CPS 475/575 Secure Application Development

Team 13 Final Project - miniFacebook

Dr. Phu Phung

Christopher Bussen - 101657219 - bussenc1@udayton.edu

Joe Durham - 101613997 - durhamj4@udayton.edu

Bitbucket URL: https://bitbucket.org/secad-team-project/secad-s22-team13-project/src/master/

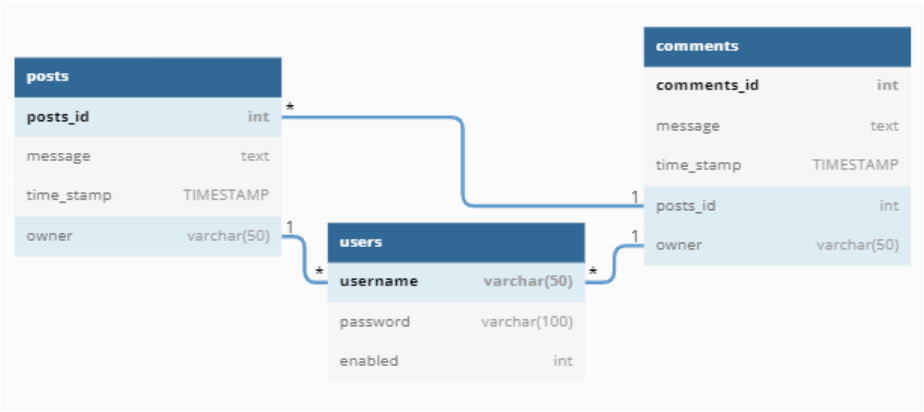
1. Introduction

Link to repository: https://bitbucket.org/secad-team-project/secad-s22-team13-project/src/master/
Link to video demo recording: https://udayton.zoom.us/rec/share/boZqnYL8mFMfH8AzxovEkbp2Lf5F5oSVoggudaALIt9CCF6U6neMgHLQiHn9zuHL.0rwb3U_pNS-5XT1Z?startTime=1651865470000 -- note: forgot to show current chat functionality in demo but as you can see, the basis is present.

In this project, we have implemented a site that allows users to create a miniFacebook account using a username and password. Users can create a new account starting from the login homepage (form.php) - - the username must be in the form of an email and the password must meet the length and character requirements provided. Additionally, users are required to retype their password when creating an account. After logging in, users will see a welcome message with their username as well as the title of the site. One of the first functionalities we added on this page was a link which allows users to change their password and it will be updated in the database, allowing them to login with the new password later. Logged in users are also able to create posts which are stored in the database, and these posts will have a time stamp and the user's name next to them for other logged in users to see whenever they reload the page. There is a spot on the homepage for users to create these posts as well as a separate page dedicated solely to making posts. Additionally, an admin account has been created with the role of superuser, whereas every other user is automatically created with the role of a regular user (unless manually added to the database as a superuser). Superuser's have the exclusive ability to view a list of all of the registered users in the system. There is also a page for users to chat with other online and logged in users - this page utilizes the code from lab 8 in class. Finally, there is a link for users to be able to easily logout of their account, and after hitting this link another link appears that directs users back to the login page.

2. Design

Database Entity Relationship Diagram



To start, we created a mysql database on our local systems called secadteam13 that would be used to store all of the necessary data. In this database, we have three tables - one named users, another named posts, and the last one for comments. In the users table, there are three variables, these being username (used to keep track of an account's username, made up of characters), password (used to keep track of an account's password, made up of characters), and enabled (an integer meant to keep track of whether or not a certain user is a superuser or a regular user). When the account passwords are stored, the password()function is used to encrypt/hash the password. The posts table is made up of four variables: posts_id (an integer that automatically increments), message (made up of characters in a text field), time_stamp (the date and time that the post was created), and owner (a set of characters that is inherited from the username of the user that creates the post). Finally, the comments tables has five variables: comments_id (another integer that automatically increments), message (as mentioned before), time_stamp (as mentioned before), posts_id (inherited from the post the comment is created on), and owner (inherited from the username of the user that creates the comment). Additionally, we included the login information for our team admin account in the database.sql file so that it is automatically inserted into the users table. As mentioned earlier, the roles of regular users and superusers are separated using the int enabled variable in the database. This number will tell the system what role the user should have, and thus what abilities each user's account should have access to. Additionally, we have hardcoded our team admin account as a superuser for ease of use.

3. Implementation & security analysis

In this project, we apply the security programming principles we learned in class in several ways, especially by using defense in depth, defense in breadth, and robust programming in addition to following several of the principles we learned in class. Some of these principles followed include (and are not limited to) the least privilege principle, which we did by only allowing users minimal permissions (can be seen through regular and super users). We also followed the economy of mechanism as we kept everything as simple and small as we could. As mentioned before, we also used defense in depth/breadth - this is because there are many different layers of security in the project and we have tested the use cases of the system to make sure they are secure and will stay that way throughout the life cycle. The first layer provided is that the system is deployed on HTTPS, thus making the data that is sent from system encrypted. The next and first layer noticed by the user is the login system - a user must first have a valid username and password recognized by the database in order to login. Additionally, we use sessions to check if a user is already logged in when they try to access a page, and if they are not, then they are redirected to login. This ensures that someone who has the site link can't access the site without credentials. There are also several other defense mechanism that defend against other common attacks (these will be mentioned later). The first database security principle we have used in our project is that we did not use root access for modifying the project's database. Instead, we created a new database user and granted it access to the necessary database so that other databases are safe if this system is successfully attacked. Another important database security principle we used is hashing and encryption in the database. We implemented this by using the built-in password() function that encrypts the user's password when storing it in the database. This way, the password is not stored in plaintext within the database, making it harder for an attacker to view the data even after a successful attack. Moving on, our code is robust and defensive as it is able to handle bad inputs without the system being harmed or crashing. This is done by checking all of the possible user inputs and making sure that they are not empty/invalid. This input validation is done through things such as regular expressions (like with creating a new user) as well as different conditional statements within the code. If an input is found to be invalid, the system will alert the user and redirect them to a new page without crashing. Additionally, the code is kept simple as we tried to break it down into smaller units that can each be properly defended. Additionally, all conditions are checked in order to prevent uncommon and unpredicted outcomes. Our project also defends against many of the known attacks. It uses sanitized HTML outputs to prevent

cross-site scripting attacks. Additionally it implements prepared statements to protect against SQL injection attacks. The project was also deployed on HTTPS, meaning that any data in transactions is encrypted and protected. Session hijacking was prevented setting the limit lifetime for the cookies as well as the secure and httponly flags. Furthermore, the browser information is stored in the session and this is checked whenever the session is checked. This prevents the attacker from using a different browser. Finally, CSRF attacks are prevented by generating a randomized secret token in the changepasswordform.php file and the changepassword.php file later checks for the presence of this same token in the session. As a result, the attacker is not able to fully construct a valid request (unless they manage to guess the token value). Finally, the roles of regular users and superusers are separated using a variable in the users table in the database which indicates a specific user's role. If this variable indicates superuser, then the user will have access to the superuser functionality - otherwise, they will not.

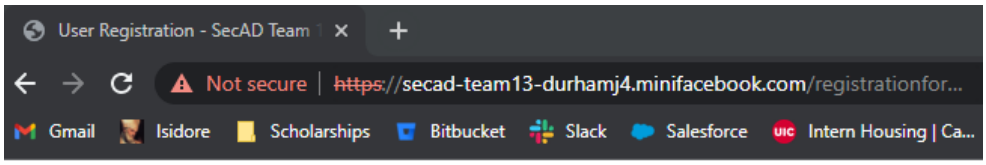
4. Demo (screenshots)

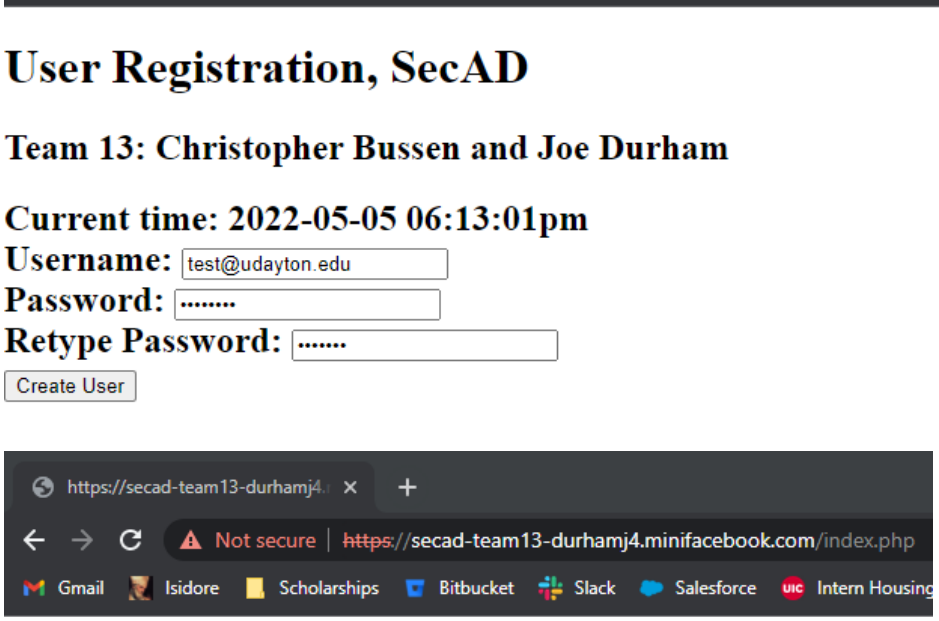
Login Form

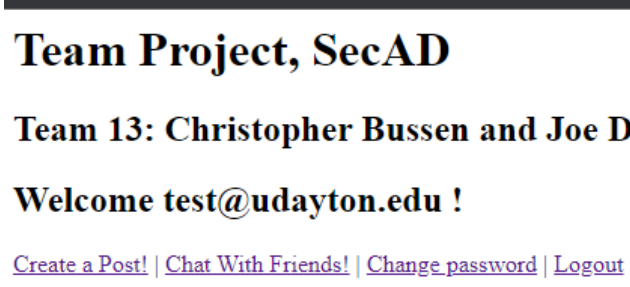
The user must enter valid credentials, such as a valid email and a password containing 8 characters, 1 special character, 1 number, 1 uppercase, and 1 lowercase. Only registered users can login; the applciation checks with the database.

New User Login

For new users, they must enter new credentials. The database is also affected after the new account is created.







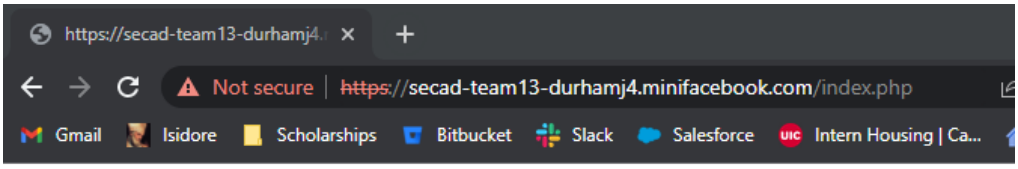
```
mysql> select * from users;
+---------------------+-------------------------------------------+---------+
| username            | password                                  | enabled |
+---------------------+-------------------------------------------+---------+
| admin               | *4FD3A5CBFB58BD0D84393E18655EC11A74F0BA8C |       1 |
| durhamj4@udayton.edu | *2255A90A3039268D9FEC1518C7DB627D2F97AB4F |    NULL |
+---------------------+-------------------------------------------+---------+
2 rows in set (0.00 sec)

mysql> select * from users;
+---------------------+-------------------------------------------+---------+
| username            | password                                  | enabled |
+---------------------+-------------------------------------------+---------+
| admin               | *4FD3A5CBFB58BD0D84393E18655EC11A74F0BA8C |       1 |
| durhamj4@udayton.edu | *2255A90A3039268D9FEC1518C7DB627D2F97AB4F |    NULL |
| test@udayton.edu    | *48B1BB7AD34484EF0632D4B9A748CC861DFBE88B |    NULL |
+---------------------+-------------------------------------------+---------+
3 rows in set (0.00 sec)
```

Home Page

On the home page, logged in users can create posts and change their password. There is a different home page for admin and non-admin users.

Non-Admin:



Team Project, SecAD

Team 13: Christopher Bussen and Joe Durham

Welcome test@udayton.edu !

Create a Post! | Chat With Friends! | Change password | Logout

What do you want to post?

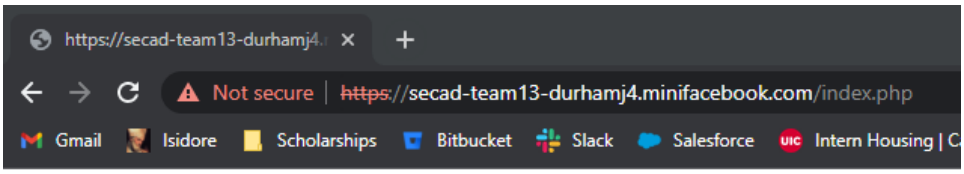
Post

Posts:

durhamj4@udayton.edu posted @ 2022-05-05 15:06:51:

hello

Admin:



Team Project, SecAD

Team 13: Christopher Bussen and Joe Durham

Welcome admin !

Create a Post! | Chat With Friends! | Change password | Logout

View List of Registered Users

What do you want to post?

Post

Posts:

test@udayton.edu posted @ 2022-05-05 18:21:32:

separate page to post

test@udayton.edu posted @ 2022-05-05 18:20:00:

hello from test user for demo

durhamj4@udayton.edu posted @ 2022-05-05 15:06:51:

hello

Change Password

Logged in users have the ability to change their password. Their results are seen after changing.

Change Password, SecAD

Team 13: Christopher Bussen and Joe Durham

Username: test@udayton.edu
New Password: ••••••••
Change Password

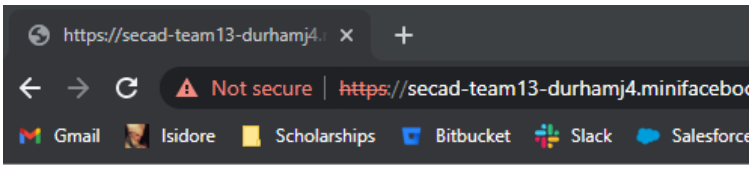
DEBUG:changepassword.php->Got: username=test@udayton.edu;newpassword=Test123!
DEBUG>prepared_sql= UPDATE users SET password=password(?) WHERE username= ?;

The new password has been set.

Home | Logout

Create Posts

Logged in users have the ability to post and view posts of other users. The posts are displayed from the database, with the most recent post at the top of the page. Users can ether post directly from the homepage, or they can post in the 'post' page.



Team Project, SecAD

Team 13: Christopher Bussen and Joe Durham

Welcome test@udayton.edu !

Create a Post! | Chat With Friends! | Change password | Logout

What do you want to post?

Post

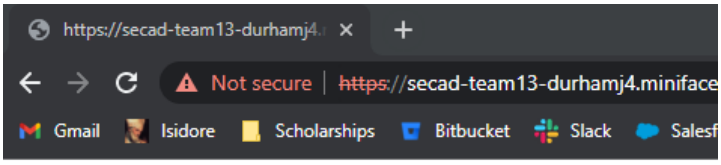
Posts:

test@udayton.edu posted @ 2022-05-05 18:20:00:

hello from test user for demo

durhamj4@udayton.edu posted @ 2022-05-05 15:06:51:

hello



What do you want to post?

separate page to post

Post

Team Project, SecAD

Team 13: Christopher Bussen and Joe Durham

Welcome test@udayton.edu !

Create a Post! | Chat With Friends! | Change password | Logout

What do you want to post?

Post

Posts:

test@udayton.edu posted @ 2022-05-05 18:21:32:

separate page to post

test@udayton.edu posted @ 2022-05-05 18:20:00:

hello from test user for demo

durhamj4@udayton.edu posted @ 2022-05-05 15:06:51:

hello

Chat With Friends

This simple real time chat shows when users are trying, and it shows the username and their message.

Home | Logout
Current time:
Thu May 05 2022 18:23:08 GMT-0400 (Eastern Daylight Time)
Type message and enter to send:
Message from server:

List of Registered Users

The admin is able to see the list of all registered users from the database with a simple click in the application.

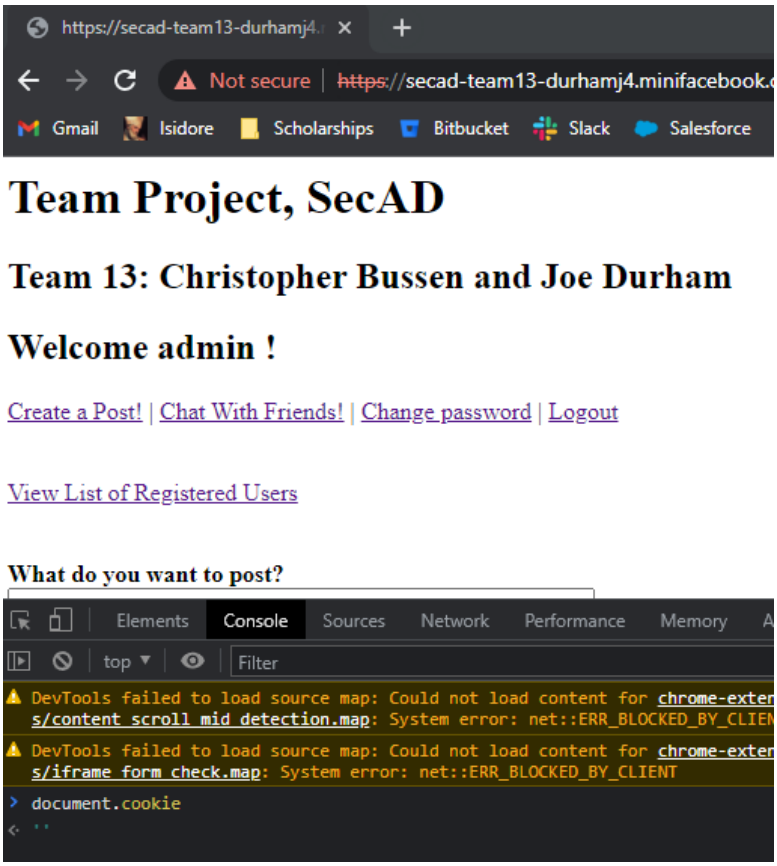
List of Registered Users, for Admin only

admin
durhamj4@udayton.edu
test@udayton.edu

Home

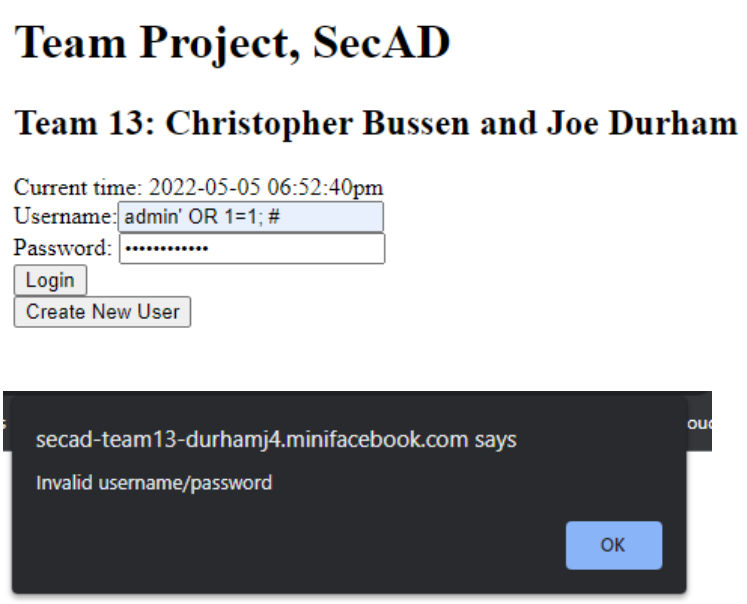
Hiding Cookies

The cookies for each session are hidden. This helps in preventing session hijacking.



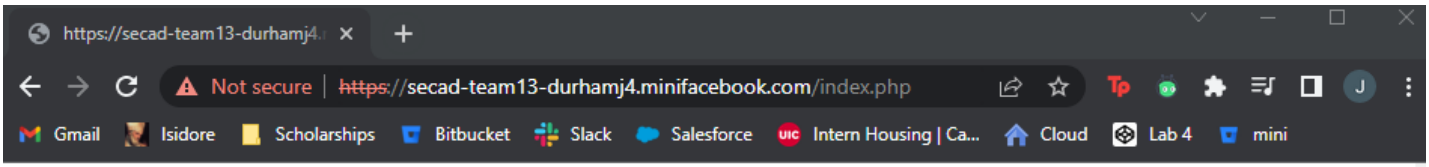
SQLi Attack Prevention

SQL Injection attacks are prevented against. The system does not allow SQL to be inject becuase of the prepared SQL statements that have been implemented.



Further Session Hijacking Prevention

Since the session ID is traced, it makes it even harder for a session hijacking attack to happen. If a cookie is inserted into a different browser, the application does not allow it to login.



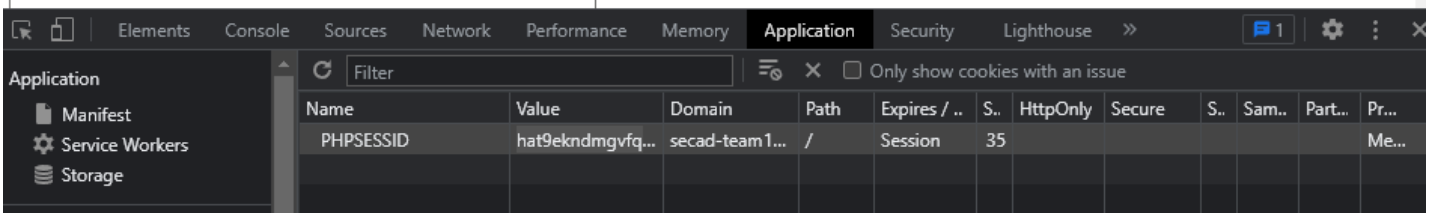
Team Project, SecAD

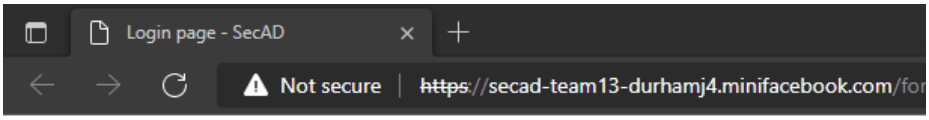
Team 13: Christopher Bussen and Joe Durham

Welcome durhamj4@udayton.edu !

Create a Post! | Chat With Friends! | Change password | Logout

What do you want to post?

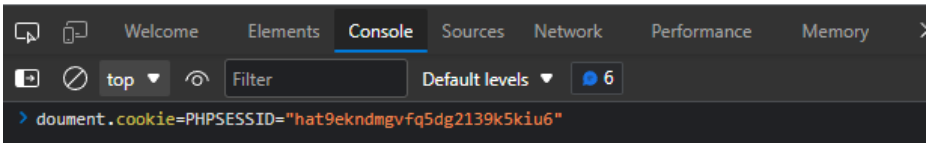




Team Project, SecAD

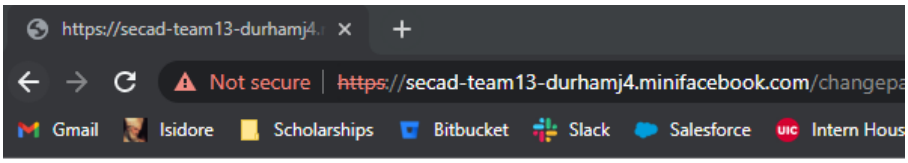
Team 13: Christopher Bussen and Joe Durham

Current time: 2022-05-05 06:57:31pm
Username:
Password:
Login
Create New User



XSS and CSRF Attacks

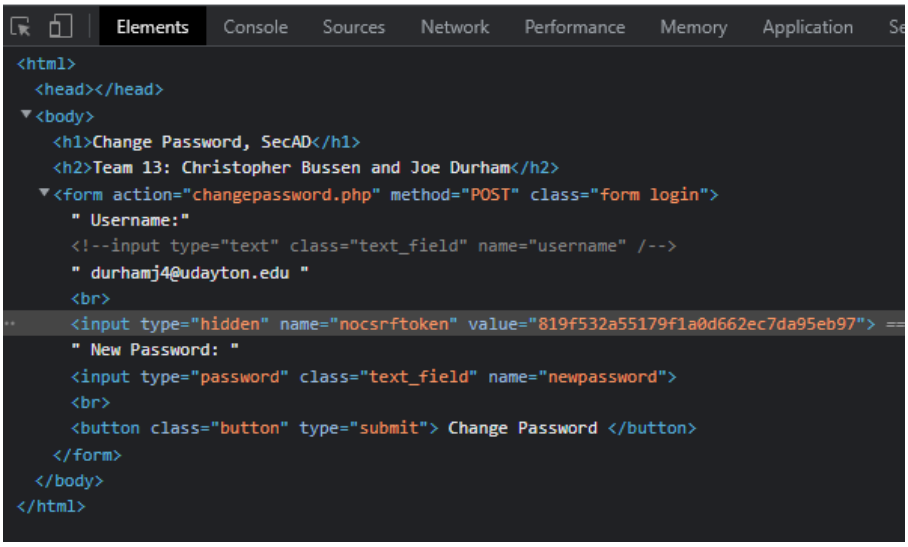
The application is protected against attacks becuase of the 'nocsrftoken' value that is changed every few seconds, or if a user refreshes the page. The images show a user refreshing the page, and the value changing.

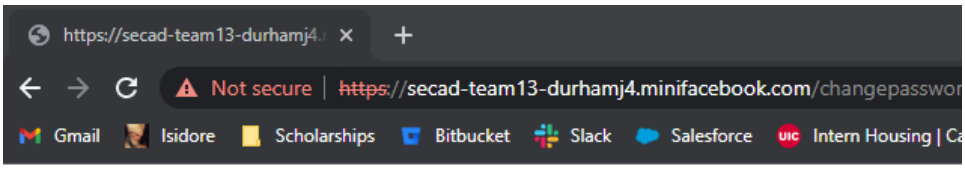


Change Password, SecAD

Team 13: Christopher Bussen and Joe Durham

Username: durhamj4@udayton.edu
New Password:
Change Password

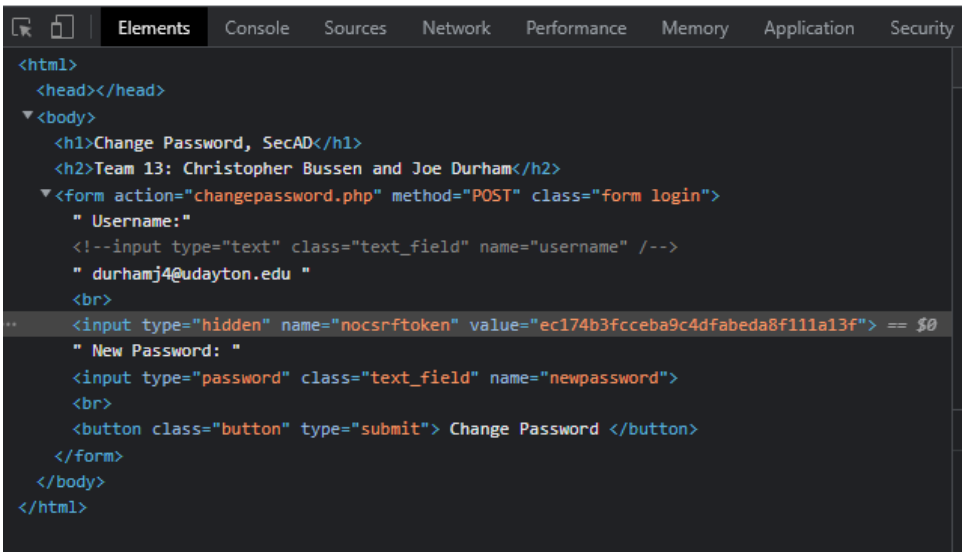




Change Password, SecAD

Team 13: Christopher Bussen and Joe Durham

Username: durhamj4@udayton.edu
New Password:
Change Password



Appendix

database.sql:

```
 -- if the table exists, delete it
DROP TABLE IF EXISTS `users`;
DROP TABLE IF EXISTS `posts`;

-- create a new table
CREATE TABLE users(
    username varchar(50) PRIMARY KEY,
    password varchar(100) NOT NULL
);

CREATE TABLE `posts`(
    message text,
    created_at datetime,
    `owner` varchar(50),
    FOREIGN KEY (`owner`) REFERENCES `users`(`username`) ON DELETE CASCADE
);

-- insert data to the table users
LOCK TABLES `users` WRITE;
INSERT INTO `users` VALUES ('admin',password('team13Admin'));
UNLOCK TABLES;
```

addnewuser.php:

```
<?php
    require "database.php";
    $username = $_POST["username"];
    $password = $_POST["password"];

    function addnewuser($username, $password) {
        global $mysqli;
        $prepared_sql = "INSERT INTO users SET username=?,password=PASSWORD(?);";
        echo "DEBUG>prepared_sql= $prepared_sql\n";
        if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
        $stmt->bind_param("ss", $username, $password);
        if(!$stmt->execute()) return FALSE;
        return TRUE;
    }

    if(isset($username) AND isset($password)){
      echo "DEBUG:addnewuser.php->Got: username=$username;password=$password\n<br>";
      if(addnewuser($username,$password)){
        echo "<h4>The new user has been created.</h4>";
      }else{
        echo "<h4>Error: Cannot create the user.</h4>";
      }
    }else{
      echo "No provided username/password to create.";
      exit();
    }
?>
<a href="index.php">Home</a> | <a href="logout.php">Logout</a> | <a href="changepasswordform.php">  Change Password</a>
```

changepassword.php:

```
<?php
  require "session_auth.php";
  require "database.php";
  $username = $_SESSION["username"];
  $newpassword = $_POST["newpassword"];
  $nocsrftoken = $_POST["nocsrftoken"];
  if(!isset($nocsrftoken) or ($nocsrftoken!=$_SESSION['nocsrftoken'])){
    echo "<script>alert('Cross-site request forgery is detected!');</script>";
    header("Refresh:0; url=logout.php");
    die();
  }
  if(isset($username) AND isset($newpassword)){
    echo "DEBUG:changepassword.php->Got: username=$username;newpassword=$newpassword\n<br>";
    if(changepassword($username,$newpassword)){
      echo "<h4>The new password has been set.</h4>";
    }else{
      echo "<h4>Error: Cannot change the password.</h4>";
    }
  }else{
    echo "No provided username/password to change.";
    exit();
  }
?>
<a href="index.php">Home</a> | <a href="logout.php">Logout</a>
```

changepasswordform.php:

```
<?php
  require "session_auth.php";
  $rand= bin2hex(openssl_random_pseudo_bytes(16));
  $_SESSION["nocsrftoken"] = $rand;
?>
<html>
        <h1>Change Password, SecAD</h1>
        <h2>Team 13: Christopher Bussen and Joe Durham</h2>

          <form action="changepassword.php" method="POST" class="form login">
                Username:<!--input type="text" class="text_field" name="username" /-->
                <?php echo htmlentities($_SESSION["username"]); ?>
                <br>
                <input type="hidden" name="nocsrftoken" value="<?php echo $rand; ?>"/>
                New Password: <input type="password" class="text_field" name="newpassword" /> <br>
                <button class="button" type="submit">
                  Change Password
                </button>
          </form>
</html>
```

client.html:

```
<!DOCTYPE html>
<html lang="en">
<meta charset="utf-8">
<script src="https://secad-team13-durhamj4.minifacebook.com:4430/socket.io/socket.io.js"></script>
<script>
function startTime() {
    document.getElementById('clock').innerHTML = new Date();
    setTimeout(startTime, 500);
}
if (window.WebSocket) {
 console.log("HTML5 WebSocket is supported");
} else {
  alert("HTML5 WebSocket is not supported");
}
var myWebSocket = io.connect("https://secad-team13-durhamj4.minifacebook.com:4430");
myWebSocket.onopen = function() {
    console.log('WebSocket opened');
}
myWebSocket.on("message", function(msg) {
    console.log('Received from server: '+ msg);
    document.getElementById("receivedmessage").innerHTML += sanitizeHTML(msg) + "<br>";
});
myWebSocket.on("typing", function(msg) {
    document.getElementById("typing").innerHTML = "Someone is typing...<br>";
    setTimeout(function(){document.getElementById("typing").innerHTML = "<br>";},500);
});
myWebSocket.onclose = function() {
    console.log('WebSocket closed');
}

function doSend(msg){
    if (myWebSocket) {
        myWebSocket.emit("message", msg);
        console.log('Sent to server: ' +msg);
    }
}
function entertoSend(e){
  //alert("keycode =" + e.keyCode);
  if(e.keyCode==13){//enter key
    var username = "<?php echo $_SESSION['username'] ?>";
    doSend(username + ":" + document.getElementById("message").value);
    document.getElementById("message").value = "";
  }
}

var sanitizeHTML = function (str) {
  var temp = document.createElement('div');
  temp.textContent = str;
  return temp.innerHTML;
};
</script>

<body onload="startTime()">
Current time: <div id="clock"></div>

Type message and enter to send: <input type = "text" id="message" size = "30" onkeypress="entertoSend(event)" onkeyup="myWebSocket.emit('typi
<br>
<div id = "typing"></div>
Message from server:
<hr>
<div id = "receivedmessage"></div>



</body>
</html>
```

client.php:

```
<?php
    require("session_auth.php");
    $username = $_SESSION["username"];
?>

<!DOCTYPE html>
<html lang="en">
<meta charset="utf-8">
<script src="https://secad-team13-durhamj4.minifacebook.com:4430/socket.io/socket.io.js"></script>
<script>

var username = <?php echo json_encode($username); ?>;

function startTime() {
    document.getElementById('clock').innerHTML = new Date();
    setTimeout(startTime, 500);
}
if (window.WebSocket) {
 console.log("HTML5 WebSocket is supported");
} else {
  alert("HTML5 WebSocket is not supported");
}
var myWebSocket = io.connect('https://secad-team13-durhamj4.minifacebook.com:4430');
myWebSocket.onopen = function() {
    console.log('WebSocket opened');
}

myWebSocket.on("message", function(msg) {
    console.log('Received from server: '+ msg);
    document.getElementById("receivedmessage").innerHTML += sanitizeHTML(msg) + "<br>";
});
myWebSocket.on("typing", function(username) {
    console.log(name);
    document.getElementById("typing").innerHTML = sanitizeHTML(name) + " is typing... <br>";
    setTimeout(function(){document.getElementById("typing").innerHTML = "<br>";},2000);
});

myWebSocket.onclose = function() {
    console.log('WebSocket closed');
}

function doSend(msg){
    if (myWebSocket) {
        msg = username + ": " + msg;
        myWebSocket.emit("message", msg);
        console.log('Sent to server: ',  msg);
    }
}

function entertoSend(e){
  //alert("keycode =" + e.keyCode);
  if(e.keyCode==13){//enter key
    var username = "<?php echo $_SESSION['username'] ?>";
    doSend(username + ":" + document.getElementById("message").value);
    document.getElementById("message").value = "";
  }
}

var sanitizeHTML = function (str) {
  var temp = document.createElement('div');
  temp.textContent = str;
  return temp.innerHTML;
};
</script>

<a href="index.php">Home</a> |
<a href="logout.php">Logout</a>
<br>

<body onload="startTime()">
```

```
Current time: <div id="clock"></div>

Type message and enter to send: <input type = "text" id="message" size = "30" onkeypress="entertoSend(event)" onkeyup="myWebSocket.emit('typi
<br>
<div id = "typing"></div>
Message from server:
<hr>
<div id = "receivedmessage"></div>



</body>
</html>
```

createcomment.php:

```
<?php
    require "session_auth.php";
    require "database.php";
    $username=$_SESSION["username"];
    $usercomment=$_REQUEST["usercomment"];
    $post_id=$_GET['post_id'];

    if (isset($username) AND isset($usercomment) AND isset($post_id)) {
        if (!createcomment($username, $usercomment, $post_id)) {
            echo "<script>alert('The post was not stored in the database');</script>";
        }
    } else {
        echo "<script>alert('Error: cannot find user or post.');</script>";
        exit();
    }
    header("Refresh:0; url=index.php");
?>
```

createcommentform.php:

```
<?php
    require "session_auth.php";
    if(isset($_GET["post_id"])){
        $post_id =  $_GET["post_id"];
    }
?>

<html>
<body>
    <div id="commentform" style="display:inline;">
        <form action="createcomment.php?post_id=<?php echo htmlentities($post_id); ?>" method="POST" style="display: inline;">
            <textarea rows="1" cols="30" name="usercomment"></textarea>
            <button class="button" type="submit">
             Comment
            </button>
        </form>
    </div>

    <script>
        function showForm(){
            var ele = document.getElementById("commentform");
            if(ele.style.display === "none"){
                ele.style.display = "block";
            }else{
                ele.style.display = "none";
            }
        }
    </script>
    </body>
    </html>
```

createpost.php:

```
<?php
    require "session_auth.php";
    require "database.php";
    $username=$_SESSION["username"];
    $userpost=$_REQUEST["userpost"];

    if (isset($username) AND isset($userpost)) {
        if (!createpost($username, $userpost)) {
            echo "<script>alert('Error: The post was not stored in the database.');</script>";
        }
    } else {
        echo "<script>alert('Error: cannot find user or post.');</script>";
        exit();
    }
    header("Refresh:0; url=index.php");
?>
```

createpostform.php:

```
<?php
    require "session_auth.php";
?>
<html>
<body>
    <form action="createpost.php" method="POST" style="display:inline" >
        <b>What do you want to post?</b> </br>
        <textarea rows="5" cols="50" name="userpost"></textarea>
        <button class="button" type="submit">
          Post
        </button>
    </form>
</body>
</html>
</br>
```

database.php:

```
<?php
    $mysqli = new mysqli('localhost','admin','team13Admin','secadteam13');
    if($mysqli->connect_errno){
        printf("Database connection failed: %s\n", $mysqli->connect_error);
        exit();
    }

    function changepassword($username, $newpassword) {
        global $mysqli;
        $prepared_sql = "UPDATE users SET password=password(?) WHERE username= ?;";
        echo "DEBUG>prepared_sql= $prepared_sql\n";
        if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
        $stmt->bind_param("ss", $newpassword,$username);
        if(!$stmt->execute()) return FALSE;
        return TRUE;
    }


    function createpost($owner, $message) {
        global $mysqli;
        $time_stamp = date('Y-m-d H:i:s');
        $prepared_sql = "INSERT INTO posts(message, time_stamp, owner) VALUES (?, ?, ?);";
        if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
        $stmt->bind_param("sss", $message, $time_stamp, $owner);
        if(!$stmt->execute()) return FALSE;
        return TRUE;
    }
?>
```

enableDisbale.php:

```
<?php
    require "session_auth.php";
    require "database.php";
    $username = $_SESSION["username"];
    $checkedlist = $_POST["checkedlist"];

    function userAbilities($checklist){
        global $mysqli;
        $prepared_sql = "UPDATE users SET enabled=0 WHERE username != 'admin';";
        if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
        if(!$stmt->execute()) return FALSE;

        foreach ($checklist as $check => $user) {
            $prepared_sql = "UPDATE users SET enabled=1 WHERE username=?;";
            if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
            $stmt->bind_param('s', $user);
            if(!$stmt->execute()) return FALSE;
        }
        return TRUE;
    }

    if (isset($username) AND isset($checkedlist)) {
        if (userAbilities($checkedlist)) {
            echo "<script>alert('The new enable/disable setting has been set.');</script>";
        } else {
            echo "<script>alert('Error: Cannot enable/disable the user');</script>";
        }
    } else {
        echo "<script>alert('You cannot disable every user from the website! Try again');</script>";
    }
    header("Refresh:0; url=index.php");
>
```

form.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Login page - SecAD</title>
</head>
<body>
        <h1>Team Project, SecAD</h1>
        <h2>Team 13: Christopher Bussen and Joe Durham</h2>

<?php
  //some code here
  echo "Current time: " . date("Y-m-d h:i:sa")
?>
          <form action="index.php" method="POST" class="form login">
                Username:<input type="text" class="text_field" name="username" /> <br>
                Password: <input type="password" class="text_field" name="password" /> <br>
                <button class="button" type="submit">
                  Login
                </button>
          </form>

          <form action="registrationform.php" method="POST" class="new user">
            <button class="button" type="submit">
              Create New User
            </button>
          </form>

</body>
</html>
```

index.php:

```
<?php
```

```
    $lifetime = 15 * 60;
    $path = "/";
    $domain = "*.minifacebook.com";
    $secure = TRUE;
    $httponly = TRUE;
    session_set_cookie_params($lifetime, $path, $domain, $secure, $httponly);
    session_start();

    $mysqli = new mysqli('localhost','admin','team13Admin','secadteam13');
    if($mysqli->connect_errno){
        printf("Database connection failed: %s\n", $mysqli->connect_error);
        exit();
    }

    if (isset($_POST["username"]) and isset($_POST["password"]) ) {
        if (securechecklogin($_POST["username"],$_POST["password"])) {
            $_SESSION["logged"] = TRUE;
            $_SESSION["username"] = $_POST["username"];
            $_SESSION["browser"] = $_SERVER["HTTP_USER_AGENT"];
        }else{
            echo "<script>alert('Invalid username/password');</script>";
            session_destroy();
            header("Refresh:0; url=form.php");
            die();
        }
    }
    if(!isset($_SESSION["logged"]) or $_SESSION["logged"] != TRUE) {
        echo "<script>alert('You have not logged in. Please login first');</script>";
        header("Refresh:0; url=form.php");
        die();
    }
    if($_SESSION["browser"] != $_SERVER["HTTP_USER_AGENT"]){
        echo "<script>alert('Session hijacking detected!');<script>";
        header("Refresh:0; url=form.php");
        die();
    }
?>
    <h1>Team Project, SecAD</h1>
    <h2>Team 13: Christopher Bussen and Joe Durham</h2>
    <h2> Welcome <?php echo htmlentities($_SESSION['username']); ?> !</h2>
    <a href="createpostform.php">Create a Post!</a> |
    <a href="client.php">Chat With Friends!</a> |
    <a href="changepasswordform.php">Change password</a> |
    <a href="logout.php">Logout</a> <br><br><br>

<?php
    if(strcmp($_SESSION["username"], "admin") == 0) {
?>
    <a href="listOfUsers.php">View List of Registered Users</a> <br><br><br>
<?php
    }
?>

<?php

    require("createpostform.php");
    echo "<h2> Posts: </h2>";

    // Display all the contents of the "posts" table
    global $mysqli;
    $prepared_sql = "SELECT posts_id, message, time_stamp, owner FROM posts ORDER BY posts_id DESC;";
    if(!$stmt = $mysqli->prepare($prepared_sql)) echo "Prepared Statement Error";
    if(!$stmt->execute()) echo "Execute Error";
    $posts_id = NULL; $message = NULL; $time_stamp = NULL; $owner = NULL;
    if(!$stmt->bind_result($posts_id, $message, $time_stamp, $owner)) echo "Binding failed";

?>

<?php
```

```
    //Getting all the posts in the database to display
    while($stmt->fetch()){
?>

            <h4><b><?php echo htmlentities($owner)?></b>
            posted @ <?php echo htmlentities($time_stamp)?>:</h4>
                <?php echo htmlentities($message)?>
            <br>
<?php
    }
?>


<?php

    function securechecklogin($username, $password) {
        global $mysqli;
        $prepared_sql = "SELECT * FROM users WHERE username= ?  AND password=password(?);";
        if(!$stmt = $mysqli->prepare($prepared_sql))
            echo "Prepared Statement Error";
        $stmt->bind_param("ss", $username,$password);
        if(!$stmt->execute()) echo "Execute Error";
        if(!$stmt->store_result()) echo "Store_result Error";
        $result = $stmt;
        if($result->num_rows ==1)
            return TRUE;
        return FALSE;
    }
?>
```

listOfUsers.php:

```
<?php
  require "session_auth.php";
  require "database.php";
  $nocsrftoken = $_POST["nocsrftoken"];

?>

<html>

 <table id="table" style="width:40%" align="center">
     <tr>
          <th><h2><b>List of Registered Users, for Admin only</b></h2></th>
     </tr>
     <tr>

     <?php

            $prepared_sql = "select username from users;";
            if(!$stmt = $mysqli->prepare($prepared_sql)){
                return FALSE;
            }
            if(!$stmt->execute()) return false;
            $username = NULL;
            if(!$stmt->bind_result($username)) echo "Binding failed";

            while($stmt->fetch()){
            ?>
                 <td><?php echo htmlentities($username)?></td>
        </tr>
        <?php
            }
        ?>

        </table>
        <a href="index.php">Home</a>
        </html>
```

logout.php:

```
<?php
    session_start();
    session_destroy();
?>
<p> You are logged out! </p>

<a href="form.php">Login again</a>
```

registrationform.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>User Registration - SecAD Team 13</title>
</head>
<body>
        <h1>User Registration, SecAD</h1>
        <h2>Team 13: Christopher Bussen and Joe Durham<h2>

<?php
  //some code here
  echo "Current time: " . date("Y-m-d h:i:sa")
?>
          <form action="addnewuser.php" method="POST" class="form login">
                Username: <input type="text" class="text_field" name="username" required pattern="^[\w.-]+@[\w-]+(.[\w-]+)*$"
                title="Please enter a valid email address as your username."
                placeholder="Your email address..."
                  onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/> <br>
                Password: <input type="password" class="text_field" name="password"
                required pattern="^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#$%^&])[\w!@#$%^&]{6,}$"
                placeholder="Your password..."
                title="Password must have at least 6 characters with 1 special symbol !@#$%^& 1 number, 1 lowercase, and 1 UPPERCASE"
                onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: ''); form.repassword.pattern = this.value;"/> <br>
                Retype Password: <input type="password" class="text_field" name="repassword"
                placeholder="Retype your password..." required
                title="Password does not match"
                onchange="this.setCustomValidity(this.validity.patternMismatch?this.title: '');"/> <br>
                <button class="button" type="submit">
                  Create User
                </button>
          </form>
</body>
</html>
```

session_auth.php:

```php
<?php
    $lifetime = 15 * 60;
    $path = "/";
    $domain = "*.minifacebook.com";
    $secure = TRUE;
    $httponly = TRUE;
    session_set_cookie_params($lifetime, $path, $domain, $secure, $httponly);
    session_start();

    if( !isset($_SESSION["logged"]) or $_SESSION["logged"] != TRUE){
        //the session is not authenticated
         echo "<script>alert('You have to login first!');</script>";
         session_destroy();
         header("Refresh:0; url=form.php");
         die();
        }

    if( $_SESSION["browser"] != $_SERVER["HTTP_USER_AGENT"]){
        //it is a session hijacking attack since it comes from a different browser
         echo "<script>alert('Session hijacking attack is detected!');</script>";
         session_destroy();
         header("Refresh:0; url=form.php");
         die();
        }
    ?>
```

socetio-chatserver.js:

```
/* A Simple HTTPS server with socket.IO in Node.js
   by Phu Phung for SecAD
   */
/* ensure that you have the key and certificate files
   copied to the /etc/ssl folder as in Lab 6.
   change the file name accordingly.*/
var xssfilter = require("xss");
var https = require('https'), fs = require('fs');
var sslcertificate  = {
  key: fs.readFileSync('/etc/ssl/secad.key'),
  cert: fs.readFileSync('/etc/ssl/secad.crt') //ensure you have these two files
};
var httpsServer = https.createServer(sslcertificate,httphandler);
var socketio = require('socket.io')(httpsServer);

httpsServer.listen(4430); //cannot use 443 as since it reserved for Apache HTTPS
console.log("HTTPS server is listenning on port 4430");

function httphandler (request, response) {
  //console.log("URL requested = " + request.url);
  //read the websocketchatclient.html file and
  //to create a HTTP Reponse regardless of the requests
  response.writeHead(200); // 200 OK
  var clientUI_stream = fs.createReadStream('./client.html');
  clientUI_stream.pipe(response);
}

socketio.on('connection', function (socketclient) {
  console.log("A new socket.IO client is connected: "+ socketclient.client.conn.remoteAddress+
              ":"+socketclient.id);

  socketclient.on("message", (data) => {
    var data = xssfilter(data);
    socketio.emit("message", data);
    console.log("Received data: " + data);
  });

  socketclient.on("typing", () => {
    socketio.emit("typing");
    console.log("Someone is typing...");
  });

});
```