CSOC 1030: Assignment #6

August 08, 2023

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

Table of Contents

[Business Logic Flaw: Revelation of Email ID and Net worth of users 2](#_Toc137283759)

[Description 2](#_Toc137283760)

[Impact 2](#_Toc137283761)

[Recommendations 2](#_Toc137283762)

[Steps to Reproduce 3](#_Toc137283763)

# Business Logic Flaw: Revelation of Email ID and Net worth of users

## Description

During the assessment of the web application within the scope, Millionaires Matching Ltd. (<http://10.5.30.60>), a business logic flaw was identified. This flaw enabled malicious actors to access and reveal email addresses and net worth information of users, which are functionalities prohibited within the web application’s intended design. Business logic flaw refers to the rules and processes that dictate how a web application should function based on its intended purpose.

## Impact

By examining the JavaScript (JS) code on the front end, a malicious actor can extract directory information, revealing numerous paths including the “/activate” path which exposes usernames. Armed with this knowledge of usernames, the attacker can proceed to infiltrate individual accounts. By employing SQL injection commands within the password field, the attacker can effectively bypass security measures, leading to disclosure of user’s Email IDs and Net worth. It’s important to note that the application’s user disclaimers explicitly state that the Email IDs and Net worth of users are strictly confidential and not shared with others.

Top of Form

## Recommendations

Avoid including directory paths or any other sensitive data in JavaScript files that are accessible to users. Implement robust validation and sanitization of user inputs. Use parameterized queries or prepared statements to prevent SQL injection attacks.

## Steps to Reproduce

The assessed application ( <http://10.5.30.60> ) landing page is shown below.

![A screenshot of a computer

Description automatically generated]()

After clicking ‘login’ on the landing page, we get a Login page protected with Username and Password.

![A screenshot of a computer login screen

Description automatically generated]()

Upon scrutinizing the web application’s frontend source code (JS), we came across an extensive list of directories, as displayed below:

![A screenshot of a computer

Description automatically generated]()

Here, we examined the “/activate” directory, which exposed a significant number of usernames crucial for accessing the login page.

![A screenshot of a computer

Description automatically generated]()

As seen below, when visiting the “/profile” directory without valid credentials, the page is loaded as blank accompanied by the Disclaimer saying, “**We’ll never reveal your email or net worth to other users of the platform**.”

![A screenshot of a computer

Description automatically generated]()

Here, we used one of the usernames found earlier, **Kaden74** as our target. The password used here is a simple SQL injection command (‘ or ‘1’=’1) for test purposes.

![A screenshot of a computer login screen

Description automatically generated]()

Contrary to our expectations, we were able to login as **Kaden74**.

**Note**: We can see the email address and net worth of the individual

![A screenshot of a computer

Description automatically generated]()