PRACTICAL FINAL

Dennis Keritsis

Problems

1. A user can merely copy and paste the URL to the Welcome Page with: (http://192.168.74.131/Final/welcome.php). New cookie isn't generated upon login. This is related to Session Management.
2. Can just press Login button with no data. This is related to Session Management and improperly validating input.
3. Verbose messaging for incorrect username and password.
4. Verbose messaging for username already take for registration.
5. Password isn't hashed. We don't want to store the password itself. We want to compare it against a hash of the password.
6. Need unique usernames to prevent username enumeration, like email addresses.
7. Welcome Page doesn't outline the user logged in.
8. The Welcome Page does not have a session time out.
9. If the Welcome Page had a session time out, it doesn't outline for how long.

Fix of Proper Session Management (1)[1]

Code Discussion(1)(a)[2]

10. For login.php code change, lines 6-8, 13-24, 28, 41-47, 50-51, 53-54, 57-60, 63-65, 67-69, 71-74, 80, and 103 were altered.
11. For welcome.php code change, lines 4-18 and 33-34 were altered.
12. HTTP is stateless and we want to use cookies to maintain that state. We want to generate a cookie once we have verified that the username and password is correct with "session_start();". Additionally, we want to use attributes within the cookie. Specifically, I choose to use "start_time", "expire_time", and "username".[3] The attributes of "start_time" and "expire_time" are like what they sound like. *See* Login.php at ll. 57-60. They are times to maintain a timer for the session after login.
13. As noted above, someone could just copy and page the URL to the welcome page and get in. We want to fix this by having the "header" function redirect the user if there isn't a proper "username" or if the username is merely empty.[4] Additionally, I wanted to take the username and display it on the Welcome page. *See* Welcome.php at ll. 7-10, 33-34.
14. I want to take the code to the next level by adding a session timer. *Compare* Welcome.php at ll. 11-18 *with* Login.php at ll. 57-60.

Security Implications (1)(b)

15. There is poor session management. Without any authentication, a user can merely copy and paste the URL to the Welcome Page with: (http://192.168.74.131/Final/welcome.php). *See*

[1] Pages of login.php and welcome.php are of primary concern in this section.
[2] **All code changes can be found in the Appendices (A) through (D).** Section (A) is the files overview. Section (B) is the login.php comparison. Section (C) is the register.php comparison. Section (D) is the welcome.php comparison.
[3] Username was an attribute already in the cookie, but I am using it now in the welcome page.
[4] This was pretty trivial since this was in the lab4c code.

Section Problems at Example (1) *supra*. Essentially, authentication is totally useless. In the Final code, a cookie is created; however, the cookie isn't properly utilized.

16. Additionally, an individual could merely just login without entering a username and password since there was no session management.[5] *See* Section Problems at Example (2). With a cookie being used to validate a username and lack of username, a cookie not properly set will redirect the user back to the login page.
17. With regards to social implications, it is important to notify *who* is logged in. By using the cookie, we are able to display who is logged in. Sometimes, if a computer is a shared computer, it is not always clear who is logged in. This notification will help users maintain their sessions. *See* Section Problems at Example (7).
18. Building future on social implications, many users stay logged in for an extended period of time. This can be problematic if the computer is shared. I set a timer for 1 minute to remedy this issue. *See* Section Problems at Examples (8-9).

Section (2) Front End[6]

Verification of Input (2)(a)

Code Discussion (2)(a)(i)

19. Changes related to Verification of input can be found in ll. 6-8, 13-24, 28, and 103 in login.php and ll. 11-15 and 49 of register.php. In the older code, POST messages were merely sent. This allowed for messages that are blanks strings of usernames and password to be sent. The fix was simple in that we wanted to make sure that empty was not TRUE for both the username and password. Additionally, the trim() function was used to remove any blank spaces. Ultimately, if these factors check out, we would send an SQL message accordingly. *See* Login.php at ll. 28-29.
20. Additionally, there was some registration quandaries that were fixed. We want username to be emails and to indeed be valid emails. *See* Register.php at ll. 11-15, 19-32.

Security Implications (2)(a)(ii)

21. As noted in Section Problems at Example (2) *supra*., an individual can just login without a username and password. This frontend code adds another layer of protection. Also, the trim function may be helpful because hackers are always coming up with crafty ways to circumvent what code we are using. It is always good to keep our code tight and know exactly what is getting POST'ed.
22. Additionally, I made all the users register with an email. This helps to mitigate any username duplications (although the backend handles this). Moreover, this helps mitigate any username enumeration hackers might use since username would be longer. For example, if I just registered dkeritsi as my username this would be weaker than dkeritsis@gmail.com or dkeritsis@hotmail.com or dkeritsis@wmich.edu. This is true because there are more permutations at hand which would increase the computational complexity. Additionally, during

[5] This can be remedied with proper front end develop and validation of username input as well.

[6] Pages of register.php and login.php are of primary concern in this section.

registration, I made sure that at least 15 characters were used in total for the email address. *See* Register.php at ll. 21-23.

23. Although this code isn't showcase it, future code implementations could send a confirmation and registration associated with the email. Also, 2-factor authentication could be code-implemented in the future.

Verbose Messaging (2)(b)

Code Discussion (2)(a)(i)

24. Login.php at ll. 41-47 and 66-70 showcase the mitigation of verbose messaging. Line 49 of Register.php deals with this verbose message, albeit obliquely, with using email addresses and noting they have been taken. The code is simple. At the login, merely do not say whether the username or the password is incorrect. Both on under the password and username boxes, I disclose: "The username or password is incorrect." These coupled messages appear whether an incorrect password is triggered or an incorrect username is triggered.

Security Implications (2)(a)(ii)

25. Verbose messaging can be used by hackers to enumerate usernames and passwords, along with the permutations. We want to give them the least amount of information as possible. Also, for username registration, we are using email. I will still note that the email is taken. A hacker could use verbose messaging to enumerate a database of users. However, by using email addresses, a brute for attack will take longer for reasons stated above given the permutations. Also, I set a limit of 12 characters for the email address.

Section (3) Cryptography[7]

Code Discussion (3)(a)

26. The code is simple. In Register.php at ll. 93-94, we want to hash the password and store it accordingly. The first argument of $password is just the password that is to be hashed accordingly. The second argument of PASSWORD_DEFAULT is just the supporting arguments. *See* https://www.php.net/manual/en/function.password-hash.php. As will be discussed later, SALT-ing is an option; however, it appears that in PHP 7.0 this option is deprecated. But it is "preferred to simply use the salt that is generated by default." *Id*. Salt-ing will be discussed under the same section. Ultimately, we store the **hash** of the password, not the password itself during registration.
27. For login.php the code is likewise simple. *See* Login.php at ll. 54-55. We pull from the database the hash of password (i.e. the registered hash) and compare it again the password the user input. This is done with the "password_verify" function.

Math Discussion (3)(b)

[7] Pages of register.php and login.php are of primary concern in this section.

28. A hash is just a oneway function. During registration the following happens: database[row, column]=Hash_with_SALT(password). The most simple type of hash is a checksum. For instance, take the checksum of 18237, which is 1+8+2+3+7=21. The number 21 can not be reversed to get back 18,237. Additionally, this mapping of one way functions is a "non-injective surjective function" also known as "surjective, not a bijection." See https://en.wikipedia.org/wiki/Injective_function. During this mapping, this leads the "collisions". Collisions are not a problem as long as hash functions are cryptographic (as opposed to non-cryptographic). Additionally, it is important to note that statistical methods may be used to determine what the mappings are. For example, the password love (an element in set X) will be hashed and mapped to some element in set Y. This is a very common password. Using statistical methods, a hacker may be able to determine what the password is (element in set X) based on frequency of elements in set Y.
29. To mitigate this, we want to use SALT. This uses a one time value in conjection with the hash. As such, the output would be able as follows: Hash(password, one-time-value).As such, Hash("love", one_time_value_1) != Hash("love", one_time_value_2) even though the same password of "love" is used.

Security Implications (3)(c)

30. PHP is handles SALT-ing with PASSWORD_DEFAULT. Moreover, PHP uses "new and stronger algorithms" as time goes on. This is helpful because many cryptographic hashes may become non-cryptographic as time goes on. As noted above, statistical methods can be used against a database with hashes if SALTing isn't use. However, once information entropy is added, it becomes must more difficult to determine the password.
31. On a more general note, we **never** want to store passwords. Someone could crack our database (or even a gray-hatter may do it) and they would have usernames and passwords. If someone got the username/hashed_password, they wouldn't be able to login because of the one-way nature of hash functions. The hashed_password essentially becomes useless to them.

Section (4) SQL injections

32. Q1. Why do MSQLI prepared statements close the door on SQL injections?
33. A1.
34. For mysql_prepare the first and second arguments are populated with $link and $sql. The $sql is the variable the SELECT query to be sent. According to Adam: "The purpose of prepared statement is to not include data in your SQL statements." *See* https://www.php.net/manual/en/mysqli.prepare.php. Also, we should "[a]lways use prepared statements. They are cleaner to use…and not prone to SQL injections. *Id*. Zeebuck notes the same thing: "Obviously more code has been put into prepared statements today to allow it ot be used to prevent sql inejctions…." *Id*. The prepared statement works by "preventing a user from executing his own SQL code and damaging the integrity of a database." Essentially, the way the prepared statement works is it separates the "user input from the SQL commands." *Id*.

35. According to tecadmin, we can use the "PHP-MySQLi Driver."[8] *See* https://tecadmin.net/prevent-sql-injection-in-php/. It outlines a number of steps: (i) use prepare, (ii) bind parameters, (iii) execute, (iv) result, and (v) fetch. It appears that our code is doing the same thing:

    (i) ($stmt = mysqli_prepare($link, $sql),

    (ii) mysqli_stmt_bind_param($stmt, "s", $param_username);,

    (iii) mysqli_stmt_execute($stmt),

    (iv) mysqli_stmt_bind_result($stmt, $username, $hashed_password), and

    (v) mysqli_stmt_fetch($stmt);.

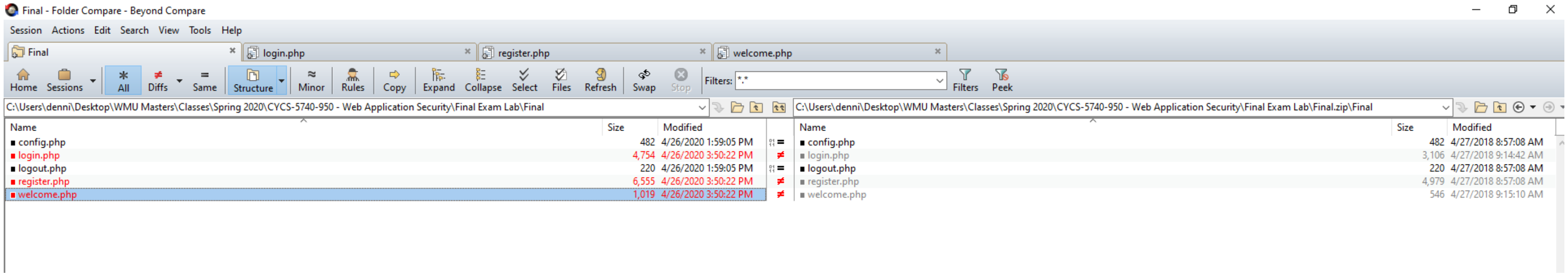
36. The last thing to do is the close the stmt connection.
37. It appears that the prepare statements are broken up into two stages: (i) prepare stage and (ii) blind and execute stage. *See* https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php. Purportedly the "blind" can be done with the mysqli_stmt_blind_param() function and this is the "**only** security feature to prevent SQL injection[.]" *Id* (emphasis added).

38. Q2. What approach to mysqli querying might you find in older, less secure applications?
39. A2. The older approach would be to use "MySQL" instead of "MySQLi". According to phppot.com, "MySQLi function mysqli_query() allows [sic] to enforce error prone queries and prevents bugs like SQL injections." *See* https://phppot.com/php/mysql-vs-mysqli-in-php/.
40. Using an older approach of MySQL, a user would simply take the $sql string and say:
    a. $result = $conn->query($sql);
41. *See* https://phppot.com/php/mysql-vs-mysqli-in-php/
42. However, just using a "MYSQLi Procedural" method might be vulnerable because it is important to use the mysqli_stmt_blind_param() approach. **Put another way, using MySQLi will not by itself prevent SQL injections.** The following may be deleterious:
    a. $result = mysqli_query($conn, $sql);
43. *See* https://www.w3schools.com/php/php_mysql_select.asp

[8] Another option is to use the PHP-PDO Driver.

Appendix A

Files Overview



Final - Folder Compare - Beyond Compare
Session Actions Edit Search View Tools Help
Final
login.php
register.php
welcome.php
Home Sessions All Diffs Same Structure Minor Rules Copy Expand Collapse Select Files Refresh Swap Stop
Filters Peek

Filters: *.*

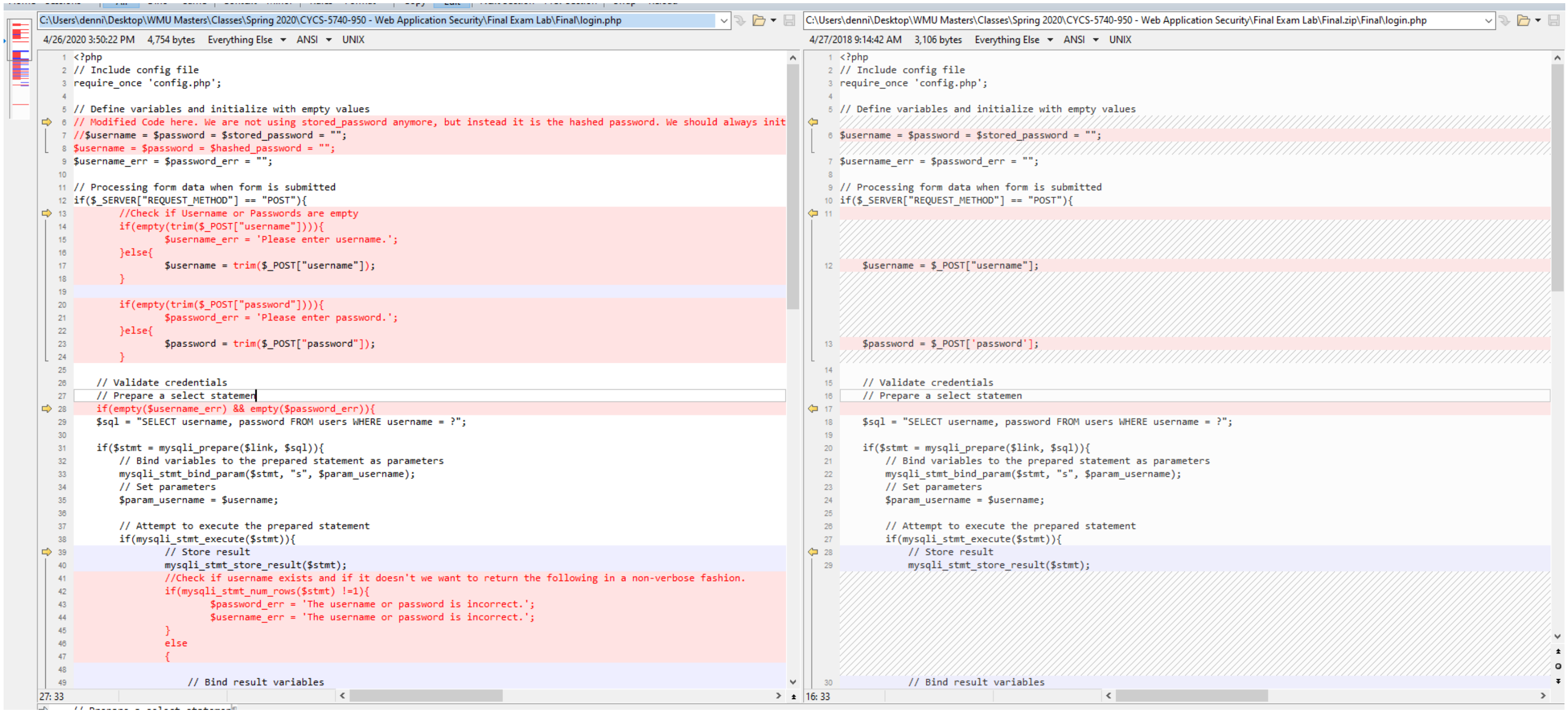
C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final

Name	Size	Modified		Name	Size	Modified
config.php	482	4/26/2020 1:59:05 PM	=	config.php	482	4/27/2018 8:57:08 AM
login.php	4,754	4/26/2020 3:50:22 PM	≠	login.php	3,106	4/27/2018 9:14:42 AM
logout.php	220	4/26/2020 1:59:05 PM	=	logout.php	220	4/27/2018 8:57:08 AM
register.php	6,555	4/26/2020 3:50:22 PM	≠	register.php	4,979	4/27/2018 8:57:08 AM
welcome.php	1,019	4/26/2020 3:50:22 PM	≠	welcome.php	546	4/27/2018 9:15:10 AM

Appendix B

Compare Login.php



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\login.php

4/26/2020 3:50:22 PM 4,754 bytes Everything Else ANSI UNIX

```php
<?php
// Include config file
require_once 'config.php';

// Define variables and initialize with empty values
// Modified Code here. We are not using stored_password anymore, but instead it is the hashed password. We should always init
//$username = $password = $stored_password = "";
$username = $password = $hashed_password = "";
$username_err = $password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){
        //Check if Username or Passwords are empty
        if(empty(trim($_POST["username"]))){
                $username_err = 'Please enter username.';
        }else{
                $username = trim($_POST["username"]);
        }

        if(empty(trim($_POST["password"]))){
                $password_err = 'Please enter password.';
        }else{
                $password = trim($_POST["password"]);
        }

    // Validate credentials
    // Prepare a select statemen
    if(empty($username_err) && empty($password_err)){
    $sql = "SELECT username, password FROM users WHERE username = ?";

    if($stmt = mysqli_prepare($link, $sql)){
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "s", $param_username);
        // Set parameters
        $param_username = $username;

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
            // Store result
            mysqli_stmt_store_result($stmt);
            //Check if username exists and if it doesn't we want to return the following in a non-verbose fashion.
            if(mysqli_stmt_num_rows($stmt) !=1){
                $password_err = 'The username or password is incorrect.';
                $username_err = 'The username or password is incorrect.';
            }
            else
            {

                // Bind result variables
```

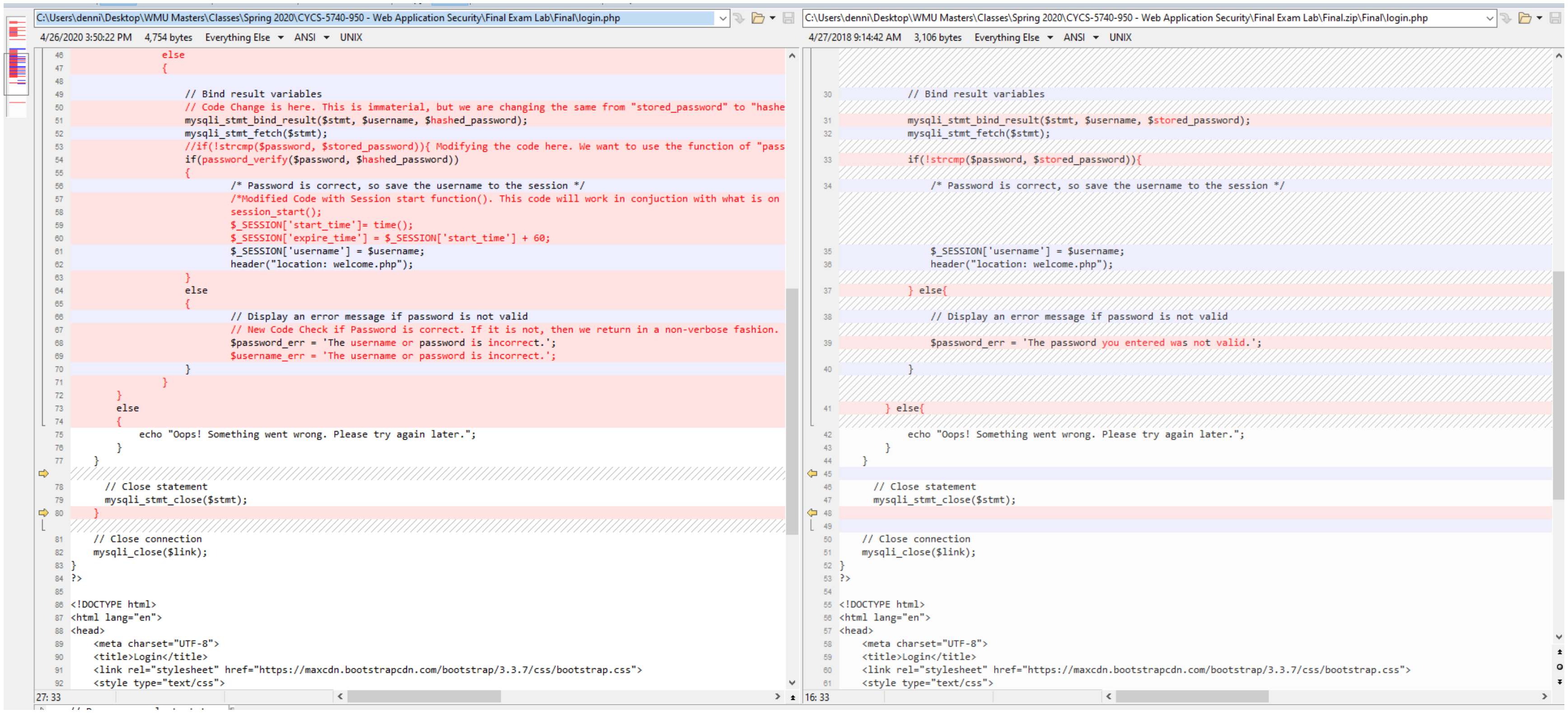
27: 33

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\login.php

4/27/2018 9:14:42 AM 3,106 bytes Everything Else ANSI UNIX

```php
<?php
// Include config file
require_once 'config.php';

// Define variables and initialize with empty values
$username = $password = $stored_password = "";
$username_err = $password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    $username = $_POST["username"];
    $password = $_POST['password'];

    // Validate credentials
    // Prepare a select statemen

    $sql = "SELECT username, password FROM users WHERE username = ?";

    if($stmt = mysqli_prepare($link, $sql)){
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "s", $param_username);
        // Set parameters
        $param_username = $username;

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
            // Store result
            mysqli_stmt_store_result($stmt);
            // Bind result variables
```

16: 33



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\login.php

4/26/2020 3:50:22 PM   4,754 bytes   Everything Else   ANSI   UNIX

```
                else
                {

                    // Bind result variables
                    // Code Change is here. This is immaterial, but we are changing the same from "stored_password" to "hashe
                    mysqli_stmt_bind_result($stmt, $username, $hashed_password);
                    mysqli_stmt_fetch($stmt);
                    //if(!strcmp($password, $stored_password)){ Modifying the code here. We want to use the function of "pass
                    if(password_verify($password, $hashed_password))
                    {
                        /* Password is correct, so save the username to the session */
                        /*Modified Code with Session start function(). This code will work in conjuction with what is on
                        session_start();
                        $_SESSION['start_time']= time();
                        $_SESSION['expire_time'] = $_SESSION['start_time'] + 60;
                        $_SESSION['username'] = $username;
                        header("location: welcome.php");
                    }
                    else
                    {
                        // Display an error message if password is not valid
                        // New Code Check if Password is correct. If it is not, then we return in a non-verbose fashion.
                        $password_err = 'The username or password is incorrect.';
                        $username_err = 'The username or password is incorrect.';
                    }
                }
            }
            else
            {
                echo "Oops! Something went wrong. Please try again later.";
            }
        }

        // Close statement
        mysqli_stmt_close($stmt);
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
```

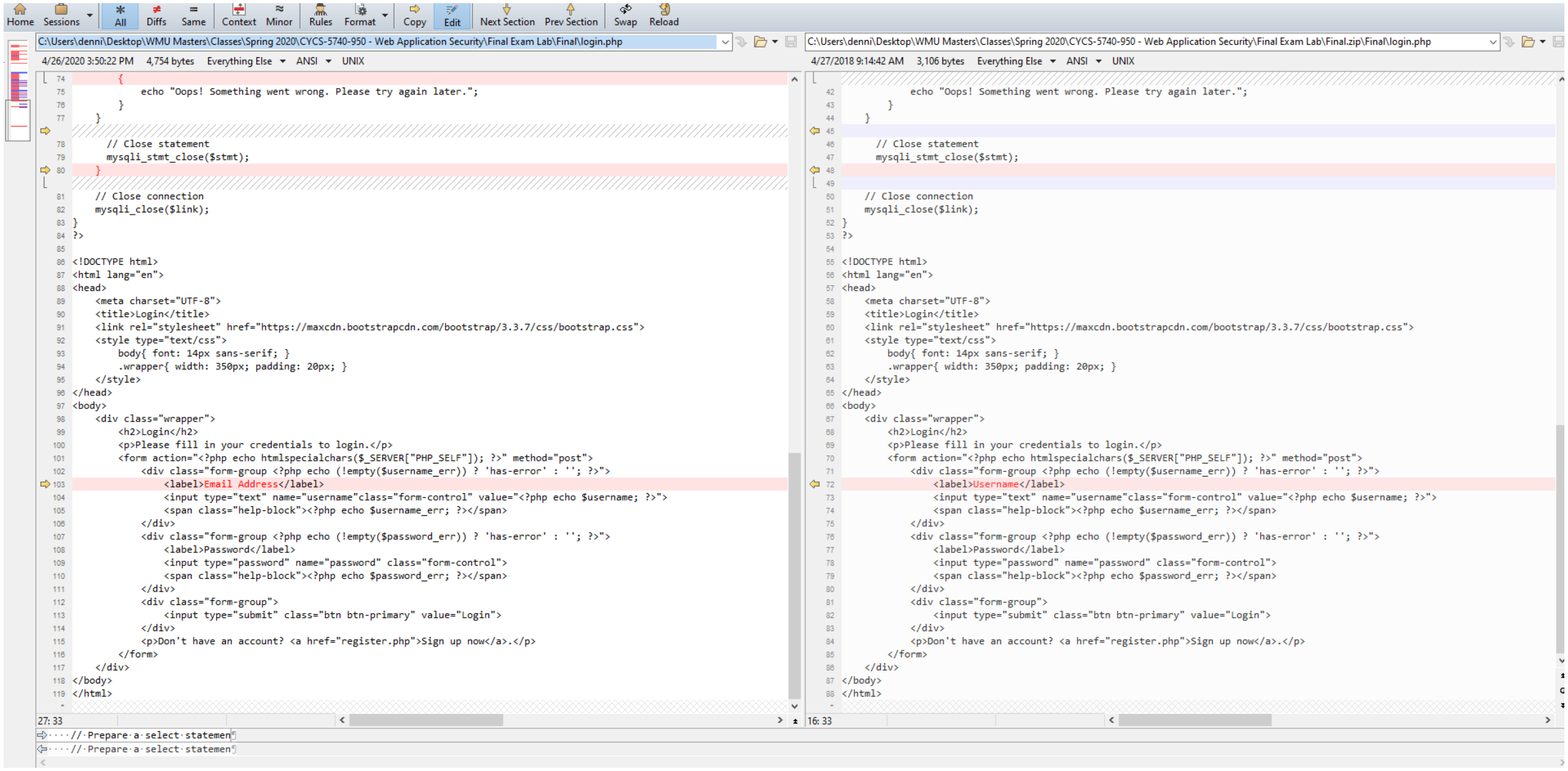
27: 33

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\login.php

4/27/2018 9:14:42 AM   3,106 bytes   Everything Else   ANSI   UNIX

```
                    // Bind result variables

                    mysqli_stmt_bind_result($stmt, $username, $stored_password);
                    mysqli_stmt_fetch($stmt);

                    if(!strcmp($password, $stored_password)){

                        /* Password is correct, so save the username to the session */

                        $_SESSION['username'] = $username;
                        header("location: welcome.php");

                    } else{

                        // Display an error message if password is not valid

                        $password_err = 'The password you entered was not valid.';

                    }

                } else{

                    echo "Oops! Something went wrong. Please try again later.";
                }
            }

        // Close statement
        mysqli_stmt_close($stmt);

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
```

16: 33



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\login.php

4/26/2020 3:50:22 PM 4,754 bytes Everything Else ANSI UNIX

```
        {
            echo "Oops! Something went wrong. Please try again later.";
        }
    }

    // Close statement
    mysqli_stmt_close($stmt);
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Login</h2>
        <p>Please fill in your credentials to login.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Email Address</label>
                <input type="text" name="username"class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control">
                <span class="help-block"><?php echo $password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary" value="Login">
            </div>
            <p>Don't have an account? <a href="register.php">Sign up now</a>.</p>
        </form>
    </div>
</body>
</html>
```

27: 33

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\login.php

4/27/2018 9:14:42 AM 3,106 bytes Everything Else ANSI UNIX

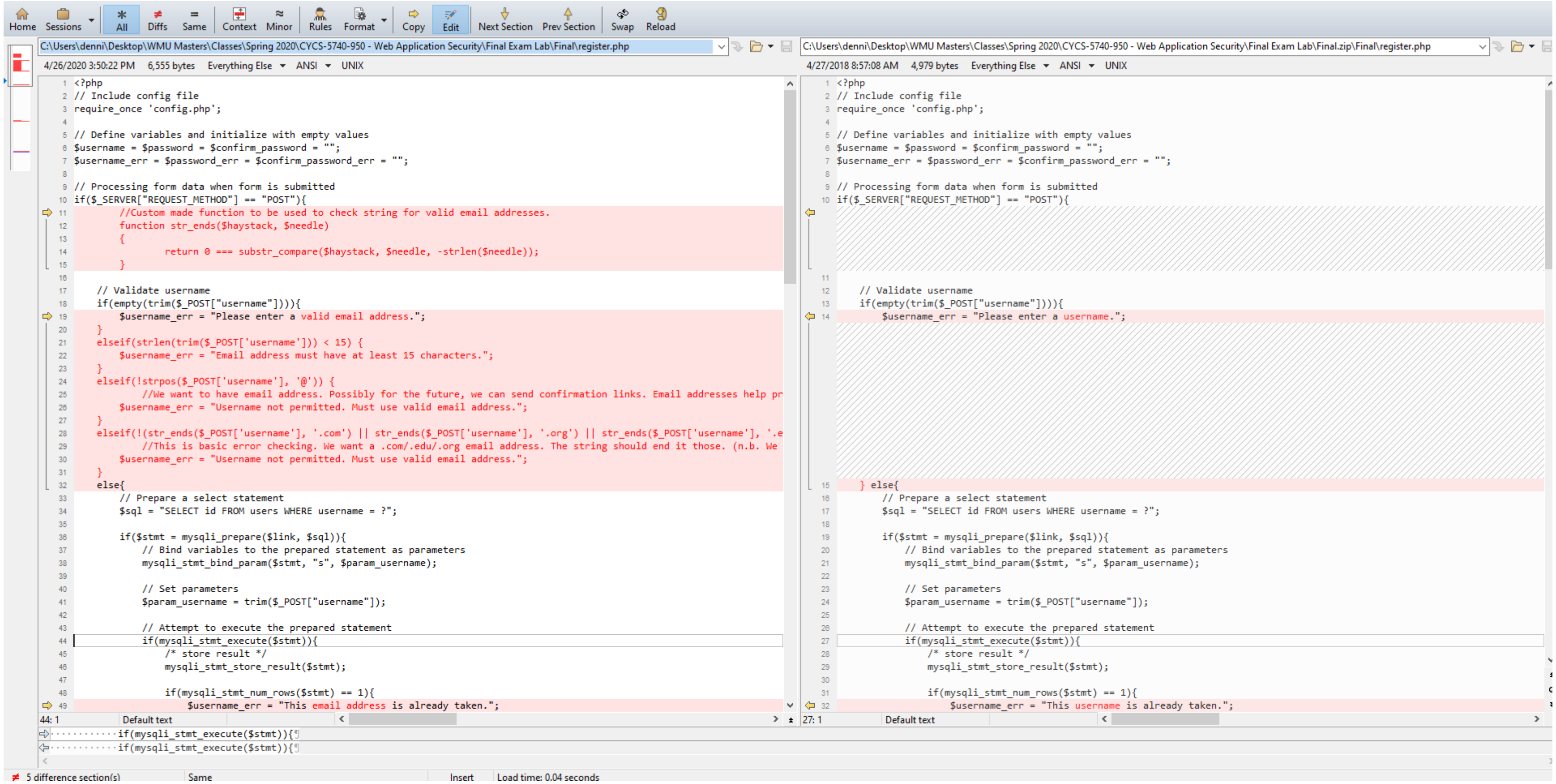
```
            echo "Oops! Something went wrong. Please try again later.";
        }
    }

    // Close statement
    mysqli_stmt_close($stmt);


    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Login</h2>
        <p>Please fill in your credentials to login.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Username</label>
                <input type="text" name="username"class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control">
                <span class="help-block"><?php echo $password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary" value="Login">
            </div>
            <p>Don't have an account? <a href="register.php">Sign up now</a>.</p>
        </form>
    </div>
</body>
</html>
```

16: 33

```
····//·Prepare·a·select·statemen¶
····//·Prepare·a·select·statemen¶
```

Appendix C

Compare Register.php



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\register.php

4/26/2020 3:50:22 PM 6,555 bytes Everything Else ANSI UNIX

```
<?php
// Include config file
require_once 'config.php';

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){
        //Custom made function to be used to check string for valid email addresses.
        function str_ends($haystack, $needle)
        {
                return 0 === substr_compare($haystack, $needle, -strlen($needle));
        }

    // Validate username
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter a valid email address.";
    }
    elseif(strlen(trim($_POST['username'])) < 15) {
        $username_err = "Email address must have at least 15 characters.";
    }
    elseif(!strpos($_POST['username'], '@')) {
            //We want to have email address. Possibly for the future, we can send confirmation links. Email addresses help pr
        $username_err = "Username not permitted. Must use valid email address.";
    }
    elseif(!(str_ends($_POST['username'], '.com') || str_ends($_POST['username'], '.org') || str_ends($_POST['username'], '.e
            //This is basic error checking. We want a .com/.edu/.org email address. The string should end it those. (n.b. We
        $username_err = "Username not permitted. Must use valid email address.";
    }
    else{
        // Prepare a select statement
        $sql = "SELECT id FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);

            // Set parameters
            $param_username = trim($_POST["username"]);

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                /* store result */
                mysqli_stmt_store_result($stmt);

                if(mysqli_stmt_num_rows($stmt) == 1){
                    $username_err = "This email address is already taken.";
```

44: 1 Default text

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\register.php

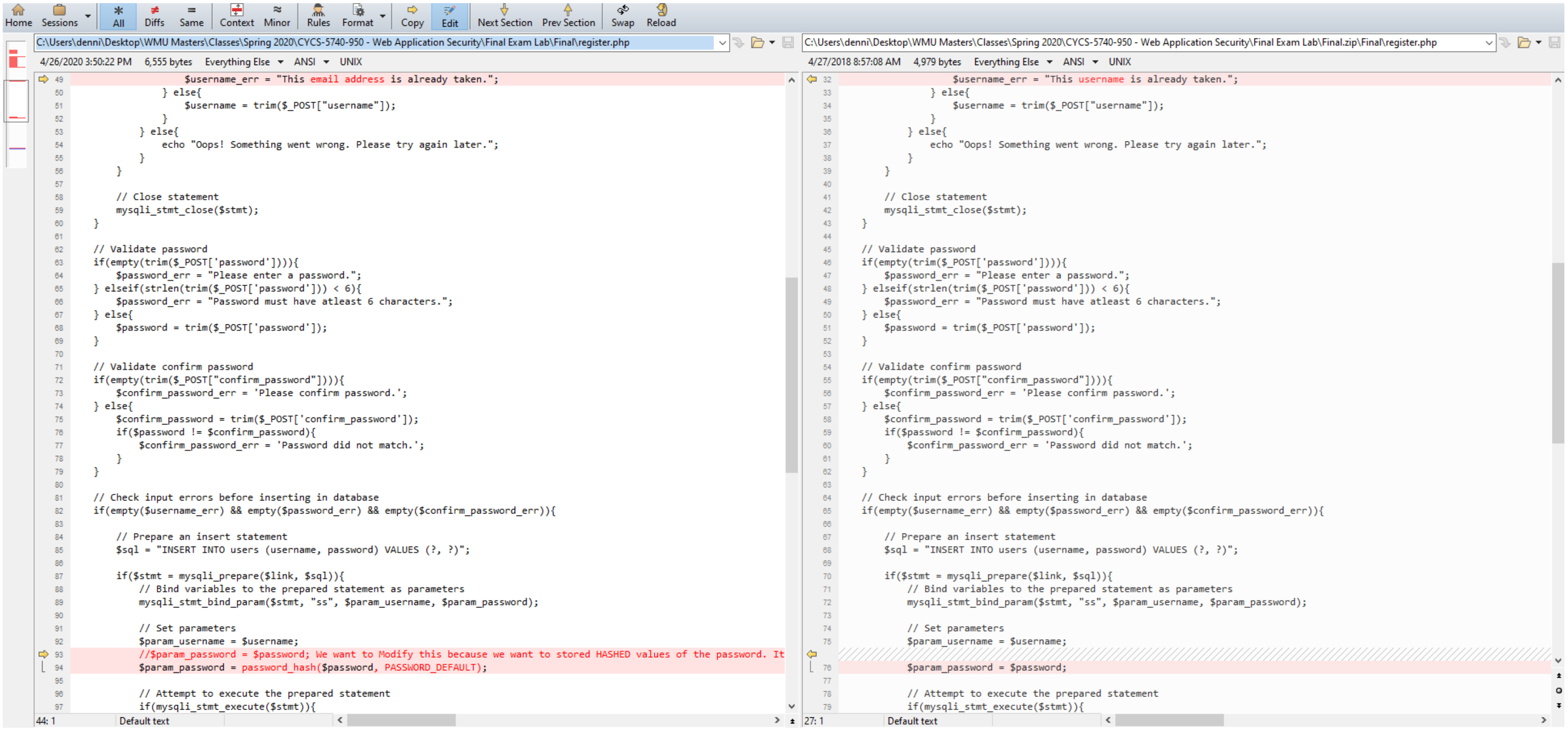
4/27/2018 8:57:08 AM 4,979 bytes Everything Else ANSI UNIX

```
<?php
// Include config file
require_once 'config.php';

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Validate username
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter a username.";
    } else{
        // Prepare a select statement
        $sql = "SELECT id FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);

            // Set parameters
            $param_username = trim($_POST["username"]);

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                /* store result */
                mysqli_stmt_store_result($stmt);

                if(mysqli_stmt_num_rows($stmt) == 1){
                    $username_err = "This username is already taken.";
```

27: 1 Default text

```
            if(mysqli_stmt_execute($stmt)){
            if(mysqli_stmt_execute($stmt)){
```

5 difference section(s) Same Insert Load time: 0.04 seconds



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\register.php

4/26/2020 3:50:22 PM 6,555 bytes Everything Else ANSI UNIX

```
                $username_err = "This email address is already taken.";
            } else{
                $username = trim($_POST["username"]);
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
        }
    }

    // Close statement
    mysqli_stmt_close($stmt);
}

// Validate password
if(empty(trim($_POST['password']))){
    $password_err = "Please enter a password.";
} elseif(strlen(trim($_POST['password'])) < 6){
    $password_err = "Password must have atleast 6 characters.";
} else{
    $password = trim($_POST['password']);
}

// Validate confirm password
if(empty(trim($_POST["confirm_password"]))){
    $confirm_password_err = 'Please confirm password.';
} else{
    $confirm_password = trim($_POST['confirm_password']);
    if($password != $confirm_password){
        $confirm_password_err = 'Password did not match.';
    }
}

// Check input errors before inserting in database
if(empty($username_err) && empty($password_err) && empty($confirm_password_err)){

    // Prepare an insert statement
    $sql = "INSERT INTO users (username, password) VALUES (?, ?)";

    if($stmt = mysqli_prepare($link, $sql)){
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "ss", $param_username, $param_password);

        // Set parameters
        $param_username = $username;
        //$param_password = $password; We want to Modify this because we want to stored HASHED values of the password. It
        $param_password = password_hash($password, PASSWORD_DEFAULT);

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
```

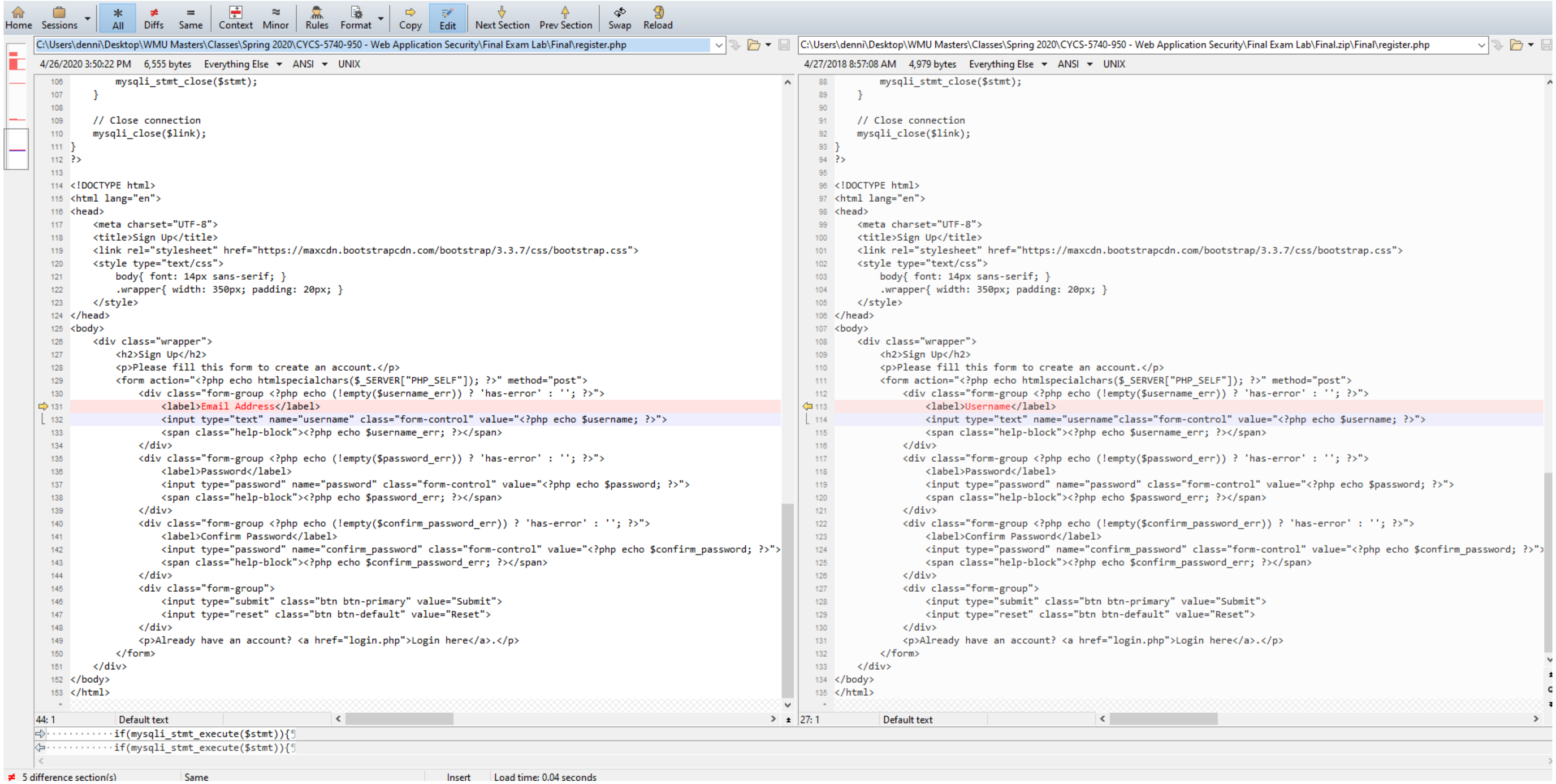
44: 1

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\register.php

4/27/2018 8:57:08 AM 4,979 bytes Everything Else ANSI UNIX

```
                $username_err = "This username is already taken.";
            } else{
                $username = trim($_POST["username"]);
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
        }
    }

    // Close statement
    mysqli_stmt_close($stmt);
}

// Validate password
if(empty(trim($_POST['password']))){
    $password_err = "Please enter a password.";
} elseif(strlen(trim($_POST['password'])) < 6){
    $password_err = "Password must have atleast 6 characters.";
} else{
    $password = trim($_POST['password']);
}

// Validate confirm password
if(empty(trim($_POST["confirm_password"]))){
    $confirm_password_err = 'Please confirm password.';
} else{
    $confirm_password = trim($_POST['confirm_password']);
    if($password != $confirm_password){
        $confirm_password_err = 'Password did not match.';
    }
}

// Check input errors before inserting in database
if(empty($username_err) && empty($password_err) && empty($confirm_password_err)){

    // Prepare an insert statement
    $sql = "INSERT INTO users (username, password) VALUES (?, ?)";

    if($stmt = mysqli_prepare($link, $sql)){
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "ss", $param_username, $param_password);

        // Set parameters
        $param_username = $username;

        $param_password = $password;

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
```

27: 1



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\register.php

4/26/2020 3:50:22 PM   6,555 bytes   Everything Else   ANSI   UNIX

```
        mysqli_stmt_close($stmt);
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Sign Up</h2>
        <p>Please fill this form to create an account.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Email Address</label>
                <input type="text" name="username" class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control" value="<?php echo $password; ?>">
                <span class="help-block"><?php echo $password_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($confirm_password_err)) ? 'has-error' : ''; ?>">
                <label>Confirm Password</label>
                <input type="password" name="confirm_password" class="form-control" value="<?php echo $confirm_password; ?>">
                <span class="help-block"><?php echo $confirm_password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary" value="Submit">
                <input type="reset" class="btn btn-default" value="Reset">
            </div>
            <p>Already have an account? <a href="login.php">Login here</a>.</p>
        </form>
    </div>
</body>
</html>
```

44: 1   Default text

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\register.php

4/27/2018 8:57:08 AM   4,979 bytes   Everything Else   ANSI   UNIX

```
        mysqli_stmt_close($stmt);
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; }
        .wrapper{ width: 350px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Sign Up</h2>
        <p>Please fill this form to create an account.</p>
        <form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group <?php echo (!empty($username_err)) ? 'has-error' : ''; ?>">
                <label>Username</label>
                <input type="text" name="username"class="form-control" value="<?php echo $username; ?>">
                <span class="help-block"><?php echo $username_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($password_err)) ? 'has-error' : ''; ?>">
                <label>Password</label>
                <input type="password" name="password" class="form-control" value="<?php echo $password; ?>">
                <span class="help-block"><?php echo $password_err; ?></span>
            </div>
            <div class="form-group <?php echo (!empty($confirm_password_err)) ? 'has-error' : ''; ?>">
                <label>Confirm Password</label>
                <input type="password" name="confirm_password" class="form-control" value="<?php echo $confirm_password; ?>">
                <span class="help-block"><?php echo $confirm_password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary" value="Submit">
                <input type="reset" class="btn btn-default" value="Reset">
            </div>
            <p>Already have an account? <a href="login.php">Login here</a>.</p>
        </form>
    </div>
</body>
</html>
```

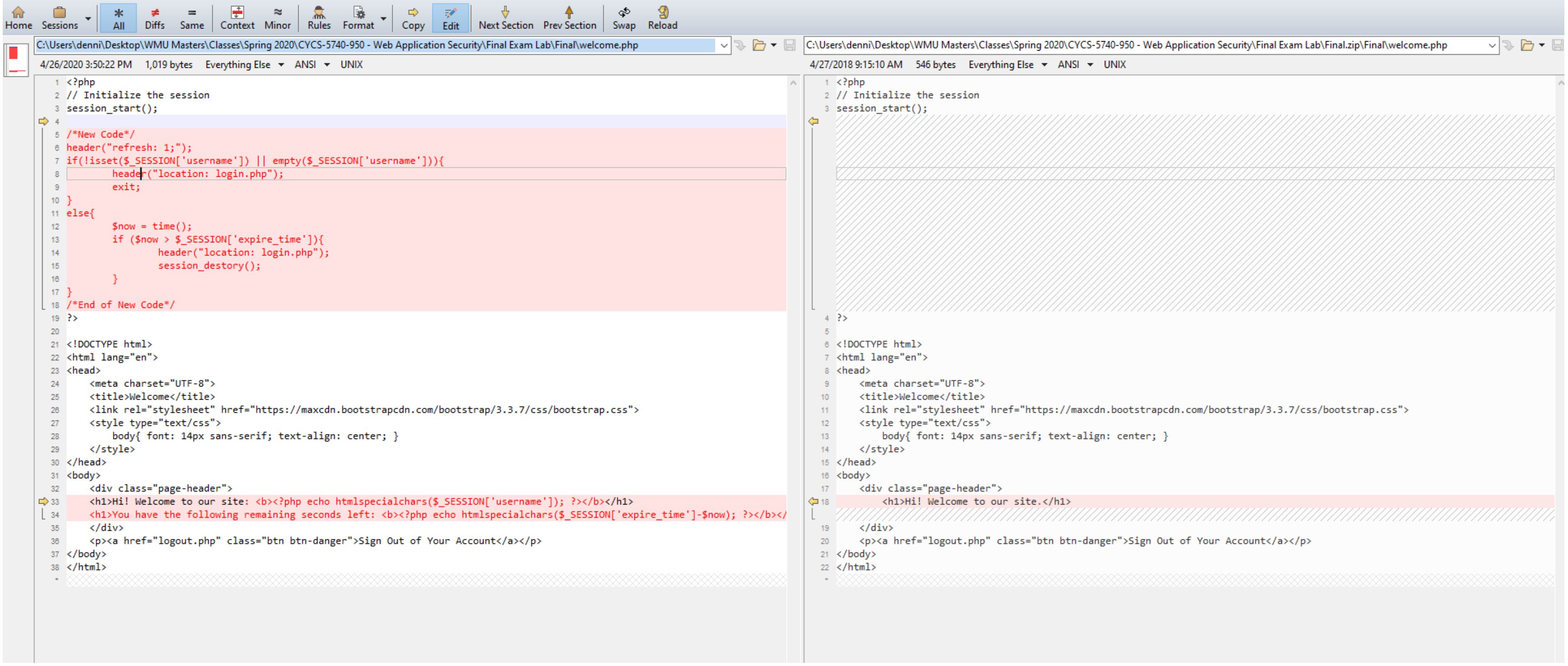
27: 1   Default text

```
············if(mysqli_stmt_execute($stmt)){¶
············if(mysqli_stmt_execute($stmt)){¶
```

5 difference section(s)   Same   Insert   Load time: 0.04 seconds

Appendix D

Compare Welcome.php



C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final\welcome.php

4/26/2020 3:50:22 PM 1,019 bytes

```
<?php
// Initialize the session
session_start();

/*New Code*/
header("refresh: 1;");
if(!isset($_SESSION['username']) || empty($_SESSION['username'])){
        header("location: login.php");
        exit;
}
else{
        $now = time();
        if ($now > $_SESSION['expire_time']){
                header("location: login.php");
                session_destory();
        }
}
/*End of New Code*/
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
    <h1>Hi! Welcome to our site: <b><?php echo htmlspecialchars($_SESSION['username']); ?></b></h1>
    <h1>You have the following remaining seconds left: <b><?php echo htmlspecialchars($_SESSION['expire_time']-$now); ?></b></
    </div>
    <p><a href="logout.php" class="btn btn-danger">Sign Out of Your Account</a></p>
</body>
</html>
```

C:\Users\denni\Desktop\WMU Masters\Classes\Spring 2020\CYCS-5740-950 - Web Application Security\Final Exam Lab\Final.zip\Final\welcome.php

4/27/2018 9:15:10 AM 546 bytes

```
<?php
// Initialize the session
session_start();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.css">
    <style type="text/css">
        body{ font: 14px sans-serif; text-align: center; }
    </style>
</head>
<body>
    <div class="page-header">
        <h1>Hi! Welcome to our site.</h1>
    </div>
    <p><a href="logout.php" class="btn btn-danger">Sign Out of Your Account</a></p>
</body>
</html>
```