

WebGoat Missing Function Level Access Controls

Twitter: @BlackSheepSpicy

Twitch: <https://twitch.tv/BlackSheepSpicy>

2. Oh good, more digging through code, thanks OWASP!

Your Mission

Find two menu items not visible in menu below that are or would be of interest to an attacker/malicious user and put the labels for those menu items (there are no links right now in the menus).

Account

My Profile

Privacy/Security

Log Out

Messages

Hidden Item 1

Hidden Item 2

Submit

I did my best to show where these hidden items are, unfortunately it just comes down to digging through code, How I found it was I hit inspect element on the account menu and worked around there:

```
<li>
  <a href="/users">Users</a>
</li>
<li>
  <a href="/config">Config</a>
```



Big thing to keep in mind with this challenge, they're looking for the **label**, not the actual endpoint:

Account

[My Profile](#)[Privacy/Security](#)[Log Out](#)

Messages

✓

Hidden Item 1

Users

Hidden Item 2

Config

Submit

Correct! And not hard to find are they?!? One of these urls will be helpful in the next lab.

3. Building off the last exercise:

Just Try It

As the previous page noted, sometimes apps rely on client controls. to control access (obscurity). If you can find items that don't have visible links, just try them, see what happens. Yes, it can be that simple!

Gathering User Info

Often times, data dumps from vulnerabilities such as sql injection, but they can also come from poor or lacking access control.

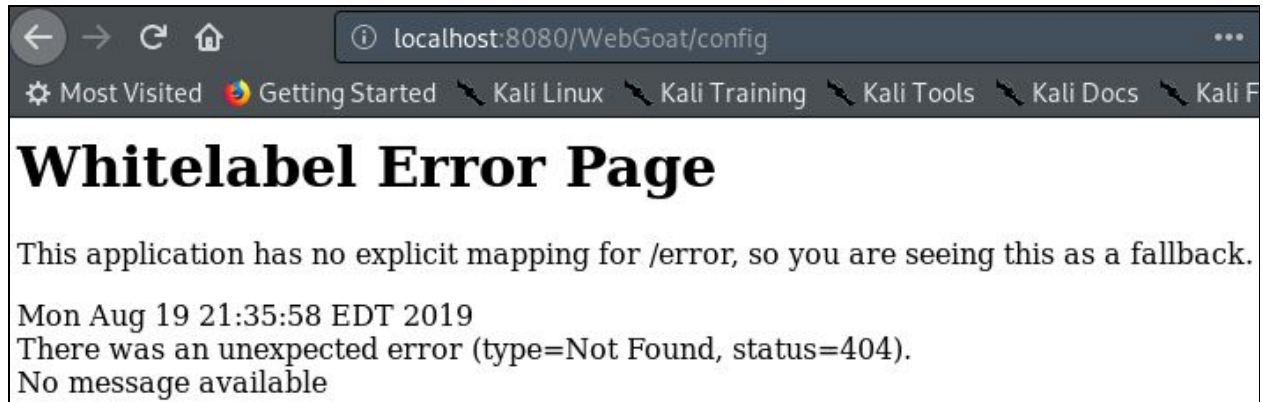
It will likely take multiple steps and multiple attempts to get this one. Pay attention to the comments, leaked info. and you'll need to guess some. You may need to use another browser/account along the way. Start with the info. you already gathered (hidden menu items) to see if you can pull the list of users and then provide the 'Hash' for your own user account.

Your Hash:

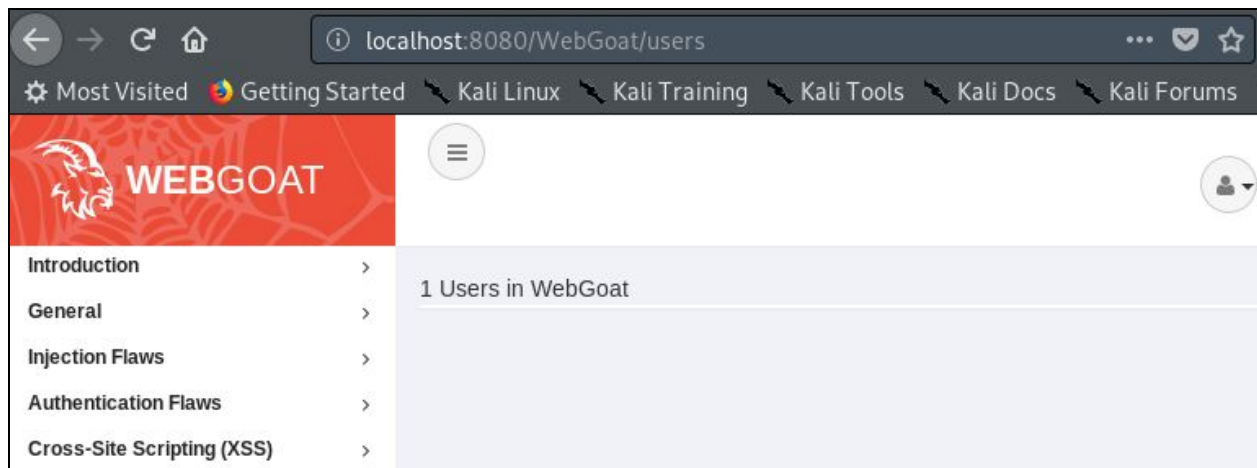
Submit



So remember those endpoints we discovered in the last challenge? They come into play here. Let's take a look at those endpoints:

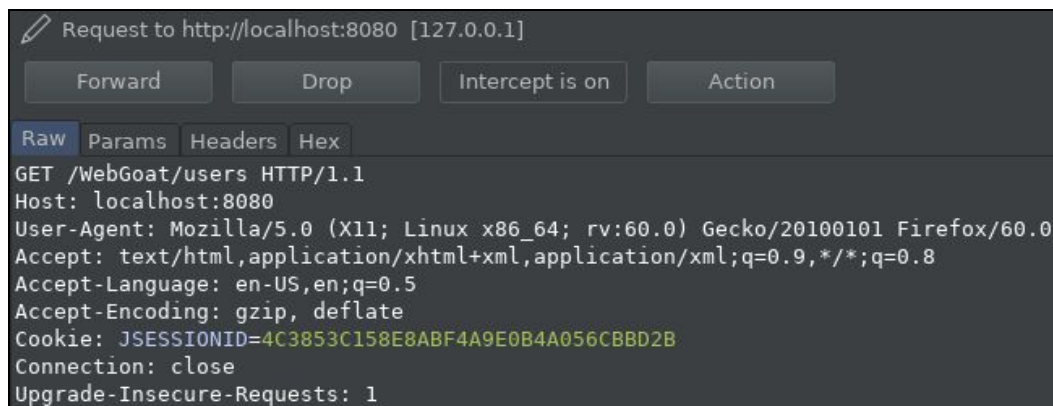


K, what's the other one?



Hey rad its actually a valid endpoint, we can play around with this a bit but honestly I fully permit reading the hints on this one because this one is **weird**.

Start by intercepting the get request to the users endpoint:



Now get this shit, I have absolutely no idea why this works... at all... but if you set a **content-type** header to **application/json** and suddenly it returns the user hash OWASP is looking for:

```
JSON  Raw Data  Headers
Save  Copy
0:
  username: "blacksheepspicy"
  admin: false
  userHash: "EYWgvrA/XIN9mbs8QgoNPPLCVz0qjiZbT/LTxHE0b5Q="
```

Seriously why does this work?

Hope you enjoyed this writeup and if you want to see me attempt these challenges live be sure to drop by my Twitch when I'm live and also follow my Twitter for some quality shitposting!

