­

Lab V – Session Hijacking Attack and Protection

CPS 499-02/592-02

Software/Language Based Security

Fall 2020

Dr. Phu Phung

Evan Krimpenfort

# Code:

<https://github.com/Krimpenfort23/autumn-2020/tree/master/cps_499/lab-5>

# Task I:

## Login Modification

Graphical user interface, text, application, email

Description automatically generated

**Figure 1: changing login credentials**

## b. Observing HTTP Reponses/Requests With Cookies – First Time

### i. Any cookie information in the request?

Graphical user interface, text, application

Description automatically generated

**Figure 2: HTTP Request**

There is no cookie information here in Figure 2 because there is no cookie to get from the server. The server right now does not have a cookie for this site. So, there is nothing to grab.

### ii. Any cookie information in the response?

### Graphical user interface, text, application, email Description automatically generated

**Figure 3: HTTP Response**

However, in Figure 3, we can see that there is cookie information under a *Set-Cookie* call because the server now wants to associate that site with some sort of ID. We can see here that the cookie is “*Set-Cookie: PHPSESSID=qr2hg7000qf5q0ot1811j2qsm1; expires=Sun, 25-Oct-2020 22:54:02 GMT; Max-Age=86400; path=/”*.

## Observing HTTP Reponses/Requests With Cookies – Second Time

### Any cookie information in the request?

Graphical user interface, text, application

Description automatically generated

**Figure 4: HTTP Request After**

In figure 4, there is cookie information. This is because the server has the cookie and it got that cookie from the previous HTTP response with the *Set-Cookie* call. The Cookie seen in the request was *“Cookie: PHPSESSID=qr2hg7000qf5q0ot1811j2qsm1”* which is the same cookie as before. Since the browser has that ID and so does the server, that check allows the HTTP Request to ask for that cookie information and see if there’s anything there.

### Any cookie information in the response?

Graphical user interface, text, application

Description automatically generated

**Figure 5: HTTP Response After**

In figure 5, we can see that there is no cookie information and that’s because the server doesn’t need any cookie information anymore from the browser. From now on, the Requests will ask for the cookie to verify any change in the server that need to be shown in the browser and the responses will remain cookieless.

# Task II: Session Hijacking Attack

## a. Performing the Attack

### i. Steal the Cookie inside of the SEED VM

Graphical user interface, text, application, email

Description automatically generated

**Figure 6: Getting the Cookie.**

In figure 6, you can see that the Cookie is ﻿*"PHPSESSID=uf9otmqf2jnrcq1nl8pb5h7a07"*.

### The Attacker Side

Graphical user interface, text, application

Description automatically generated

**Figure 7: To the Desktop: Can’t log in now**

### Perform the Attack

Graphical user interface, text, application

Description automatically generated

**Figure 8: Attack executed**

## Attack Explanation

This attack happens because I stole the ID that was associated with a session of a successful login. So, when the browser sends an HTTP request with the cookie information, the information received back was making the “logged” variable true so that the user could log in. Because the attacker was able to steal the cookie and place it into their own browser, that was the only reason this could happen.

# Task III: Fix the Session Hijacking Vulnerability

## Code revision and comparison

A picture containing calendar

Description automatically generated

**Figure 9: Code for the Hijacking protection in the index page**

A picture containing logo

Description automatically generated

**Figure 10: Code for the Hijacking protection on the login page**

## Attack Prevention

Graphical user interface, application

Description automatically generated

**Figure 11: Hijacking on the index page**

Graphical user interface, application

Description automatically generated

**Figure 12: Hijacking on the login page**