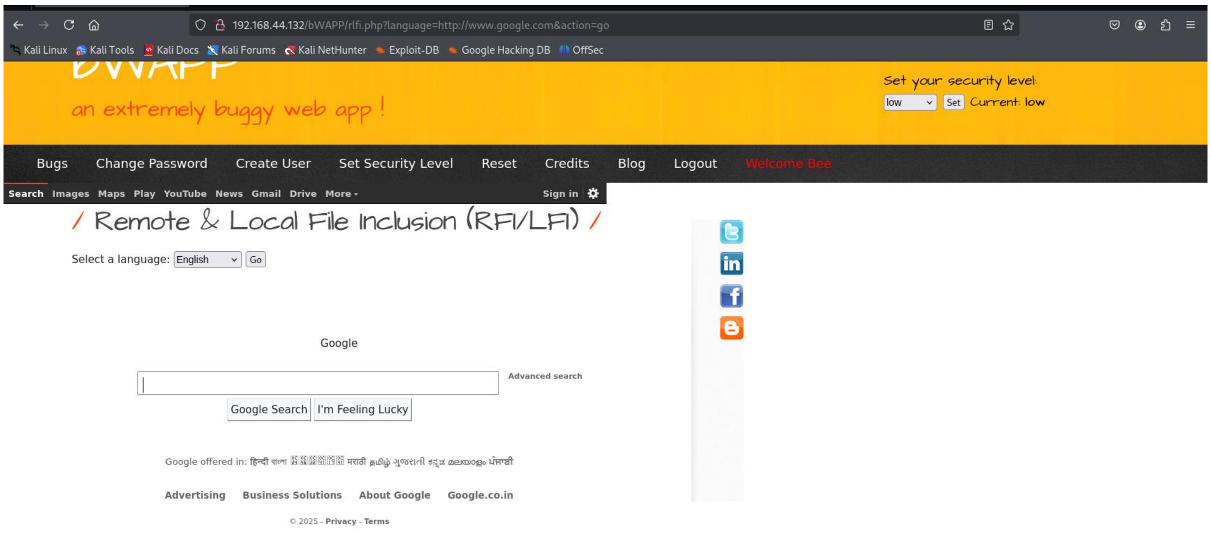
---

## Compromised Links::

http://192.168.44.132/bWAPP/rlfi.php?language=http://www.google.com&action=go

---

## SAMPLE IMPLEMENTATION



192.168.44.132/bWAPP/rlfi.php?language=http://www.google.com&action=go

MEDIUM

## 7.16 Vulnerability Name : Insecure Transmission of Data

---

### Overview

The web application transmits data over an **unencrypted HTTP** channel rather than a secure HTTPS connection. This exposes sensitive information — such as login credentials, session identifiers, and personal data — to potential interception by attackers monitoring the network.

Without **SSL/TLS encryption**, data in transit is vulnerable to **Man-in-the-Middle (MITM) attacks**, session hijacking, and unauthorized tampering or disclosure.

---

### Security Risk Score

- **CVSS Version**: 3.0
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:N / S:C / C:L / I:L / A:N
- **CVSS Base Score**: **6.1 (Medium)**
- **CVSS ID**: **CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:L/I:L/A:N**

---

### Consequence

If an attacker intercepts traffic on the network (e.g., via a public Wi-Fi), they can:

- Steal user credentials and session cookies
- Gain unauthorized access to user accounts
- Manipulate or alter transmitted data
- Compromise application integrity and user trust

---

### Risk Reduction Measures

1. **Implement SSL/TLS**: Obtain and install a valid TLS certificate to enforce HTTPS on all endpoints.
2. **Enable HSTS**: Configure HTTP Strict Transport Security to enforce HTTPS in browsers.
3. **Redirect HTTP to HTTPS**: Ensure all HTTP requests are redirected to their HTTPS equivalents.

4. **Secure Cookies**:
   - Set the Secure flag to ensure cookies are only sent over HTTPS.
   - Set the HttpOnly flag to prevent client-side access to cookies.
5. **Use Strong TLS Configurations**: Enforce TLS 1.2 or TLS 1.3 with strong cipher suites.

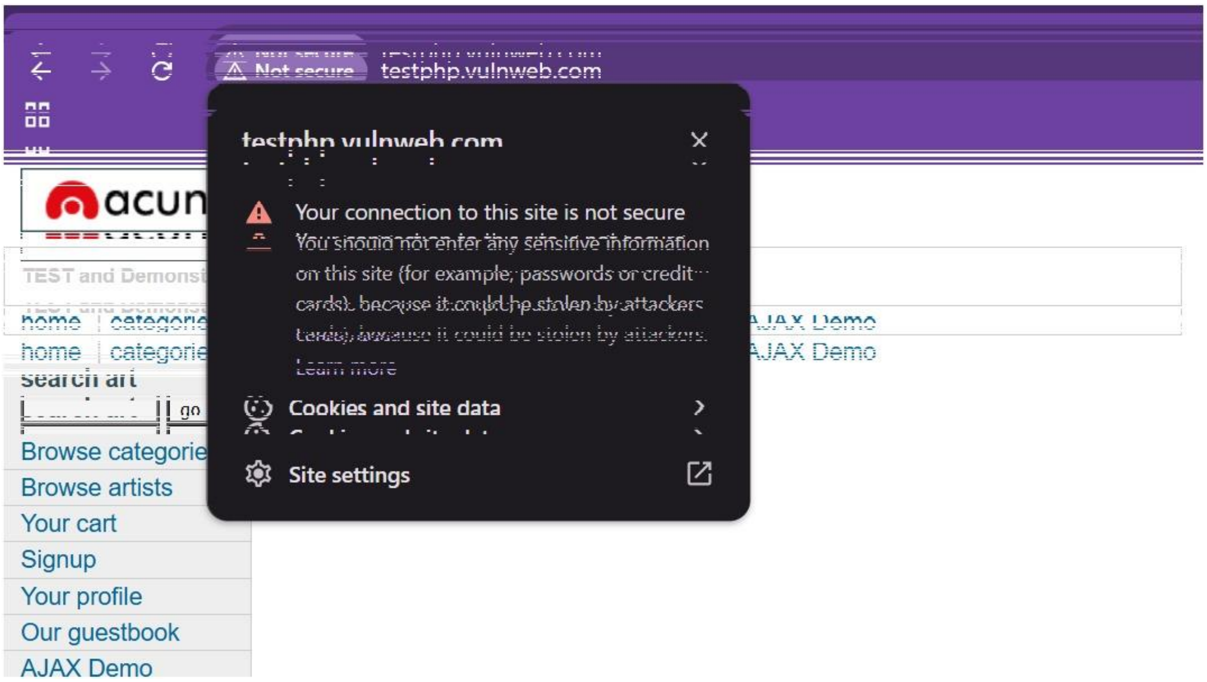
## Evidence Sources

- **CWE-319**: Cleartext Transmission of Sensitive Information
  https://cwe.mitre.org/data/definitions/319.html
- **OWASP Top 10 - A02:2021**: Cryptographic Failures
  OWASP Transport Layer Protection Cheat Sheet

## Compromised Links::

- http://testphp.vulnweb.com/

## SAMPLE IMPLEMENTATION


testphp.vulnweb.com

Your connection to this site is not secure

You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers.

Learn more

## 7.17 Vulnerability Name : Cross-Site Tracing (XST) Vulnerability

### Overview

The application accepts and responds to the **TRACE** HTTP method at the endpoint:

http://192.168.204.137/bWAPP/sm_xst.php

When a TRACE request is issued, the server reflects the entire request, including headers such as **cookies and authentication tokens**, back to the client. This behavior enables **Cross-Site Tracing (XST)** attacks in certain scenarios.

XST is a security issue where attackers can exploit the TRACE method in conjunction with **reflected XSS** (in outdated or misconfigured browsers or proxies) to access session cookies — even when the HttpOnly flag is set.

### Security Risk Score

- **CVSS Version**: 3.1
- **CVSS Base Vector**: AV:N / AC:L / PR:N / UI:R / S:C / C:L / I:L / A:N
- **CVSS Base Score**: **6.1 (Medium)**
- **CVSS ID**: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

### Consequence

- **Cookie Theft**: When combined with a reflected XSS vulnerability, cookies may be accessed despite HttpOnly protection.
- **Session Hijacking**: Leads to unauthorized access to authenticated user sessions.
- **Header Leakage**: Can reveal internal headers via proxy chains.
- **Security Policy Violation**: Use of TRACE method violates secure server configuration best practices.
- **Environment Fingerprinting**: May expose implementation details useful in further attacks.

### Risk Reduction Measures

1. **Disable TRACE Method on the Web Server**:
   - **Apache**:
     Set TraceEnable off in httpd.conf
2. **Secure Cookies**:
   - Set HttpOnly, Secure, and SameSite=Lax or SameSite=Strict attributes.
3. **Prevent Reflected XSS**:
   - Validate and sanitize all user input.
   - Use Content Security Policy (CSP) headers.
4. **Harden Server Configuration**:
   - Disable unused HTTP methods globally.
   - Regularly audit web server configurations.

---

## Evidence Sources

- **CWE-724**: Improper Handling of HTTP TRACE Method
  https://cwe.mitre.org/data/definitions/724.html
- **OWASP**:
  - Cross-Site Tracing (XST) Attack
  - Testing for HTTP Methods (WSTG)

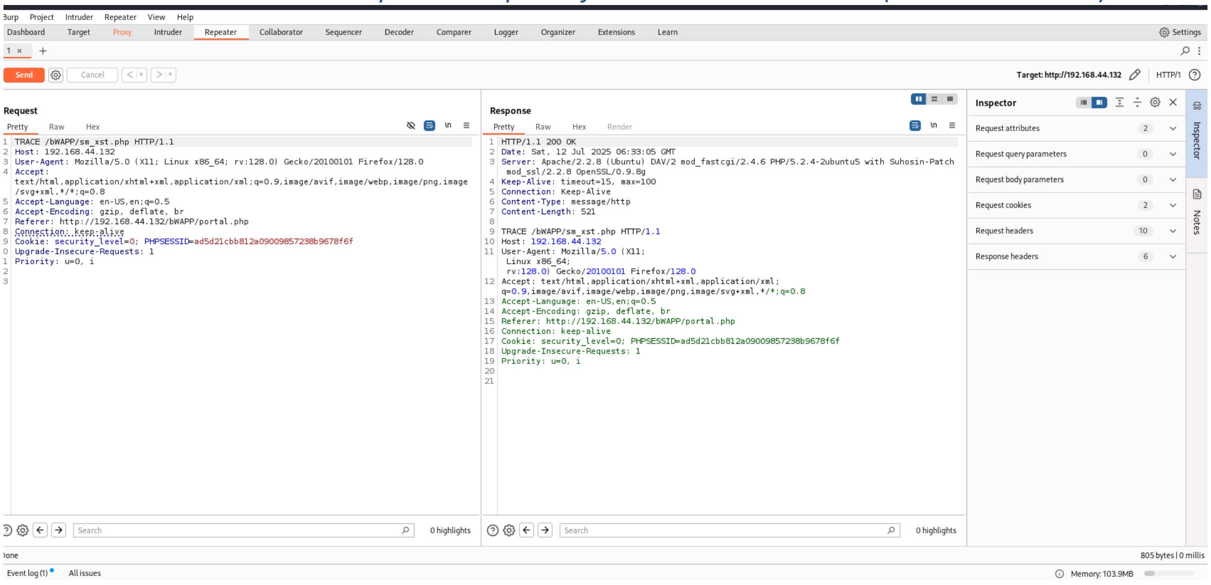
---

## Compromised Links::

http://192.168.44.132/bWAPP/sm_xst.php

---

## SAMPLE IMPLEMENTATION



Request

```
TRACE /bWAPP/sm_xst.php HTTP/1.1
Host: 192.168.44.132
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image
/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.44.132/bWAPP/portal.php
Connection: keep-alive
Cookie: security_level=0; PHPSESSID=ad5d21cbb812a09009857238b9678f6f
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response

```
HTTP/1.1 200 OK
Date: Sat, 12 Jul 2025 06:33:05 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch
 mod_ssl/2.2.8 OpenSSL/0.9.8g
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: message/http
Content-Length: 521

TRACE /bWAPP/sm_xst.php HTTP/1.1
Host: 192.168.44.132
User-Agent: Mozilla/5.0 (X11;
 Linux x86_64;
 rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;
q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://192.168.44.132/bWAPP/portal.php
Connection: keep-alive
Cookie: security_level=0; PHPSESSID=ad5d21cbb812a09009857238b9678f6f
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Target: http://192.168.44.132

LOW

## 7.18 Vulnerability Name: HTML Injection

### Overview

The application fails to properly sanitize HTML input submitted by the user. As a result, arbitrary HTML code is rendered in the browser. This allows attackers to manipulate the page layout and potentially trick users with deceptive content, such as fake forms or phishing links.

### Security Risk Score

- **CVSS Version:** 3.1
- **CVSS Base Vector:** AV:N / AC:L / PR:N / UI:R / S:C / C:N / I:L / A:N
- **CVSS Base Score:** 3.5 (Low)
- **CVSS ID:** CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:N/I:L/A:N

### Consequence

HTML Injection does **not** typically allow script execution (unlike XSS), but it can still be leveraged for:

- Injecting fake login forms
- Spoofing UI elements (e.g., buttons or messages)
- Social engineering attacks (e.g., phishing)
- Altering the visual structure or content of the page

### Risk Reduction Measures

1. **HTML Entity Encoding:**
   Encode characters like <, >, " using HTML entities (&lt;, &gt;, &quot;).
2. **Strict Input Validation:**
   Apply **whitelisting** techniques to only allow expected and safe inputs.
3. **Templating Engine Security:**
   Use frameworks with built-in HTML sanitization (e.g., React JSX, Django Templates).
4. **Content Security Policy (CSP):**
   Deploy CSP headers to restrict the execution and injection of HTML and scripts.

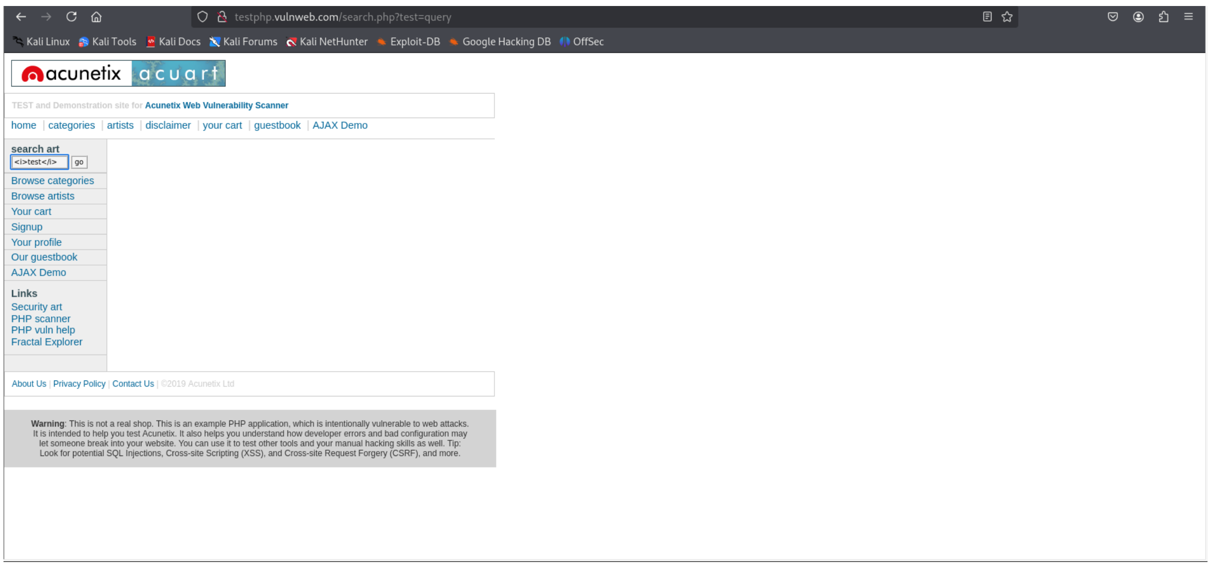
## Evidence Sources

- **OWASP:** HTML Injection
- **CWE-80:** Improper Neutralization of Script-Related HTML Tags

## Compromised Links::

http://testphp.vulnweb.com/search.php?test=query

## SAMPLE IMPLEMENTATION



testphp.vulnweb.com/search.php?test=query

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

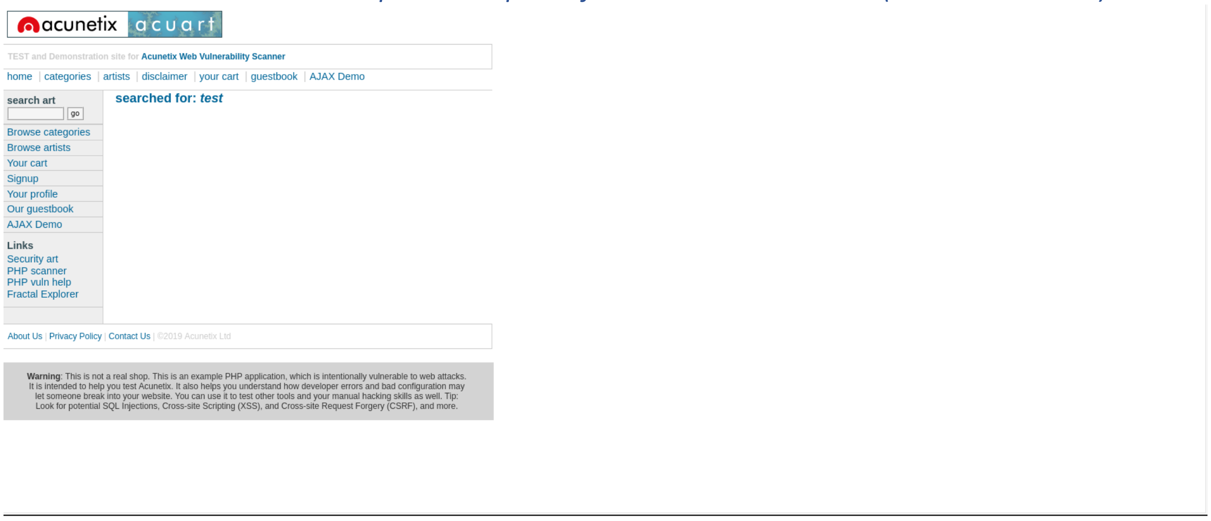
**search art**
<i>test</i> go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

**Links**
Security art
PHP scanner
PHP vuln help
Fractal Explorer

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. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.



acunetix acuart

TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

**search art**

go

Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo

**Links**
Security art
PHP scanner
PHP vuln help
Fractal Explorer

**searched for: *test***

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. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

# 8.Conclusion

**Overview:**
The security assessment uncovered 23 vulnerabilities across the target web applications, ranging from low to critical severity.

---

**Key Findings:**

- **Critical vulnerabilities include:**
    - SQL Injection
    - Remote Code Execution
    - Command Injection
    - Unrestricted File Upload
    - Broken Authentication
      These can allow unauthorized access, data breaches, or full server compromise**.**
- **Configuration Weaknesses such as:**
    - Outdated components
    - Exposed sensitive files
    - Unsafe HTTP methods
    - Lack of encryption during transmission
      These increase the attack surface and compromise user privacy and data integrity.

---

**Remediation & Recommendations:**

- Immediate Action Required:
  Prioritize Critical and High severity issues for urgent remediation.
- Development Best Practices:
    - Enforce input validation
    - Secure file handling
    - Strengthen session management
- Strategic Improvements:
    - Adopt defense-in-depth security models
    - Conduct regular security audits