# Cross Site Scripting

## Computer Security CSE 484

Tatiana Ensslin - November 2, 2015

## Introduction

The goal of this lab is to become familiar with cross-site scripting (XSS). This is a type of vulnerability found in web applications. This vulnerability makes it possible for attackers to inject malicious code into a web browser in order to steal the victim's credentials, for example, session cookies. In order to illustrate what attackers can do by exploiting XSS vulnerabilities we will use Elgg in our pre-built Ubuntu VM image to show the vulnerability which we will launch an XSS attack into. This exemplifies the type of attack received by MySpace in 2005—the Samy worm.

## 2.1 Environment Configuration

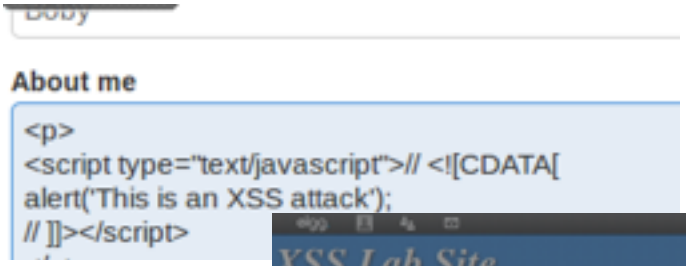
In this lab, we need three tihngs, a web browser, and Apache web server, and the Elgg website. We will use the LiveHTTPHeaders extension in Firefox to inspect the HTTP requests and responses. To begin, we start by starting up the Apache server.

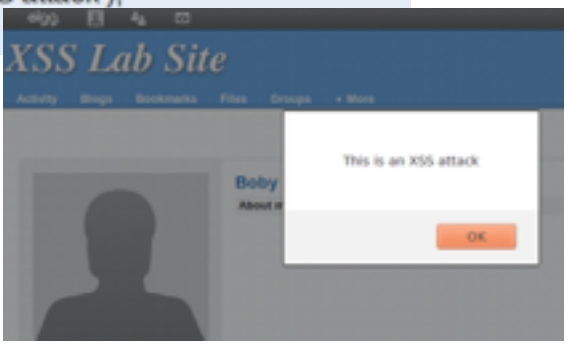
```
[11/02/2015 17:39] seed@ubuntu:~$ sudo service apache2 start
[sudo] password for seed:
 * Starting web server apache2
httpd (pid 1077) already running
```

Next, ensure that the DNS and the Apache server is configured. This has already been done for the URL needed for this lab.

## 3.1 Task 1: Posting a Malicious Message to Display and Alert Window

The objective of this task is to embed a JavaScript programming the Elgg profile, so that when another profile views your profile, the JavaScript program will be executed and an alert window will be displayed. By embedding the JavaScript code in the profile, any user who views your profile will see the alert window. Here we ass the javascript to the about me section, and when a visitor visits Boby's page, the alert arises.



### 3.2 Task 2: Posting a Malicious Message to Display Cookies

The objective of this task is to embed a JavaScript program in the ELgg profile that will display the users cookies in an alert window. To do this we insert the JavaScript into the about me section, therefore when a user looks are Boby's profile, their cookies are reveled.

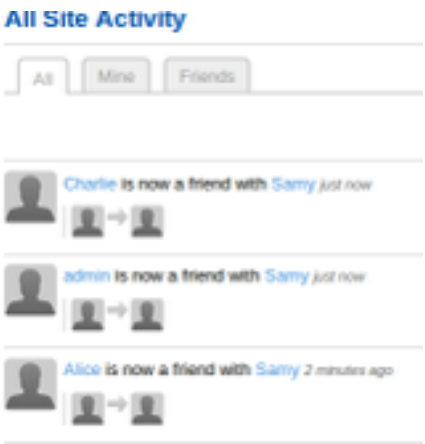
```
<p>
<script type="text/javascript">// <![CDATA[
alert (document.cookie);
// ]]></script>
```

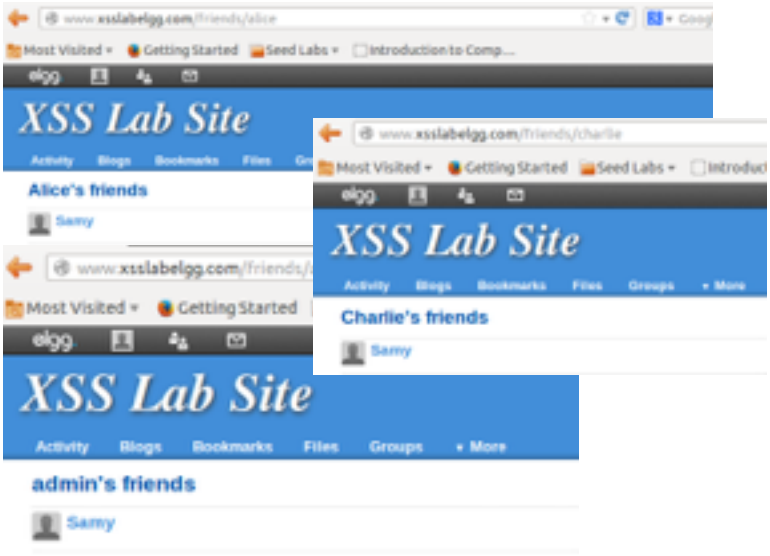

### 3.5 Task 5: Writing an XSS Worm

Now we will preform an attack similar to what Samy did in MySpace in 2005. First we will write an XSS worm that does no self propagate. Now we will write a malicious JavaScript program that forges HTTP requests direction from the victim's browser, without the intervention of the attacker. The objective of the attack is to modify the victims profile and add Samy a friend to the victim. To do this, we use the Ajax, partially provided for us and modify the access tokens to validate the add friend URL HTTPrequest and add this to Samy's about me section on his profile.

```
var sendurl="http://www.xsslabelgg.com/action/friends
/add?friend=42"+"&"+"__elgg_ts="+elgg.security.token.__elgg_ts+"&"+"__elgg_token="+elgg.security.
token.__elgg_token;
```

Above, we modify the sendurl to add in the elgg.security.elgg_ts and elgg.securit.elgg_token in order to create a propagating worm that will befriend the victim when Samy's page is viewed. Not how everyone who views Samy's profile is now his friend:



### 3.6 Task 6: Writing a Self-Propagating XSS Worm

In order to actually become a real worm-it needs to be able to self propagate. In this section, we modify the worm code to allow it to self propagate. By modifying the sending URL and sending a post request to edit the description of a persons about me, we are able to spread the worm. For example, Charlie got the worm from Alice, as displayed in my demo, and it can be viewed from Charlie's profile, seen below, because it states, "Samy is my HERO."