ISO 27001 Compliant Incident Management Report - SQL Injection Vulnerability

Introduction

This report details the identification and exploitation of an SQL injection vulnerability in the Damn Vulnerable Web Application (DVWA). The test was conducted in a controlled environment to demonstrate a common vulnerability and its potential impact on application security.

Incident Description

During the security assessment of DVWA, an SQL injection vulnerability was discovered in the "SQL Injection" module. This vulnerability allows an attacker to inject malicious SQL queries through the web application's input fields, thereby compromising the integrity and confidentiality of the data stored in the database.

SQL Injection Method Used

To replicate and demonstrate the vulnerability, the following SQL payload was used in the "User ID" field:

sql

```
' 1' OR '1'='1
```

This payload exploits the vulnerability to modify the original SQL query in such a way that it returns the usernames and passwords stored in the users table, specifically for the user with id = 1. By successfully executing this SQL injection, the target user's credentials are obtained without authorization.

Obtained results:

```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

Incident Impact

Exploiting this vulnerability could allow an attacker to:

- Access and extract confidential information from the database, including user credentials.
- Modify, delete, or compromise sensitive data stored in the application.

This represents a significant risk to the confidentiality, integrity, and availability of the data and services provided by DVWA.

Recommendations

Based on the findings of this security assessment, the following corrective and preventive measures are recommended:

1. **Input Validation**: Implement strict input validations for all user-supplied data, using secure parameters in SQL queries to prevent SQL injection.

2. **Use a Web Application Firewall (WAF)**

3. **Penetration Testing**: Conduct regular security audits, including penetration tests, to identify and mitigate security vulnerabilities before they are exploited by attackers.

4. **Education and Awareness**: Train technical and non-technical staff on secure application development practices and raise awareness of the risks associated with security vulnerabilities.

Conclusions

The identification and successful exploitation of the SQL injection vulnerability in DVWA underscores the importance of proactive security in the development and maintenance of web applications. Implementing robust security controls such as WAF for instance and following best cybersecurity practices are essential to protect critical assets and ensure business continuity.