# **CPS 475/575 - Secure Application Development, Spring 2020**

Dr. Phu Phung

Department of Computer Science,  
University of Dayton

Lab 5 - Secure Web Application Development in PHP with Cookies and Sessions and Databases (MySQL)

Instruction released: 3/3/2020

Deadline: to be updated

Introduction

In this lab, you will learn the basic web application development in the PHP language with the concepts of cookies, sessions, and databases (MySQL). You will also do the experiments security vulnerabilities including session hijacking, cross-site scripting (XSS) and SQL Injection (SQLi) attacks, and on how to protect your web application from these potential attacks.

**Preparation:** The code skeleton is provided in the course repository. You need to pull the course repo to get the latest update. Suppose you cloned the course repo in your local SEED VM at ~/secad:

$ ﻿cd ~/secad && git pull

Copy the new folder **labs/lab5** to your private repo, and change the directory to there, e.g.:

$ ﻿cd ~/secad-pphung1/labs/lab5

**Source code organization:** You need to push all your code for this lab under a folder labs/lab5 within your private repository. You need to include the URL of this folder at the beginning of your report. For example, in my case, the URL is:  
<https://bitbucket.org/phu-udayton/secad-pphung1/src/master/labs/lab5/>

**You will lose 5% of this lab grade if this URL is missing in your report and 5% if the code is not in your repository.**

This lab consists of three (3) tasks as follows.

**Task 1 (8 points): Simple session test with PHP**

**Task 2 (12 points): A simple yet secure login system in PHP with MySQL**

**~~Task 3 (10 points): Secure Authentication with Session~~**

### Report and submission

You need to write your report to submit via Isidore. Your report must be submitted in PDF file (reports not in PDF format will be returned without grading for this lab). Your report MUST have the course number, course name, the instructor, your name and UD email. You MUST organize your report as in this instruction, for example:

Task 1. xxx

1. Yyy

You need to demonstrate your experiments by screenshots, therefore, **it is important to capture screenshots when you perform the lab**. **Your screenshots must have a short description or a caption to explain the task.**

I recommend you to use Google Docs to write your report as it is convenient to include the screenshots for your report. *f you use Google Docs to write the report, you can use an add-on such as “Code Blocks” (*[*https://gsuite.google.com/marketplace/app/code\_blocks/100740430168*](https://gsuite.google.com/marketplace/app/code_blocks/100740430168)*) to format and indent your code.*

**Task 1 (8 points): Simple session test with PHP**

(Note: this task was introduced in Lecture 15 3/3/2020)

sessiontest.php is a simple PHP application that stores a variable in session. The code is provided from the course repository. Its content is listed below:  


1. (2 points) Deploy and test sessiontest.php   
   Steps:

* Create lab5 folder on the web server root directory:

$ ﻿sudo mkdir /var/www/html/lab5

* Deploy the sessiontest.php to the lab5 folder:

$ ﻿sudo cp sessiontest.php /var/www/html/lab5

* Test from the Firefox browser within the VM:<http://localhost/lab5/sessiontest.php>
  + Refresh the page couple of times
* Test again from a browser in your laptop (and perform the same steps as above):<http://192.168.56.100/lab5/sessiontest.php>
  + Note that the IP address might be different in your SEED VM, to get the correct IP address, try $ ifconfig from the terminal

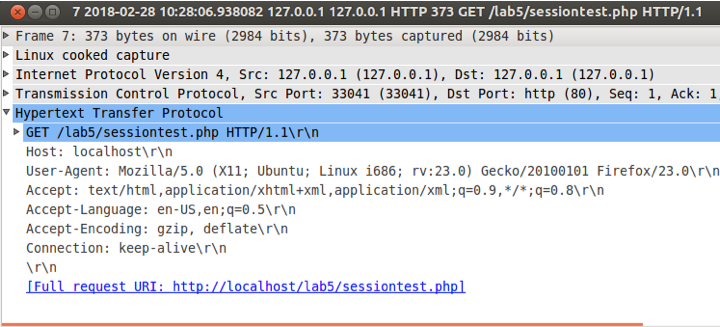
Capture the screenshot similar to the below screenshot from the two browsers to include in your report (1 point). Explain the results (1 point).

1. (4 points) Observe the Session-Handshaking using Wireshark

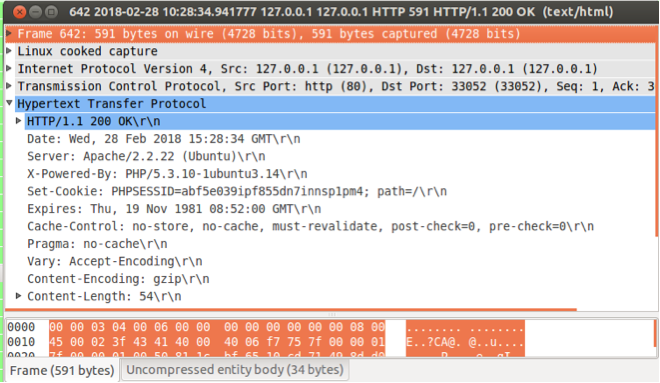
Steps to perform:

* Clear the Cookies in the Firefox browser (History -> Clear All History -> Clear Now)
* Open the Wireshark program and start observing (As in Lab 1)
* Refresh the webpage
* Refresh the webpage again
* Stop the Wireshark and observe the first and the second HTTP Request/Response

(1 p) Capture the first HTTP Request to include in your report (see the sample screenshot below) and answer the following question: Is there any cookie information sent by the browser? Why?



(1 p) Capture the first HTTP Response to include in your report (see the sample screenshot below) and answer the following question: Is there any cookie information sent in the HTTP Response? If so, what is the value?



(1 p) Capture the second HTTP Request to include in your report (see the sample screenshot below) and answer the following question: Is there any cookie information sent by the browser? Copy the line of the cookie in your report. Is this the same cookie information set by the server?



(1 p) From the above experiments, write a short paragraph to express your understanding of sessions and cookies in PHP (or in a Web application in general). Note: this is a frequently asked job interview question in web application development.

1. (2 p) Hijack a session

Steps to perform:

* Refresh the webpage on the Firefox browser to e.g.,20 times
  + Open the Firebug to copy the session id (PHPSESSID)
* Open the webpage in Chrome browser in your laptop
  + Ensure that the views are 1 time (by clear the cookies)
  + Right-click Inspect -> Console. Copy the screenshot to include in your report.
  + Type from the console and enter:
    - document.cookie="PHPSESSID=<the-value-you-have-copied>"
  + Refresh the webpage in the Chrome browser

Capture the last screenshot to include in your report and answer the following questions: what is the value of the times on the webpage? Where does this value come from and why? (1p)

Suppose that session ID is stored in the cookie and stolen by an attacker and there is no session protection mechanism, what can the attacker do with that session ID? (1 p)

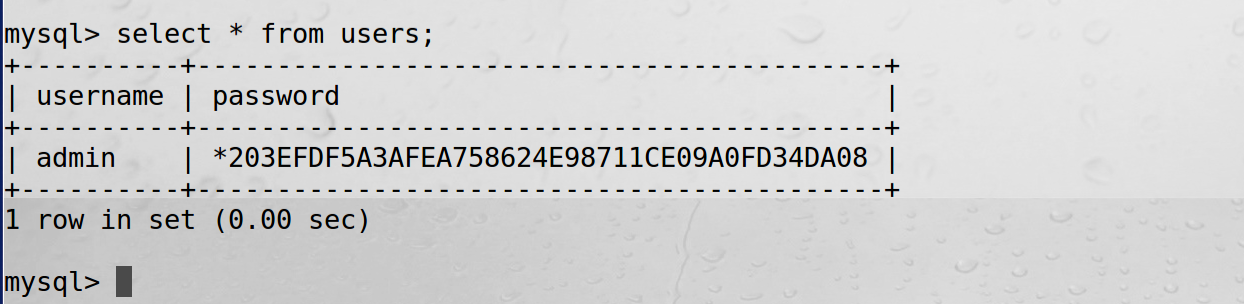
**Task 2 (12 points): A simple yet secure login system in PHP with MySQL**

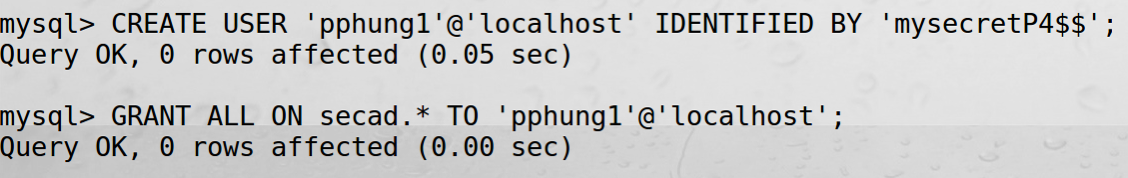
(Note: these sub-tasks are introduced in Lecture 16 and 17. Updated: 3/10/2020)

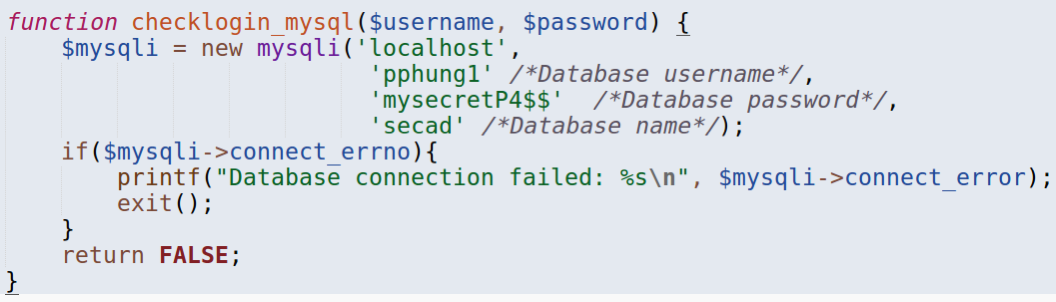
1. Database setup and management

Refer to Lecture 16 (3/10/2020) for this sub-task. The steps include:

* 1. (2p) Create a new database in the MySQL database server in the SEED VM and create a table “users” in the database and insert a new record username/password in the database. The password must be hashed with the password function in MySQL. Capture the screenshots of the results similar to below:



* 1. (1p) Create a new database user and grant permission to access the database created in (i). Capture the screenshot to include in your report.   
     

If you have done this during the class but did not capture the screenshot, you can just capture the screenshot of the credentials from the mysqli connection, i.e.:  


1. (2 p) Insecure login system with PHP/MySQL:  
   You are provided a simple login system with 2 PHP files in the course repository (already pulled with Task 1):

form.php: User type username and password and submit

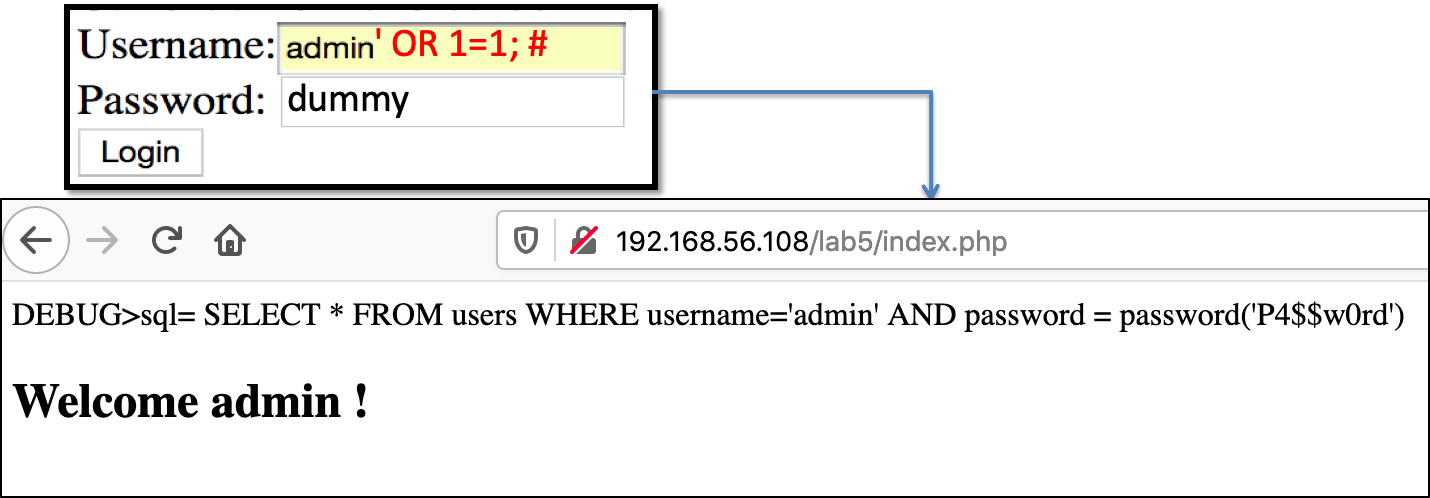
index.php: check the provided username/password to allow the login or not

In this sub-task, you need to clone the function checklogin in the index.php application and rename it to checklogin\_mysql. This new checklogin\_mysql function will connect to the database, construct a SQL query from the given username/password to check if it is matching the record in the database. Revise the code to call this new function. Refer to Lecture 16 for the steps. Include the code of the code (in plaintext) new checklogin\_mysql function in your report.

Deploy these two PHP files to the Apache Web Server and test with the case of the valid username/password and capture the screenshot to include in your report.



1. Performing XSS and SQL Injection attacks
   1. (2 p) SQL Injection attacks

Instead of typing the correct username/password, you need to provide a SQL Injection Code in the username field to bypass the password check. The password can be whatever. Capture the screenshot to demonstrate the success of this attack (1 point), e.g.:  
and explain why the attack is successful (1 point).

* 1. (1 p) Cross-site Scripting (XSS) Attacks

XSS is a vulnerability in web applications that allows attackers to inject JavaScript code that will be executed on the origin of the web site. This current web application is also vulnerable to this attack since it outputs the user input directly without validating. Repeat the similar attack as in Task 1.b, however, after the #, provide a piece of JavaScript code such as:  
admin' OR 1=1;# <script>alert(document.cookie);</script>  
Perform the attack and capture the screenshot to demonstrate it and explain why this attack happened, e.g, wherein the PHP source code causing this vulnerability.

1. Prepared Statement Implementation

Copy and paste the entire checklogin\_mysql function in the index.php rename the function to securechecklogin. Revise the code to implement the Prepared Statement as introduced in Lecture17. Change the call from checklogin\_mysql to securechecklogin.

Include the code of this new function in your report (1p). Redeploy the new code to the server and test to ensure that it can prevent SQLi attacks. Capture the screenshot to include in your report (1p) and explain why the attacks can be prevented (2p).

**Task 3 (10 points): Secure Authentication with Session**

Update: Moved to Lab 6