­

Lab VIII – Cross-Site Request Forgery Attacks

CPS 499-02/592-02

Software/Language Based Security

Fall 2020

Dr. Phu Phung

Evan Krimpenfort

# Task I: Understanding HTTP Request Parameters

## What is the full URL?

The full URL would be <http://www.myblog.com/admin/index.php>. The admin is important because that is where the .php file is located.

## What is the HTTP Method of request?

The method of request is going to be “POST” as seen by the method attribute inside of the form header.

## What field names are used in this request?

The field names used in this request are **Title** and **Text**. That can be seen from looking inside of the form header.

# Task II: Construct the webpage to perform the CSRF

## The Revised Code

Text

Description automatically generated

**Figure 1: Revised Code**

## Alert Message

Graphical user interface, text, application

Description automatically generated

**Figure 2: Alert Message from me**

## Capture the Scenario

Graphical user interface, application

Description automatically generated

**Figure 3: Alert message directs to the login page**

# Task III: Perform the CSRF attack using XSS

## Construct the XSS Code

Graphical user interface, text, application, email

Description automatically generated

**Figure 4: Code before injected**

## b. Simulate the Attack

### i. The CSRF Request happens

Graphical user interface, text

Description automatically generated

**Figure 5: Code that was injected**

### ii. The data is injected into the system

Graphical user interface, text, application

Description automatically generated

**Figure 6: The attack was done**



**Graphical user interface

Description automatically generated**



**Figure 7: The post was made**

# Task IV: CSRF Defenses

## Why does Task III happen?

When the user on the windows machine clicks on the link that takes them to the attack html page, the server does not distinguish the request referrer. This allows that site to request any HTTP request it wants and in this particular case, it adds a post.

## Describe a solution to prevent such attacks

A solution that could be implemented in order to prevent this attack would be a way to validate the referrer. This involves Inspecting the HTTP Request header for a URL that is allowed to make the requests. In this case, it would be only from the URL <http://www.myblog.com/admin/index.php> and never from my personal IP address.