CSOC 1030: Assignment #4

July 26, 2023

**Prepared By**: *Ashish Kishor Hedau*

Table of Contents

[SQL injection vulnerability leading to sensitive data exposure 2](#_Toc137283759)

[Description 2](#_Toc137283760)

[Impact 2](#_Toc137283761)

[Recommendations 2](#_Toc137283762)

[Steps to Reproduce 3](#_Toc137283763)

# SQL Injection vulnerability leading to Sensitive Data Disclosure

## Description

The assessed web application “<http://10.5.30.40>” is designed to get classified information about “Area 51 inventory”. The web application was found to have a SQL Injection vulnerability. The vulnerability lies in the “threat\_level” parameter, which does not properly validate and sanitize user-supplied data before processing it, allowing attackers to execute malicious SQL queries on the database.

## Impact

The SQL Injection vulnerability in the web application allows attackers to execute arbitrary SQL commands on the database. This could lead to unauthorized access, modification, exfiltration, or deletion of sensitive data. Given the application’s purpose of accessing classified information, the exploitation of this vulnerability could potentially compromise the physical security of Area 51 if critical asset information is exposed to the public.

## Recommendations

Instead of embedding user input directly into SQL statements, use parameterized queries or prepared statements. This will allow the database to distinguish between code and data, preventing SQL injection. In addition, utilized stored procedures to define specific database operations that can be called with parameters. This separates the SQL logic from the application code and reduces the risk of injection.

## Steps to Reproduce

The application in scope ( <http://10.5.30.40> ) was found to have a ‘search parameter’ for Area 51 inventory accompanied by ‘threat levels’, ‘order by’ and ‘order’ parameters. The website is designed to give classified information of Area 51 when a user enters inventory name in the parameter mentioned above.

![A screenshot of a computer

Description automatically generated]()

The image below shows the inventory list when a user enters “A” in the search parameter.

![A screenshot of a computer

Description automatically generated]()

Passing the ‘POST’ request through a proxy revealed that the parameter was encoded by “URL Encoding” and “Base64” on client-side.

![A screenshot of a computer

Description automatically generated]()

The image below shows the encoded (‘) as Jw%3d%3d (single quote) being sent to the application via threat\_level parameter. We get the 500 internal server error showcasing that the server was unable to handle this request.

![A screenshot of a computer error

Description automatically generated]()

“Union Select” payload is displayed in the image, which would give us the table name, user, column name from the database exposed

![A screenshot of a computer

Description automatically generated]()

Here we are injecting the encoded “Union Select” payload in the threat\_level parameter mentioned in the image above.

![A screenshot of a computer

Description automatically generated]()

To see if we could dump the whole database, we used SQL map.

Note: We use “–tamper=base64encode” flag as we couldn’t dump the database with normal SQLmap query.

![A screenshot of a computer

Description automatically generated]()

We were able to successfully dump the whole database, if done by a threat actor the Area 51 security posture would be compromised entirely.

![A screenshot of a computer

Description automatically generated]()