Student's name: Matheus Henrique Schaly

Professor's name: Carlisle Adams

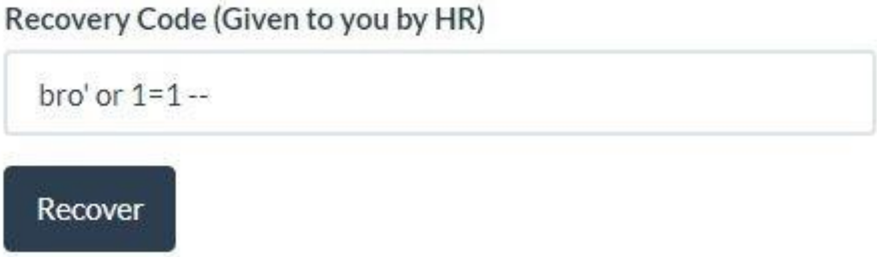
Course: CSI 4139 Design of Secure Computer Systems

Due date: 18th November, 2019

Report #4

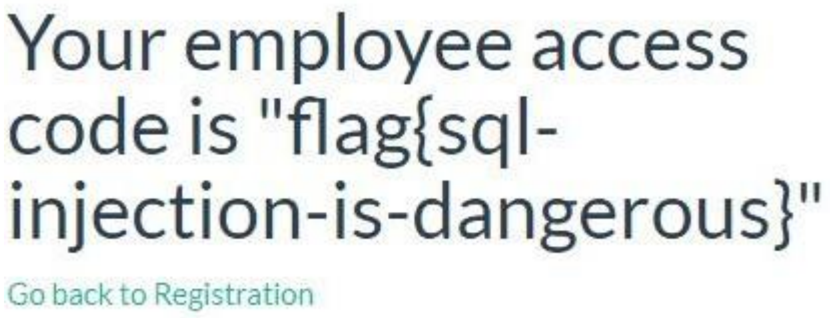
1) A SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information [1]. In this case the attack was used to get the password. Besides that, a flag was discovered which was needed to register an account.

Recommendation: There are many ways to reduce the treats of a SQL injection attack. Those are: 1) Don't use dynamic SQL, don't construct queries with user input, 2) update and patch, 3) firewall, 4) trust no one, 5) reduce your attack surface, 6) use appropriate privileges, 7) continuously monitor SQL statements from database-connected application, and so on [2].



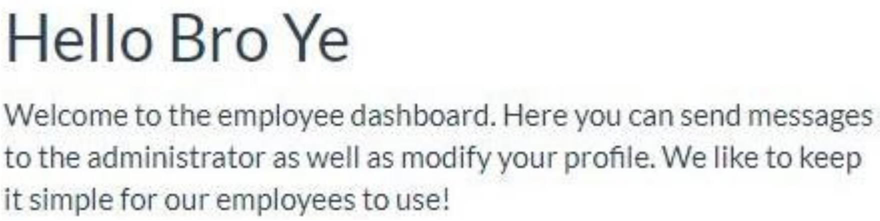
Recovery Code (Given to you by HR)

bro' or 1=1 --



Your employee access code is "flag{sql-injection-is-dangerous}"

Go back to Registration

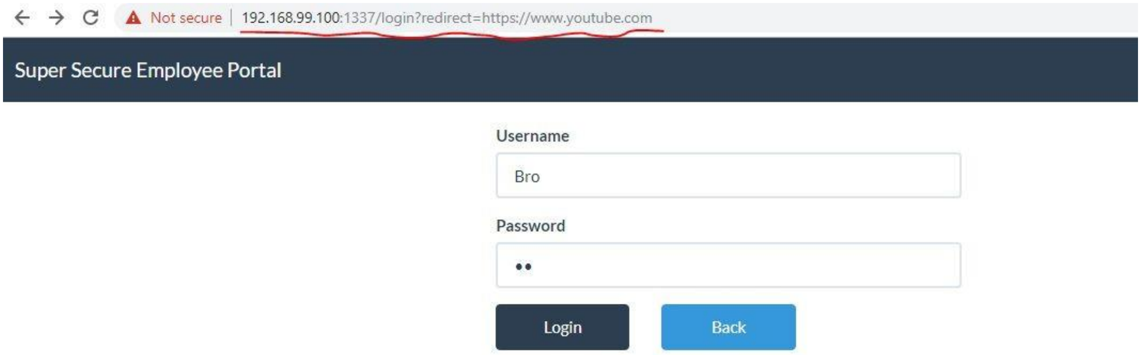


Hello Bro Ye

Welcome to the employee dashboard. Here you can send messages to the administrator as well as modify your profile. We like to keep it simple for our employees to use!

2) No flags were generated in this part.

Recommendation: Instead of using GET request, hardcode the path variable in the code, so that when the login button gets clicked, the variable containing the correct URL is called.



Not secure | 192.168.99.100:1337/login?redirect=https://www.youtube.com

Super Secure Employee Portal

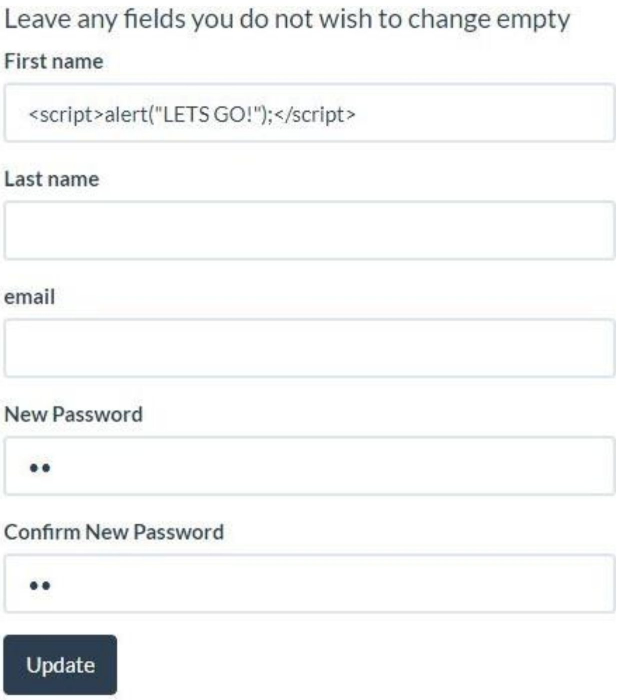
Username: Bro

Password: ••

3) We used a JavaScript code in the first name field to generate the pop up on the dashboard page. Then, another flag was discovered.

Recommendation: There are some ways to prevent XSS vulnerabilities. These are some of them: 1) “Escaping” data. Which means taking the data an application has received and ensuring it’s secure before rendering it for the end user. 2) Validating input.

That is, ensuring an application is rendering the correct data and preventing malicious data from doing harm to the site. 3) Sanitizing. Which means ensuring data received can do no harm to users as well as your database by scrubbing the data clean of potentially harmful markup, changing unacceptable user input to an acceptable format [3].



Leave any fields you do not wish to change empty

First name

<script>alert("LETS GO!");</script>

Last name

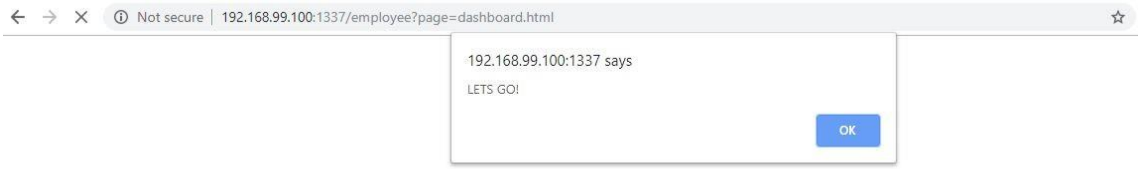
email

New Password

●●

Confirm New Password

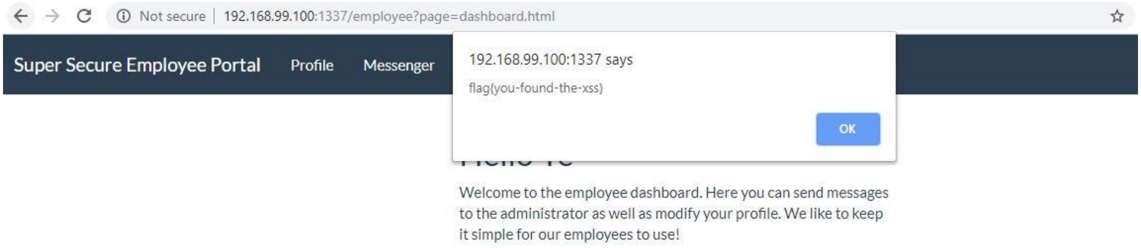
●●



Not secure | 192.168.99.100:1337/employee?page=dashboard.html

192.168.99.100:1337 says

LETS GO!



Not secure | 192.168.99.100:1337/employee?page=dashboard.html

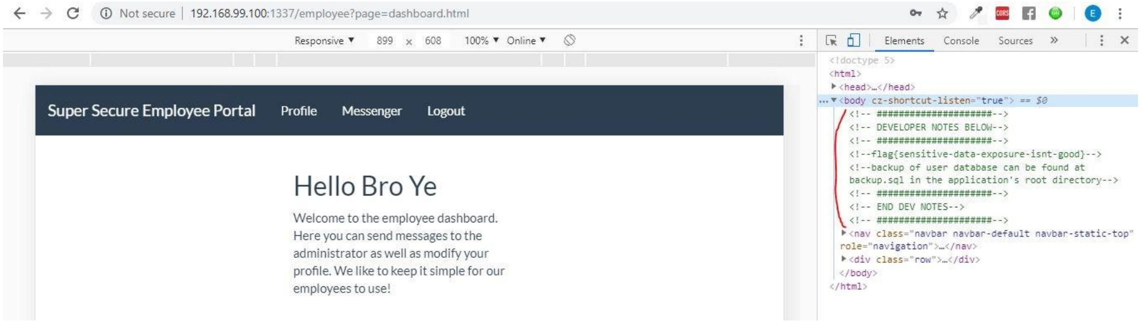
192.168.99.100:1337 says

flag{you-found-the-xss}

Welcome to the employee dashboard. Here you can send messages to the administrator as well as modify your profile. We like to keep it simple for our employees to use!

4) The step was on the comments in the dashboard page’s HTML code, under the “developer notes”: “Backup of user database can be found at backup-sql in the application’s root directory”.

Recommendation: Any unauthorized user shouldn't have access to sensitive data. The administrator shouldn't expose sensitive information in code that is available to the user.


Not secure | 192.168.99.100:1337/employee?page=dashboard.html
Responsive ▾ 899 x 608 100% ▾ Online ▾
Elements Console Sources »
Super Secure Employee Portal Profile Messenger Logout

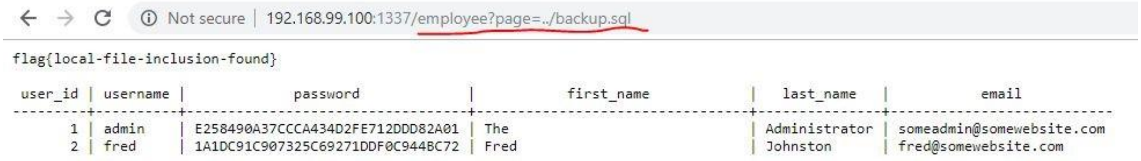
Hello Bro Ye

Welcome to the employee dashboard. Here you can send messages to the administrator as well as modify your profile. We like to keep it simple for our employees to use!

```
<!doctype 5>
<html>
 ▸<head>…</head>
…▾<body cz-shortcut-listen="true"> == $0
    <!-- #####################-->
    <!-- DEVELOPER NOTES BELOW-->
    <!-- #####################-->
    <!--flag{sensitive-data-exposure-isnt-good}-->
    <!--backup of user database can be found at
    backup.sql in the application's root directory-->
    <!-- #####################-->
    <!-- END DEV NOTES-->
    <!-- #####################-->
   ▸<nav class="navbar navbar-default navbar-static-top"
   role="navigation">…</nav>
   ▸<div class="row">…</div>
   </body>
</html>
```

5) Yes, and a new flag was discovered.

Recommendation: Unauthorized users shouldn't have access to sensitive files. Access control could be used to deny access of unauthorized users, only allowing them to visualize public and irrelevant (in the security perspective) data.

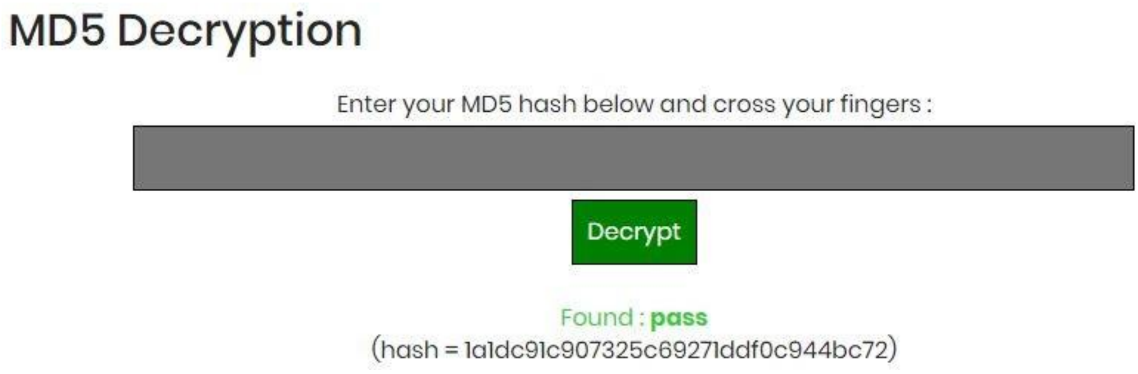

Not secure | 192.168.99.100:1337/employee?page=../backup.sql

```
flag{local-file-inclusion-found}

 user_id | username |             password             |            first_name            |   last_name   |           email
---------+----------+----------------------------------+----------------------------------+---------------+---------------------------
       1 | admin    | E258490A37CCCA434D2FE712DDD82A01 | The                              | Administrator | someadmin@somewebsite.com
       2 | fred     | 1A1DC91C907325C69271DDF0C944BC72 | Fred                             | Johnston      | fred@somewebsite.com
```

6) By using the *backup.sql* file and a decryption related website [4], we could decrypt Fred's password and access his account.

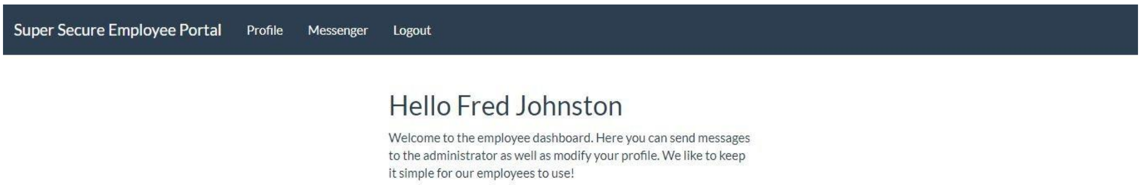
Recommendation: Fred's password was extremely weak. The administrator should force strong password creation. Furthermore, using more secure and widespread hashing algorithms would also improve the system's security.



MD5 Decryption

Enter your MD5 hash below and cross your fingers :

Found : **pass**
(hash = 1a1dc91c907325c69271ddf0c944bc72)



Hello Fred Johnston

Welcome to the employee dashboard. Here you can send messages to the administrator as well as modify your profile. We like to keep it simple for our employees to use!

References:

[1] Academy, W. and injection, S. (2019). *What is SQL Injection? Tutorial & Examples*. [online] Portswigger.net. Available at: https://portswigger.net/web-security/sql-injection [Accessed 15 Nov. 2019].

[2] Attacks, H. and Rubens, P. (2019). *How to Prevent SQL Injection Attacks*. [online] Esecurityplanet.com. Available at: https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html [Accessed 15 Nov. 2019].

[3] Checkmarx. (2019). *3 Ways to Prevent XSS*. [online] Available at: https://www.checkmarx.com/2017/10/09/3-ways-prevent-xss/ [Accessed 15 Nov. 2019].

[4] Md5online.org. (2019). *MD5 Online | Free MD5 Decryption, MD5 Hash Decoder*. [online] Available at: https://www.md5online.org/md5-decrypt.html [Accessed 15 Nov. 2019].