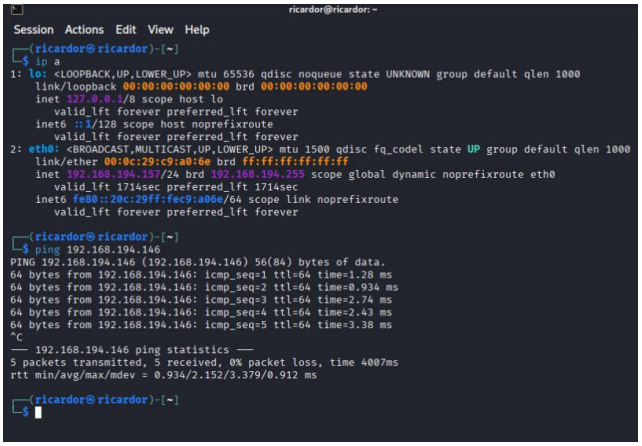
Course	COMP3: Advanced Pentesting
Lab Assignment	THE MEGA HACKING MEGA GROUP PROJECT OF DEATH
Instructor	Adam Abernethy
Student Name	Manuel Manrique Lopez Ricardo Y. Rubin Uriostegui
Submission Date	10/December/2025

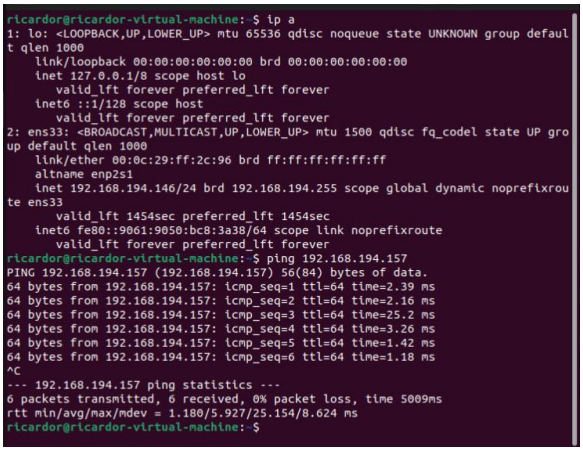
THE MEGA HACKING MEGA GROUP PROJECT OF DEATH

Juice Shop

Network diagram

2 vms





Connection between both vms

vm segment: 192.168.194.0/24 (NAT)

Role	VM Name	IP Address	Relevant Port	Network Adapter
Attacker	Ubuntu	192.168.194.146	8080	NAT
Victim	Kali Linux	192.168.194.157	3000	NAT

VM Specifications

VM Ubuntu – Attacker

IP Ubuntu: 192.168.194.146

Memory: 8.9GB

Network Adapter: NAT

BurpSuite proxy: 8080

VM Kali - Victim

IP Kali: 192.168.194.157

Memory: 8.9GB

Network Adapter: NAT

Juice shop: 3000

Installing docker (old version but works for the lab)

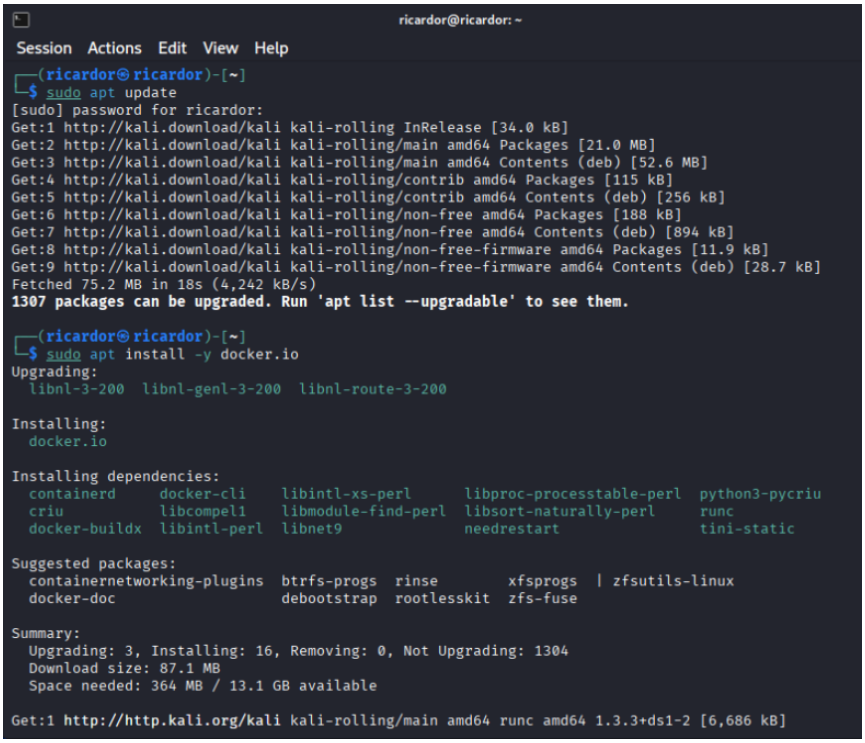


Figure 1 - Installing docker.

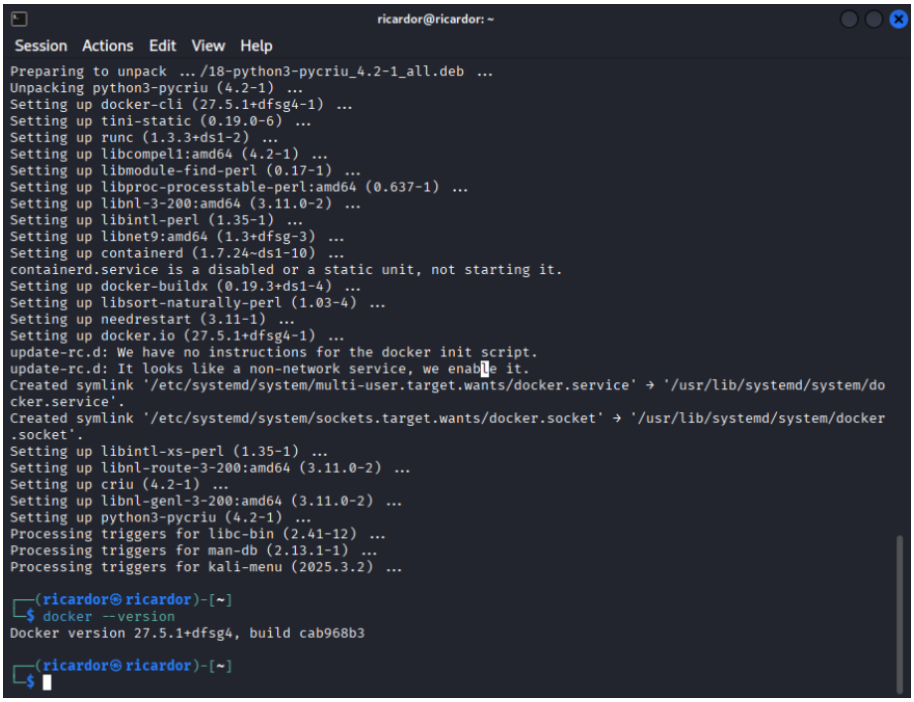


Figure 2 - Confirming dockers was successfully installed.

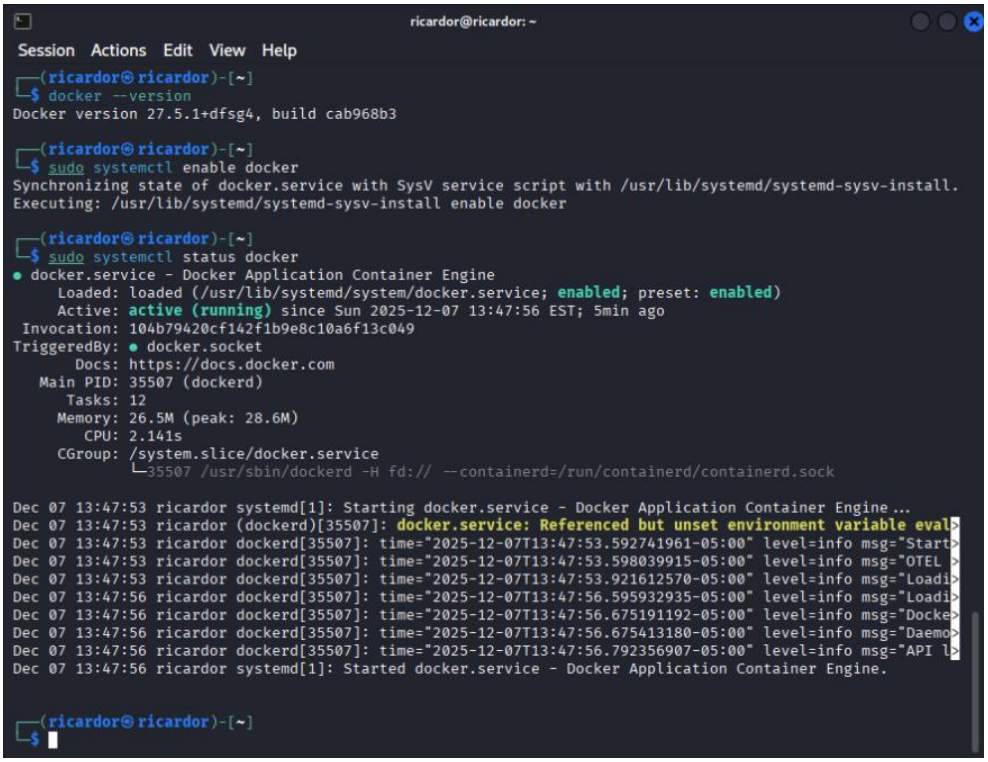


Figure 3 - Starting docker and confirming the status is active.

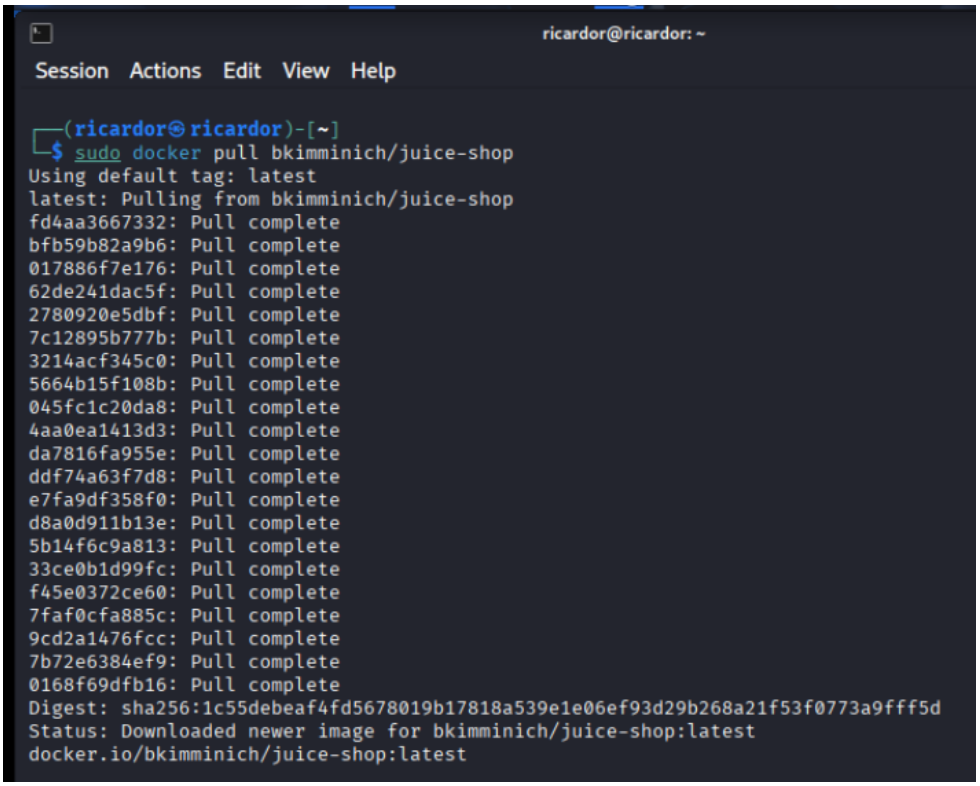
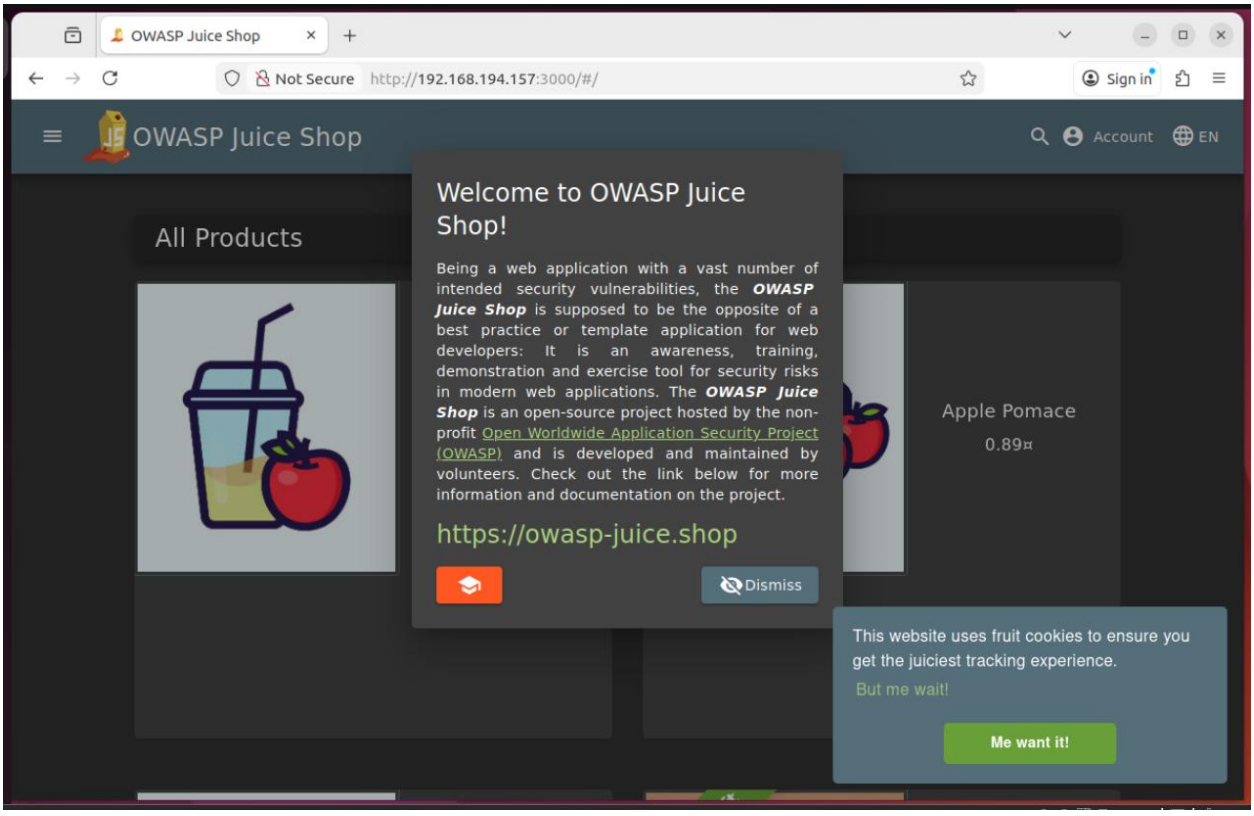


Figure 4 - Pulling and extracting the juice shock docker image.

```
┌──(ricardor㉿ricardor)-[~]
└─$ sudo docker run --rm -p 3000:3000 bkimminich/juice-shop
info: Detected Node.js version v22.21.1 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Entity models 20 of 20 are initialized (OK)
info: Required file server.js is present (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file main.js is present (OK)
info: Required file tutorial.js 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 validated (OK)
info: Server listening on port 3000
```

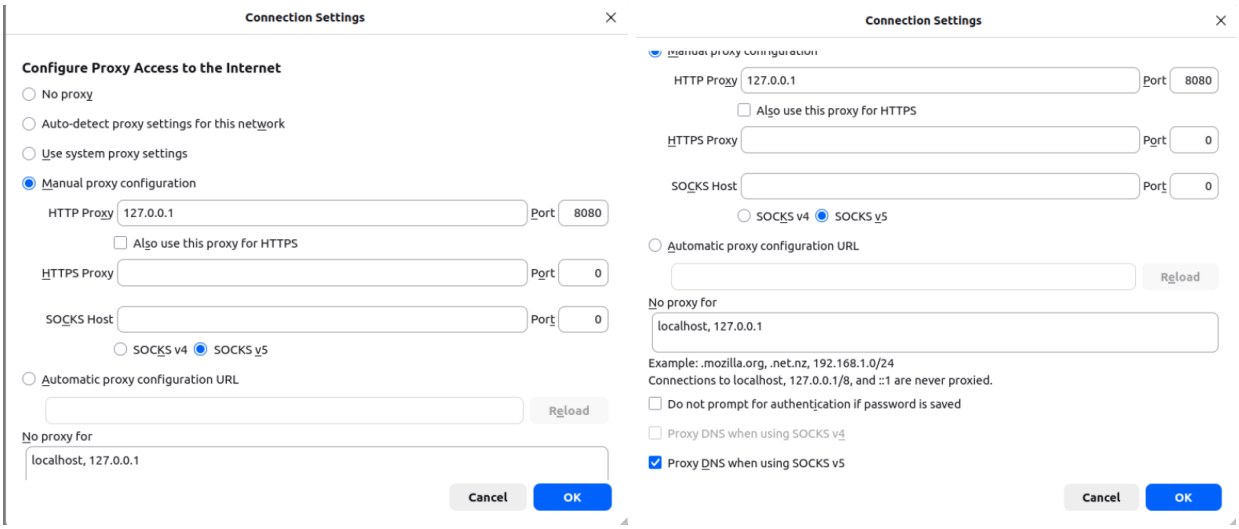
Figure 5 - Running the docker container.

On the attacker machine. I went to http://192.168.194.157:3000



On the attacker machine

Firefox -> Settings -> Network Settings



Then click ok

Now in Firefox we typed: http:192.168.194.157:3000

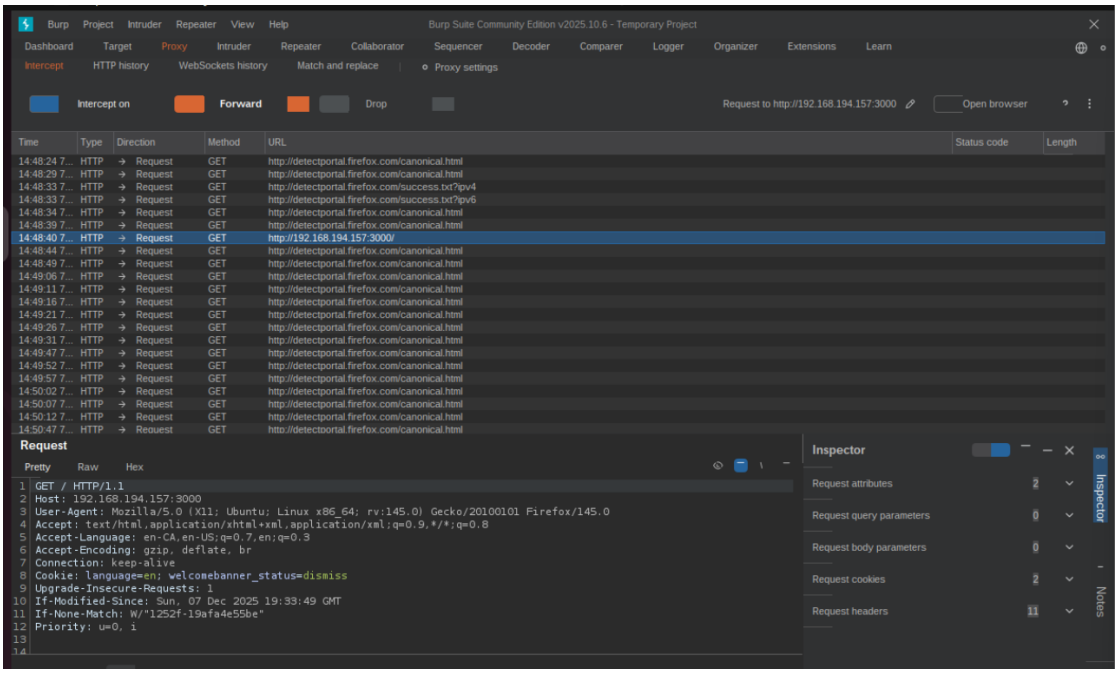


Figure 6 - And here we confirm the request.

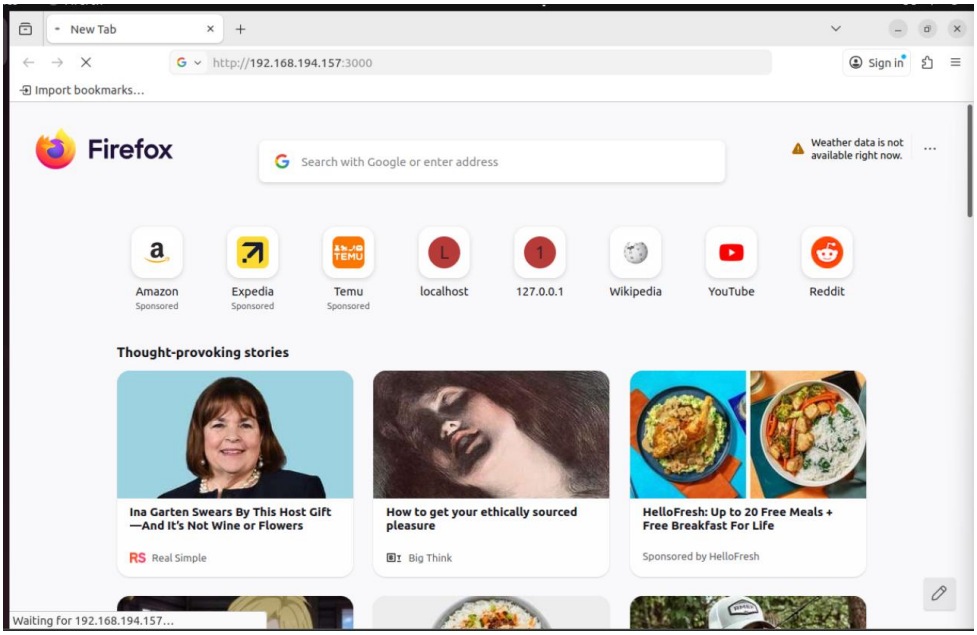


Figure 7 - Website tries to load. But it won't until we forward the request from burp suite.

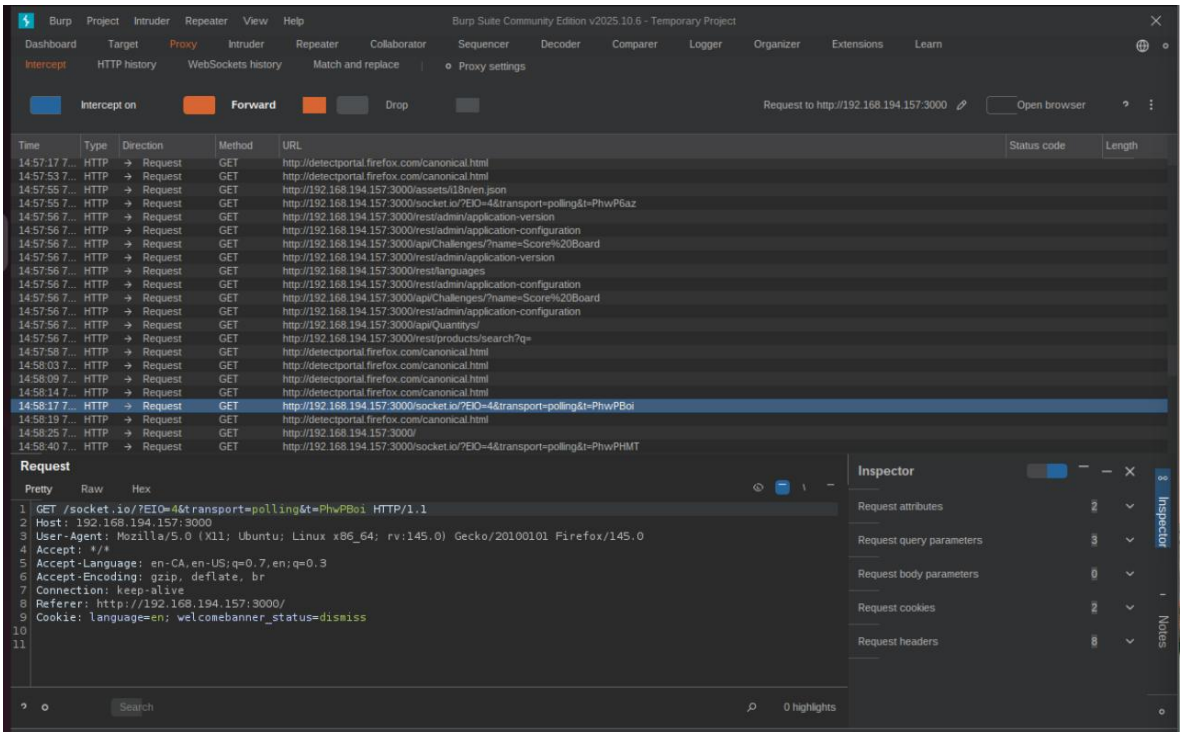


Figure 8 - After clicking Forward.

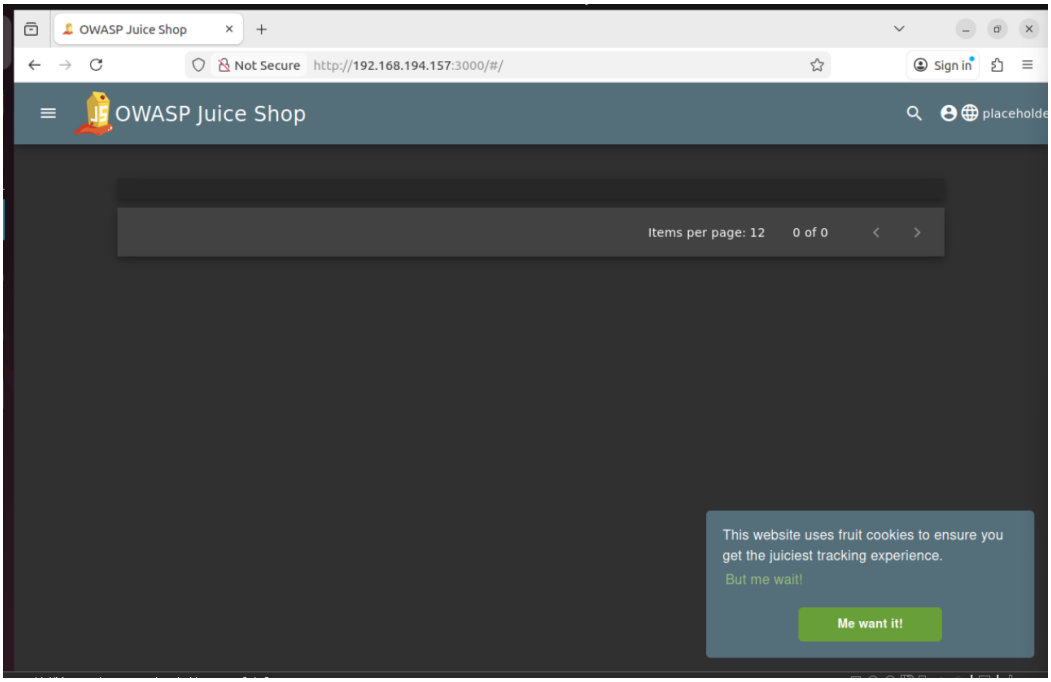


Figure 9 - Now we see the website.

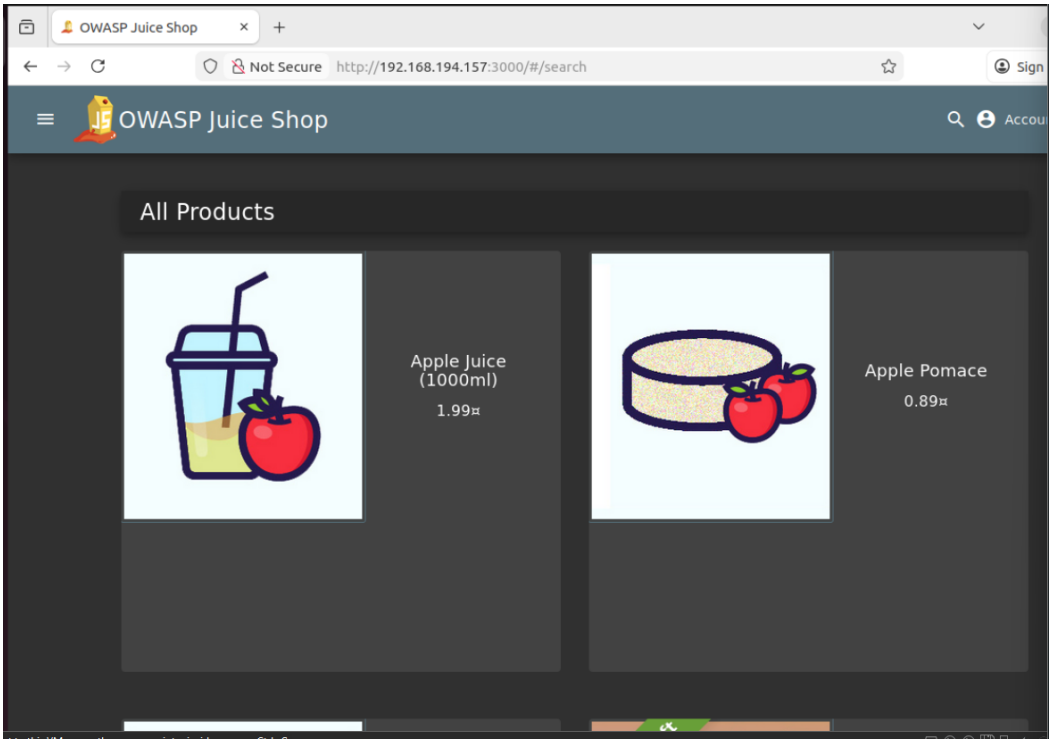


Figure 10 - In order to see the pictures, we turned off the intercept and the refresh the website.

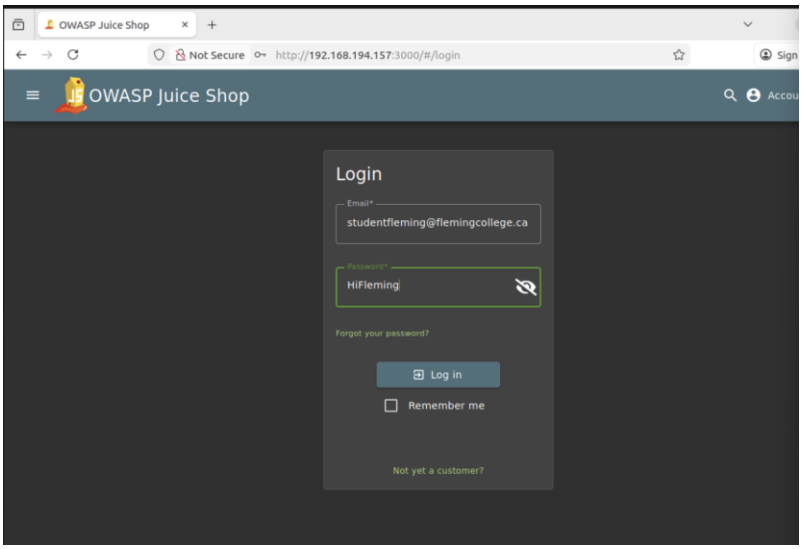


Figure 11 - False credentials used.

And in Burpsuite we can see the false credentials being intercepted.

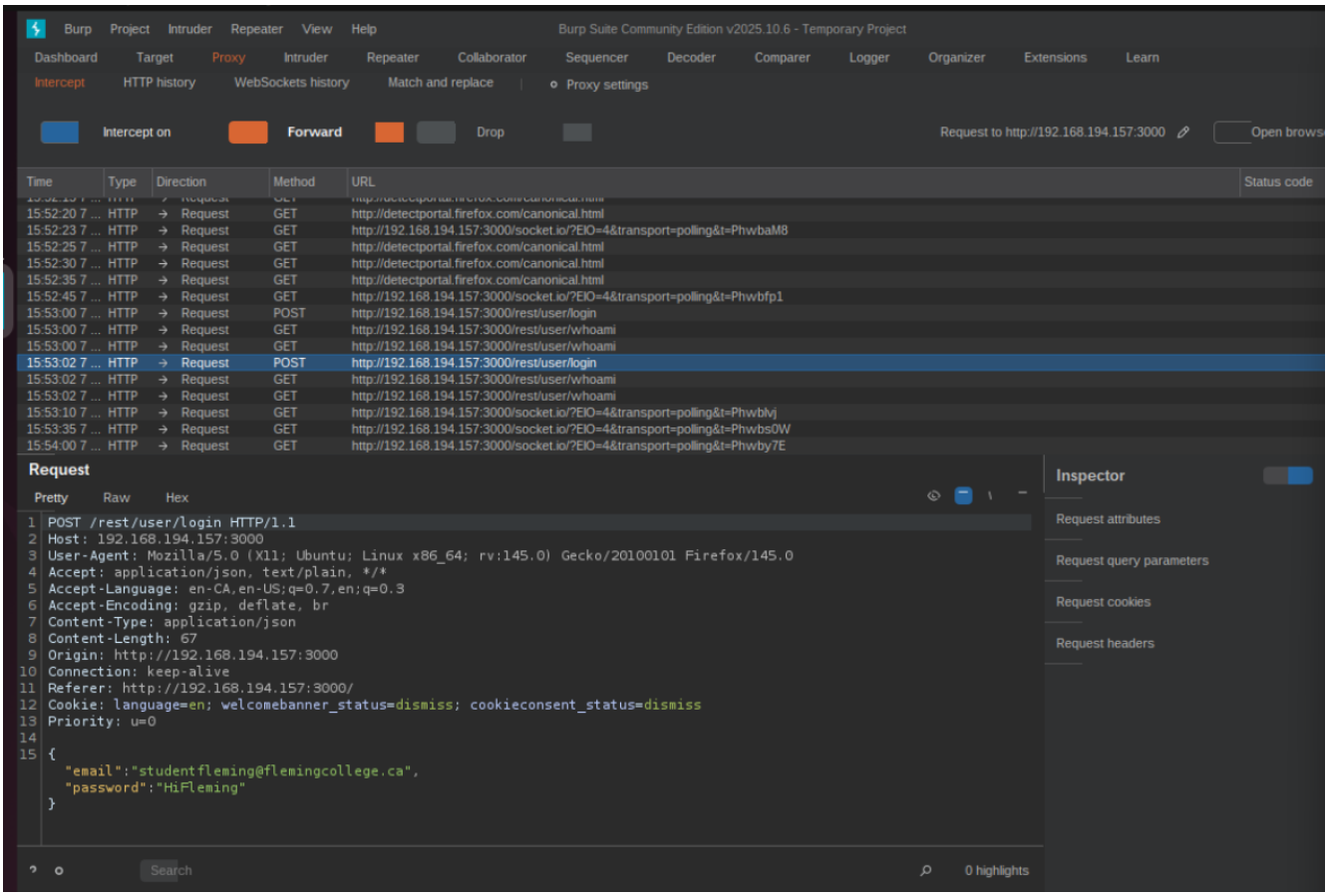


Figure 12 - Credentials intercepted.

The Attack

And now we will perform the attack for the vulnerability:

Username enumeration via different response (Sopyan, 2025).

The goal is to identify valid usernames through differential error responses.

How? Using BurpSuite -> Intruder to send multiple logins attempts with the emails we are going to add to search for the existing usernames.

The 401 and 500 http error will help us to identify which emails exists. And the impact that this can cause is that it can allows an attacker to confirm valid usernames or accounts and then he can use other techniques to figure it out the passwords.

Once we had the intercepted credentials. Then we are going to modified it. To know if an email is already registered.

Click Intruder. And we modified the last line. We select studentfleming@flemingcollege.ca and then click on “Add 5”

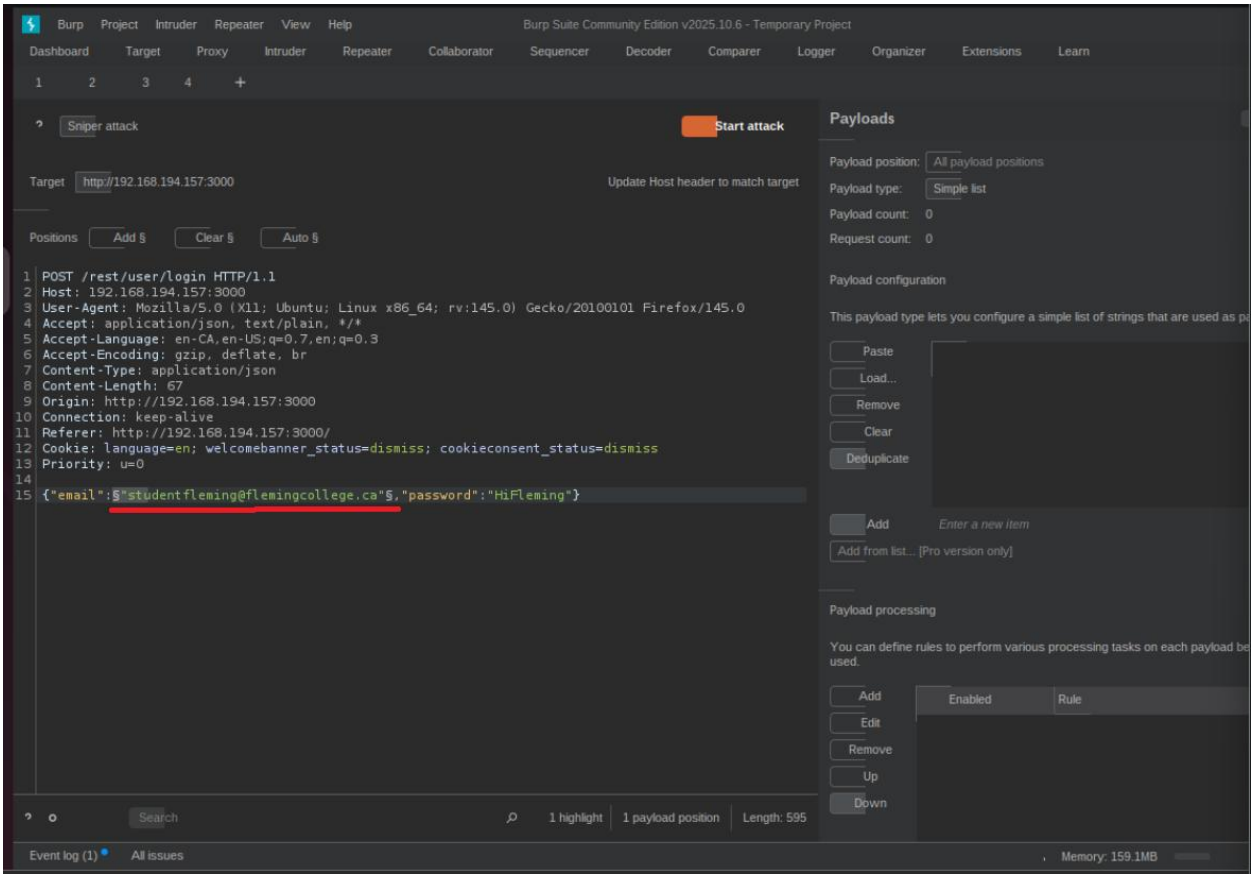


Figure 13 – Line we change.

Then we added emails to the list. On the right side we see payloads.

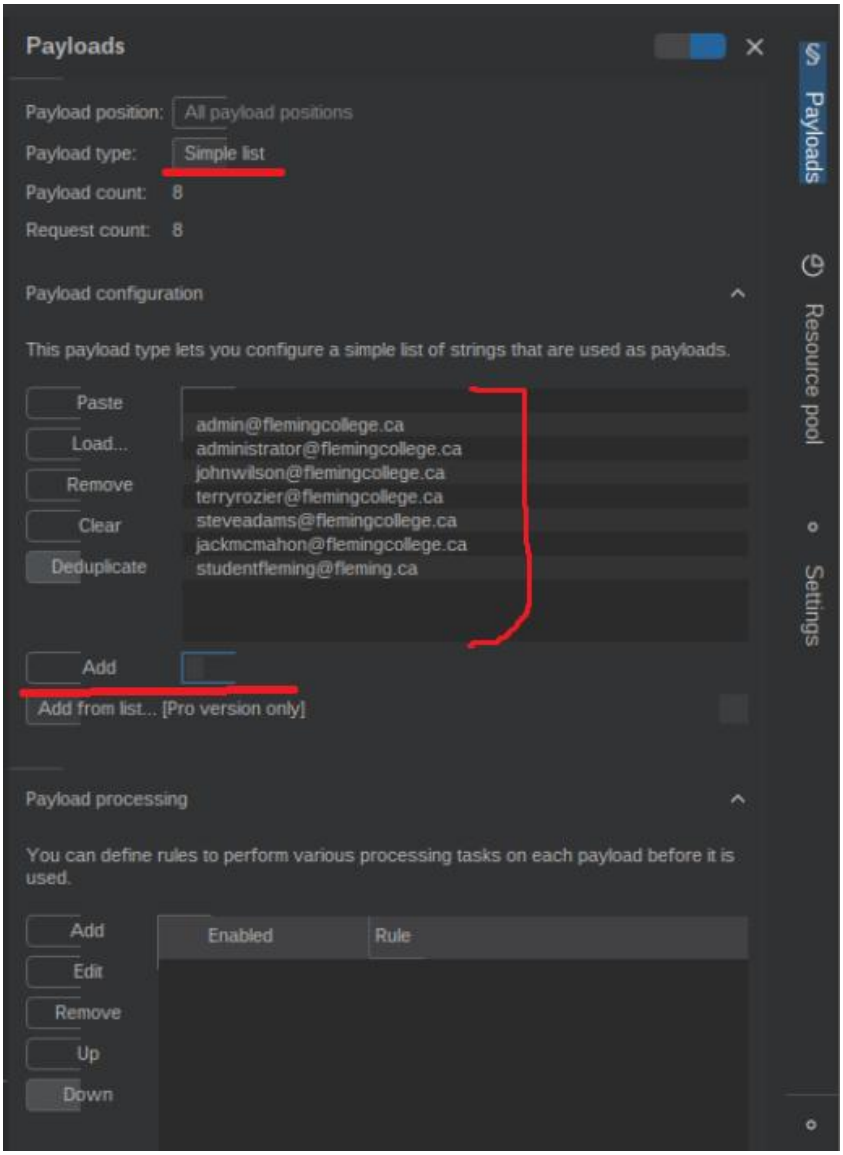


Figure 14 - Adding emails.

After adding the emails. We Clicked on “Start Attack”. And it opens the following window:

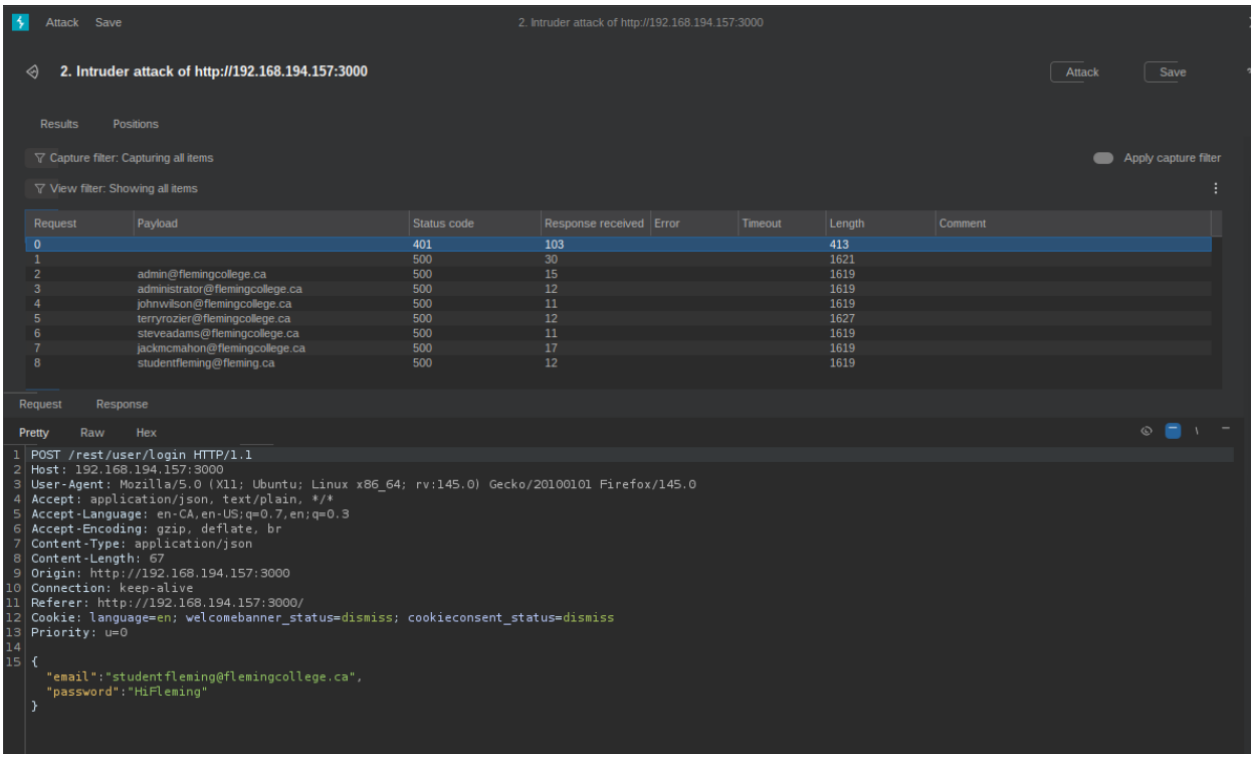
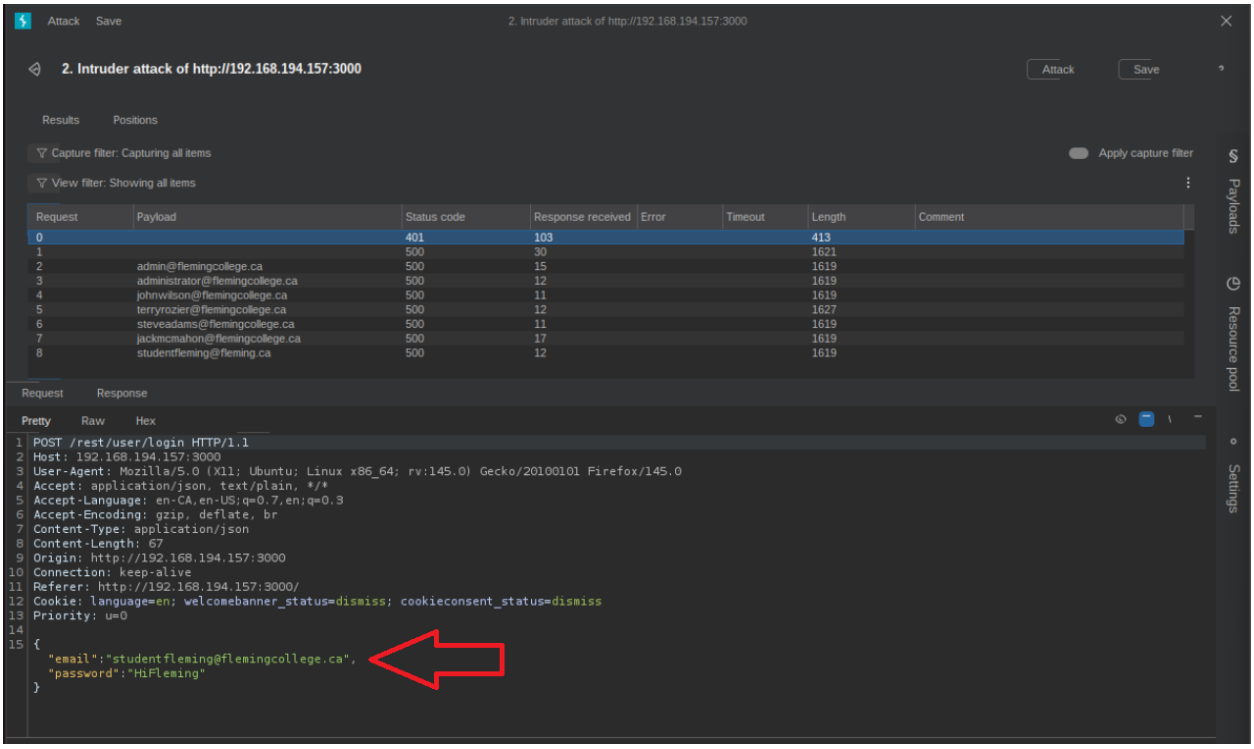


Figure 15 - Intruder attack window.

Here we can see how the emails we added are giving us the Status code as 500. Which means the internal server error, and it can be because the email doesn’t exist and they are trying to validate a user that doesn’t exit. (MDN n.d.)

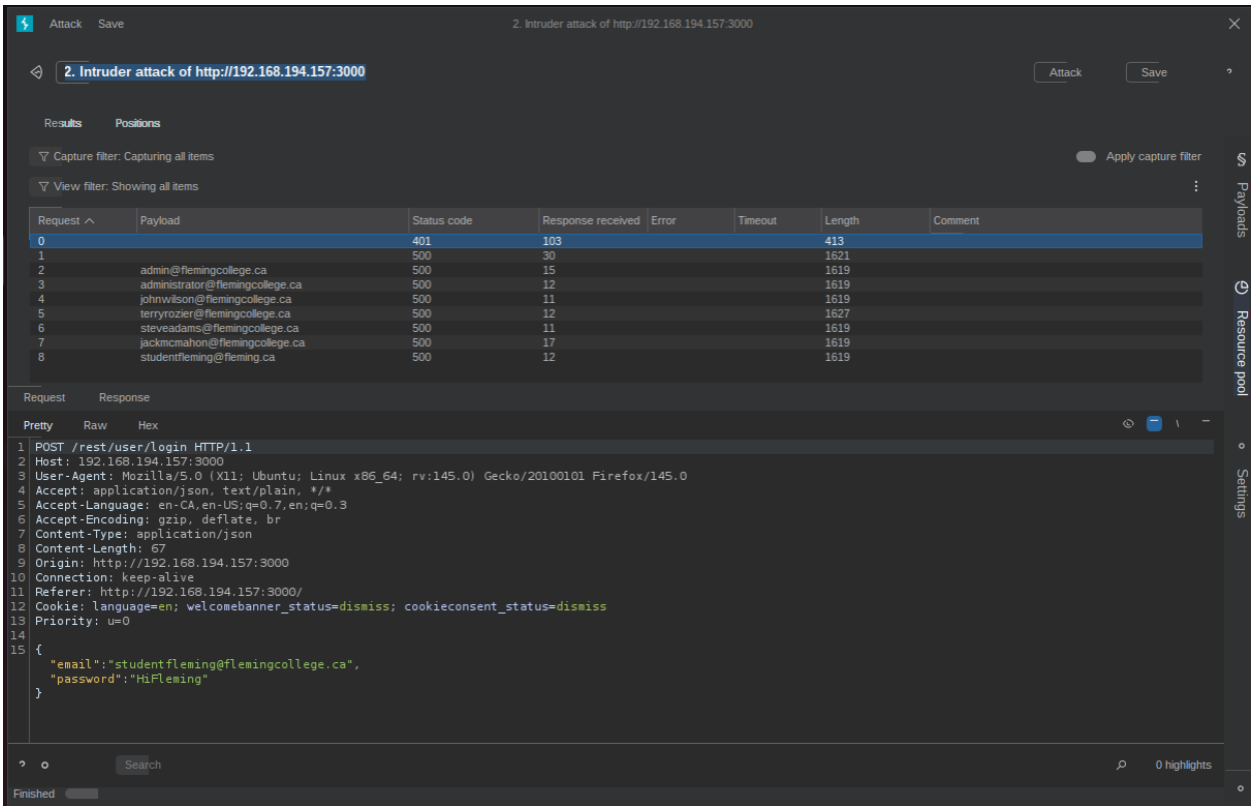
On the other hand, we can see that the first line has the status code as 401. Which means that the user exists, but the password is invalid. (MDN n.d.)



Therefore, we can confirm that the email address studentfleming@flemingcollege.ca is a valid email address registered.

Mitigation

During the attack, we used Burp intruder to test the /user/login endpoint. The server gave us different http status code when the email existed vs when it did not. This difference allows us to enumerate valid accounts by looking at the status code.



Now the goal is to remove all observable differences between valid and invalid user accounts so an attacker can no longer identify existing users during a brute-force enumeration attack.

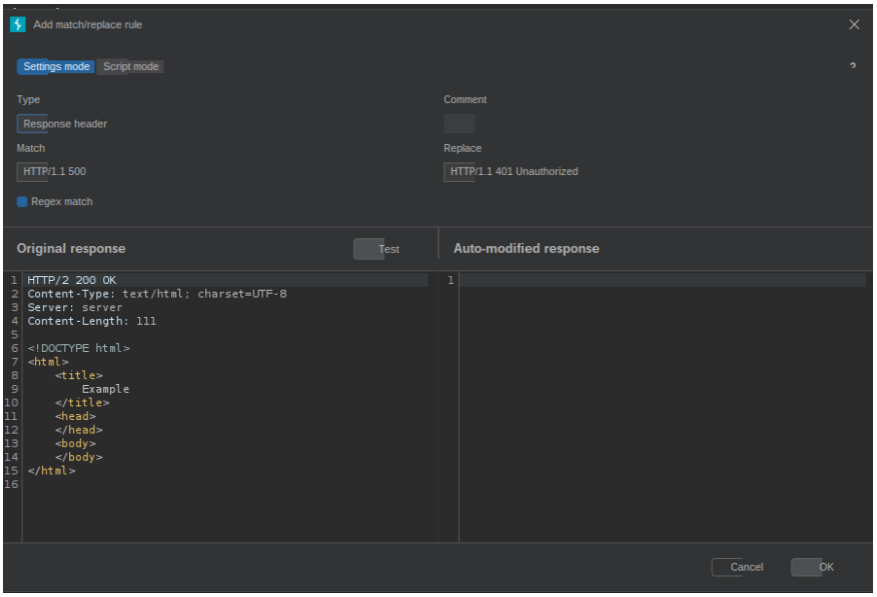
In order to achieve this, the login endpoint must always return the same type of response for every failed login, regardless of whether the email exists.

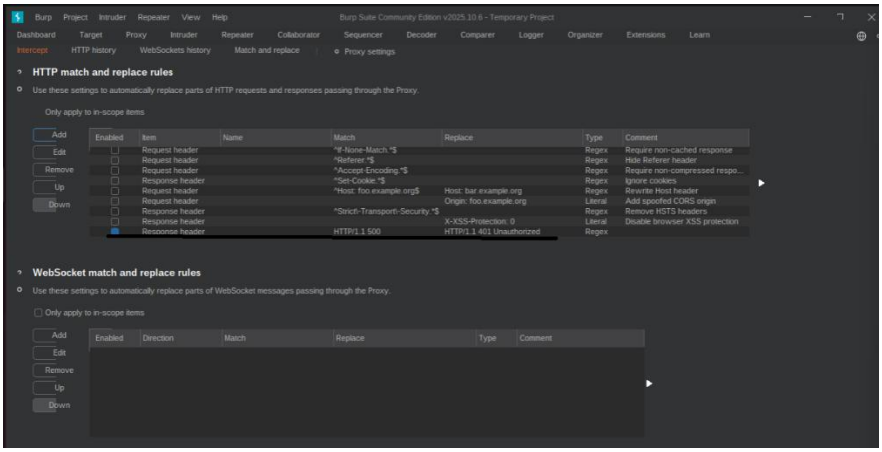
Security control

To fix this problem, we can apply a security control that makes the login endpoint respond the exact same way for every failed attempt. The problem happened because the server returned different HTTP status codes depending on whether the email existed or not. And because the attacker can see this information it is easy to identify which accounts were valid.

We forced every 500 status response to look like a 401 Unauthorized:

We would use “Response first line” on the match and replace rules to change the http status code. However, this option is only available in the burpsuite pro version. So, because we are using the community edition, we applied the closest possible control by using the “Response header rule” that forces every 500 responses to look like a 401 Unauthorized. So, this can remove the observable difference attackers were using to enumerate accounts.





References

Kimminich, B., & Hollenbach, J. (n.d.). *Juice-shop/juice-shop: Owasp Juice Shop: Probably the most modern and sophisticated insecure web application*. GitHub. https://github.com/juice-shop/juice-shop#docker-container

MDN. (n.d.). *HTTP response status codes*. https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status

Sopyan, A. (2025, June 4). *Authentication part-1 : Username enumeration via different responses*. Medium. https://medium.com/@AhmadSopyan/authentication-part-1-username-enumeration-via-different-responses-8ffc76acf511