University of Dayton
Department of Computer Science

CPS 475/575
Secure Application Development

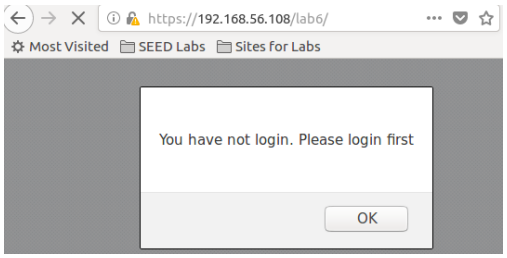
Lecture 21 – Secure Database Modification and Access Control

Phu Phung
4/7/2020

Review: The web login system – Current Implementation: `index.php` & `form.php`

- Users with valid username/password can be authenticated and logged in
  - Authenticated users can be logged in automatically without providing username/password

- Invalid username/password or authenticated users must be redirected to the login form:



Review: Security of the current login system

(Lecture 16/Lab 5.1)

(Lecture 17/Lab 5.2)

(Lecture 18/Lab 6.1)

(Lecture 19+20/Lab 6.2)

A New Use case: Database Modification

- A common scenario in a database-related application, e.g., the login system, is to modify the data
  - Example: Logged in users can change their passwords
    - Any web application systems should allow and implement such a scenario
      - E.g., You should be able to change the passwords of your UD account, Google accounts, Facebook accounts, etc…
- We will design and implement password changing use case for our current login system
  - We will consider the security aspects during the development stage

Today's Agenda

- Database Modification: A new use case – Changing password
  - Common Session Authentication
  - Database Modification Implementation
  - Access control and authorization security issues
    - Unauthorized access demos
      - Change the password of another user
      - CSRF attack

Review: Lab 6

Lab instructions: http://bit.ly/secad-s20-lab6 (**Task 3 has been added**)

- Task 1: Data Protection and HTTPS (Lecture 18)
  - Follow Lecture 18 to setup HTTPS for your web server
  - Copy Lab 5 code and deploy as Lab 6
- Task 2: Secure Session Authentication (Lecture 19+20)
  - Revised Login System with Session Management and Protection
- Task 3: Secure Database Modification
  a. Database Modification and Access Control (Today)
  b. Cross-site Request Forgery Attack and Prevention (Next Lecture)

Database Modification – The form (Homework)

- A new use case: the logged-in user wants to change the password
  - We need to create a new PHP page with a form with username and the new password inputs
    - Copy the current `form.php` file and name it to `changepasswordform.php`:
      `$ cp form.php changepasswordform.php`
    - Add the link to this page from the `index.php`:
      `<a href="changepasswordform.php">Change password</a>`
    - Change the HTML content and the form in `changepasswordform.php` as below (new/revised contents are highlighted in boxes)

```
<h1>Change Password, SecAD</h1>
<h2>By Phu Phung</h2>
<?php
  //some code here
  echo "Current time: " . date("Y-m-d h:i:sa")
?>
        <form action="changepassword.php" method="POST" class="form login">
              Username:<input type="text" class="text_field" name="username">
            <br>
            New password: <input type="password" class="text_field" name="newpassword" /> <br>
              <button class="button" type="submit">
                Change password
              </button>
        </form>
```

Database Modification – The action (Homework)

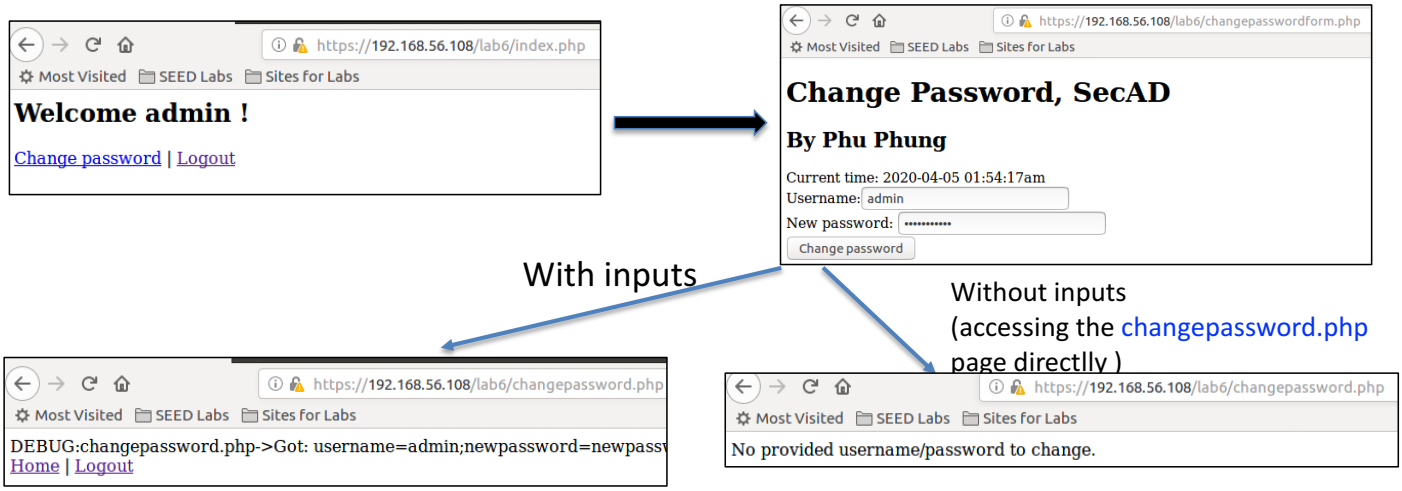
- We need to create a new PHP page to get the username and new password inputs
  - Connect to the database to execute a corresponding SQL query, i.e., UPDATE users SET password = … WHERE username= … to change the password
  - Hands-on: Create the `changepassword.php` file and just print out the inputs as the code below:
    $ `subl changepassword.php`

$_REQUEST can be used for both $_GET & $_POST

```php
<?php
    $username= $_REQUEST["username"];
    $newpassword = $_REQUEST["newpassword"];
    if (isset($username) AND isset($newpassword)) {
        echo "DEBUG:changepassword.php->Got: username=$username;newpassword=$newpassword\n<br>";
    }else{
        echo "No provided username/password to change.";
        exit();
    }
?>
<a href="index.php">Home</a> | <a href="logout.php">Logout</a>
```

Testing the Skeleton

- Deploy all files to the server **(ensure that you are in your lab6 private repo folder)** :
  $ `sudo cp *.php /var/www/html/lab6`
- Test: index.php -> changepasswordform.php->changepassword.php



changepasswordform.php changepassword.php

A. No input validation
B. No authentication, everyone can access the pages
C. Vulnerable to SQLi Attacks
D. A and B
E. A, B, and C

Access Control Testing

- Open the two new files directly (type from the address bar) from another browser that you have not logged in before:
  - https://192.168.56.108/lab6/changepasswordform.php
  - https://192.168.56.108/lab6/changepassword.php
- You can access these pages normally
  - There is no authentication for these two PHP pages

Secure Database Modification – Security Requirement

- Only authenticated users can access these two changepasswordform.php and changepassword.php pages
  - However, we should not ask the user to login again after authenticated (the user should not provide username/password again)
    - We should check the session
      - If the session is not logged in, redirect to the login form
  - We have implemented this mechanism in the index.php page
    - Ad-hoc solution: Copy the code to these two new pages
      - Not convenience or scalable:
        - » We will add more pages in the system
        - » We might revise the authentication code -> need to change everywhere
    - Modular solution: Create a new PHP page that only does the secure session authentication, e.g., session_auth.php
      - Require this file in any PHP pages that need authentication

Common Session Authentication (Homework)

- Create a new PHP file, namely `session_auth.php`
  $ `subl session_auth.php`
- Copy the code of session settings, validation and session protection from your current `index.php` file to this `session_auth.php` file:

```php
<?php
    $lifetime = 15 * 60;
    $path = "/lab6";
    $domain = "192.168.56.108";
    $secure = TRUE;
    $httponly = TRUE;
    session_set_cookie_params($lifetime,$path,$domain,$secure,$httponly);
    session_start();

    //check the session
    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();
    }
?>
```

Include session authentication

- Add
  `require "session_auth.php";`
  to the beginning of the two pages
  changepasswordform.php and changepassword.php

changepasswordform.php

```php
<?php
  require "session_auth.php";
?>
<html>
```

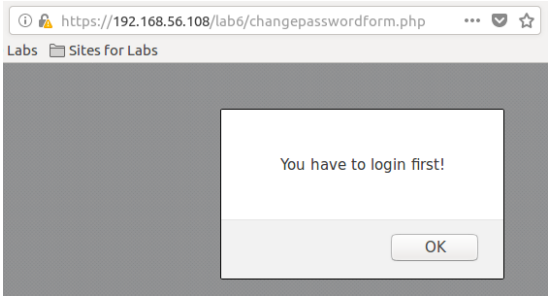
changepassword.php

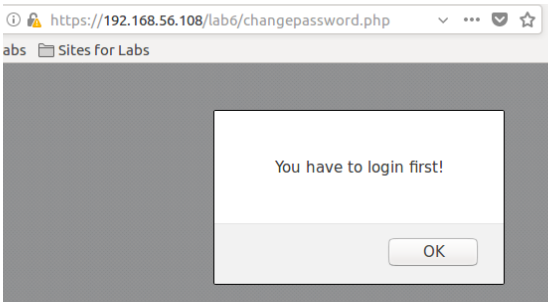
```php
<?php
    require "session_auth.php";
```

- Deploy the new files and test again
  – The user is required to login if she did not logged in before

Demo: Session authentication

- The two pages are now only accessible for authenticated users





Database Modification (Change Password) Implementation

- We need to create a Prepared Statement in UPDATE SQL, bind the inputs to the parameters and execute
  - Prepared Statements will prevent potential SQLi Attacks
- Instead of implementing this function in the changepassword.php page, we create a new page e.g., database.php
  - Later we can add more functions to handle database manipulation

database.php – The database handling module

- Create the page:
  $ `subl database.php`
- Copy the whole function `securechecklogin(..)` from the `index.php` page to this page and rename it to `changepassword(..)`
  - Move the `$mysqli` variable outside of the function
    - It can be used for other functions without redefining (please note the credentials)
  - Define to use the global variable `$mysqli` inside the `changepassword(..)` function:
    global `$mysqli;`

database.php | changepassword.php — old-repos/.../lab7 | changepassword.php — private-repos/.../lab6 | logout.php | index.php

```
1  <?php
2      $mysqli = new mysqli('localhost','[redacted]', '[redacted]', '[redacted]');
3      if($mysqli->connect_errno){
4          printf("Database connection failed: %s\n", $mysqli->connect_error);
5          exit();
6      }
7      function changepassword($username, $newpassword) {
8          global $mysqli;
```

Prepared Statement for UPDATE

- Create an UPDATE Prepared Statement:

```
$prepared_sql = "UPDATE users SET password=password(?) WHERE username= ?;";
echo "DEBUG>prepared_sql= $prepared_sql\n";
```

- Prepare and bind the parameters

```
if(!$stmt = $mysqli->prepare($prepared_sql)) return FALSE;
$stmt->bind_param("ss",$newpassword,$username);
```

Note the order of arguments

- Execute and catch error:

```
    if(!$stmt->execute()) return FALSE;
    return TRUE;
}
```

Database Modification (Password change) – Full code

database.php | changepassword.php — old-repos/.../lab7 | changepassword.php — private-repos/.../lab6 | logout.php | index.php

```php
<?php
    $mysqli = new mysqli('localhost','[redacted]1', '[redacted]$', '[redacted]');
    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;
    }
?>
```

Using the `database.php` page

- In `changepassword.php` page, include (require) the `database.php` page and call the `changepassword(..)` function accordingly

```php
<?php
    require "session_auth.php";
    require 'database.php';
    $username= $_REQUEST["username"];
    $newpassword = $_REQUEST["newpassword"];
    if (isset($username) AND isset($newpassword)) {
        echo "DEBUG:changepassword.php->Got: username=$username;newpassword=$newpassword\n";
        if (changepassword($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();
    }
?>
```