**SQL Security**

**Intro to Database Management Systems**

April 7th, 2019

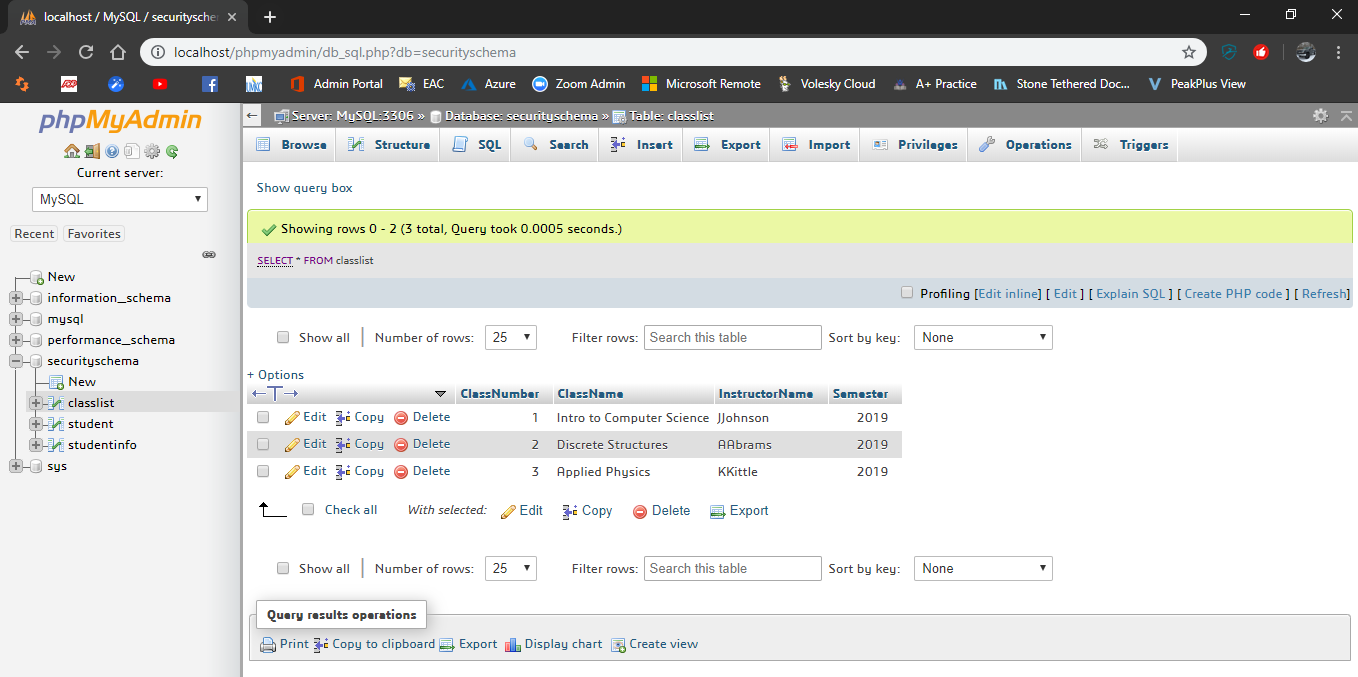
University of Missouri-Kansas City

Brandon Volesky

**Part 1**

Database security is an extremely important and often ignored component of applications development. For this assignment you will create a small database of just a few tables with data in them. The design and purpose of those will be left up to you. Using your favorite programming language write a desktop, web, or console application to allow the user to query a table. Using this query show that SQL injection can be introduced. You should show a couple of different exploits. One exploit should delete a table, return database information, or catalog information. The other only needs to return extra information not included in the query.

The relational database management system that I selected to perform these tasks was MySQL. I used WampServer software stack of Apache web server, OpenSSL, MySQL, PHP, and PHPMyAdmin.



After creating the database and populating it with data, I finished writing the PHP document that is stored in the WAMP directory. This gives the end user the ability to search for a persons name in the database.

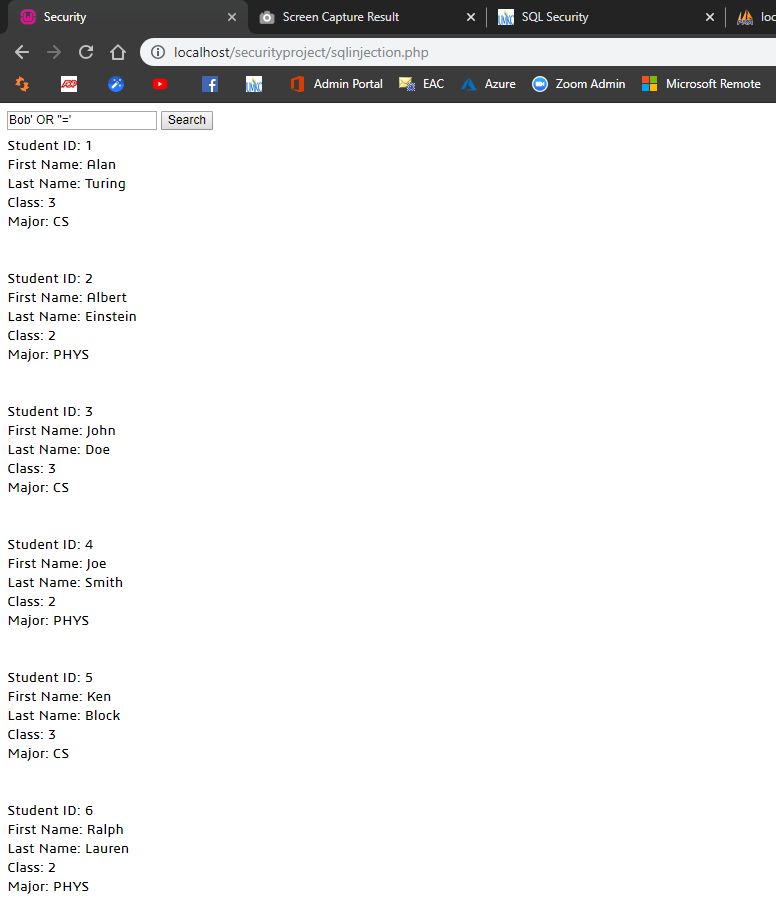
I tested the SQL injection by feeding the input box the following:  
  
***Bob’ OR ‘’=’***

This the code was intended to run:

SELECT \* from Student where Student.Last LIKE \*user input\*

But instead ran:

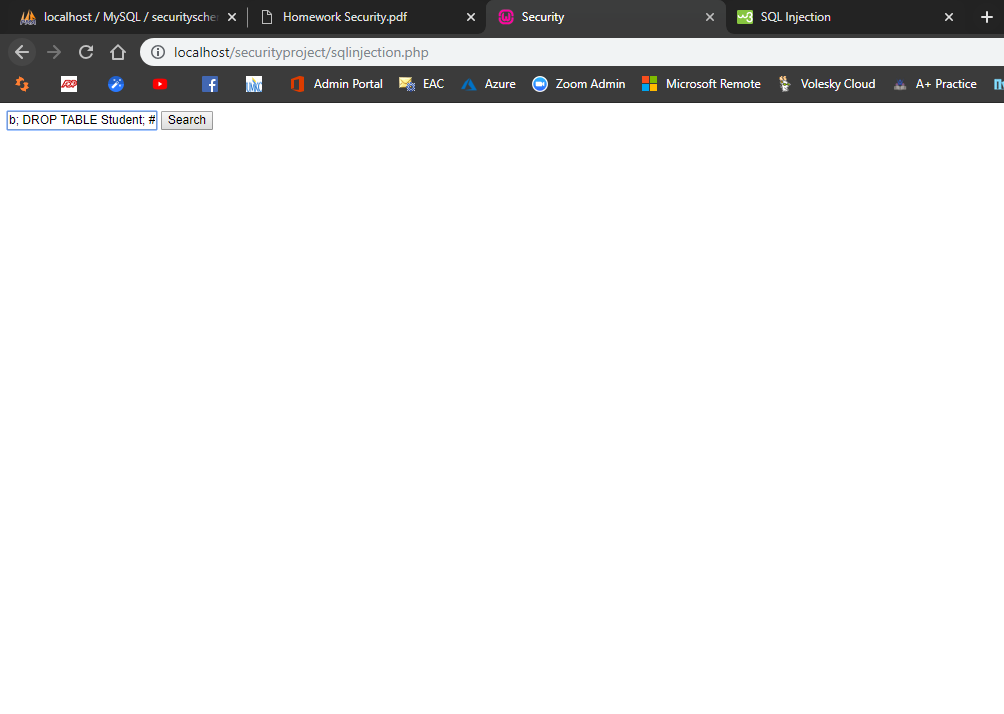
SELECT \* from Student where Student.Last LIKE ‘Bob’ OR ‘’=’’



I also inputted the following statement designed to drop the table:

***b; DROP TABLE Student; #***

The b; allows the original statement to complete as intended, but then it is directy followed up with another statement that drops the table “Student”. The semicolon ends the statement, and the pound sign comments out the ‘ that is added at the end of the query. This dropped the table and was confirmed in PHPMyAdmin.

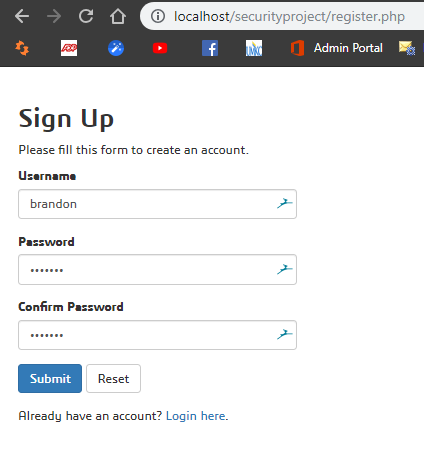


**Part 2**

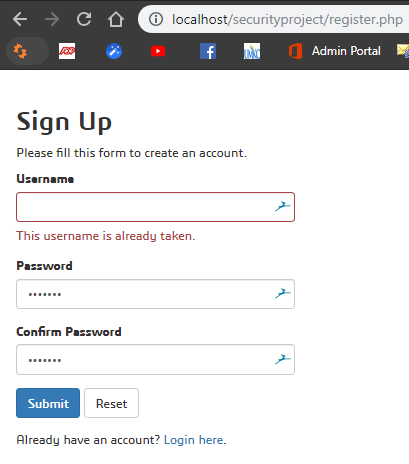
Next you want to demonstrate how RBAC or MAC can increase the security of your application. Create a view that only accesses a subset of the table your application is querying. Create a new user or role and only give them only access that user or role may require. Change your application to use that user and show the results of the SQL injection attack. The application should no longer have access to the database catalog information or portions of the query table not allowed by the view. You should also illustrate that sql injection is still possible, but only limited.

To enforce Role Based Access Control in my setup, I had to distinguish who was authenticating into that role. I created a login in PHP that allowed the user to Sign Up and Log In to the database.

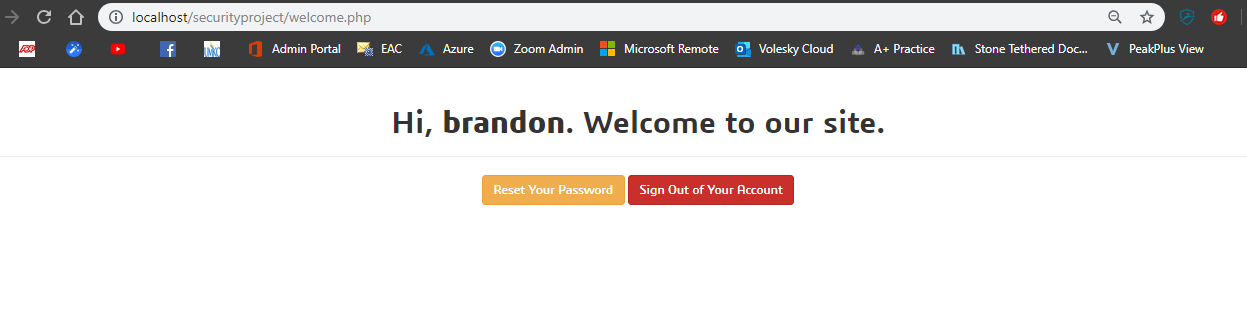
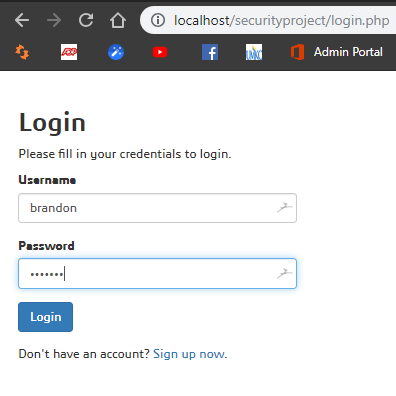
This snip shows how the user can sign up for an account.



This snip shows how the username is checked to make sure the username doesn’t already exist



The user can log into the site and is given the lowest level of database access until the dba assigns the user that specific role with access to that view.



**Part 3**

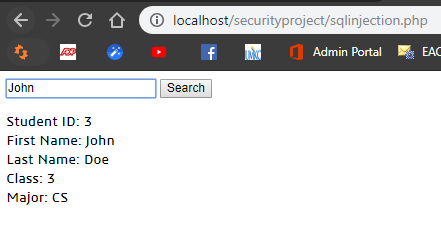
Create a stored procedure for your query. Modify your application to show use the stored procedure for the query. Attempt to use SQL injection on your query now.

I found that stored procedures were very effective against SQL injection. Instead of the user being able to run code for whatever input they submit, the input they submit is saved into a variable. This variable is then passed into the intended statement and ran within the confines of that variable.

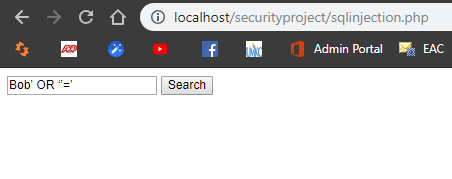
I have modified the code to run this statement when searching for individuals:

**$sql="SELECT \* from Student where First like '%$search\_value%'";**

The user input is passed into the variable $search\_value instead of being ran directly. This eliminated the sql injection abilities.



When the command Bob’ OR ‘’=’ is ran while using the stored procedures the user is given the following empty query:



**Part 4**

Take a look at how you stored the username and password for the database connection access in your program. Is storing this information in source code secure? Find an article on the best practices for storing database connection properties in the language and solution you’ve created. Include the link and description of how best to store it in your language of choice.

I searched for the best practices for storing the username and password for the stored database connection and found an article by Jonathan Soder and Ash Skilton found at the following link:

<https://www.binpress.com/using-php-with-mysql/>  
  
Ash explains that the source code to PHP scripts are not visible to people that are inspecting your source code. He states that DB passwords are generally secure in PHP, with some exceptions.

One example is a database connection that is created manually with the credentials in the same script running the query (see below).

1. <?php
2. $username="username";
3. $password="password";
4. $database="your\_database";
5. $field1-name=$\_POST['Value1'];
6. $field2-name=$\_POST['Value2'];
7. $field3-name=$\_POST['Value3'];
8. $field4-name=$\_POST['Value4'];
9. $field5-name=$\_POST['Value5'];
10. mysql\_connect(localhost,$username,$password);
11. @mysql\_select\_db($database) or die( "Unable to select database");
12. $query = "INSERT INTO tablename VALUES('','$field1-name','$field2-name',
13. '$field3-name','$field4-name','$field5-name')";
14. mysql\_query($query);
15. mysql\_close();
16. ?>

The best solution is to abstract the connection credentials so there is a separation. This

separation allows for centralized management and is important is the files are saved in

the document root directory.

The first step into correcting this is to **test** the connection to the database to ensure

there are no errors in it using variables. Errors thrown could allow the credentials to be

accessible in an error message.

1. *// Try and connect to the database*
2. $connection = mysqli\_connect('localhost',$username,$password,$dbname);
4. *// If connection was not successful, handle the error*
5. if($connection === false) {
6. *// Handle error - notify administrator, log to a file, show an error screen, etc.*
7. }

Then creating the configuration file outside of the root directory.

1. [database]
2. username = root
3. password = 1234
4. dbname = mydb

The next steps are to abstract the credentials from the connection, then put the

connection process inside a function.

1. *// Load configuration as an array. Use the actual location of your configuration file*
2. $config = parse\_ini\_file('../config.ini');
4. *// Try and connect to the database*
5. $connection =mysqli\_connect('localhost',$config['username'],$config['password'],$config['dbname']);
7. *// If connection was not successful, handle the error*
8. if($connection === false) {
9. *// Handle error - notify administrator, log to a file, show an error screen, etc.*
10. }
11. function db\_connect() {
13. *// Define connection as a static variable, to avoid connecting more than once*
14. static $connection;
16. *// Try and connect to the database, if a connection has not been established yet*
17. if(!isset($connection)) {
18. *// Load configuration as an array. Use the actual location of your configuration file*
19. $config = parse\_ini\_file('../config.ini');
20. $connection =mysqli\_connect('localhost',$config['username'],$config['password'],$config['dbname']);
21. }
23. *// If connection was not successful, handle the error*
24. if($connection === false) {
25. *// Handle error - notify administrator, log to a file, show an error screen, etc.*
26. return mysqli\_connect\_error();
27. }
28. return $connection;
29. }

**Part 5**

Finally, research a few data breaches in the last decade. Look at onces that could have been

mitigated by proper database security. Include a link and a short description of the exploit

and data loss.

More detailed information on the yahoo hack can be found here: <https://www.csoonline.com/article/3180762/inside-the-russian-hack-of-yahoo-how-they-did-it.html>

Yahoo is off the charts in terms of records breached. They have an all time high of 3 billion

accounts vulnerable to malicious outsiders.

* In a nutshell, the yahoo hack occurred because of a spear phishing email sent to an employee.
* This allowed the attacker access to the network and in turn allowed the installation of a

backdoor.

* The attacker made a coy of the database and transferred it to his computer
* The attacker gained access to the account management tool and with the help of recovery email addresses was able to target accounts.
* Using stolen “nonces” allowed to generate access cookies through a script installed on one of Yahoo’s servers, allowing the bypass of passwords.

Even though most of the data that was stolen did not directly contain SSN or Credit Card numbers, the information of the users contained in the database such as names, phone numbers, password recovery emails, cryptographic values unique to each account, and **password challenge questions.** This allowed attackers to social engineer other companies and website into giving up information and access to accounts. An attacker could use the information stolen to modify accounts associated with the same challenge question answers.

The company said the "vast majority" of the passwords involved had been hashed using the

robust bcrypt algorithm. The company was valued at $100 billion but was eventually sold to

Verizon for 4.48 billion.

Social Media breaches far outweigh Financial breaches in terms of records breached. Social

breaches allow for attackers to use spear phishing and social engineering to steal identities,

steal money and destroy lives.

**References:**

* Huang, E. (2019). Simple search bar. Retrieved from <https://codepen.io/huange/pen/rbqsD>
* Tutorial Republic. “PHP MySQL Login System.” TutorialRepublic, 2019, [www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php](http://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php).
* Lionite, -. “Using PHP with MySQL - The Right Way.” BinPress, 2019, www.binpress.com/using-php-with-mysql/.