**Incident Report**

Date of Incident: 31/10/24

Prepared by: Mary Bisola Olalekan (of team 3)

**Overview**

This report describes a security vulnerability test conducted using Burp Suite, a tool for testing web application security. I tested the Web Security Academy’s sample website to see if it had any weaknesses in how it manages user accounts. This report includes the steps I took, the methods used, what I found, and how these issues could be fixed.

**Objective**

The purpose of this test was to see if it was possible to make unauthorized changes to a user account by manipulating web requests. I wanted to understand if a user could delete or modify accounts without proper permission.

**Test Scope**

The test focused on the /api/user/wiener account endpoint within the Web Security Academy's controlled environment. The goal was to see if sensitive actions, like deleting an account, could be performed without special permission.

**Steps and Techniques Used**

**1. Getting Set Up:**

I set up Burp Suite to capture all the interactions between my computer and the web application.

My first task was to explore the web app and look at the information that gets sent and received when I try to access my account details.

**2. Trying Different HTTP Methods:**

I used PATCH and DELETE HTTP requests in Burp Suite’s Repeater tool to see if I could modify or delete the /api/user/wiener account.

**Result:** The DELETE request returned a message "User deleted", which showed that the server did not block my request, even though deleting an account should normally require special permissions.

**3. Session and Cookie Testing:**

Each request included a session cookie that looked like **session=RCuhG3pzlWAjNC2rWaHyIsz4EbMDjx**.

By examining the cookies, I found that the web app wasn’t properly checking if I was an authorized user for actions like deleting accounts.

**Key Findings**

**1. Missing Permission Checks:**

The web app allowed me to delete or change accounts without confirming my permission. This means anyone logged in could potentially delete any account, which is a major security issue.

**2. Weak Session Management:**

The app relies on session cookies, but it doesn’t seem to verify them properly. This lets users potentially reuse or alter sessions to perform unauthorized actions.

**Suggested Fixes**

Here are some basic suggestions that could help prevent these kinds of issues:

**1. Set Up Role-Based Access Control (RBAC):**

This means setting up different levels of access for different users. For example, only administrators should be able to delete accounts. Every action should be checked to make sure the user has the correct permission.

**2. Strengthen Session Management:**

Use unique, temporary session cookies that expire when the user logs out or after a short period of inactivity. Also, re-check the user’s identity when they try to perform sensitive actions.

**3. Limit HTTP Methods on Endpoints:**

Restrict what each user can do with the web app by limiting HTTP methods like PATCH or DELETE to users with special access only.

**4. Monitor and Alert:**

Set up a system to watch for unusual actions (like deleting accounts) and alert the security team. Logs can help trace any suspicious activity.

**Conclusion**

This test showed that the Web Security Academy’s sample app had weaknesses in user account security. By adding checks on permissions, improving session handling, and monitoring for suspicious actions, the app could better protect user accounts and prevent unauthorized changes.