Chad Ballay

CYBR350-342N

Week 8

08/24/2020

WebGoat – Journal pt 4 (A5pt2-A7)

Please see previous doc for full notes.

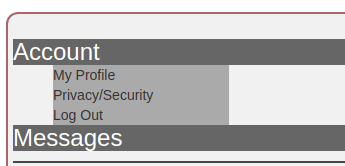
<https://github.com/Chad-Ballay/Cybersecurity-work-class/blob/master/classes/CYBR350/week6/journal_notes.docx>

<https://github.com/Chad-Ballay/Cybersecurity-work-class/blob/master/classes/CYBR350/week7/journal_week7.docx>

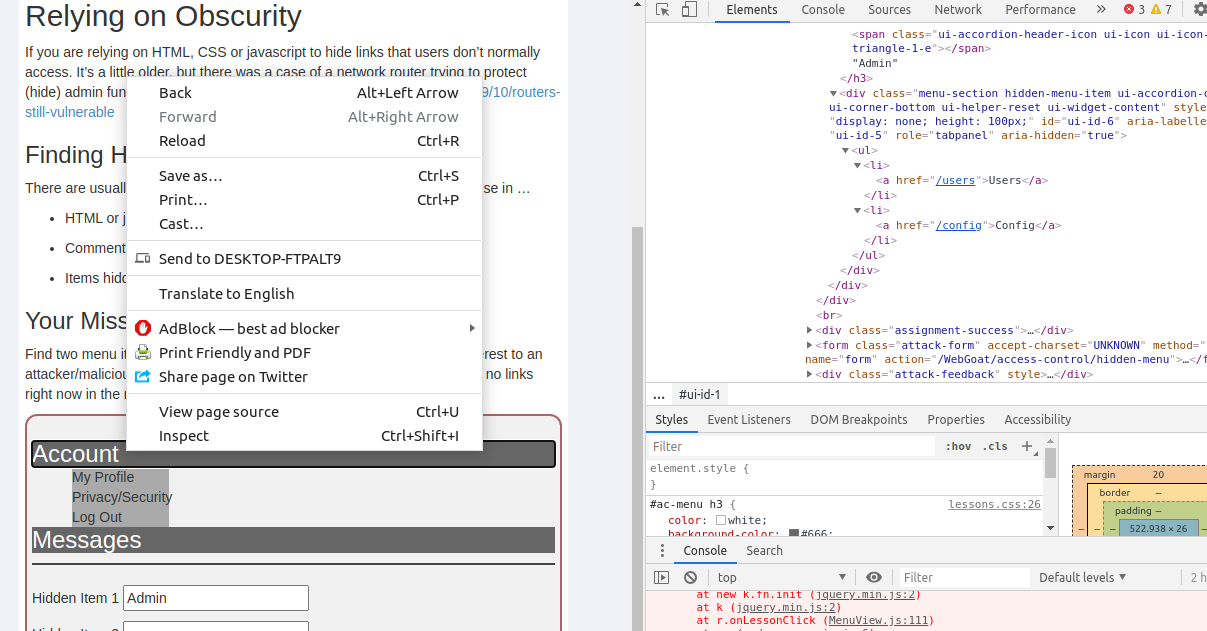
# Broken Access Control

## Missing Function Level Access Control

Starts off with the absolutely must cover dictate about Security Through Obscurity is a horrible idea. It provides no real coverage because the wisdom of the crowds works against you. There will always be more brainpower on the attack then you can muster on the defense.

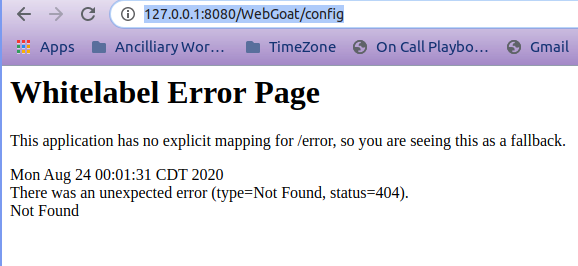


Inspecting that pulls up the code that shows menu items that are just hidden. /users and /config

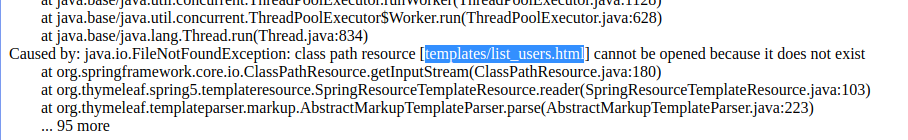


The reference those two endpoints being useful for the next stage.

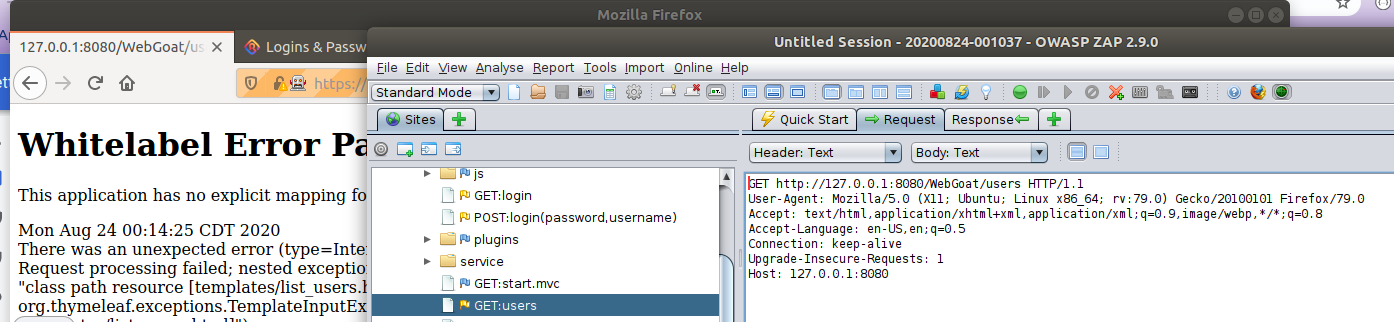
/config looks to be a dead endpoint. Returns a 404.



/users throws an error so I'm thinking that might be something. Bombs out because a file is not found.

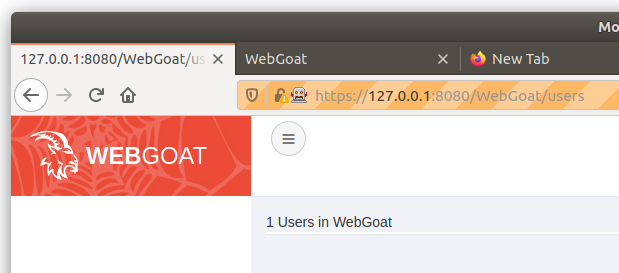
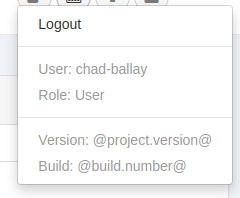


But I can't seem to figure out how to make use of that. Starting to flail and decided to inspect that actual call. It's a plain GET with very little else.



Editing it to be a PUT, TRACE, etc... returns a 500 error. A POST redirects to the login page. I gave up and started googling for the solution. I'm getting an error that doesn't jive with other people's writeups. Not able to proceed currently. <https://youtu.be/eTlTAM6oaFg?t=60>

If I downgrade to a previous version, it starts working. <https://github.com/WebGoat/WebGoat/releases/tag/v8.0.0.M26>



Was able to follow those example solutions but no real understanding of how it works. Changing Accept and Content to JSON.

Accept: application/json,\*/\*;q=0.8

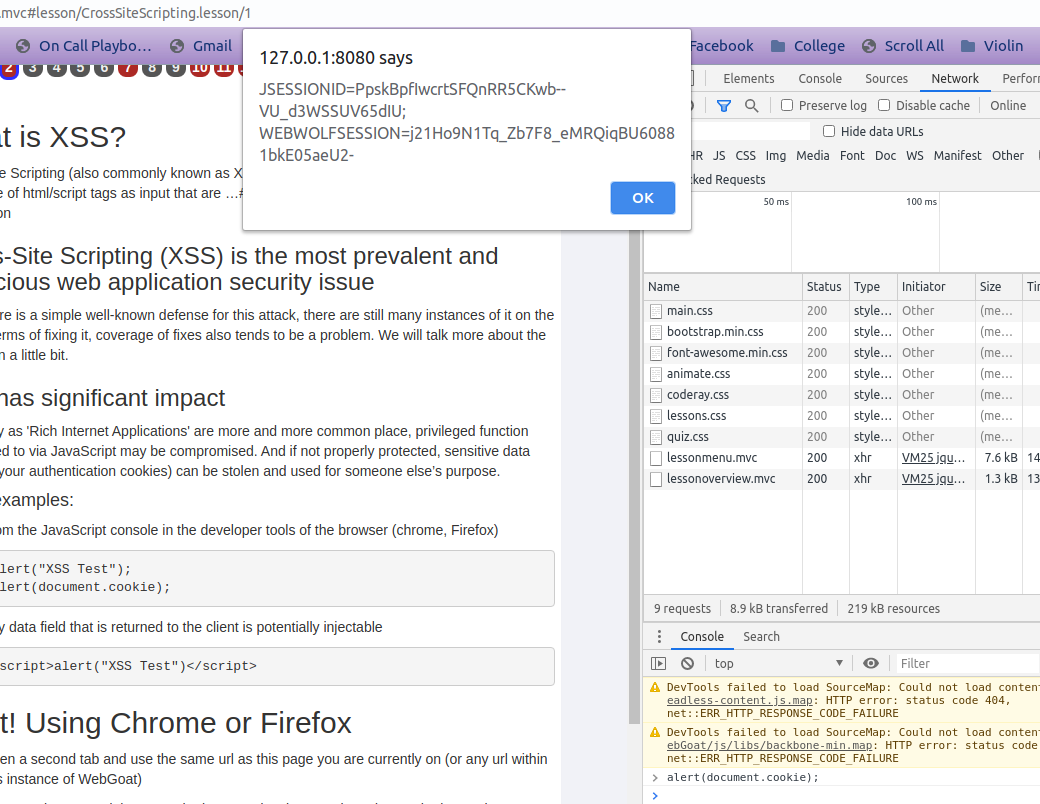
Content-Type: application/json;charset=UTF-8

# Cross-Site Scripting (XSS)

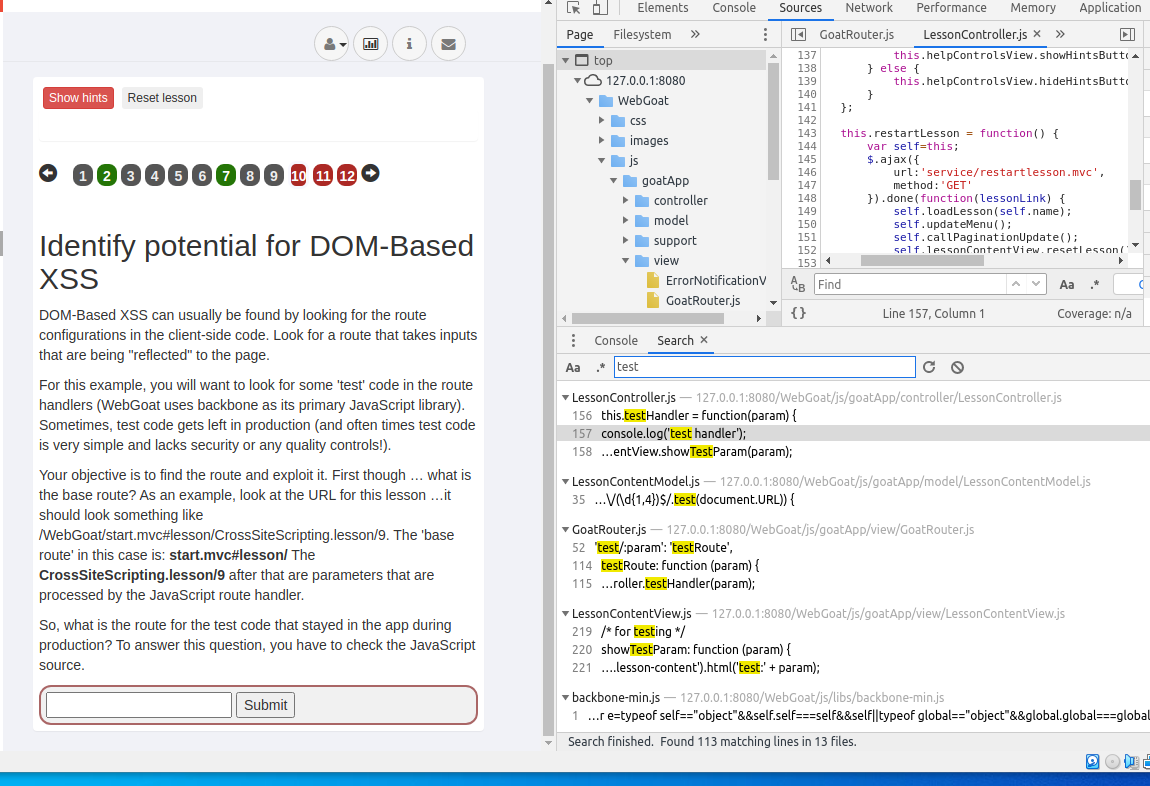
## Cross Site Scripting

Side Note: The lack of consistency with whether Cross-Site or Cross Site in two menu options right next to each other is annoying. Whether it should or shouldn't be hyphenated doesn't matter, they just should be consistent. Wasted an hour trying to figure out how to make Google Searches respect punctuation in exact searches to see which one is the correct way by vote of the crowd.

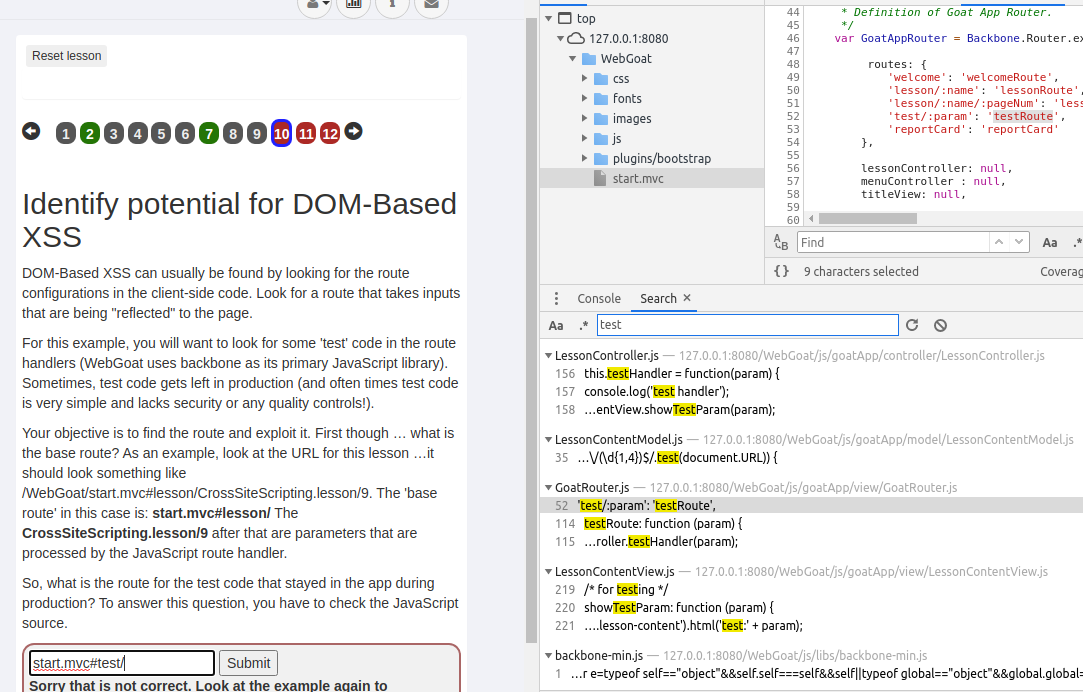
Demonstration that different tabs have access to the same cookie. (Reading up on this likened it to variable scope a little in my head. Session vs Persistent also came up.)



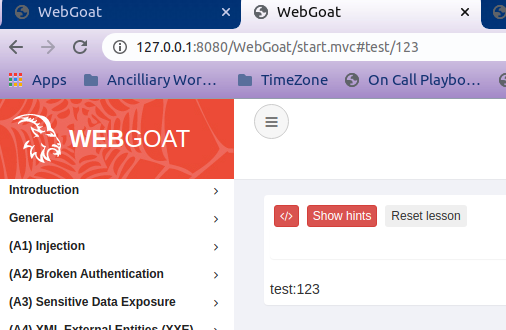
For the next one it took a bit of trial and error. It gave the hint to use either console.log() or alert(). Since the input fields for the quantity were set to type="NUMBER", they wouldn't accept text. That left the CC and Access Code field. Editing the CC field would get the webpage to display the console.log("Test") string on the page but it wasn't being interpreted. Wasn't until I thought it through and wrapped it with a <script></script> tag that I remembered. Javascript code needs to tell the browser what it is.

This next one starts with a reference to test code that made it to production. For that I viewed the sources and started looking for the string "test". Dumb but simple. 

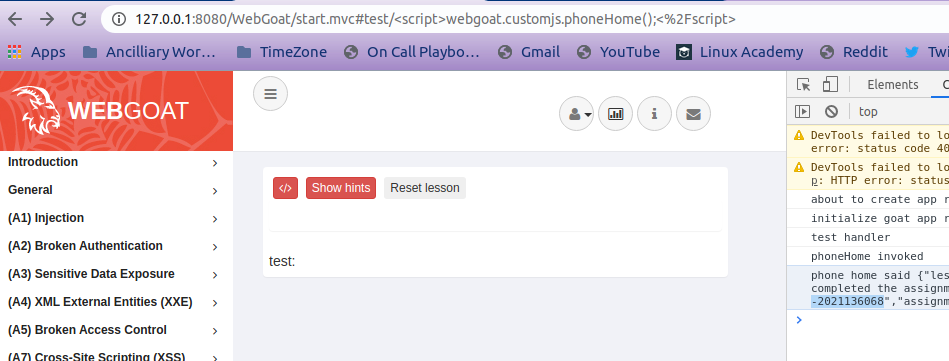
From there I started trying to pick apart the code to work my way back to where the routes were defined that would execute that code. Found the definition eventually. Then from their I had to figure out what the code was asking for vs what I was seeing.



This next one I stumbled into while figuring out the previous one. While trying to figure it out I noticed that whatever I typed after the #test/ part of the url ended up being displayed.



So for this one I'll just put <script>webgoat.customjs.phonehome()</script> and be done. Or so I thought. Didn't load the random number into the console. Beat my head against it over and over and gave up. Turns up the answer was staring me in the face. I needed to urlencode the backslash. "/" vs "%2F".....



(So close yet so far for solving this on my own. (Lesson learned that urlencoding is a must understand on when it is needed.)