COS430 – Project Deliverable 3 – Guidelines on Firefox's Developer Tools

Note: The materials presented in this project are inspired by and partially taken from SEED labs, with permission from the author, Dr. Du.

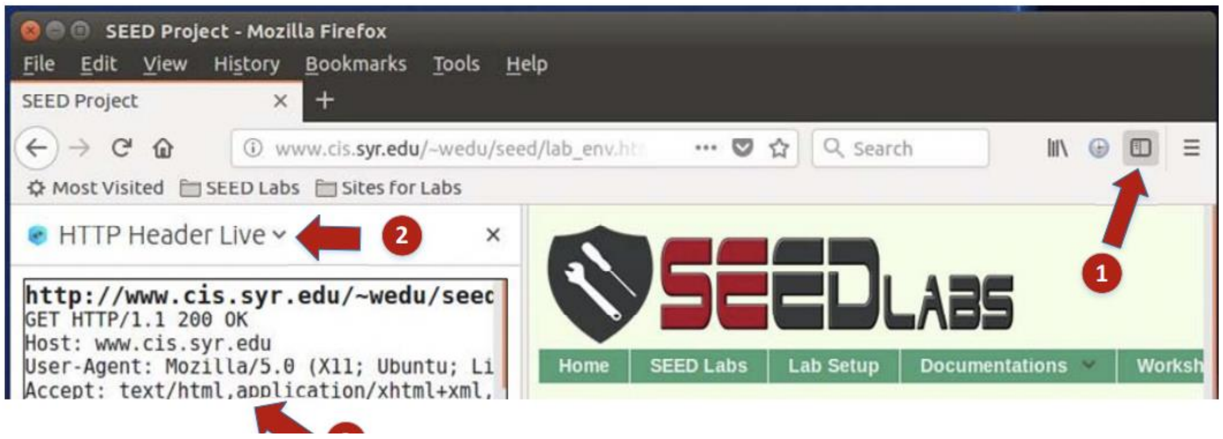
Guidelines:

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).



```
http://www.cis.syr.edu/~wedu/seec
GET HTTP/1.1 200 OK
Host: www.cis.syr.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu; Li
Accept: text/html,application/xhtml+xml,
```

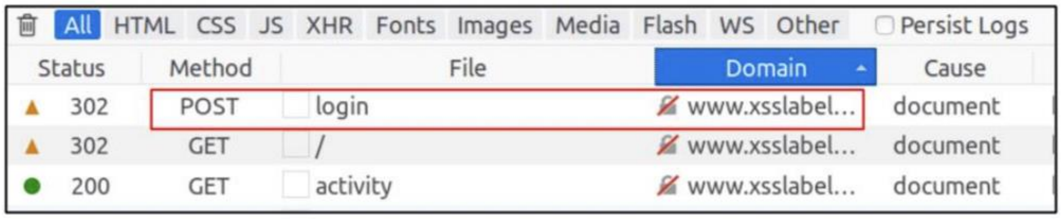
Figure 1 - Enable the HTTP Header Live Add-on

2. Using the Web Developer Tool to Inspect HTTP Headers

There is another tool provided by Firefox that can be quite useful in inspecting HTTP headers. The tool is the *Web Developer Network Tool*. In this section, we cover some of the important features of the tool. The *Web Developer Network Tool* can be enabled via the following navigation:

```
Click Firefox's top right menu --> Web Developer --> Network
or
Click the "Tools" menu --> Web Developer --> Network
```

We use the user login page in Elgg as an example. Figure 2 shows the Network Tool showing the HTTP POST request that was used for login.



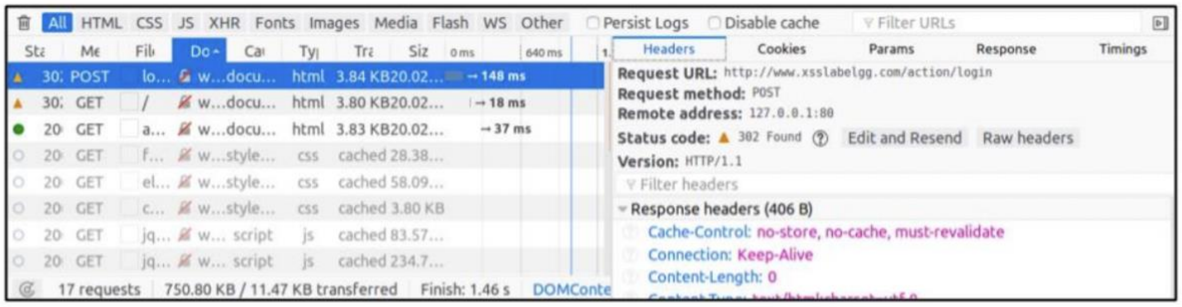
Status	Method	File	Domain	Cause
302	POST	login	www.xsslabel...	document
302	GET	/	www.xsslabel...	document
200	GET	activity	www.xsslabel...	document

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).

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, you 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.

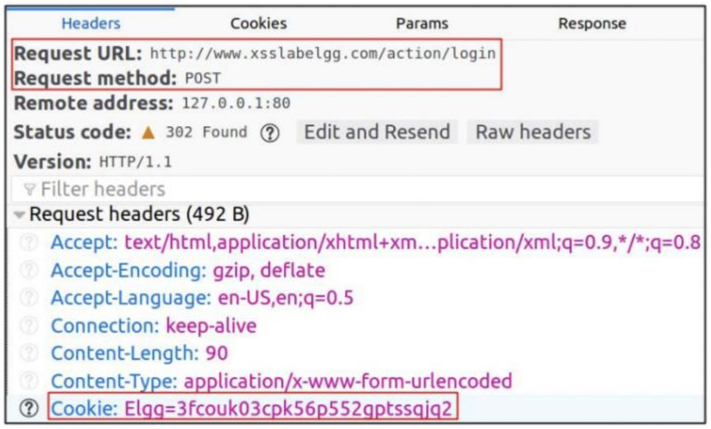

All HTML CSS JS XHR Fonts Images Media Flash WS Other
Persist Logs
Disable cache
Filter URLs
Headers Cookies Params Response Timings
Edit and Resend
Raw headers
Filter headers

Sta	Me	Fil	Do	Ca	Ty	Tra	Siz	0 ms	640 ms
30:	POST	lo...	w...docu...	html	3.84 KB	20.02...		→ 148 ms	
30:	GET	/	w...docu...	html	3.80 KB	20.02...		→ 18 ms	
20	GET	a...	w...docu...	html	3.83 KB	20.02...		→ 37 ms	
20	GET	f...	w...style...	css	cached	28.38...			
20	GET	el...	w...style...	css	cached	58.09...			
20	GET	c...	w...style...	css	cached	3.80 KB			
20	GET	jq...	w...	script	js	cached	83.57...		
20	GET	jq...	w...	script	js	cached	234.7...		

17 requests | 750.80 KB / 11.47 KB transferred | Finish: 1.46 s | DOMConte

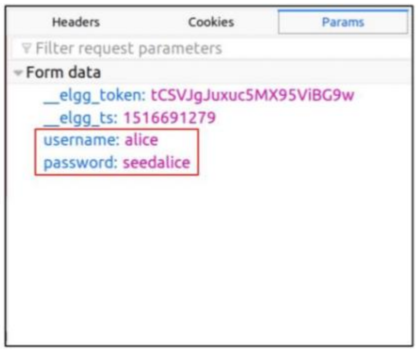
Request URL: http://www.xsslabelgg.com/action/login
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Response headers (406 B)
Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Length: 0

Figure 3 - HTTP Request and Request Details in Two Pages



Request URL: http://www.xsslabelgg.com/action/login
Request method: POST
Remote address: 127.0.0.1:80
Status code: 302 Found
Version: HTTP/1.1
Request headers (492 B)
Accept: text/html,application/xhtml+xm...plication/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Length: 90
Content-Type: application/x-www-form-urlencoded
Cookie: Elgg=3fcouk03cpk56p552gptssqjq2

(a) HTTP Request Headers



Form data
__elgg_token: tCSVJgJuxuc5MX95ViBG9w
__elgg_ts: 1516691279
username: alice
password: seedalice

(b) HTTP Request Parameters

Figure 4 - HTTP Headers and Parameters

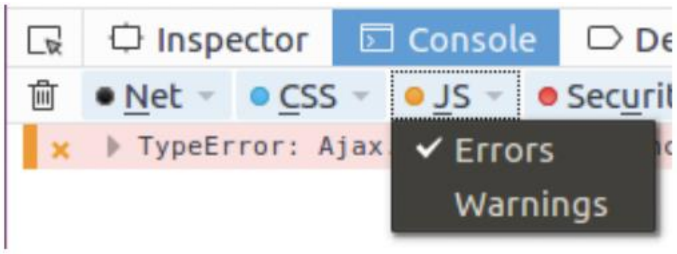
3. JavaScript Debugging

We may also need to debug our JavaScript code. Firefox's Developer Tool can also help debug JavaScript code. It can point us to the precise places where errors occur. The following instruction shows how to enable this debugging tool:

```
Click the "Tools" menu --> Web Developer --> Web Console
or use the Shift+Ctrl+K shortcut
```

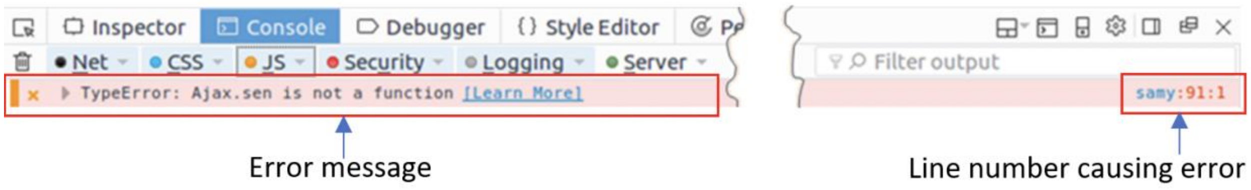
Once we are in the web console, click the `JS` tab. Click the downward pointing arrowhead beside `JS` and ensure there is a check mark beside `Error`. If you are also interested in Warning messages, click `Warning` (See Figure 5).

If there are any errors in the code, a message will display in the console. The line that caused the error appears on the right side of the error message in the console. Click on the line number and you will be taken to the exact place that has the error (See Figure 6).



```
TypeError: Ajax
```

Figure 5 - Debugging JavaScript Code (1)



```
TypeError: Ajax.sen is not a function [Learn More]                samy:91:1
```

Figure 6 - Debugging JavaScript Code (2)