

BURPSUITE SETUP

BY JAMES ROBERSON



PROFESSIONAL | CYBERSECURITY | MARCH 9, 2024

WHAT HAPPENED?

Our focus here today is to use Burpsuite to connect to juice-shop (an intentionally vulnerable website, enriched with clues, and many challenges to solve; each showcasing a particular skillset). Firstly, I began with Burpsuite installation. In my steps, I learned that installing Burpsuite isn't always as simple as `sudo apt install Burpsuite`. The installation that the Kali Rolling repository pulls from didn't work for my setup. Each time I would go to click 'open browser' through Burpsuite, I would receive this message," open browser is not enabled". Seeing how this wasn't going away, I decided to reinstall Burpsuite but through the community site, Portswigger.net.

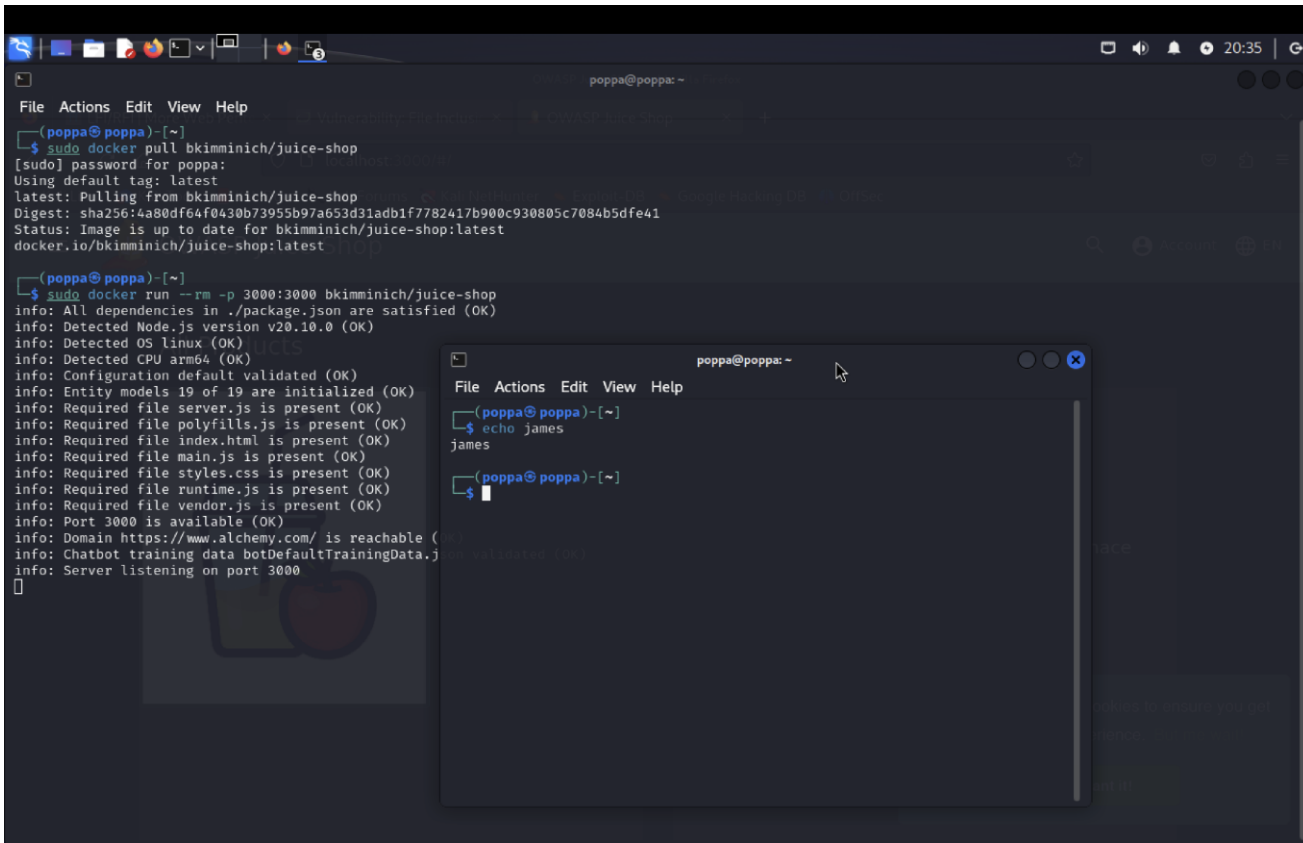
- Installed and configured Burpsuite. Check.
- Enabled 'open browser', checked intercept off, and pulled docker image for juice-shop. Check.
- Turned intercept on and played around a bit. Check and check.

Upon further research, I found that juice-shop isn't just endowed with many challenges, but that those challenges and your progress of each are championed on a scoreboard hidden somewhere in the site. One must poke around to find any clues, hints, or suggestions that may lead to its discovery.

There is also an authentication challenge that may require bruteforce attempts; Burpsuite is excellent for this due to its ability to intercept requests and forward them back with necessary changes. However, it is the necessary changes we'll make to the TCP request that we are focusing on here. Burpsuite's powerful ability to run payloads against sites according to certain specs will play a big role in this setup/walkthrough.

PROOF:

(PART I)



```
poppa@poppa: ~  
File Actions Edit View Help  
(poppa@poppa)-[~]  
$ sudo docker pull bkimminich/juice-shop  
[sudo] password for poppa:  
Using default tag: latest  
latest: Pulling from bkimminich/juice-shop  
Digest: sha256:4a80df64f0430b73955b97a653d31adb1f7782417b900c930805c7084b5dfe41  
Status: Image is up to date for bkimminich/juice-shop:latest  
docker.io/bkimminich/juice-shop:latest  
  
(poppa@poppa)-[~]  
$ sudo docker run --rm -p 3000:3000 bkimminich/juice-shop  
info: All dependencies in ./package.json are satisfied (OK)  
info: Detected Node.js version v20.10.0 (OK)  
info: Detected OS linux (OK)  
info: Detected CPU arm64 (OK)  
info: Configuration default validated (OK)  
info: Entity models 19 of 19 are initialized (OK)  
info: Required file server.js is present (OK)  
info: Required file polyfills.js is present (OK)  
info: Required file index.html is present (OK)  
info: Required file main.js is present (OK)  
info: Required file styles.css is present (OK)  
info: Required file runtime.js is present (OK)  
info: Required file vendor.js is present (OK)  
info: Port 3000 is available (OK)  
info: Domain https://www.alchemy.com/ is reachable (OK)  
info: Chatbot training data botDefaultTrainingData.json is validated (OK)  
info: Server listening on port 3000  
[ ]
```

Figure 1. In this step, I pull the juice-shop website using docker and hosted it on my local server, port 3000.

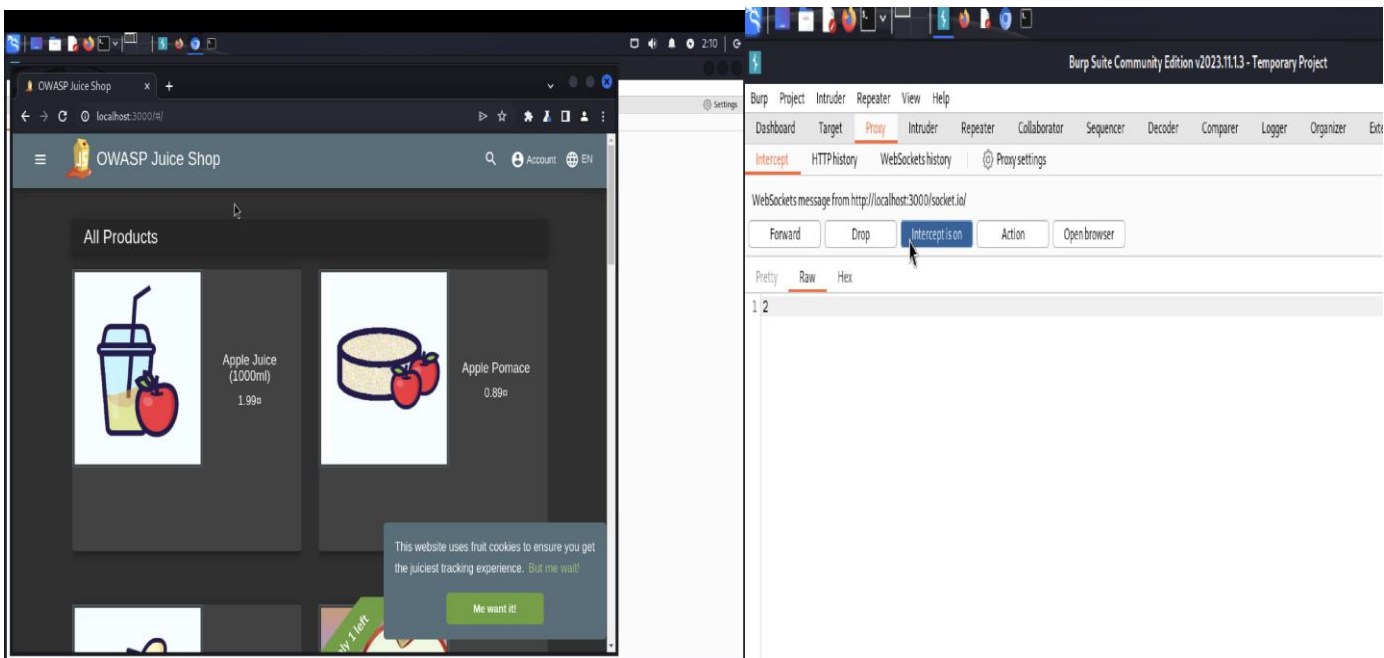


Figure 2.

Here we can confirm that juice-shop is in fact up and running by navigating to our local site at port 3000 like such, `localhost:3000`. With that confirmation, I opened up Burpsuite in my Kali VM, selected default settings, selected 'open browser', navigated to our local server at port 3000, and toggled 'intercept' to be turned on.

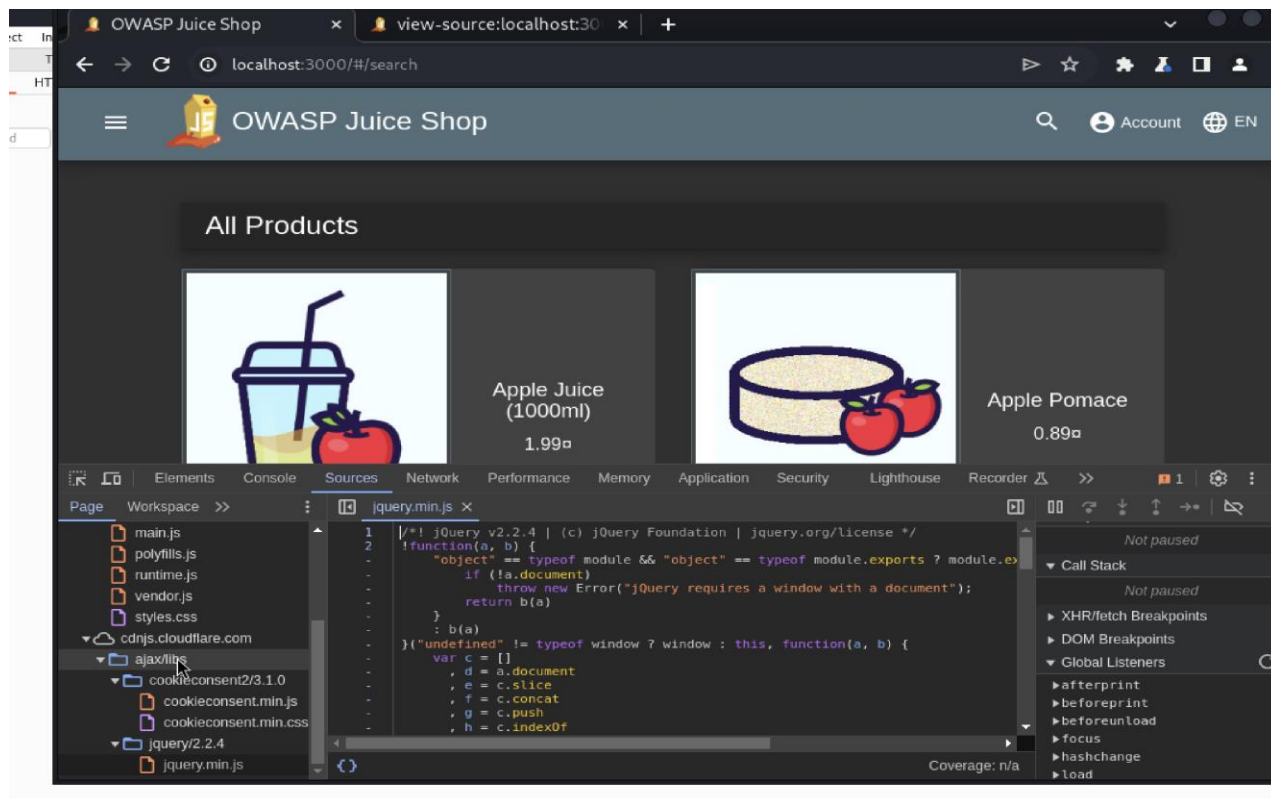


Figure 3.

Having realized I turned my intercept on prematurely, I went back to toggle it off and then deep dove into juice-shop to find anything useful. First stop, site's source code.

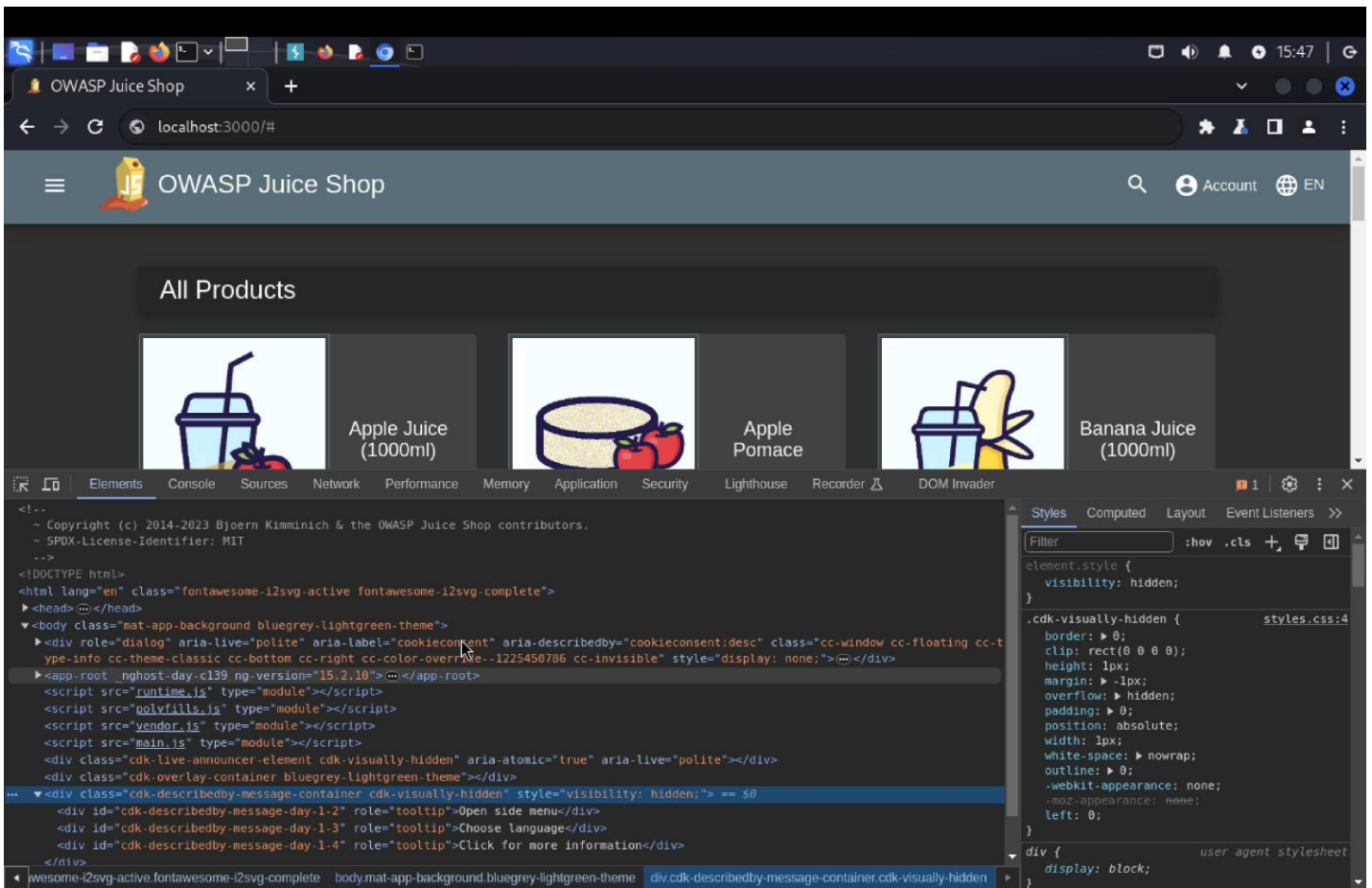


Figure 4.

I went cascading through any tab and folder I could find while peering through the source code. However, nothing useful. And when the home site didn't prove to be any further help either, I decided to alter my focus towards the URL.

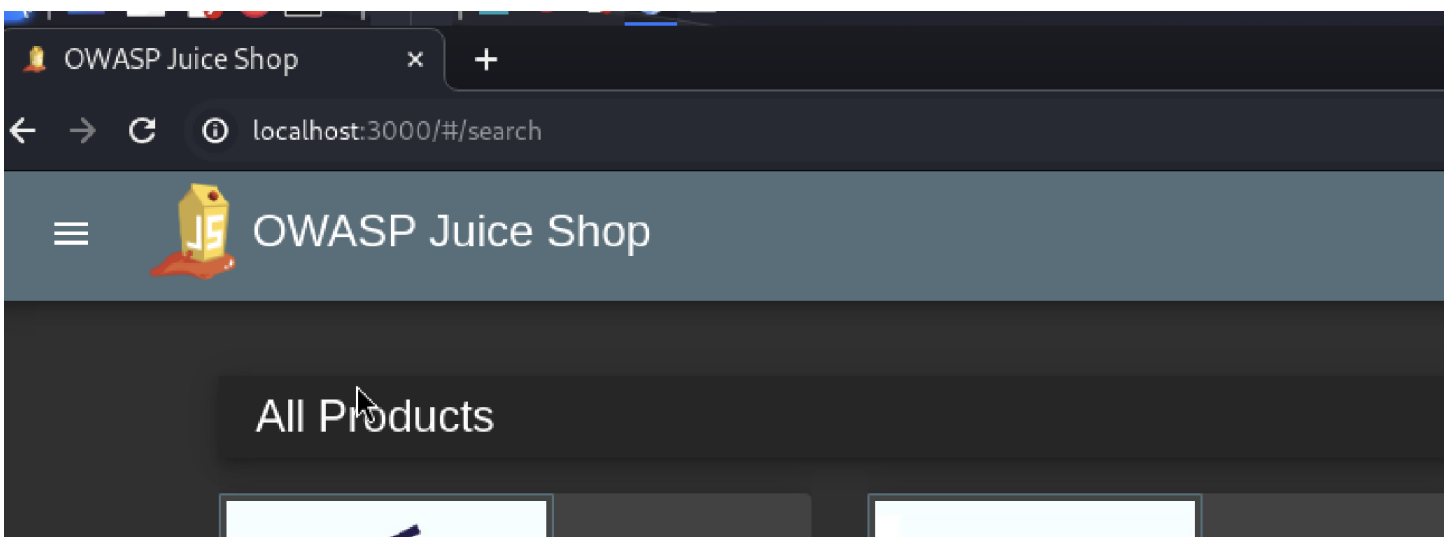


Figure 5. This is the original URL before my input.

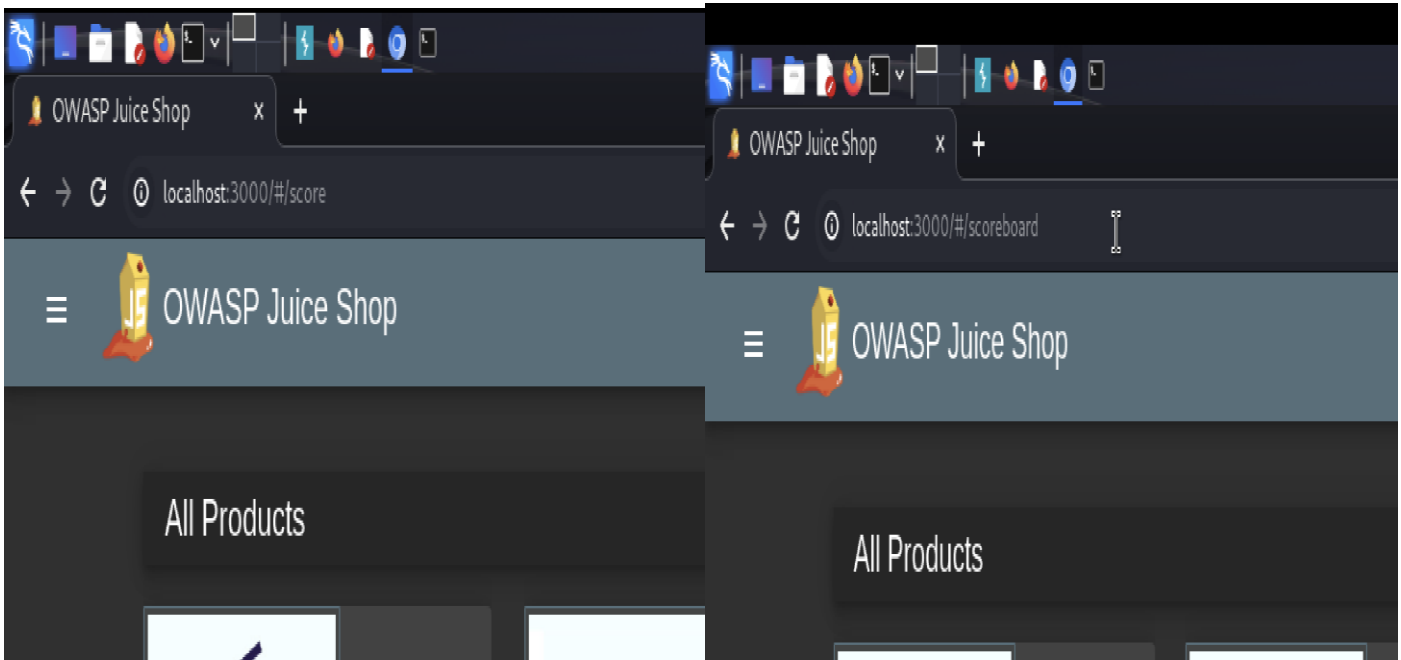


Figure 6. I tried different combinations including the one from Figure 7.

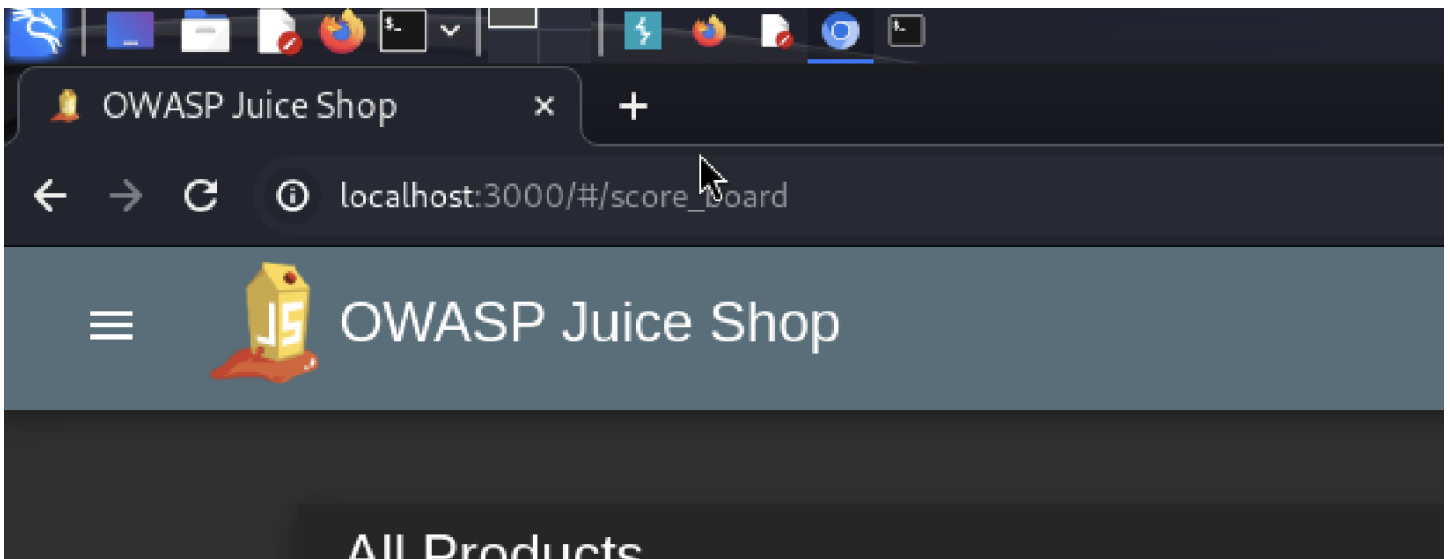


Figure 7. The final input before the actual input that worked.

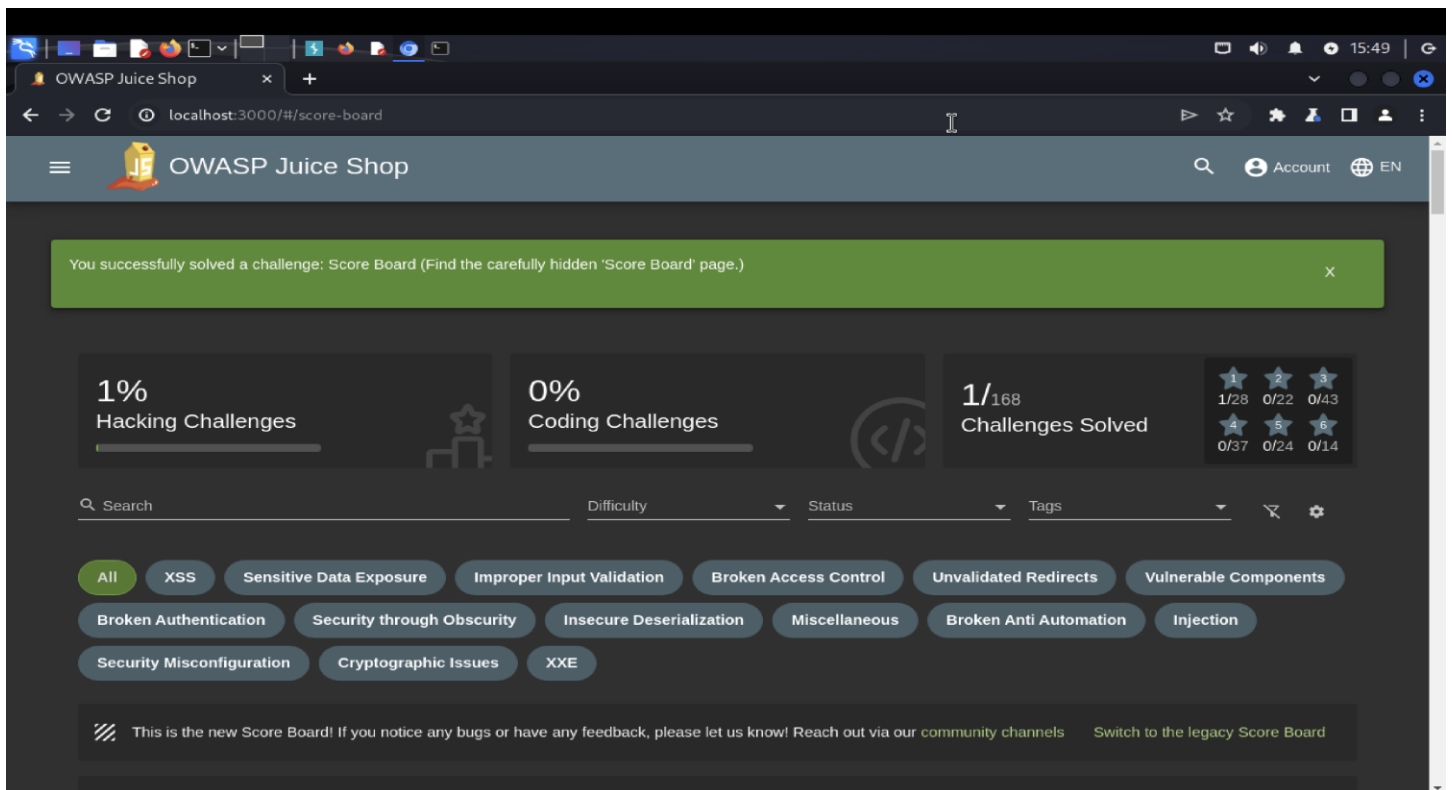


Figure 8. YAHTZEE! Found the score page or whatever.

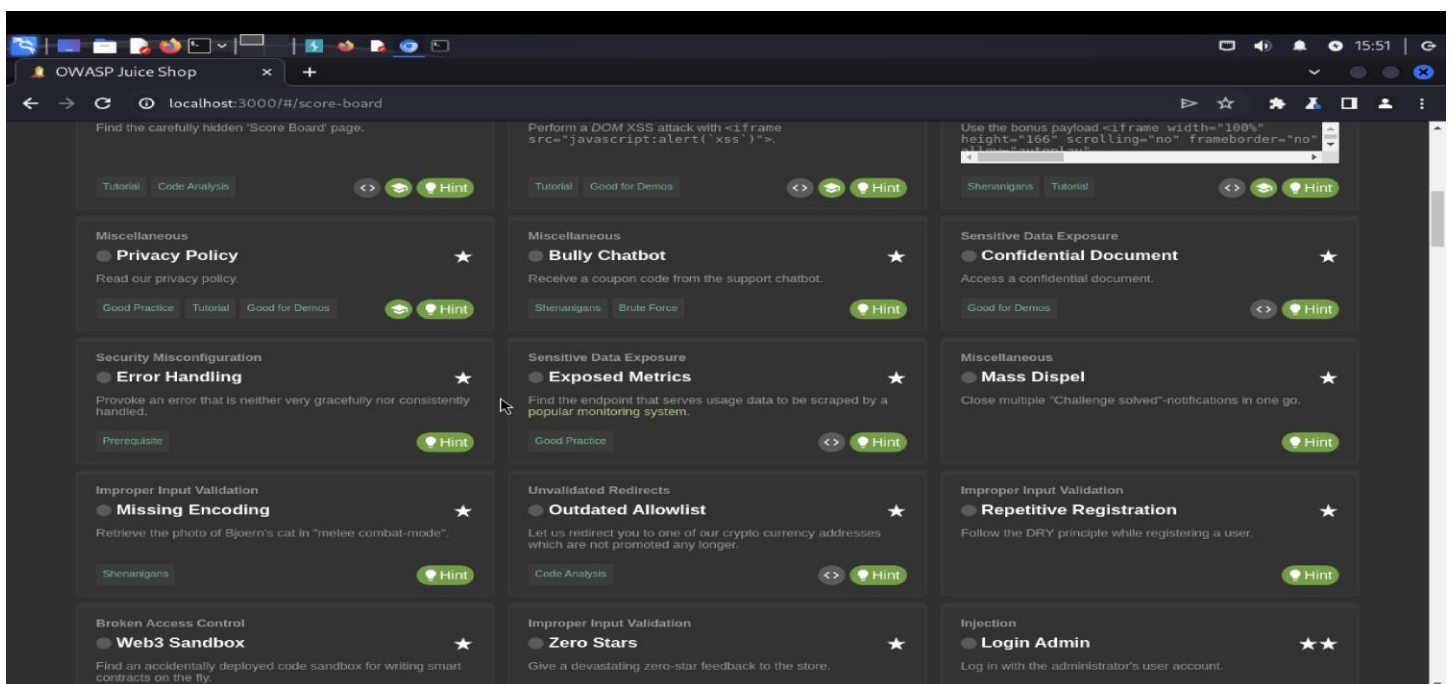


Figure 9.

Here are the many challenges that can be discovered throughout the juice-shop website; the challenges also include hints, explanations, and even topics filtered based on level of security being exploited.

(PART 2)

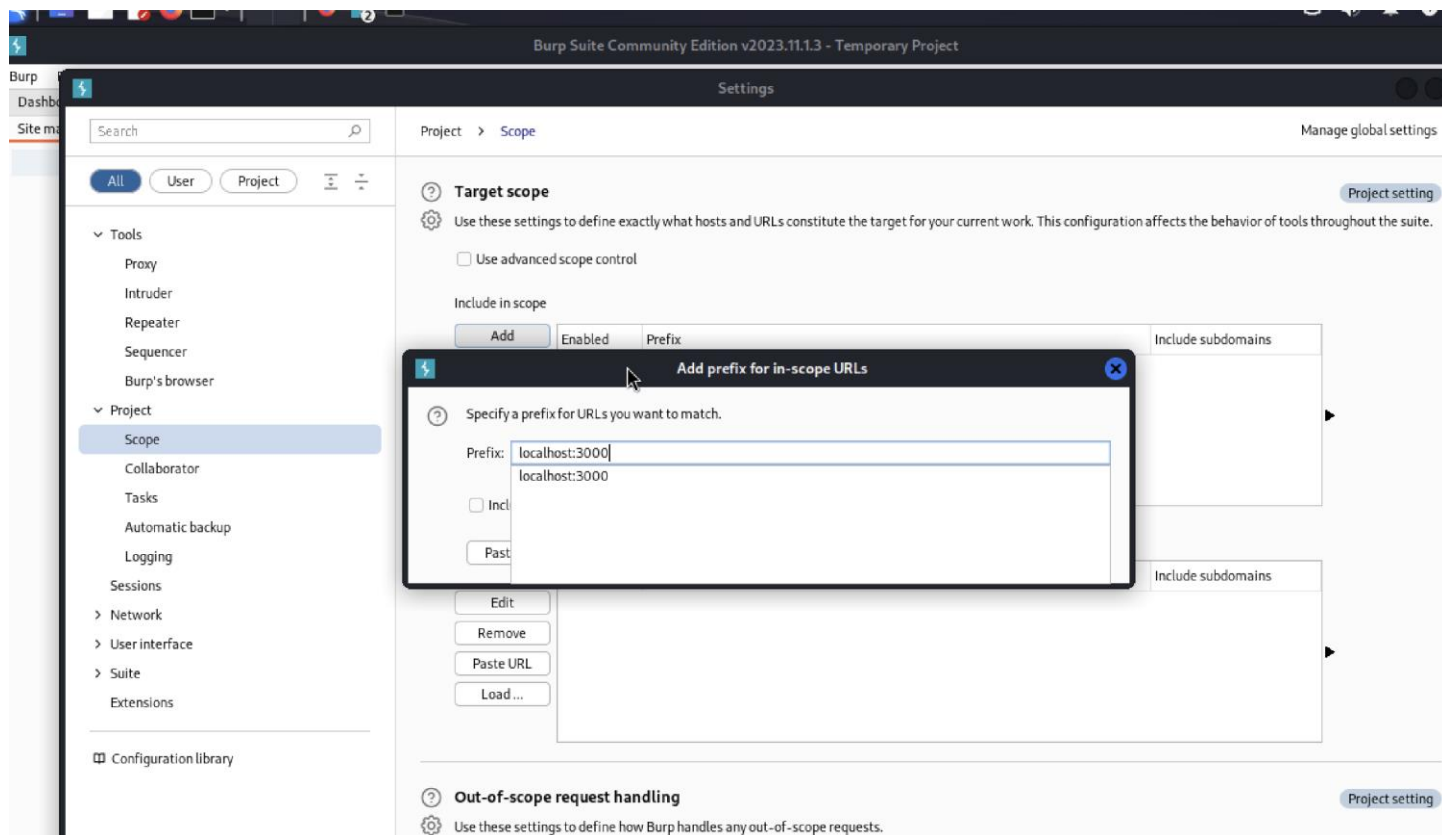


Figure 10.

This section highlights a bruteforce attack taking place on the user `admin@juice-sh.op`. Here, I set the target scope to be my local host since that is where I have juice-shop running; specifically, port 3000.

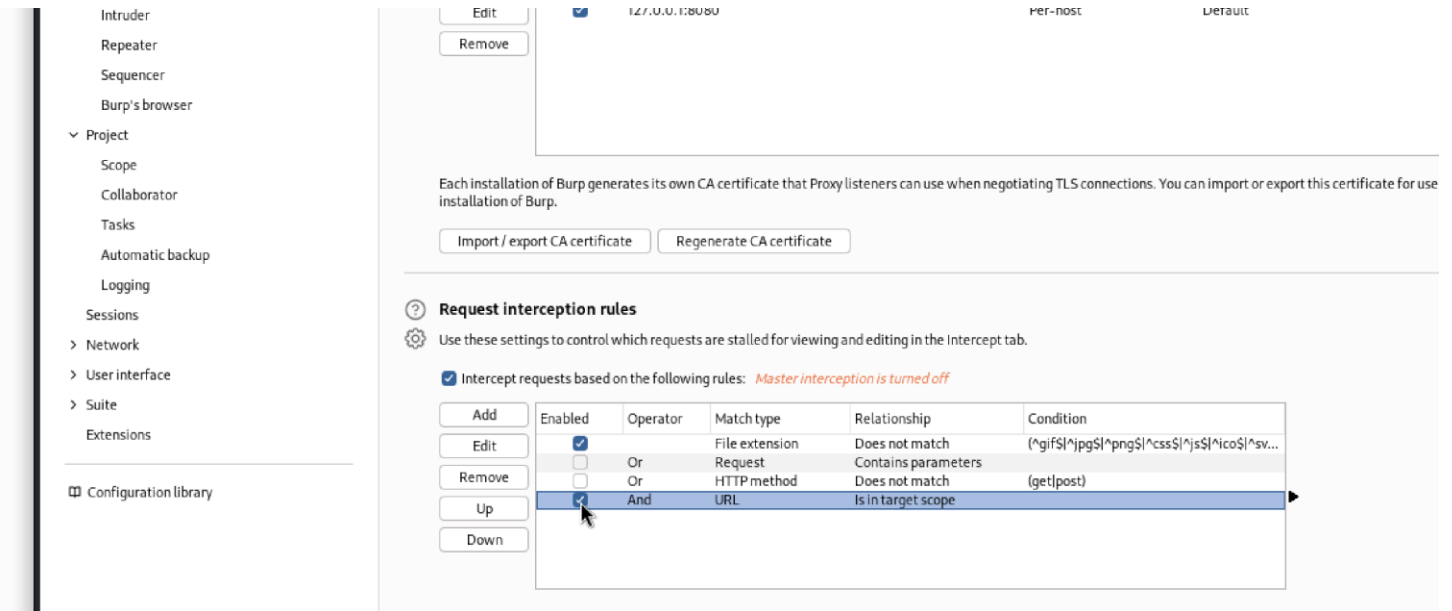


Figure 11.

I also must make sure that I check the box for 'Is in target scope' within the Proxy settings so that requests are not blocked.

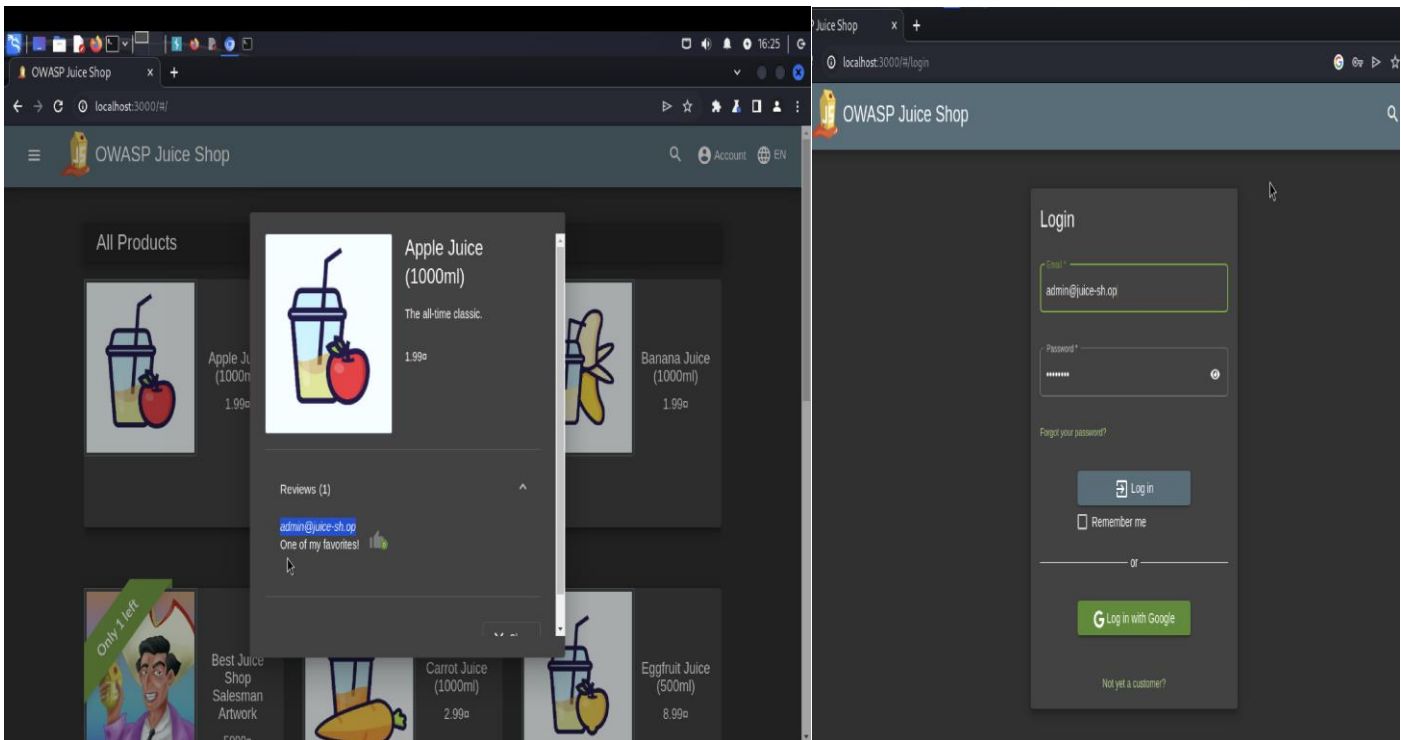


Figure 12. Next, I took the email of the admin user we're looking to exploit and input that data into the login page.

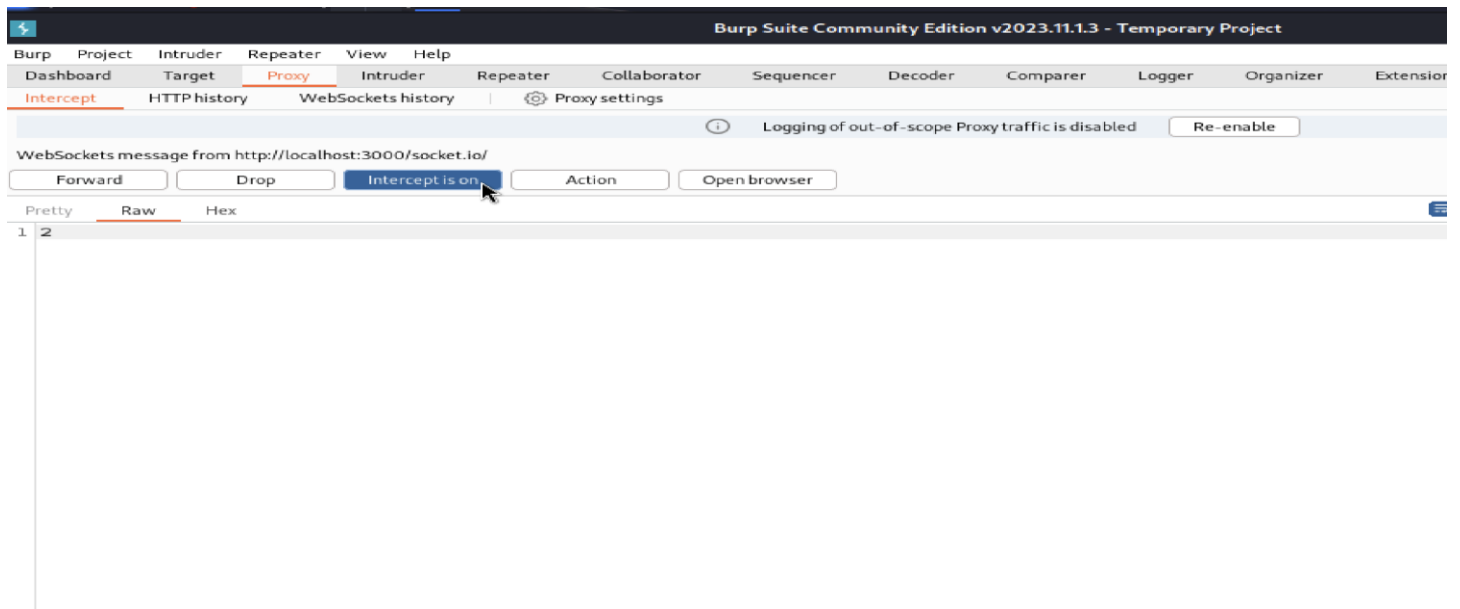


Figure I3. Before clicking ‘Log In’ from Figure I2, I turned on the Intercept.

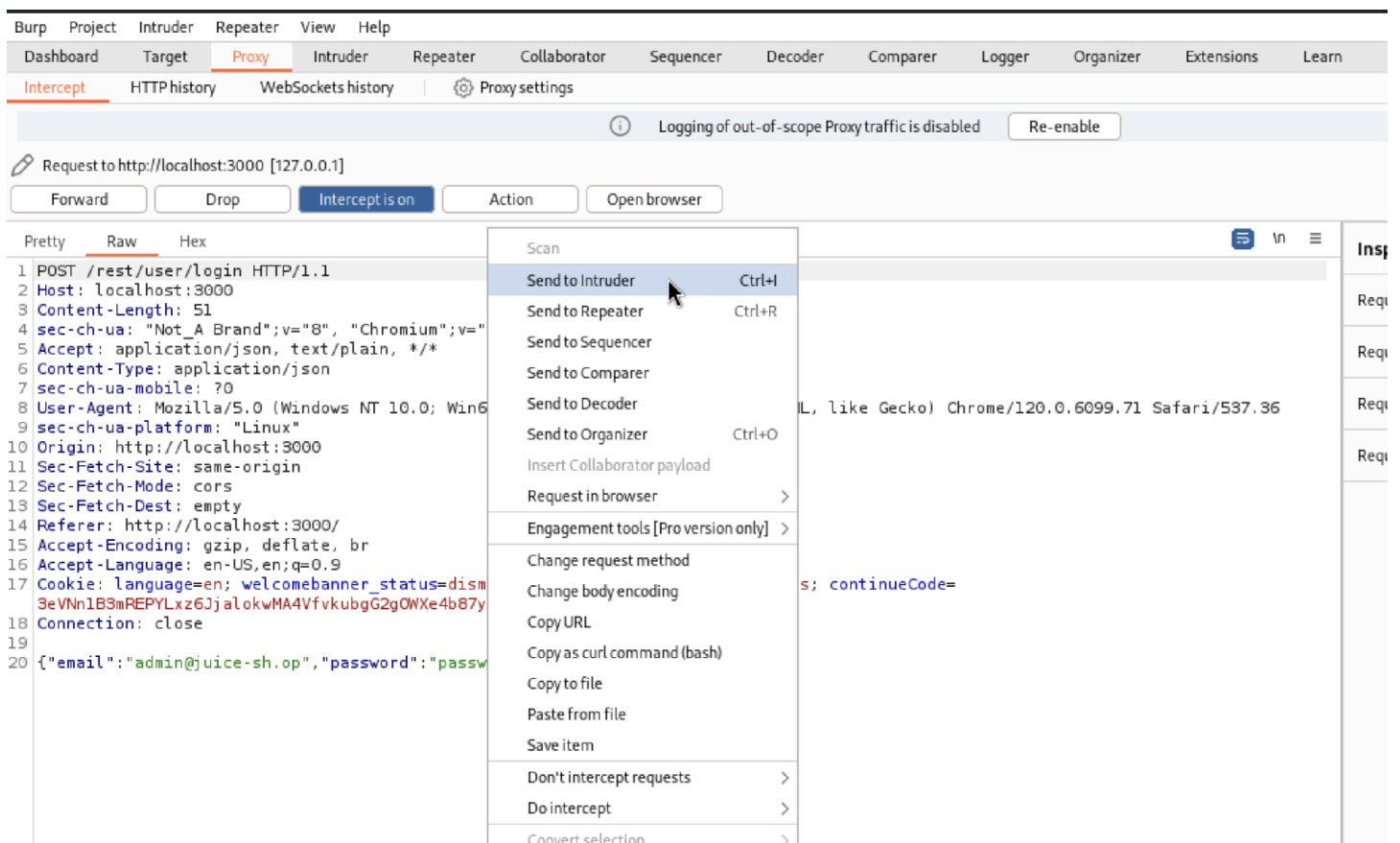


Figure I4. The request filtered into Burpsuite, and then I sent it to the Intruder.



Figure 15.

I went to the Payloads tab within the Intruder tab so that I could Add in the desired payload position, which was 'password'; or in this case, after clicking 'Add', "\$password\$". As shown in Figure 16.



Figure 16. Results from Figure 15.

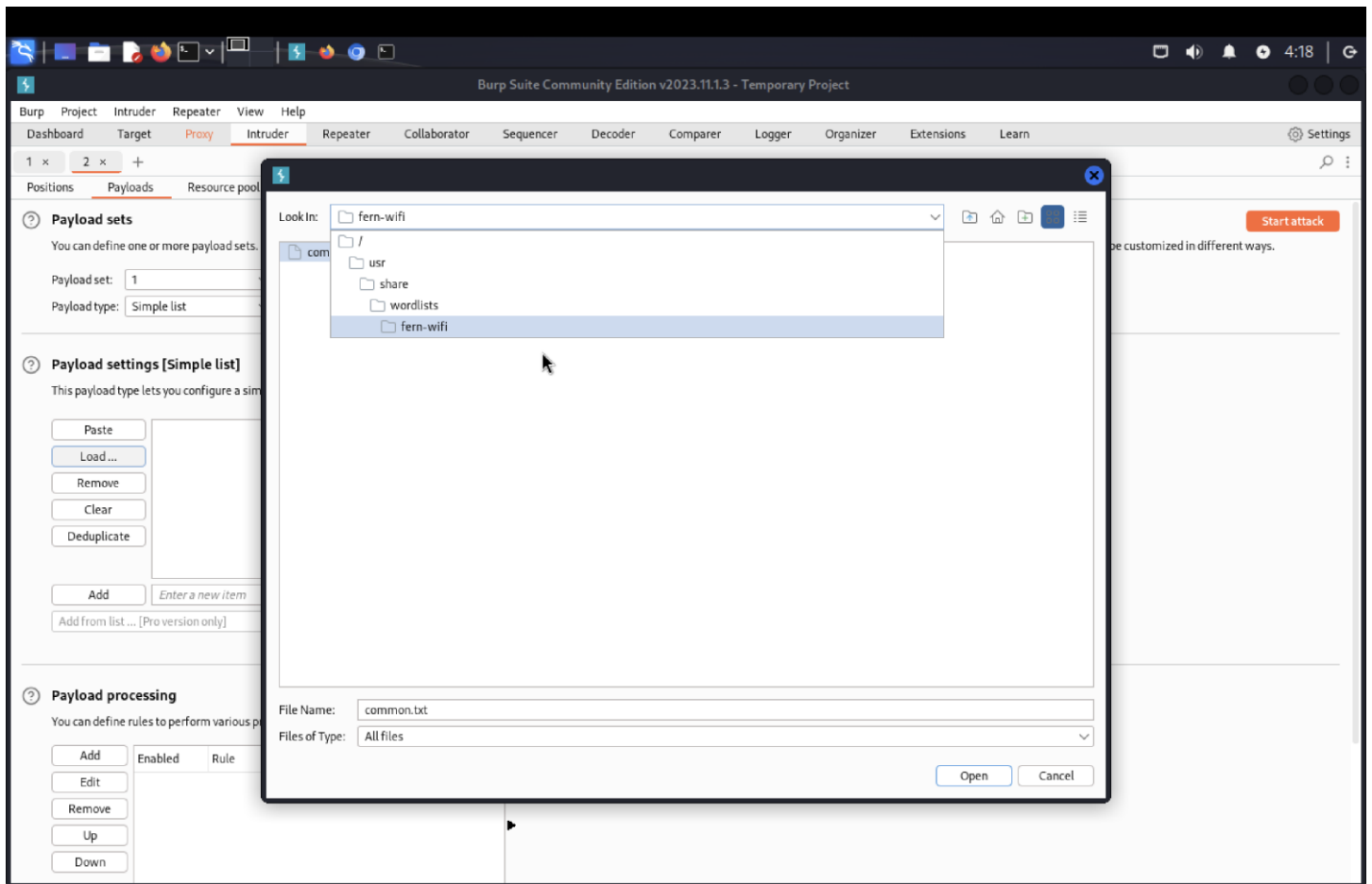


Figure 17.

Next step was setting the actual payload (the list of passwords that will run against the payload position we established in Figure 15). This list is readily available in Kali within the /wordlists/ directory as shown above. The payload I decided for this is commons.txt.

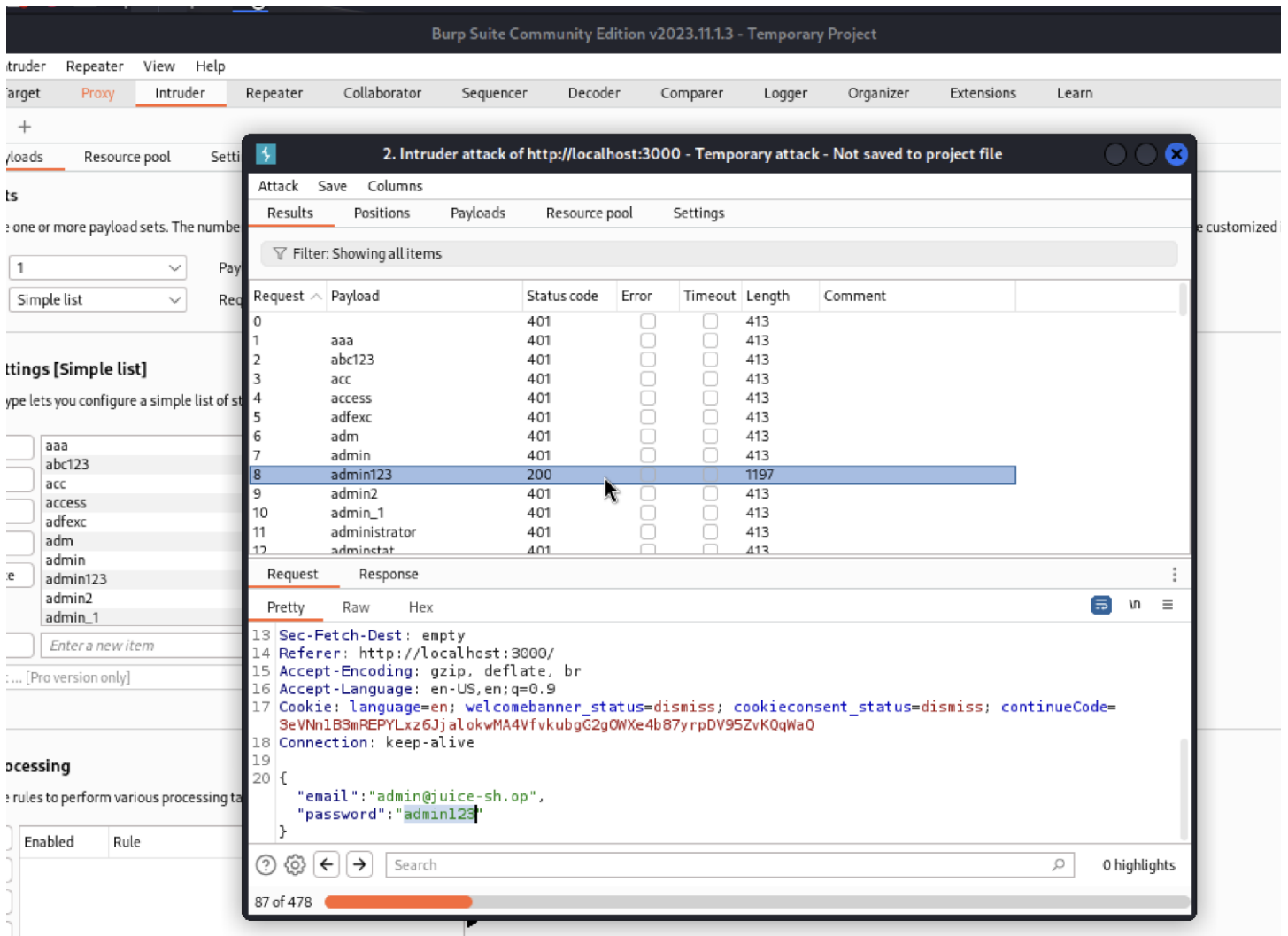


Figure 18.

I selected my payload and clicked 'Start'. As you can see above, Status code:200 was sent for Request 8, admin123. We can confirm that admin123 is the password because in the 'Pretty' tab, under the header 'Connection', it says keep-alive. This otherwise would say close.

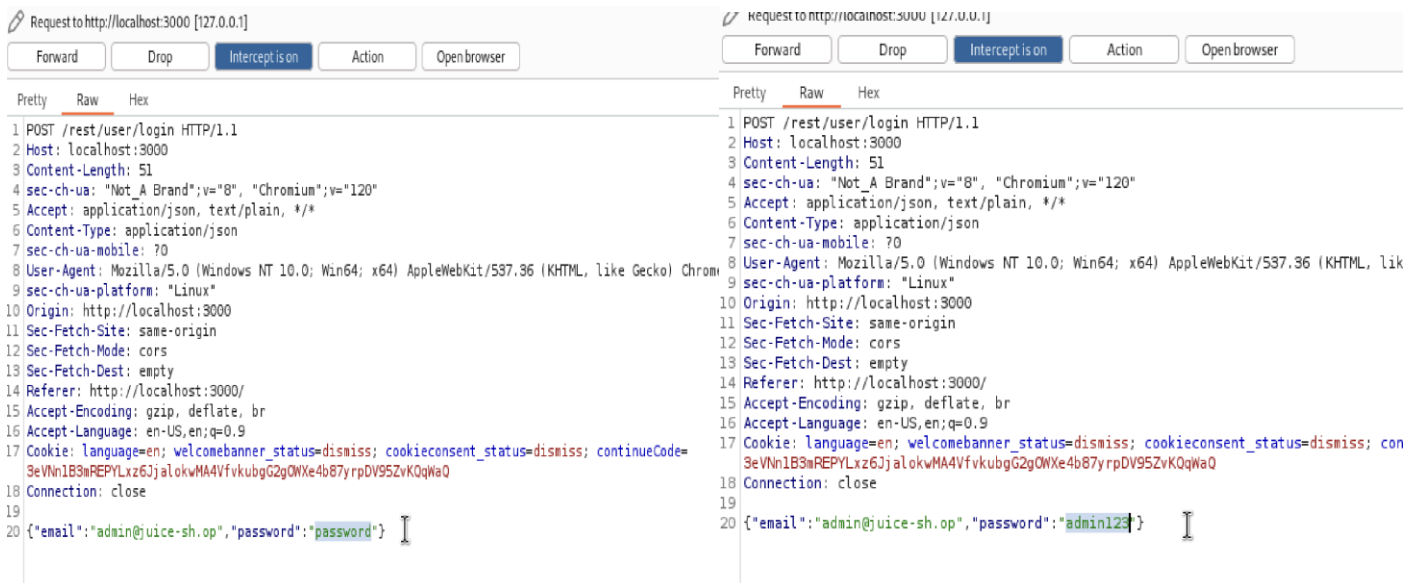


Figure 19.

Finally, I went back to the proxy tab where I had intercepted the initial request, changed the password header from 'password' to 'admin123', and forwarded the request back to my local server at port 3000.

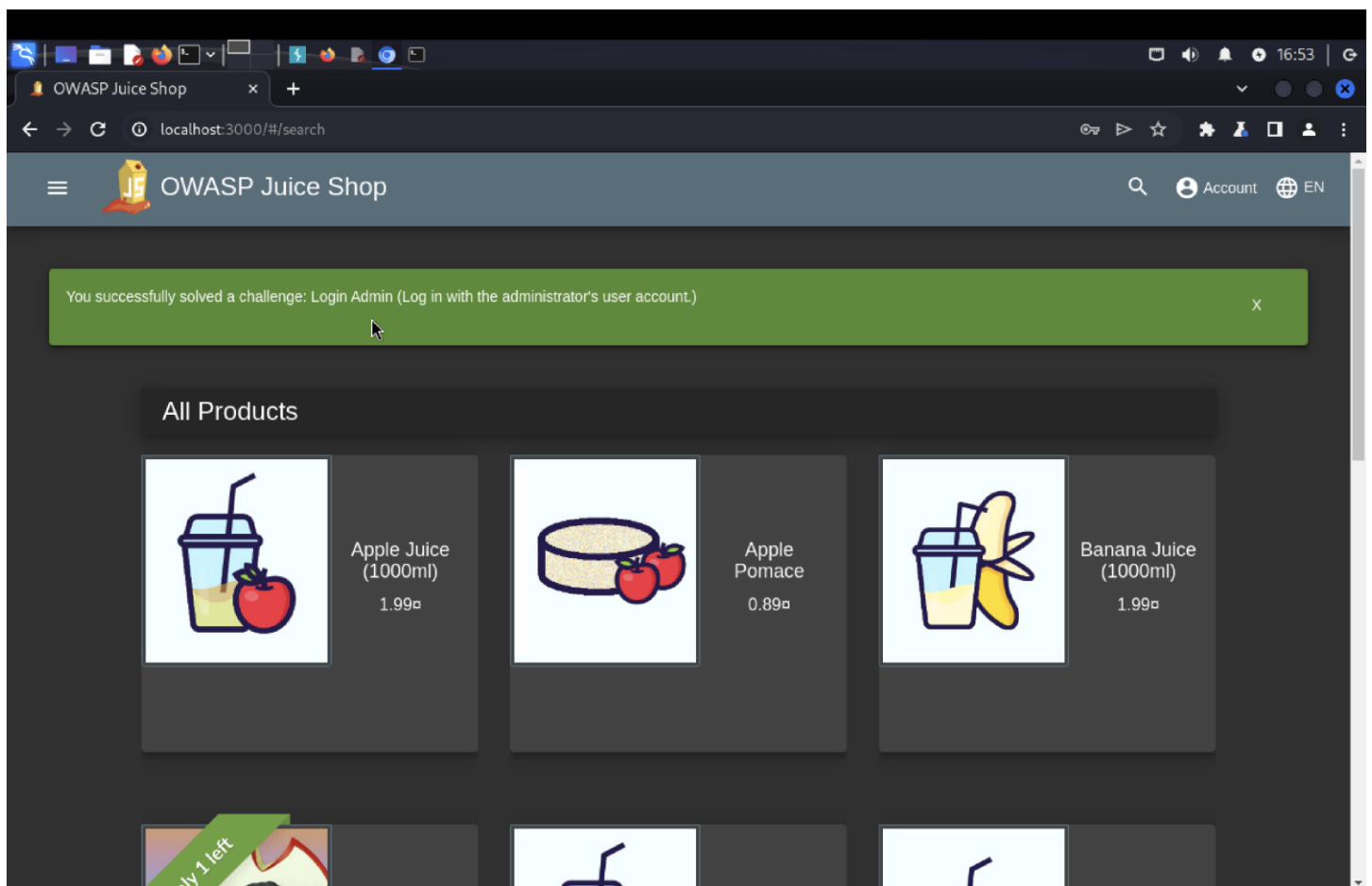


Figure 20. YAHTZEE! Yet another challenge accepted and solved.

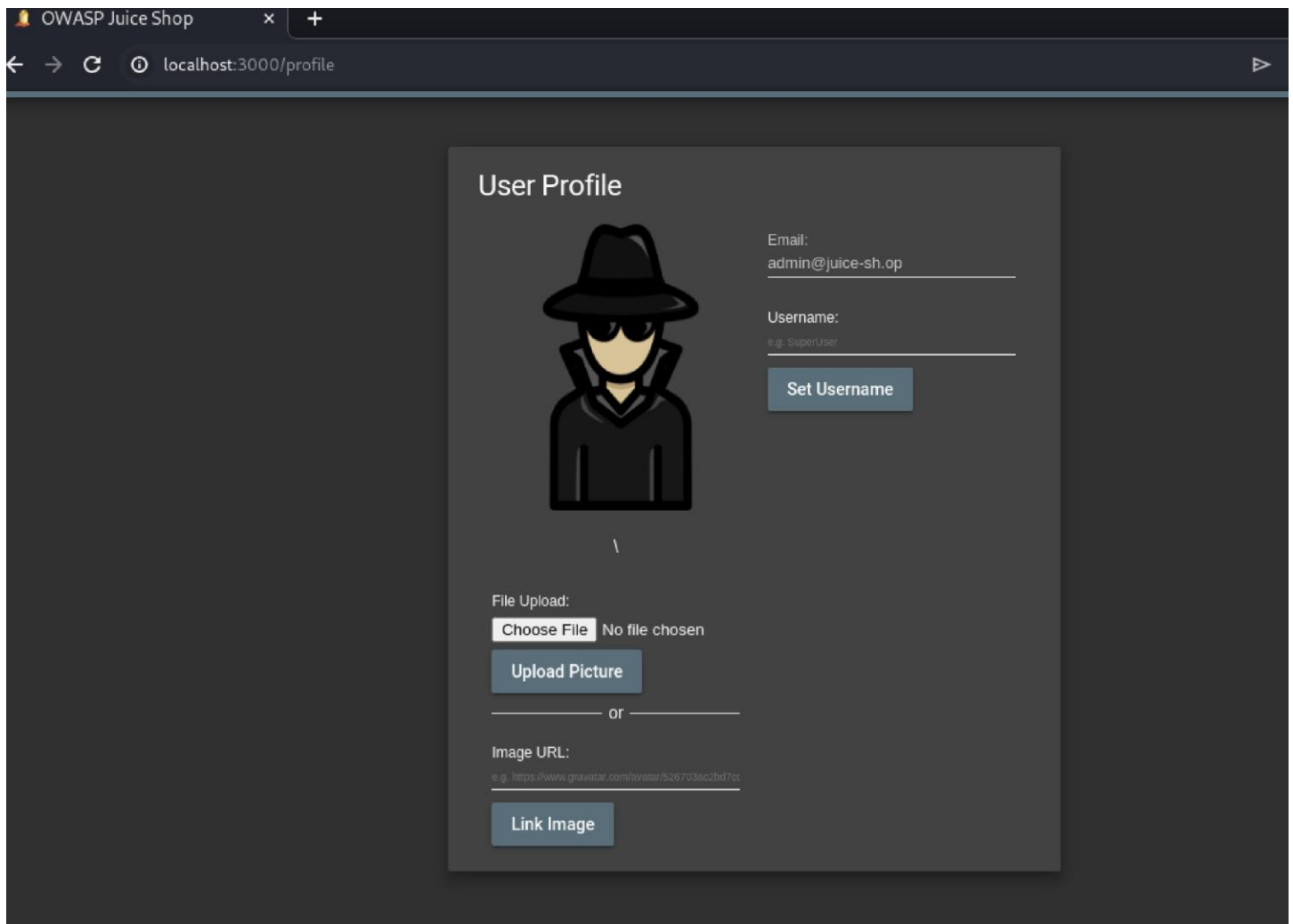


Figure 21. Our Admin Account.