

## Lab #2

**Due date & Time** 5:00pm on April 17th, 2009. The service for the lab will be shut down at the due time.

**Late Policy:** Late labs will not be accepted.

### 1 Overview

In this lab, you will get some experience of exploiting web application vulnerabilities, such as cross site scripting and cross site request forgery.

#### 1.1 The environment

We have set up a web-based forum using phpBB. We modified the software to introduce vulnerabilities in this forum. You need to exploit this vulnerability by posting some malicious messages to the message board; users who view these malicious messages will become victims. The attackers' goal is to post forged messages for the victims and to modify the victims' profiles.

The forum is hosted on <http://forest.cs.purdue.edu/phpBB2>. Please register two accounts at the forum as soon as possible. The account name must be *lastname\_1* and *lastname\_2*, where *lastname* is your last name in lower case. Please choose strong passwords to keep your accounts secure.

Once registered, please send an email to TA with subject *cs526 lab2 setup lastname*. TA will create a private forum for you and grant the access to your accounts. The name of the private forum is the same as your last name. All your works should be done in the private forum.

One account is generally enough to complete the lab. Having two accounts could be more convenient for testing and debugging. You can use one account as the adversary and the other account as the victim user. However, do NOT make the attack specific to your victim account in the final submission. You should expect an arbitrary victim user unless explicitly specified in the task descriptions.

NOTE:

1. Please be patient when registering on the forum. It may take up to 1 minute. Other operations are pretty responsive.
2. Please strictly follow the account naming rules. Don't tell TA your passwords.

## 1.2 Submission

Your submission are just messages posted in your private forum. The title of the submission should be named as *final\_attack\_k*, where *k* is the task number. The postings with other names won't be tested in the grading. It does not matter which account is used to post the submission messages.

You are recommended to do this lab using Firefox. TA will use Firefox for testing and grading your submissions by default. Please contact TA if you have difficulties in getting and using Firefox. TA can test your submissions using other browsers if requested.

You need to submit a detailed lab report to describe what you have done and what you have observed. Send the report to the TA by email before the due time.

## 1.3 Important Notes

1. Please strictly follow the account naming rules and submission naming rules.
2. You are free to use Google and any other reference you can find in this lab.
3. Please do NOT try to attack the web application in ways other than those specified in this description.

## 2 Preparation

To succeed in this lab, you should understand the basic concept of the cross-site scripting and cross-site request forgery and how to exploit those vulnerabilities. In addition, you should have some knowledge about the HTTP protocol and JavaScript programming. A good tutorial for JavaScript can be found at <http://www.w3schools.com/JS/default.asp>.

An important step in this lab is to examine the HTTP requests and responses exchanged between the browser and the server. This can be done by using a local web proxy. You are recommended to use Fiddler, a free web debugging proxy, which can be downloaded from <http://www.fiddler2.com/Fiddler2/>. After running the software, you can change the browser's proxy configuration to 127.0.0.1:8888. Then all HTTP requests and responses sent and received by the browser will be collected in Fiddler. You can examine the content of every HTTP packet. A quick introduction to Fiddler can be found at <http://msdn.microsoft.com/en-us/library/bb250446.aspx>.

If you would like to debug the JavaScript in the browser, you can use FireBug, a Firefox extension for web application debugging, which can be downloaded at <https://addons.mozilla.org/en-US/firefox/addon/1843>.

## 3 The Tasks

### 3.1 Task 1: Popup an alert window (10 pts)

The objective of this task is to post a malicious message that contains JavaScript to popup an alert window. The alert window should include the text “Hello Task 1”.

Hint: Use the JavaScript function `alert()` to popup an alert window.

### 3.2 Task 2: Display cookies (10 pts)

The objective of this task is to post a malicious message on the message board containing a JavaScript code that prints out the cookies set by the web-application for the victim user. You can use the alert window to display the cookies.

Hint: Use `document.cookie` to access the cookies in JavaScript.

### 3.3 Task 3: Post on the victim’s behalf (15 pts)

In this task, you need to post a malicious message containing a JavaScript that forges a HTTP request in the victim’s browser. The HTTP request will post a new message titled “Hello Task 3” on the victim’s behalf.

Please use the `XMLHttpRequest` for sending the request. A good tutorial on AJAX programming can be found at <http://www.xul.fr/en-xml-ajax.html>.

Hint: You may need to get a sample request for posting a message. To do that, follow the following steps. First post a regular message to the forum in the browser pointing to the local Fiddler proxy. Second, examine the content of the HTTP request for posting in the Fiddler proxy.

### 3.4 Task 4: Repeat task 3 without using JavaScript (15 pts)

In this task, you need to post a malicious message that achieve the same effect as that in Task 3 without using JavaScript. When a victim user views the message, a new message titled “Hello Task 4” is posted on the victim’s behalf.

Hint: Use a tag with a `src` attribute to trigger the request.

### 3.5 Task 5: Modify the victim’s profile (15 pts)

In this task, you need to post a malicious message. When the victim user views the message, his profile is changed. In particular, the victim’s signature will be changed to “Hello Task 5”.

For this task, you can assume you target a specific victim user. The victim user account is named “victim\_user”.

Hint: You can lookup the `user_id` and the email of the victim user by clicking the “Memberlist” in the top of the page.

Hint: Use `XMLHttpRequest` to send a POST request.

### 3.6 Task 6: Repeat task 5 without using JavaScript (15 pts)

In this task, you need to post a malicious message that achieve the same effect as that in Task 5 without using using JavaScript. When the victim user views the message, his profile is changed. In particular, the victim's signature will be changed to "Hello Task 6". You can assume the victim user is willing to click a button in your message.

You also target a specific victim user. The victim user account is named "victim\_user".

Hint: Use a hidden form to send the POST request.

### 3.7 Task 7: Repeat task 5 for any victim user(20 pts)

In this task, you need to make the attack in Task 5 work for any victim. When an arbitrary victim user views the malicious message, his profile is changed. In particular, the victim's signature will be changed to "Hello Task 7".

It should work for any victim user.

Hint: Use AJAX to get the victim's account name, `user_id` and email. You can try to grab the standard profile updating page of the victim user, and then pase the page to get the necessary information.

### 3.8 Task 8: Write a Self-Propagating XSS Worm (Bonus 20 pts)

In this task, you need to post a malicious message (denoted as P1) that is a self-propagating XSS worm. When a victim user views the message P1, a new message (denoted as P2) is posted on the victim's behalf. In addition, the message P2 is another instance of the XSS worm. If a victim user views the message P2, another new message (P3) is posted. And when P3 is clicked another message P4 is posted, ..., it goes on forever.

Typically, all the messages (P1, P2, ...) are having the same contents. The worm will propagate as people click on the messages.

We will give 20 bonus points if you can accomplish this task.

Note that in order to prevent spam, the forum will block a posting request that immediately follows a previous posting if the time interval is too short. So the malicious posting request might be rejected by the forum if you click the worm too frequently. Please remember this restriction when you test and debug your worm. You do NOT need to attempt to bypass this restriction.

## 4 Grading

For each task, 80% of the credit is given for the programming and 20% is given for the report. No credit will be given for the report corresponding to a failed attack.