William Dahl
ICSI 424 Information Security
Lab 6
October 26th, 2018

Task1: In this screen shot I use the HTTP Header Live tool to inspect an HTTP Header and I catapulted a GET request

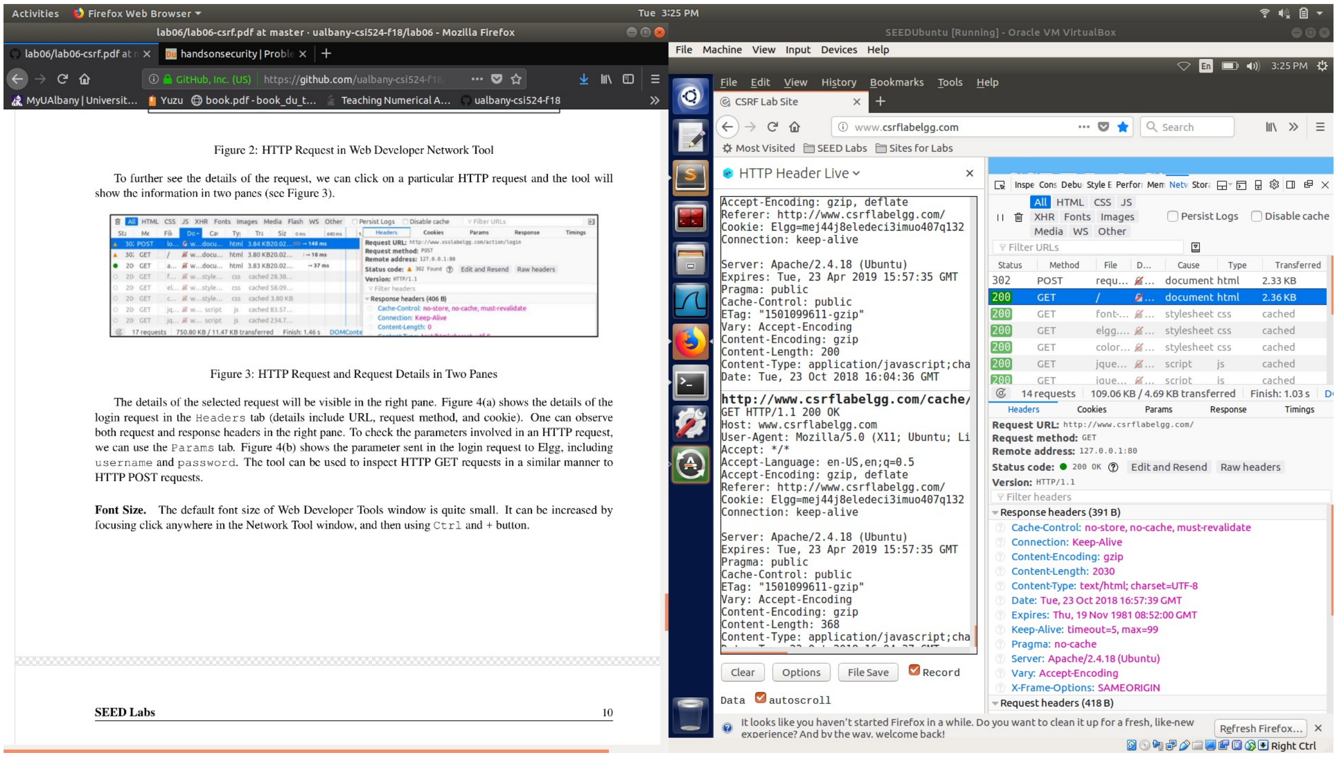


Figure 2: HTTP Request in Web Developer Network Tool

To further see the details of the request, we can click on a particular HTTP request and the tool will show the information in two panes (see Figure 3).

Figure 3: HTTP Request and Request Details in Two Panes

The details of the selected request will be visible in the right pane. Figure 4(a) shows the details of the login request in the Headers tab (details include URL, request method, and cookie). One can observe both request and response headers in the right pane. To check the parameters involved in an HTTP request, we can use the Params tab. Figure 4(b) shows the parameter sent in the login request to Elgg, including username and password. The tool can be used to inspect HTTP GET requests in a similar manner to HTTP POST requests.

Font Size. The default font size of Web Developer Tools window is quite small. It can be increased by focusing click anywhere in the Network Tool window, and then using Ctrl and + button.

SEED Labs 10

HTTP Header Live

```
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=mej44j8eledeci3imuo407q132
Connection: keep-alive

Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 23 Apr 2019 15:57:35 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;cha
Date: Tue, 23 Oct 2018 16:04:36 GMT

http://www.csrflabelgg.com/cache/
GET HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=mej44j8eledeci3imuo407q132
Connection: keep-alive

Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 23 Apr 2019 15:57:35 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;cha
```

Status	Method	File	D...	Cause	Type	Transferred
302	POST	requ...	...	document	html	2.33 KB
200	GET	/	...	document	html	2.36 KB
200	GET	font-...	...	stylesheet	css	cached
200	GET	elgg....	...	stylesheet	css	cached
200	GET	color...	...	stylesheet	css	cached
200	GET	jque...	...	script	js	cached
200	GET	jque...	...	script	js	cached

14 requests | 109.06 KB / 4.69 KB transferred | Finish: 1.03 s

Request URL: http://www.csrflabelgg.com/
Request method: GET
Remote address: 127.0.0.1:80
Status code: ● 200 OK
Version: HTTP/1.1

Response headers (391 B)

- Cache-Control: no-store, no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Encoding: gzip
- Content-Length: 2030
- Content-Type: text/html; charset=UTF-8
- Date: Tue, 23 Oct 2018 16:57:39 GMT
- Expires: Thu, 19 Nov 1981 08:52:00 GMT
- Keep-Alive: timeout=5, max=99
- Pragma: no-cache
- Server: Apache/2.4.18 (Ubuntu)
- Vary: Accept-Encoding
- X-Frame-Options: SAMEORIGIN

Request headers (418 B)

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

In this screen shot I captured a POST request

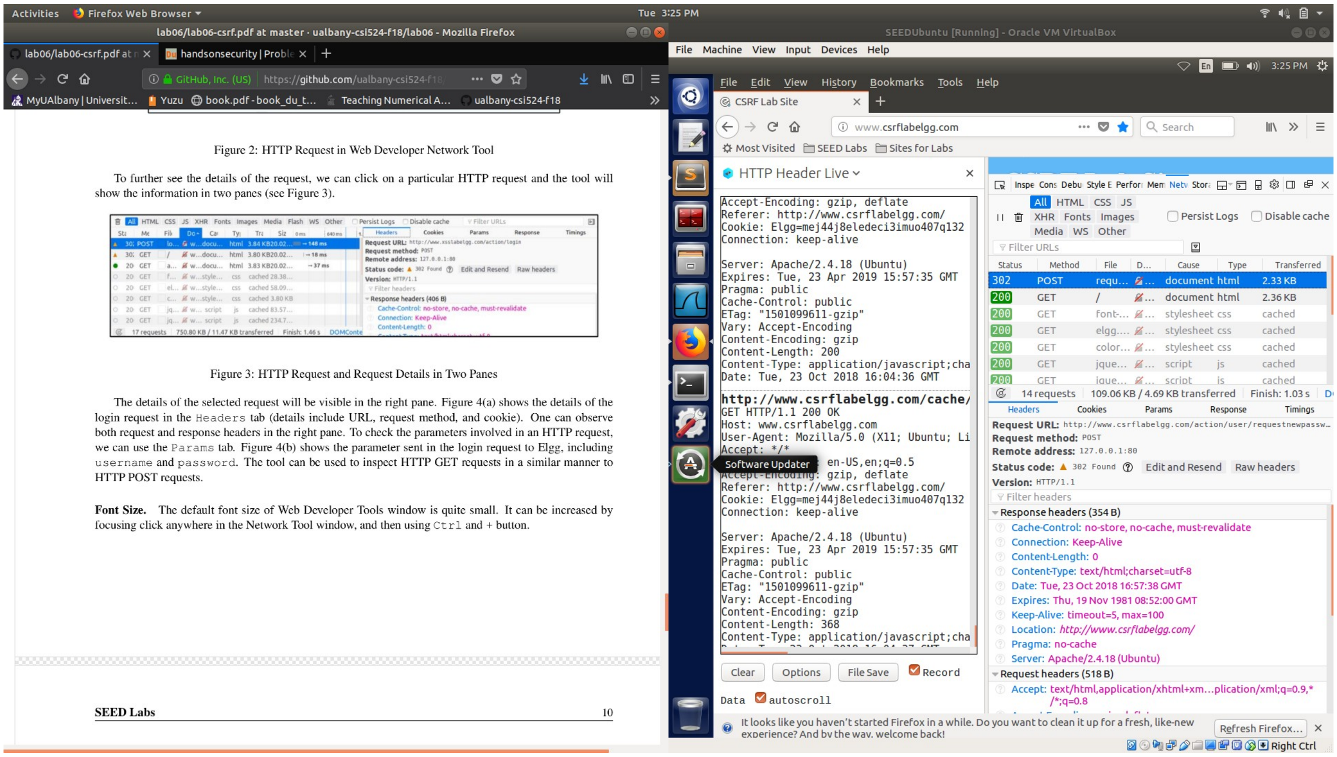


Figure 2: HTTP Request in Web Developer Network Tool

To further see the details of the request, we can click on a particular HTTP request and the tool will show the information in two panes (see Figure 3).

Figure 3: HTTP Request and Request Details in Two Panes

The details of the selected request will be visible in the right pane. Figure 4(a) shows the details of the login request in the Headers tab (details include URL, request method, and cookie). One can observe both request and response headers in the right pane. To check the parameters involved in an HTTP request, we can use the Params tab. Figure 4(b) shows the parameter sent in the login request to Elgg, including username and password. The tool can be used to inspect HTTP GET requests in a similar manner to HTTP POST requests.

Font Size. The default font size of Web Developer Tools window is quite small. It can be increased by focusing click anywhere in the Network Tool window, and then using Ctrl and + button.

SEED Labs 10

HTTP Header Live

Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=mej44j8eledeci3imuo407q132
Connection: keep-alive

Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 23 Apr 2019 15:57:35 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 200
Content-Type: application/javascript;cha
Date: Tue, 23 Oct 2018 16:04:36 GMT

http://www.csrflabelgg.com/cache/
GET HTTP/1.1 200 OK
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: */*
en-US,en;q=0.5
gzip, deflate
Referer: http://www.csrflabelgg.com/
Cookie: Elgg=mej44j8eledeci3imuo407q132
Connection: keep-alive

Server: Apache/2.4.18 (Ubuntu)
Expires: Tue, 23 Apr 2019 15:57:35 GMT
Pragma: public
Cache-Control: public
ETag: "1501099611-gzip"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 368
Content-Type: application/javascript;cha

Status	Method	File	D...	Cause	Type	Transferred
302	POST	requ...	...	document	html	2.33 KB
200	GET	/	...	document	html	2.36 KB
200	GET	font-...	...	stylesheet	css	cached
200	GET	elgg....	...	stylesheet	css	cached
200	GET	color...	...	stylesheet	css	cached
200	GET	jque...	...	script	js	cached
200	GET	jque...	...	script	js	cached

14 requests | 109.06 KB / 4.69 KB transferred | Finish: 1.03 s

Request URL: http://www.csrflabelgg.com/action/user/requestnewpassw...
Request method: POST
Remote address: 127.0.0.1:80
Status code: ▲ 302 Found
Version: HTTP/1.1

Response headers (354 B)

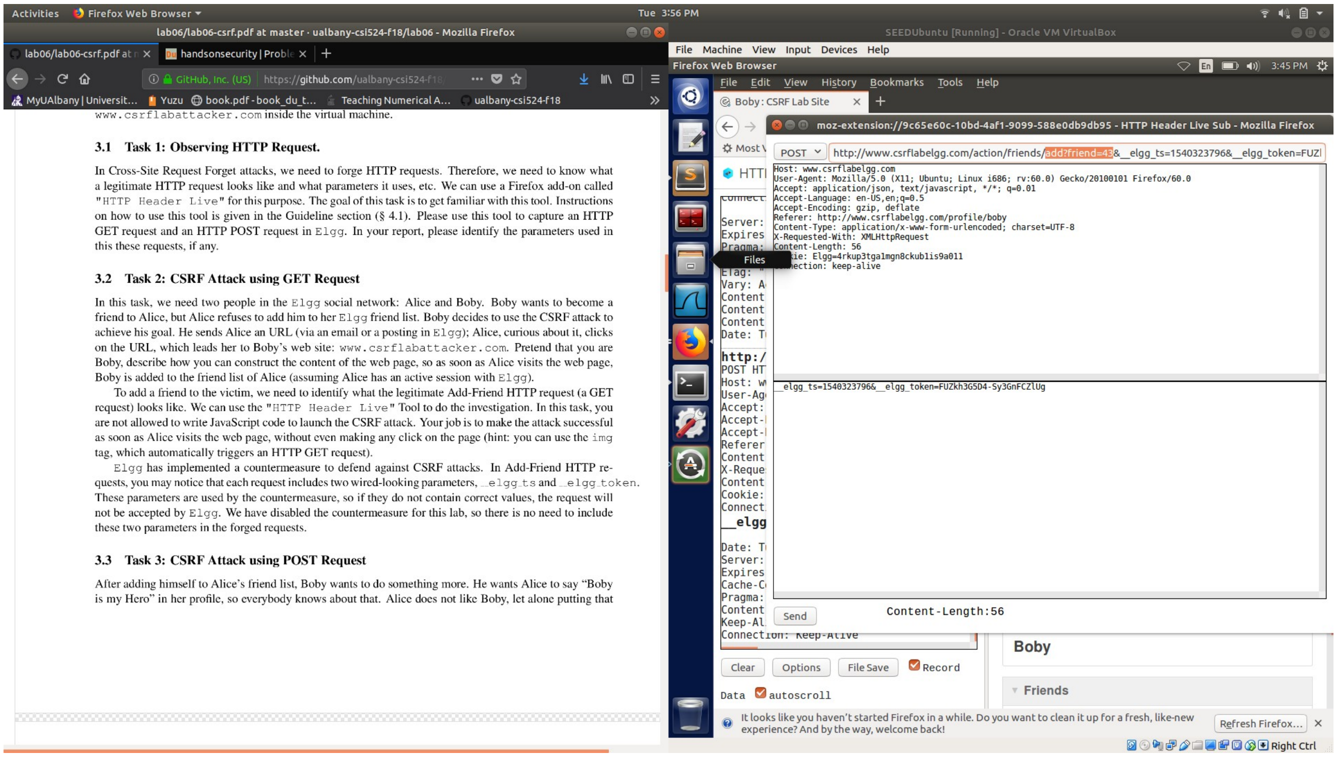
- Cache-Control: no-store, no-cache, must-revalidate
- Connection: Keep-Alive
- Content-Length: 0
- Content-Type: text/html;charset=utf-8
- Date: Tue, 23 Oct 2018 16:57:38 GMT
- Expires: Thu, 19 Nov 1981 08:52:00 GMT
- Keep-Alive: timeout=5, max=100
- Location: *http://www.csrflabelgg.com/*
- Pragma: no-cache
- Server: Apache/2.4.18 (Ubuntu)

Request headers (518 B)

- Accept: text/html,application/xhtml+xm...plication/xml;q=0.9,*/*;q=0.8

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Task 2: In this screen shot I used the “HTTP Header Live” tool to show how boby can identify what the legitimate Add-Friend HTTP request looks like.


Activities
Firefox Web Browser
Tue 3:56 PM
lab06/lab06-csrf.pdf at master · ualbany-csi524-f18/lab06 - Mozilla Firefox
https://github.com/ualbany-csi524-f18/
SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
3:45 PM
File Edit View History Bookmarks Tools Help
Boby : CSRF Lab Site
moz-extension://9c65e60c-10bd-4af1-9099-588e0db9db95 - HTTP Header Live Sub - Mozilla Firefox
Files
Send
Clear
Options
File Save
Record
Boby
Friends

www.csrflabattacker.com inside the virtual machine.

3.1 Task 1: Observing HTTP Request.

In Cross-Site Request Forget attacks, we need to forge HTTP requests. Therefore, we need to know what a legitimate HTTP request looks like and what parameters it uses, etc. We can use a Firefox add-on called "HTTP Header Live" for this purpose. The goal of this task is to get familiar with this tool. Instructions on how to use this tool is given in the Guideline section (§ 4.1). Please use this tool to capture an HTTP GET request and an HTTP POST request in Elgg. In your report, please identify the parameters used in this these requests, if any.

3.2 Task 2: CSRF Attack using GET Request

In this task, we need two people in the Elgg social network: Alice and Boby. Boby wants to become a friend to Alice, but Alice refuses to add him to her Elgg friend list. Boby decides to use the CSRF attack to achieve his goal. He sends Alice an URL (via an email or a posting in Elgg); Alice, curious about it, clicks on the URL, which leads her to Boby’s web site: www.csrflabattacker.com. Pretend that you are Boby, describe how you can construct the content of the web page, so as soon as Alice visits the web page, Boby is added to the friend list of Alice (assuming Alice has an active session with Elgg).

To add a friend to the victim, we need to identify what the legitimate Add-Friend HTTP request (a GET request) looks like. We can use the "HTTP Header Live" Tool to do the investigation. In this task, you are not allowed to write JavaScript code to launch the CSRF attack. Your job is to make the attack successful as soon as Alice visits the web page, without even making any click on the page (hint: you can use the img tag, which automatically triggers an HTTP GET request).

Elgg has implemented a countermeasure to defend against CSRF attacks. In Add-Friend HTTP requests, you may notice that each request includes two wired-looking parameters, __elgg_ts and __elgg_token. These parameters are used by the countermeasure, so if they do not contain correct values, the request will not be accepted by Elgg. We have disabled the countermeasure for this lab, so there is no need to include these two parameters in the forged requests.

3.3 Task 3: CSRF Attack using POST Request

After adding himself to Alice’s friend list, Boby wants to do something more. He wants Alice to say “Boby is my Hero” in her profile, so everybody knows about that. Alice does not like Boby, let alone putting that

POST http://www.csrflabelgg.com/action/friends/add?friend=43&__elgg_ts=1540323796&__elgg_token=FUZ

```
Host: www.csrflabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.csrflabelgg.com/profile/boby
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 56
kie: Elgg=4rkup3tga1mgn8ckub1is9a011
ection: keep-alive
```

```
__elgg_ts=1540323796&__elgg_token=FUZkh3G5D4-Sy3GnFCZlUg
```

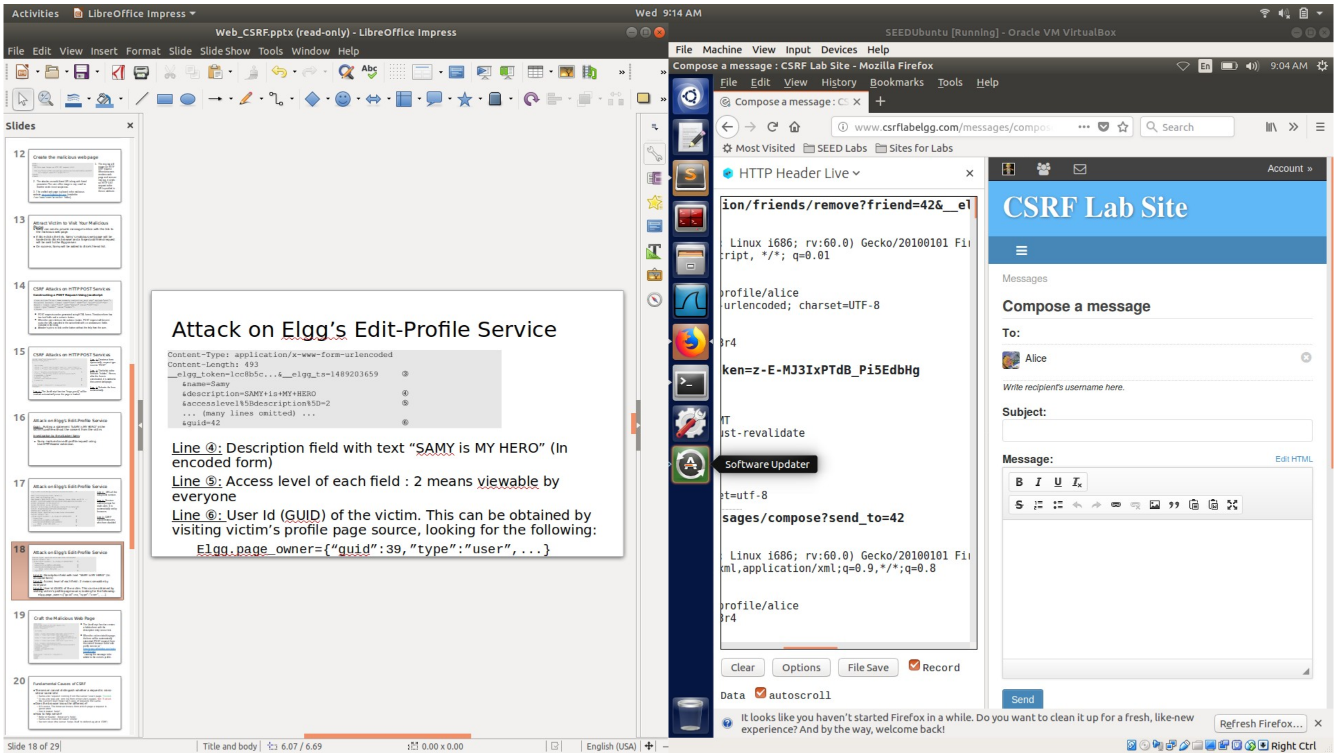
Content-Length:56

Data ☑ autoscroll

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox…

One way Boby can add himself to Alice's friends is by sending her a link to his malicious website via the messenger built in on the Elgg website. In the websites HTML code would be the line: <img src=”http://ww.csrflabelgg.com/action/friends/add?friend=43” alt=”image” width=”1” height=”1”/> This would invoke a HTTP GET request to the Elgg website causing boby to be added to Alice’s friends. He can get his id number by creating a new account and adding him self as a fried and then by inspecting the HTTP Header, he can get his id which is 43.

Task 3: In this screen shot I shown that by clicking to send a message, the HTTP header shows the id of Alice.


Activities
LibreOffice Impress
Wed 9:14 AM
Web_CSRF.pptx (read-only) - LibreOffice Impress
File Edit View Insert Format Slide Slide Show Tools Window Help
Slides
Slide 18 of 29
Title and body
English (USA)
SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Compose a message : CSRF Lab Site - Mozilla Firefox
9:04 AM
File Edit View History Bookmarks Tools Help
Compose a message : CS
www.csrflabelgg.com/messages/compos
Search
Most Visited
SEED Labs
Sites for Labs
HTTP Header Live
Software Updater
Clear
Options
File Save
Record
Data
autoscroll
Account »
Edit HTML
Send
Refresh Firefox…
Right Ctrl

Attack on Elgg's Edit-Profile Service

```
Content-Type: application/x-www-form-urlencoded
Content-Length: 493
__elgg_token=lcc8b5c...&__elgg_ts=1489203659      ③
   &name=Samy
   &description=SAMY+is+MY+HERO                   ④
   &accesslevel%5Bdescription%5D=2                ⑤
   ... (many lines omitted) ...
   &guid=42                                       ⑥
```

Line ④: Description field with text "SAMY is MY HERO" (In encoded form)

Line ⑤: Access level of each field : 2 means viewable by everyone

Line ⑥: User Id (GUID) of the victim. This can be obtained by visiting victim's profile page source, looking for the following:

```
Elgg.page_owner={"guid":39,"type":"user",...}
```

```
ion/friends/remove?friend=42&__el

 Linux i686; rv:60.0) Gecko/20100101 Fi
ript, */*; q=0.01

rofile/alice
urlencoded; charset=UTF-8

r4

ken=z-E-MJ3IxPTdB_Pi5EdbHg

T
st-revalidate

t=utf-8
sages/compose?send_to=42

 Linux i686; rv:60.0) Gecko/20100101 Fi
ml,application/xml;q=0.9,*/*;q=0.8

rofile/alice
r4
```

CSRF Lab Site

Messages

Compose a message

To:

Alice

Write recipient's username here.

Subject:

Message:

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

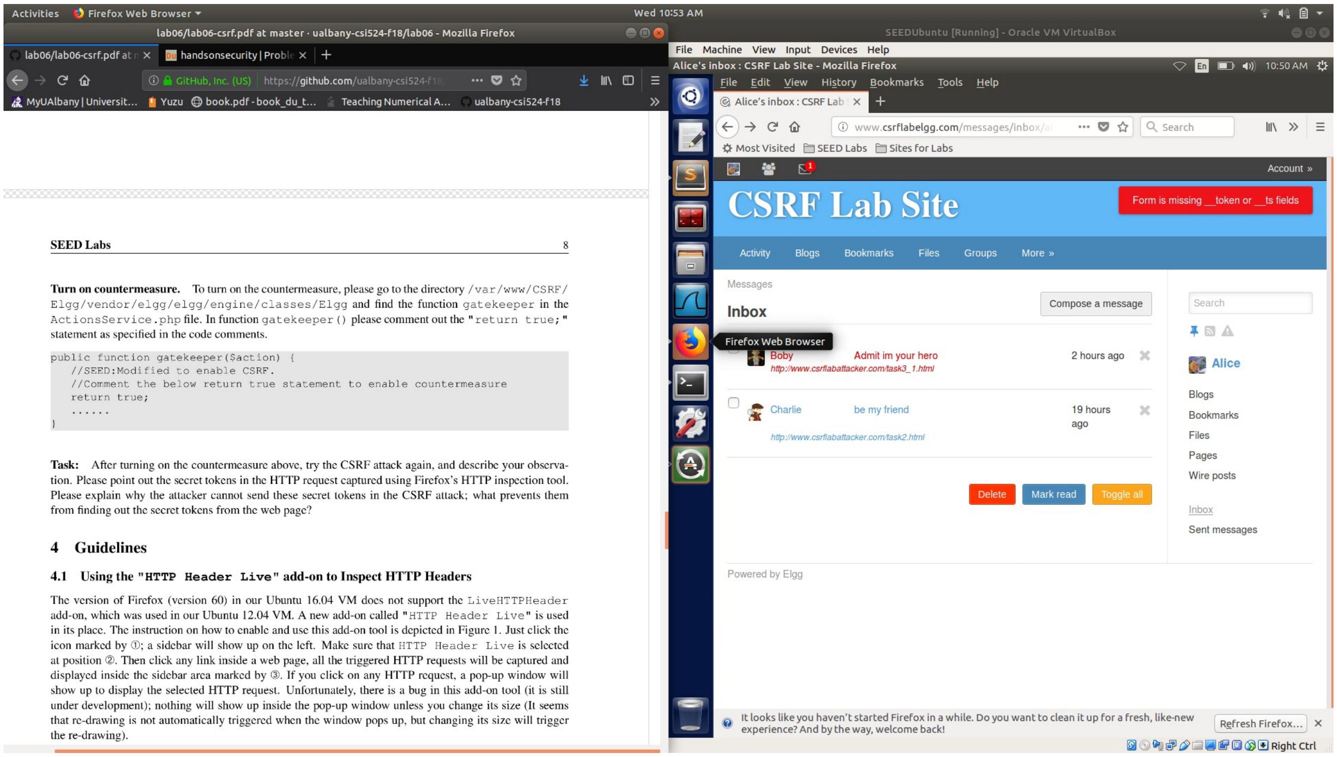
To launch the POST attack the first that needs to be done is that the fields in the JavaScript script need to be filled. The name variable in the first field needs to be 'name' because we are setting the name field in the HTML body. The value for the name field will be Alice because that is the name of the target page owner. The second field is the description field which we fill with the value 'BOBY is MY HERO'. This is what will be displayed on the page after it is edited. The next field is the accesslevel field which we set to '2' because 2 will make it view-able by anybody. Next id the guid field which we set to 42 which is Alice's id number. We can find her id number by performing any action with Alice's page and then inspect the HTTP header and see the id in the GET variables. The next thing to do within the script is to create a form so that a POST request can be sent to the target website. p.action is the action taken by the form which will open the web page http://www.csrflabelgg.com/action/profile/edit. It will then populate the web page with the fields just created and then send the request via POST. The next step in the attack would be to get the target (Alice) to open the malicious website by getting them to click on the link through a message sent to them via the elgg messenger. When the victim visits this page, the form will be automatically submitted (POST request) from the victim's browser to the edit-profile service at "http://www.csrflabelgg.com/action/profile/edit" causing the message to be added to the victim's profile.

Questions:

1.) We can find her id number by performing any action with Alice's page and then inspect the HTTP header and see the id in the GET variables in the URL. For example, when Boby goes to send a message to alice, using the HTTP Header tool he can vew the HTTP Header for the message and see that in the URL is http://www.csrflabelgg.com/messages/compose?send_to=42. This shows that the id of Alice is 42.

2.) No, Boby wont be able to carry out the same attack because in order for it to be successful he needs to know the id of the person clicking on the link, other wise the information given wont match up and the attack will fail.

Task 4: In this screen shot I ran the CSRF attack again but I got the message “form is missing _tokens or _ts fields”.


Activities
Firefox Web Browser
Wed 10:53 AM
lab06/lab06-csrf.pdf at master · ualbany-csi524-f18/lab06 - Mozilla Firefox
SEEDUbuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Alice's inbox : CSRF Lab Site - Mozilla Firefox
10:50 AM
File Edit View History Bookmarks Tools Help
www.csrflabelgg.com/messages/inbox/
Most Visited
SEED Labs
Sites for Labs
Account »
Activity
Blogs
Bookmarks
Files
Groups
More »
Firefox Web Browser

SEED Labs 8

Turn on countermeasure. To turn on the countermeasure, please go to the directory `/var/www/CSRF/Elgg/vendor/elgg/elgg/engine/classes/Elgg` and find the function `gatekeeper` in the `ActionsService.php` file. In function `gatekeeper()` please comment out the "`return true;`" statement as specified in the code comments.

```
public function gatekeeper($action) {
   //SEED:Modified to enable CSRF.
   //Comment the below return true statement to enable countermeasure
   return true;
   ......
}
```

Task: After turning on the countermeasure above, try the CSRF attack again, and describe your observation. Please point out the secret tokens in the HTTP request captured using Firefox's HTTP inspection tool. Please explain why the attacker cannot send these secret tokens in the CSRF attack; what prevents them from finding out the secret tokens from the web page?

4 Guidelines

4.1 Using the "`HTTP Header Live`" add-on to Inspect HTTP Headers

The version of Firefox (version 60) in our Ubuntu 16.04 VM does not support the `LiveHTTPHeader` add-on, which was used in our Ubuntu 12.04 VM. A new add-on called "`HTTP Header Live`" is used in its place. The instruction on how to enable and use this add-on tool is depicted in Figure 1. Just click the icon marked by ①; a sidebar will show up on the left. Make sure that `HTTP Header Live` is selected at position ②. Then click any link inside a web page, all the triggered HTTP requests will be captured and displayed inside the sidebar area marked by ③. If you click on any HTTP request, a pop-up window will show up to display the selected HTTP request. Unfortunately, there is a bug in this add-on tool (it is still under development); nothing will show up inside the pop-up window unless you change its size (It seems that re-drawing is not automatically triggered when the window pops up, but changing its size will trigger the re-drawing).

CSRF Lab Site

Form is missing __token or __ts fields

Messages

Inbox

Compose a message

Boby Admit im your hero 2 hours ago
http://www.csrflabattacker.com/task3_1.html

□ Charlie be my friend 19 hours ago
http://www.csrflabattacker.com/task2.html

Delete Mark read Toggle all

Alice

Blogs
Bookmarks
Files
Pages
Wire posts

Inbox
Sent messages

Powered by Elgg

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox…

The attacker cannot send these secret tokens in the CSRF attack or find out what the secret tokens are from the web page because, The server embeds a random secret value inside each webpage. When a request is initiated from the page, the secret value is included in the request. The server checks the value to see weather it is a cross-site request or not. Pages from a different origin will not be able to access the secret value because of the browsers same origin policy. The secret token is randomly generated and is different for different users. So, there is no way for attackers to guess or find out the secret token.