ISO 27001 Compliant Incident Report – SQL Injection Vulnerability

---

Introduction

This report documents the identification and exploitation of an SQL injection vulnerability in the Damn Vulnerable Web Application (DVWA). The tests were conducted in a secure lab environment using a Debian virtual machine.

Incident Description

During the security assessment of DVWA, the "SQL Injection" module was tested under different security levels: Low, Medium, High, and Impossible. It was discovered that at the Low security level, the application was vulnerable to classic SQL injection attacks, allowing unauthorized data retrieval by injecting SQL code into input fields.

Reproduction Process

Environment Setup

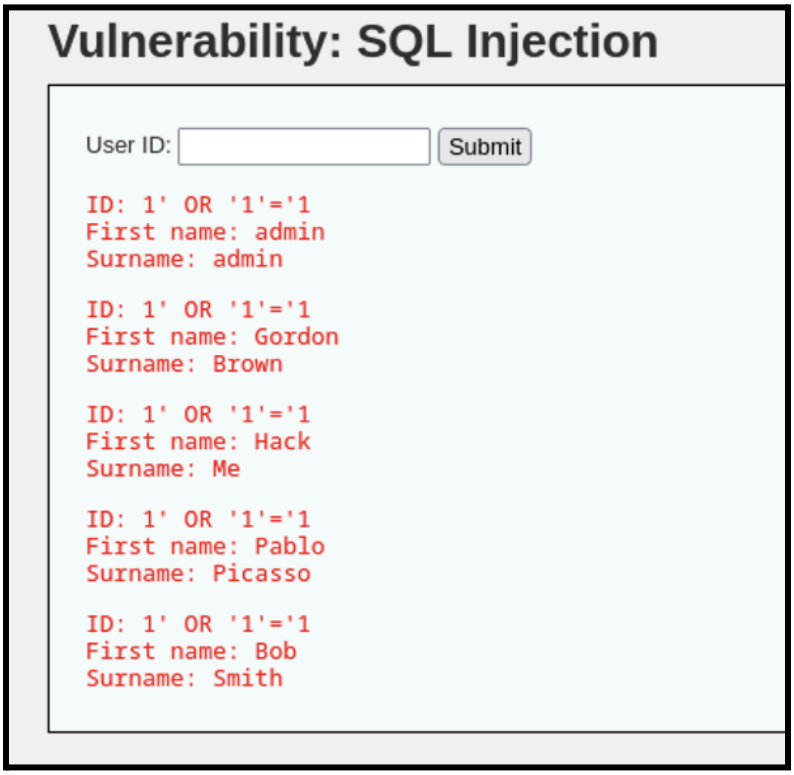
- DVWA installed on Debian VM (LAMP stack configured)
- MariaDB database created with a dedicated user for DVWA

Steps to Reproduce

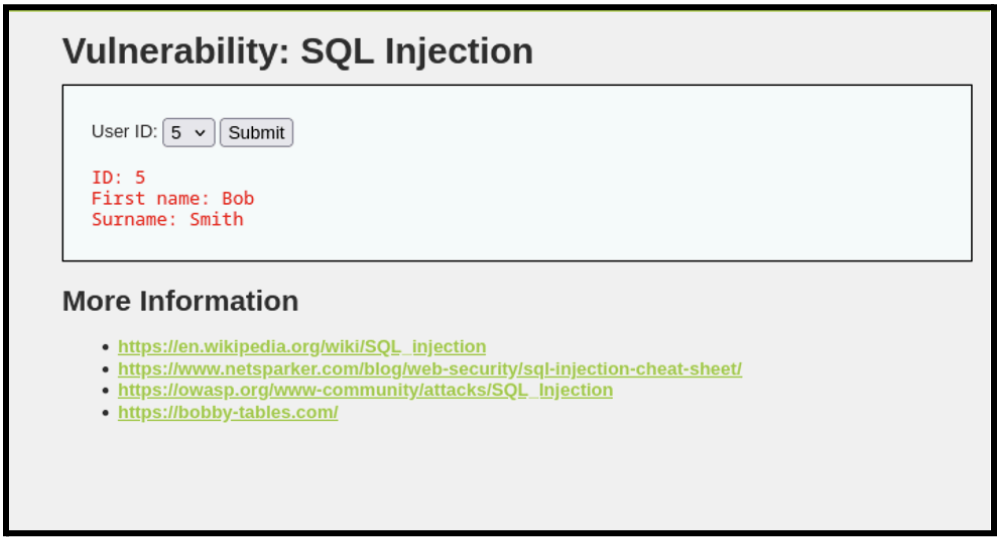
1. Navigate to DVWA at: `http://localhost/DVWA`
2. Login with credentials:
    - Username: `admin`
    - Password: `password`
3. Set the security level to **Low** from the "DVWA Security" tab.
4. Go to the "SQL Injection" module.
5. In the "User ID" input field, enter the SQL injection payload:
   `1' OR '1'='1`
6. Submit the form and observe the database response.

Observed Behavior

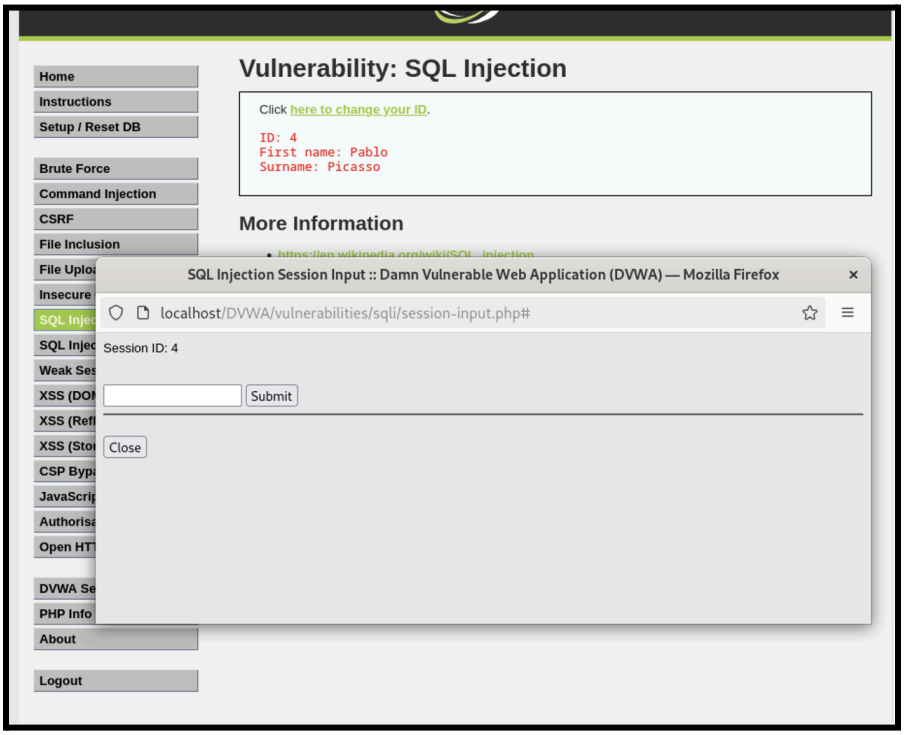
- At the Low security level, the injection returns **all user records** from the database instead of just one. Output:



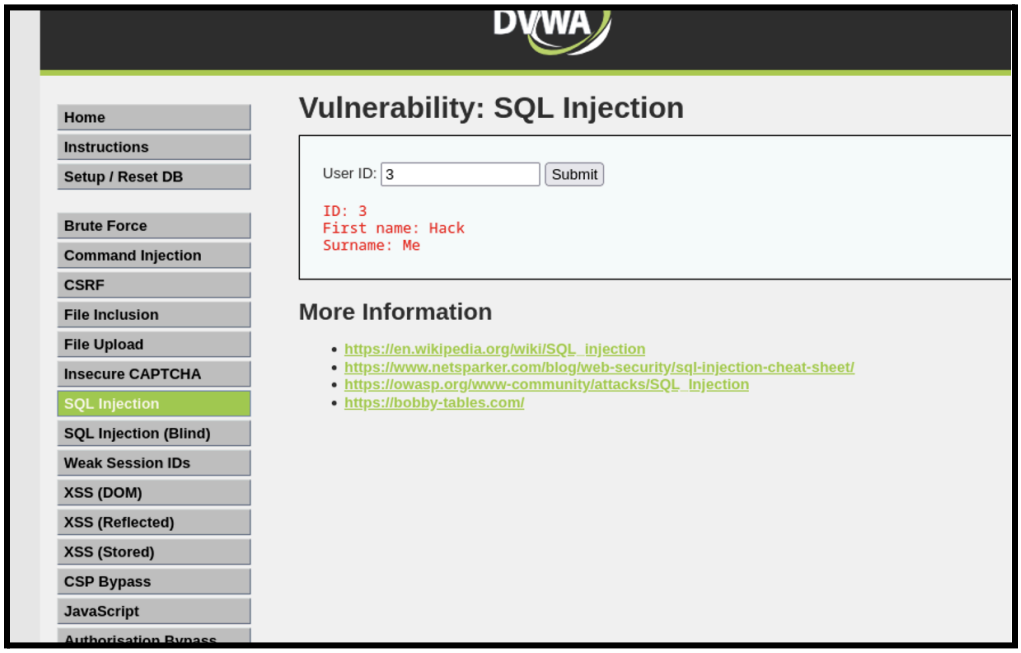
- At **Medium** security, a dropdown menu is provided for User IDs, reducing the attack surface but not eliminating it. Output:



- At **High** security, a pop-up window requests the User ID, adding an additional layer of input validation. Output:



- At **Impossible** security, only valid single numeric values are accepted, effectively blocking the injection. Output:



Incident Impact

The SQL injection vulnerability at the Low security level allows an attacker to:

- Retrieve sensitive information from the database without authorization.
- Potentially escalate to more damaging attacks such as data modification or deletion.
- Compromise the confidentiality and integrity of application data.

This vulnerability poses a serious risk to application security and data protection compliance.

Recommendations

To remediate and prevent SQL injection vulnerabilities:

1. **Input Validation and Sanitization:**
   Validate and sanitize all user inputs on both client and server sides to accept only expected data types and formats.
2. **Security Level Enforcement:**
   Configure the application to enforce higher security levels (High or Impossible) in production environments.
3. **Regular Security Testing:**
   Conduct regular penetration testing and code reviews to detect vulnerabilities early.
4. **Awareness and Training:**
   Train developers and staff on secure coding practices and the risks of injection attacks.

Conclusion

The exercise demonstrated identification and exploitation of a critical SQL injection vulnerability in DVWA. While DVWA does intentionally expose these flaws,, the results still highlight the importance of secure coding and rigorous testing in real-world applications in order to safeguard sensitive data.