Incident Report: SQL Injection Vulnerability in DVWA

Introduction

This report documents a simulated SQL injection vulnerability assessment conducted in the DVWA (Damn Vulnerable Web Application) environment. The objective was to understand and exploit SQL injection to gain insights into the security implications of such attacks. DVWA was set up on a virtual machine to provide a controlled and secure environment for the exercise.

Incident Description

The SQL injection vulnerability was identified in the "User ID" input field of the DVWA application under the "SQL Injection" section. By manipulating the SQL query, it was possible to bypass authentication controls and retrieve sensitive database information. This type of vulnerability, if left unaddressed in real-world applications, could lead to unauthorized data access, data leakage, and potentially full database compromise.

Reproduction Process

1. **Environment Setup**:
    - DVWA was installed on a virtual machine, configured with a local web server (Apache) and database (MySQL).
    - Security level was set to **Low** in DVWA to minimize protections.
2. **Execution of SQL Injection**:
    - Navigated to the **SQL Injection** section of DVWA.

Entered the following payload in the "User ID" input field:
1' OR '1'='1

    - Clicked the **Submit** button.

3. **Observation of Results**:
    - The SQL query was manipulated to return all rows from the database table.
    - Sensitive user information, including names and IDs, was displayed, demonstrating the successful exploitation of the vulnerability.

Incident Impact

The successful execution of the SQL injection highlighted significant risks that could arise if such vulnerabilities existed in a production environment:

1. **Data Breach**: Unauthorized access to sensitive user data.
2. **Reputation Damage**: Loss of trust from clients and users if the vulnerability is exploited in real-world scenarios.
3. **Regulatory Compliance Issues**: Violations of data protection regulations like GDPR or CCPA, leading to legal and financial repercussions.

Recommendations

To mitigate SQL injection vulnerabilities, the following measures are recommended:

1. **Input Validation**: Implement strict validation on user inputs to ensure only expected data formats are accepted.
2. **Parameterized Queries/Prepared Statements**: Use parameterized queries or prepared statements to prevent SQL commands from being executed as part of user input.
3. **Database User Privileges**: Limit the privileges of database user accounts to restrict access to sensitive data and operations.
4. **Web Application Firewall (WAF)**: Deploy a WAF to detect and block SQL injection attempts in real-time.
5. **Security Patching**: Regularly update software and frameworks to address known vulnerabilities.
6. **Education and Training**: Provide developers and administrators with training on secure coding practices and SQL injection prevention.

Conclusion

The exercise demonstrated the critical impact of SQL injection vulnerabilities and emphasized the importance of robust security measures in application development. While this simulation was conducted in a safe and controlled environment, it underscores the need for vigilant security practices to safeguard real-world systems. By implementing the recommended measures, organizations can significantly reduce the risk of SQL injection attacks.
