



# Fleming College

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



## Installing docker (old version but works for the lab)

```
ricardor@ricardor: ~  
Session Actions Edit View Help  
ricardor@ricardor)-[~]  
$ sudo apt update  
[sudo] password for ricardor:  
Get:1 http://kali.download/kali kali-rolling InRelease [34.0 kB]  
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [21.0 MB]  
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [52.6 MB]  
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [115 kB]  
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [256 kB]  
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [188 kB]  
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [894 kB]  
Get:8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [11.9 kB]  
Get:9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents (deb) [28.7 kB]  
Fetched 75.2 MB in 18s (4,242 kB/s)  
1307 packages can be upgraded. Run 'apt list --upgradable' to see them.  
ricardor@ricardor)-[~]  
$ sudo apt install -y docker.io  
Upgrading:  
  libnl-3-200  libnl-genl-3-200  libnl-route-3-200  
Installing:  
  docker.io  
Installing dependencies:  
  containerd  docker-cli  libintl-xs-perl  libproc-processtable-perl  python3-pycrui  
  criu  libcompel1  libmodule-find-perl  libsort-naturally-perl  runc  
  docker-buildx  libintl-perl  libnet9  needrestart  tini-static  
Suggested packages:  
  containernetworking-plugins  btrfs-progs  rinse  xfsprogs | zfsutils-linux  
  docker-doc  debootstrap  rootlesskit  zfs-fuse  
Summary:  
  Upgrading: 3, Installing: 16, Removing: 0, Not Upgrading: 1304  
  Download size: 87.1 MB  
  Space needed: 364 MB / 13.1 GB available  
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.3.3+ds1-2 [6,686 kB]
```

Figure 1 - Installing docker.

```
ricardor@ricardor: ~  
Session Actions Edit View Help  
Preparing to unpack .../18-python3-pycrui_4.2-1_all.deb ...  
Unpacking python3-pycrui (4.2-1) ...  
Setting up docker-cli (27.5.1+dfsg4-1) ...  
Setting up tini-static (0.19.0-6) ...  
Setting up runc (1.3.3+ds1-2) ...  
Setting up libcompel1:amd64 (4.2-1) ...  
Setting up libmodule-find-perl (0.17-1) ...  
Setting up libproc-processtable-perl:amd64 (0.637-1) ...  
Setting up libnl-3-200:amd64 (3.11.0-2) ...  
Setting up libintl-perl (1.35-1) ...  
Setting up libnet9:amd64 (1.3+dfsg-3) ...  
Setting up containerd (1.7.24-ds1-10) ...  
containerd.service is a disabled or a static unit, not starting it.  
Setting up docker-buildx (0.19.3+ds1-4) ...  
Setting up libsort-naturally-perl (1.03-4) ...  
Setting up needrestart (3.11-1) ...  
Setting up docker.io (27.5.1+dfsg4-1) ...  
update-rc.d: We have no instructions for the docker init script.  
update-rc.d: It looks like a non-network service, we enable it.  
Created symlink '/etc/systemd/system/multi-user.target.wants/docker.service' → '/usr/lib/systemd/system/docker.service'.  
Created symlink '/etc/systemd/system/sockets.target.wants/docker.socket' → '/usr/lib/systemd/system/docker.socket'.  
Setting up libintl-xs-perl (1.35-1) ...  
Setting up libnl-route-3-200:amd64 (3.11.0-2) ...  
Setting up criu (4.2-1) ...  
Setting up libnl-genl-3-200:amd64 (3.11.0-2) ...  
Setting up python3-pycrui (4.2-1) ...  
Processing triggers for libc-bin (2.41-12) ...  
Processing triggers for man-db (2.13.1-1) ...  
Processing triggers for kali-menu (2025.3.2) ...  
ricardor@ricardor)-[~]  
$ docker --version  
Docker version 27.5.1+dfsg4, build cab968b3  
ricardor@ricardor)-[~]  
$
```

Figure 2 - Confirming dockers was successfully installed.

```
ricardor@ricardor: ~  
Session Actions Edit View Help  
(ricardor@ricardor)-[~]  
$ docker --version  
Docker version 27.5.1-dfsg4, build cab968b3  
(ricardor@ricardor)-[~]  
$ sudo systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
(ricardor@ricardor)-[~]  
$ sudo systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)  
   Active: active (running) since Sun 2025-12-07 13:47:56 EST; 5min ago  
 Invocation: 104b79420cf142f1b9e8c10a6f13c049  
 TriggeredBy: ● docker.socket  
    Docs: https://docs.docker.com  
   Main PID: 35507 (dockerd)  
     Tasks: 12  
  Memory: 26.5M (peak: 28.6M)  
    CPU: 2.141s  
   CGroup: /system.slice/docker.service  
           └─35507 /usr/sbin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
Dec 07 13:47:53 ricardor systemd[1]: Starting docker.service - Docker Application Container Engine ...  
Dec 07 13:47:53 ricardor (dockerd)[35507]: docker.service: Referenced but unset environment variable eval>  
Dec 07 13:47:53 ricardor dockerd[35507]: time="2025-12-07T13:47:53.592741961-05:00" level=info msg="Start>  
Dec 07 13:47:53 ricardor dockerd[35507]: time="2025-12-07T13:47:53.598039915-05:00" level=info msg="OTEL >  
Dec 07 13:47:53 ricardor dockerd[35507]: time="2025-12-07T13:47:53.921612570-05:00" level=info msg="Loadi>  
Dec 07 13:47:56 ricardor dockerd[35507]: time="2025-12-07T13:47:56.595932935-05:00" level=info msg="Loadi>  
Dec 07 13:47:56 ricardor dockerd[35507]: time="2025-12-07T13:47:56.675191192-05:00" level=info msg="Docke>  
Dec 07 13:47:56 ricardor dockerd[35507]: time="2025-12-07T13:47:56.675413180-05:00" level=info msg="Daemo>  
Dec 07 13:47:56 ricardor dockerd[35507]: time="2025-12-07T13:47:56.792356907-05:00" level=info msg="API l>  
Dec 07 13:47:56 ricardor systemd[1]: Started docker.service - Docker Application Container Engine.  
(ricardor@ricardor)-[~]  
$
```

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

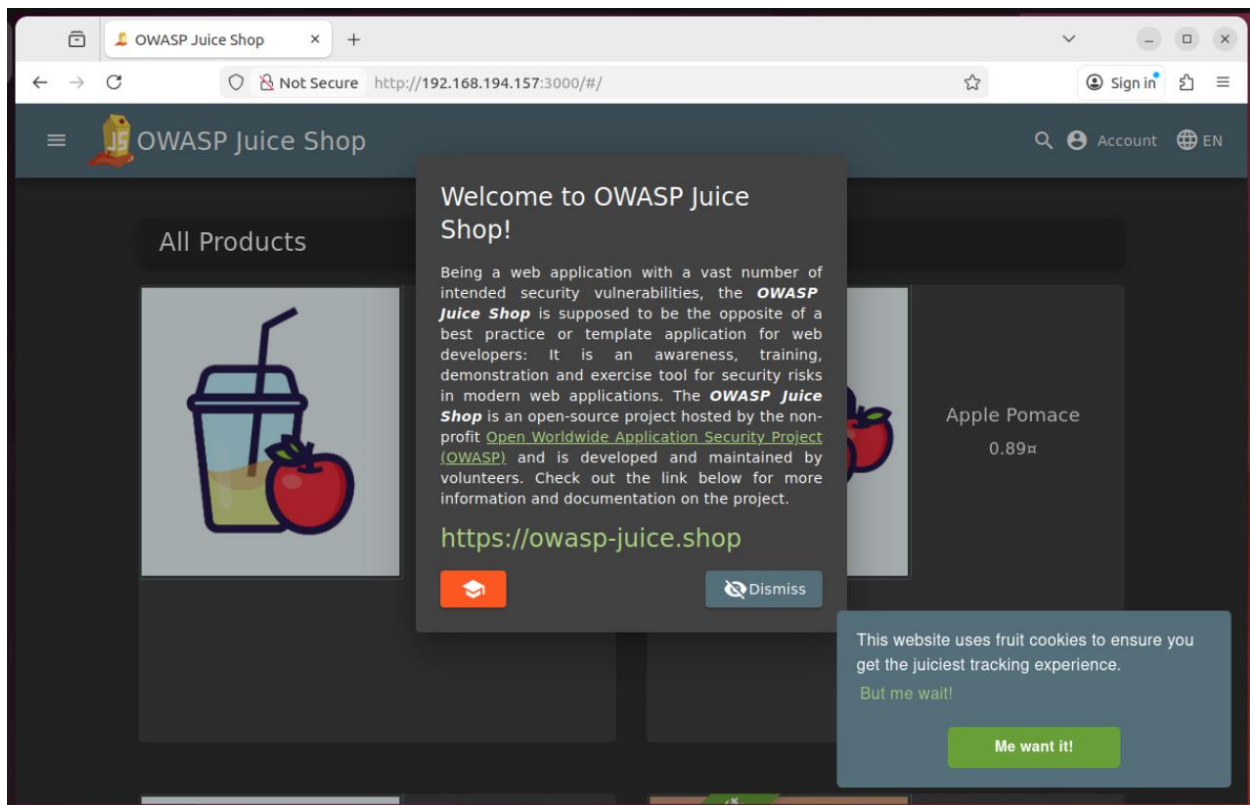
```
ricardor@ricardor: ~  
Session Actions Edit View Help  
(ricardor@ricardor)-[~]  
$ sudo docker pull bkimminich/juice-shop  
Using default tag: latest  
latest: Pulling from bkimminich/juice-shop  
fd4aa3667332: Pull complete  
bfb59b82a9b6: Pull complete  
017886f7e176: Pull complete  
62de241dac5f: Pull complete  
2780920e5dbf: Pull complete  
7c12895b777b: Pull complete  
3214acf345c0: Pull complete  
5664b15f108b: Pull complete  
045fc1c20da8: Pull complete  
4aa0ea1413d3: Pull complete  
da7816fa955e: Pull complete  
ddf74a63f7d8: Pull complete  
e7fa9df358f0: Pull complete  
d8a0d911b13e: Pull complete  
5b14f6c9a813: Pull complete  
33ce0b1d99fc: Pull complete  
f45e0372ce60: Pull complete  
7faf0cfa885c: Pull complete  
9cd2a1476fcc: Pull complete  
7b72e6384ef9: Pull complete  
0168f69dfb16: Pull complete  
Digest: sha256:1c55debeaf4fd5678019b17818a539e1e06ef93d29b268a21f53f0773a9fff5d  
Status: Downloaded newer image for bkimminich/juice-shop:latest  
docker.io/bkimminich/juice-shop:latest
```

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.

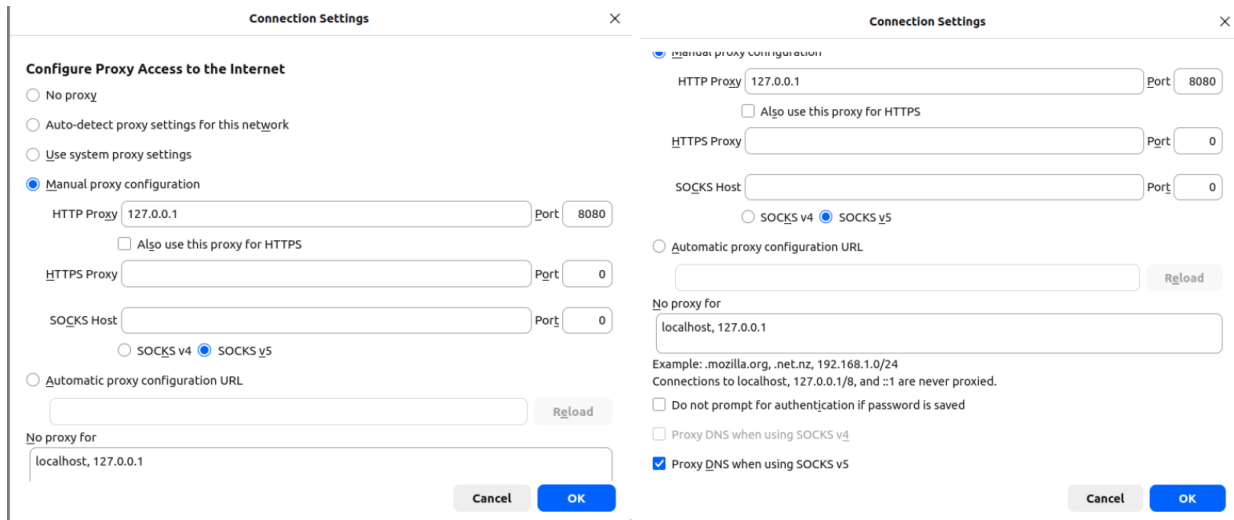
On the attacker machine. I went to <http://192.168.194.157:3000>





## On the attacker machine

Firefox -> Settings -> Network Settings



Then click ok

Now in Firefox we typed: http:192.168.194.157:3000

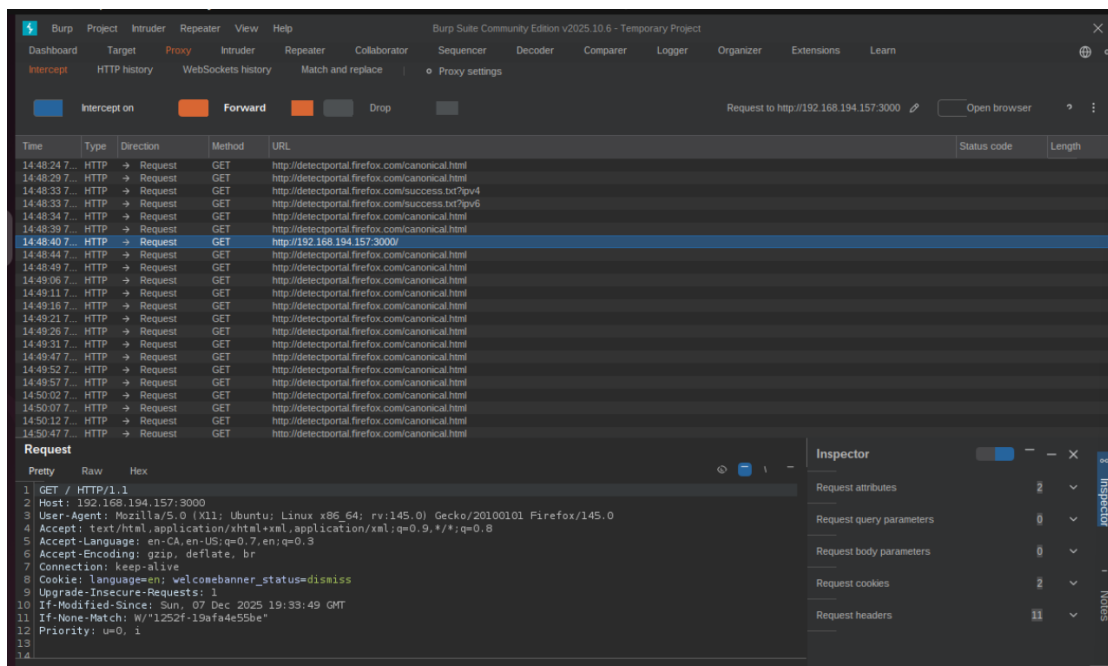


Figure 6 - And here we confirm the request.

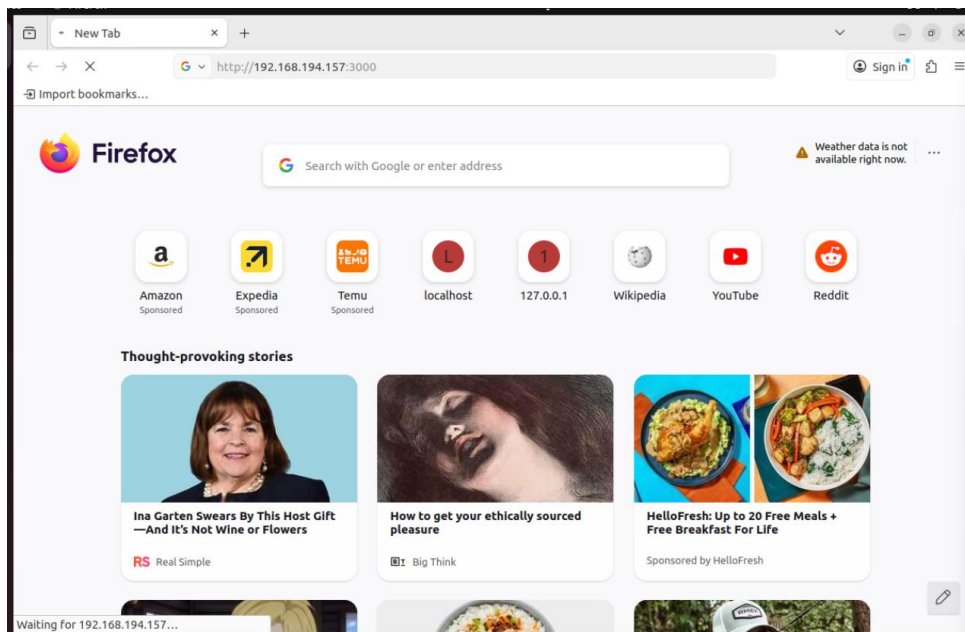


Figure 7 - Website tries to load. But it won't until we forward the request from burp suite.

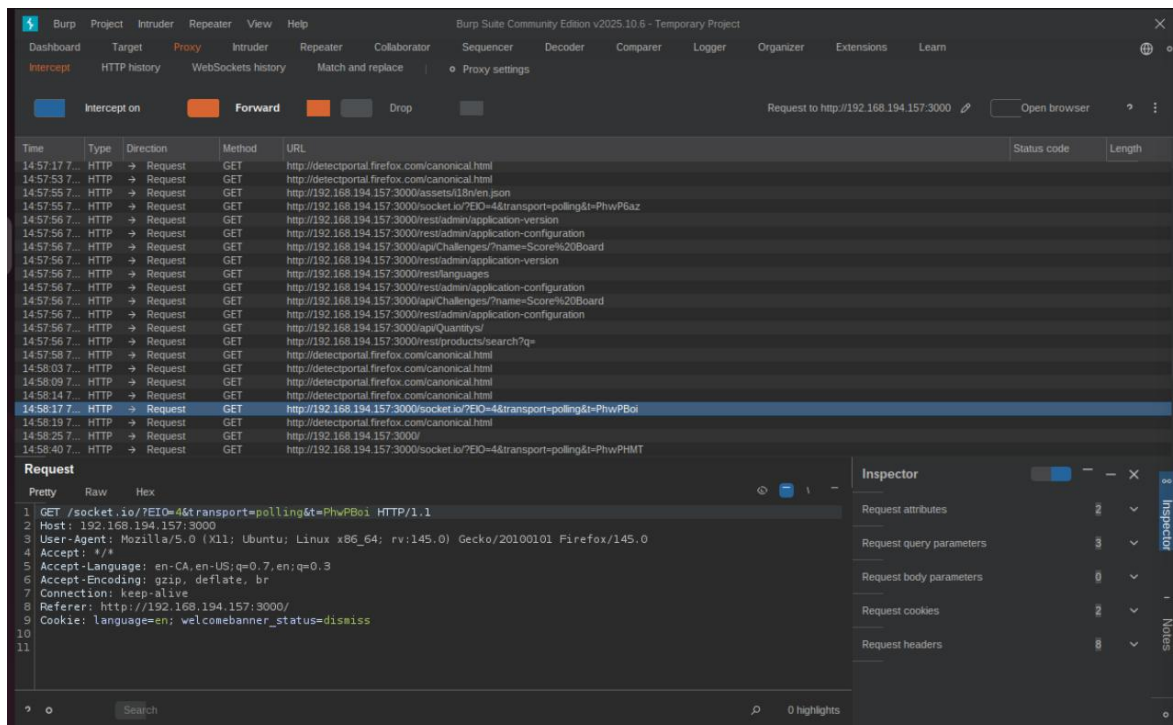


Figure 8 - After clicking Forward.

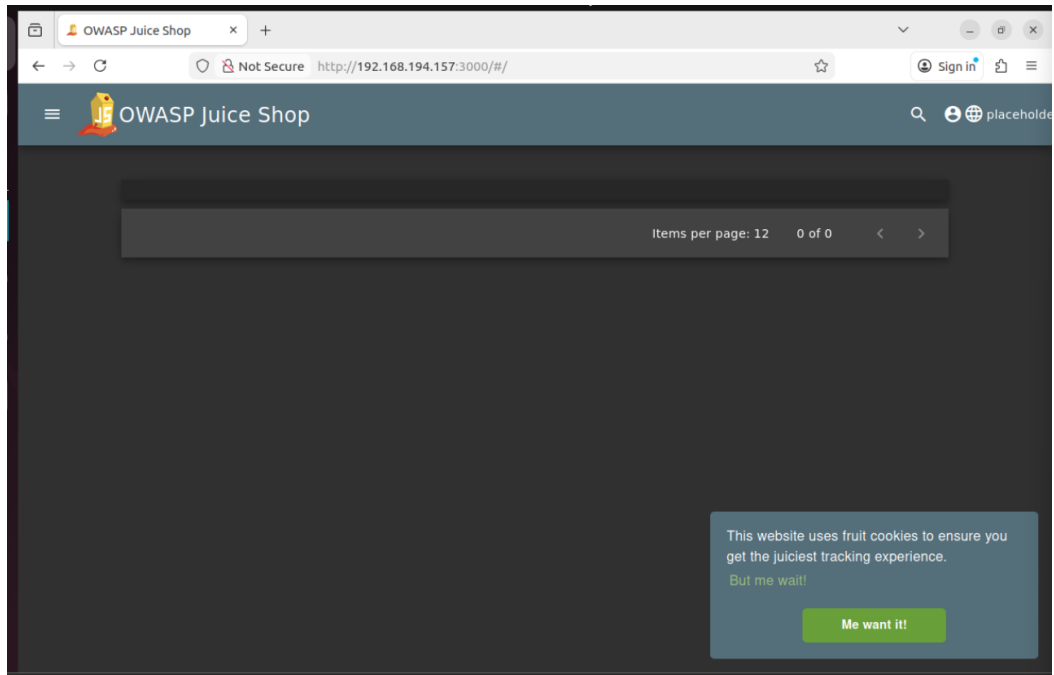


Figure 9 - Now we see the website.

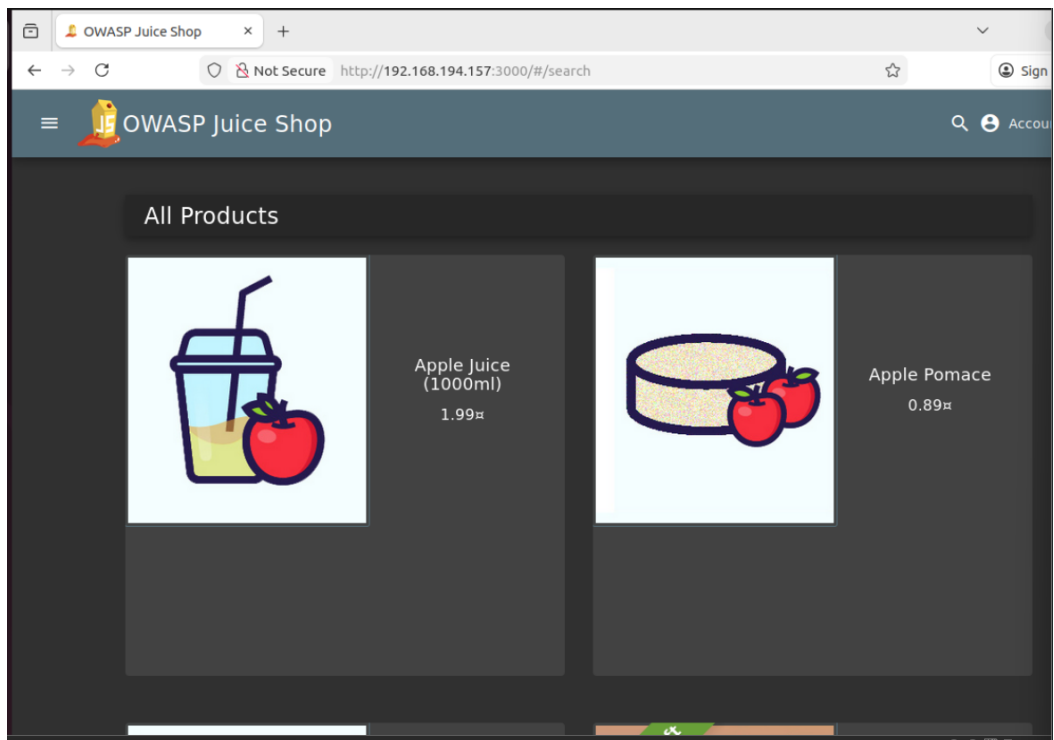


Figure 10 - In order to see the pictures, we turned off the intercept and the refresh the website.



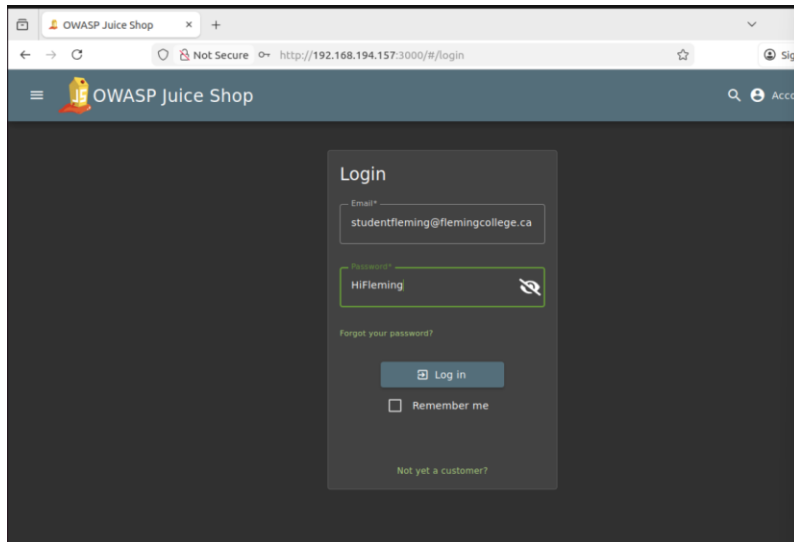


Figure 11 - False credentials used.

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

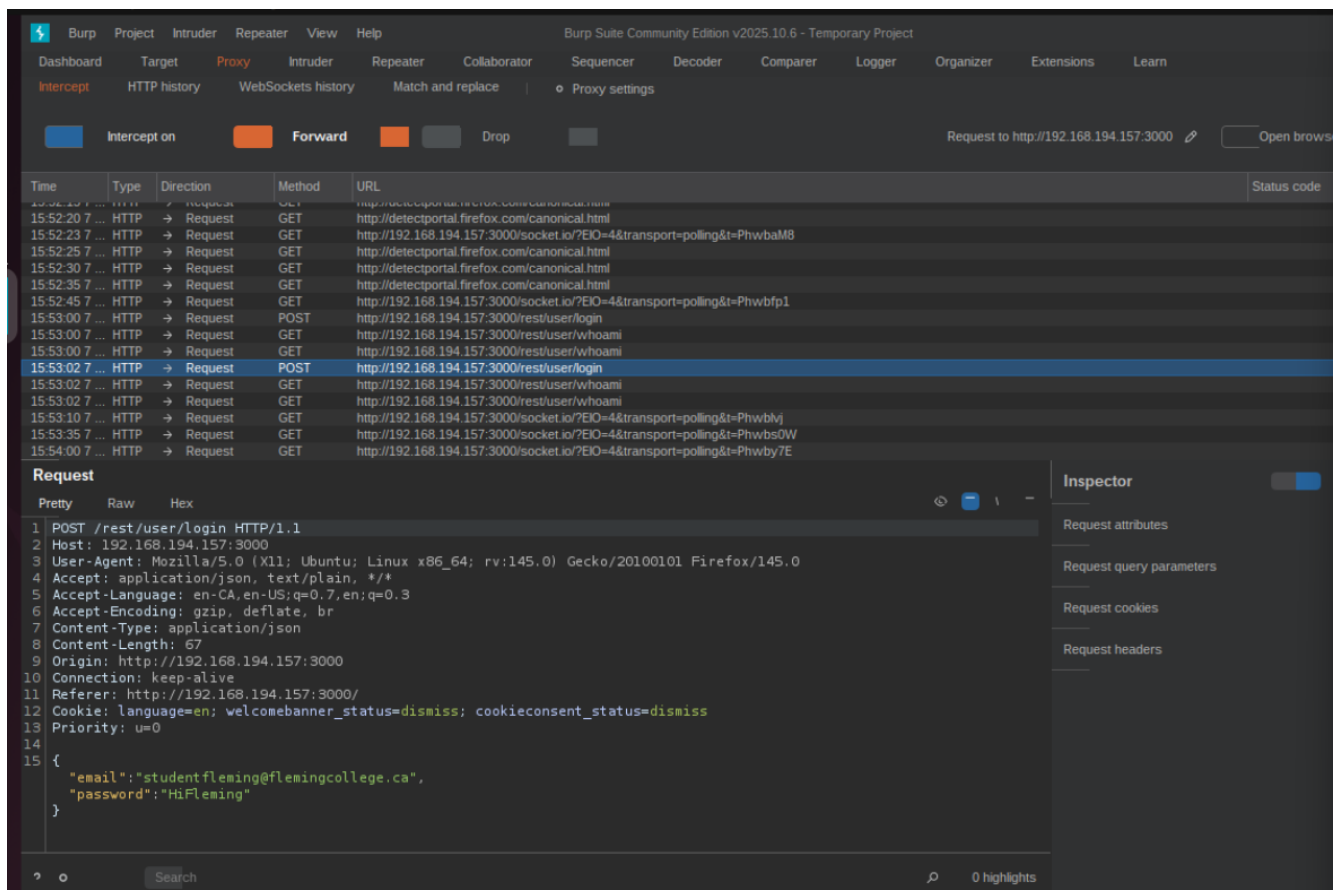


Figure 12 - Credentials intercepted.

## 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](mailto:studentfleming@flemingcollege.ca) and then click on "Add 5"

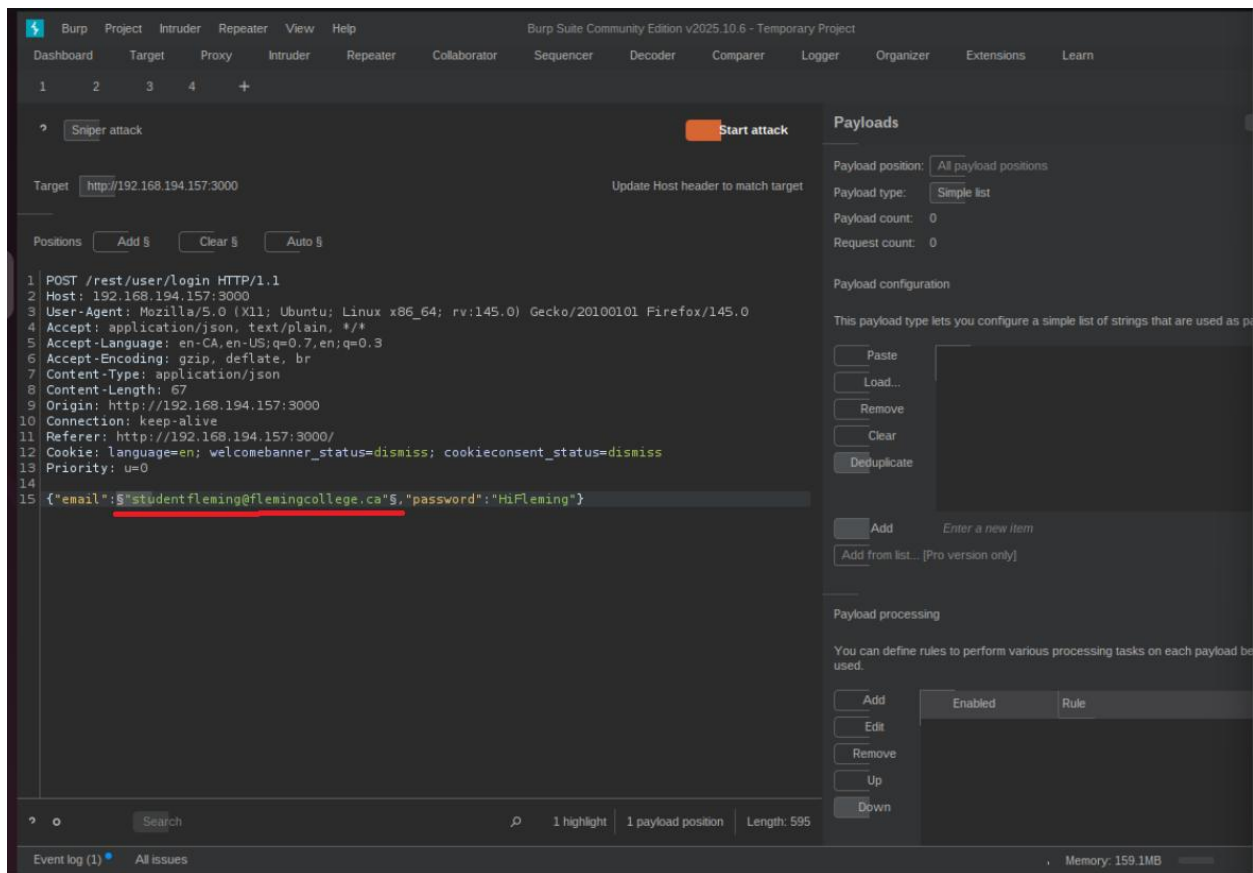


Figure 13 – Line we change.

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

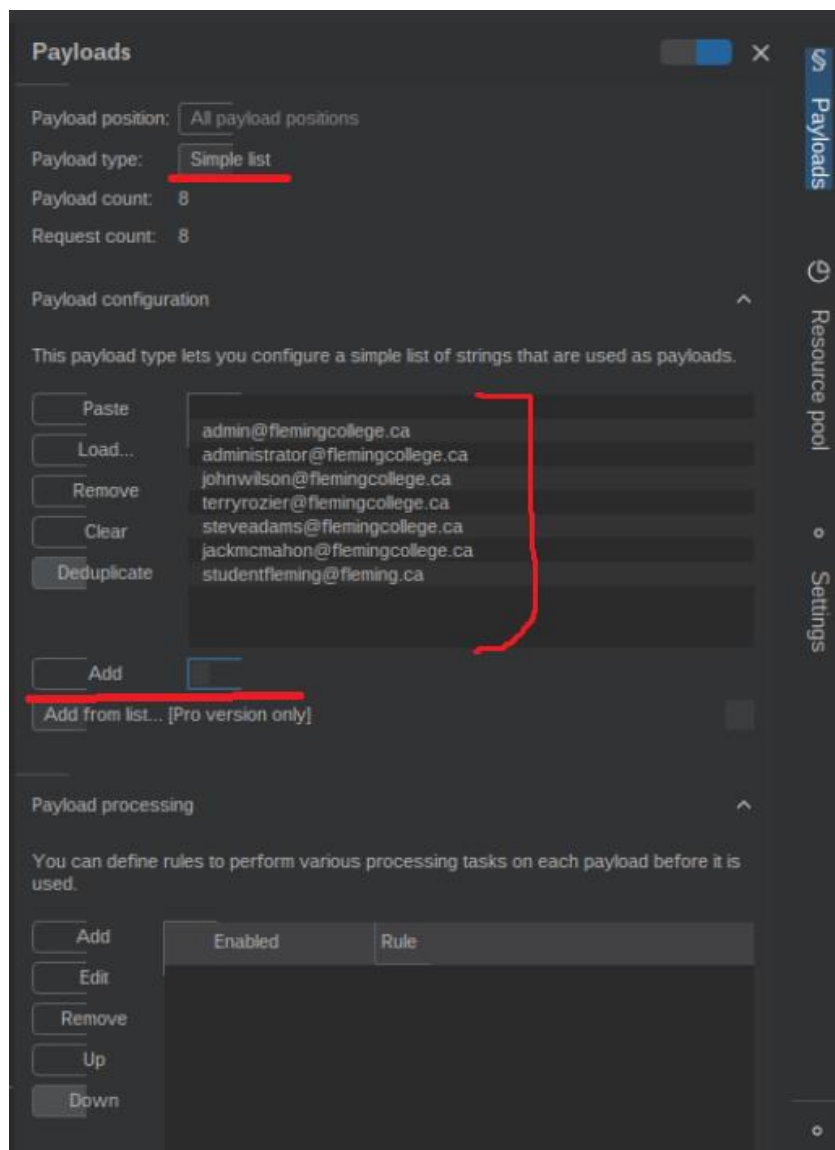


Figure 14 - Adding emails.

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

The screenshot shows the 'Intruder attack of http://192.168.194.157:3000' window. It features a table with columns: Request, Payload, Status code, Response received, Error, Timeout, Length, and Comment. The first row (index 0) shows a status code of 401 and a response length of 413. Subsequent rows (indices 1-8) show status codes of 500 and response lengths of 1621, 1619, 1619, 1627, 1619, 1619, and 1619 respectively. Below the table, the 'Request' tab is selected, showing a POST request to /rest/user/login with a JSON payload: {"email": "studentfleming@flemingcollege.ca", "password": "HIFleming"}.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		401	103			413	
1		500	30			1621	
2	admin@flemingcollege.ca	500	15			1619	
3	administrator@flemingcollege.ca	500	12			1619	
4	jhnwolson@flemingcollege.ca	500	11			1619	
5	terryrozler@flemingcollege.ca	500	12			1627	
6	steveadams@flemingcollege.ca	500	11			1619	
7	jackmcmahon@flemingcollege.ca	500	17			1619	
8	studentfleming@fleming.ca	500	12			1619	

```
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.194.157:3000
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:145.0) Gecko/20100101 Firefox/145.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 67
9 Origin: http://192.168.194.157:3000
10 Connection: keep-alive
11 Referer: http://192.168.194.157:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
13 Priority: u=0
14
15 {
16   "email": "studentfleming@flemingcollege.ca",
17   "password": "HIFleming"
18 }
```

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 exist. (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.)

Attack Save 2. Intruder attack of http://192.168.194.157:3000

Results Positions

Capture filter: Capturing all items Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		401	103			413	
1		500	30			1621	
2	admin@flemingcollege.ca	500	15			1619	
3	administrator@flemingcollege.ca	500	12			1619	
4	johnrwilson@flemingcollege.ca	500	11			1619	
5	tennyzozer@flemingcollege.ca	500	12			1627	
6	stevcadams@flemingcollege.ca	500	11			1619	
7	jackmcmahon@flemingcollege.ca	500	17			1619	
8	studentfleming@fleming.ca	500	12			1619	

Request Response

Pretty Raw Hex

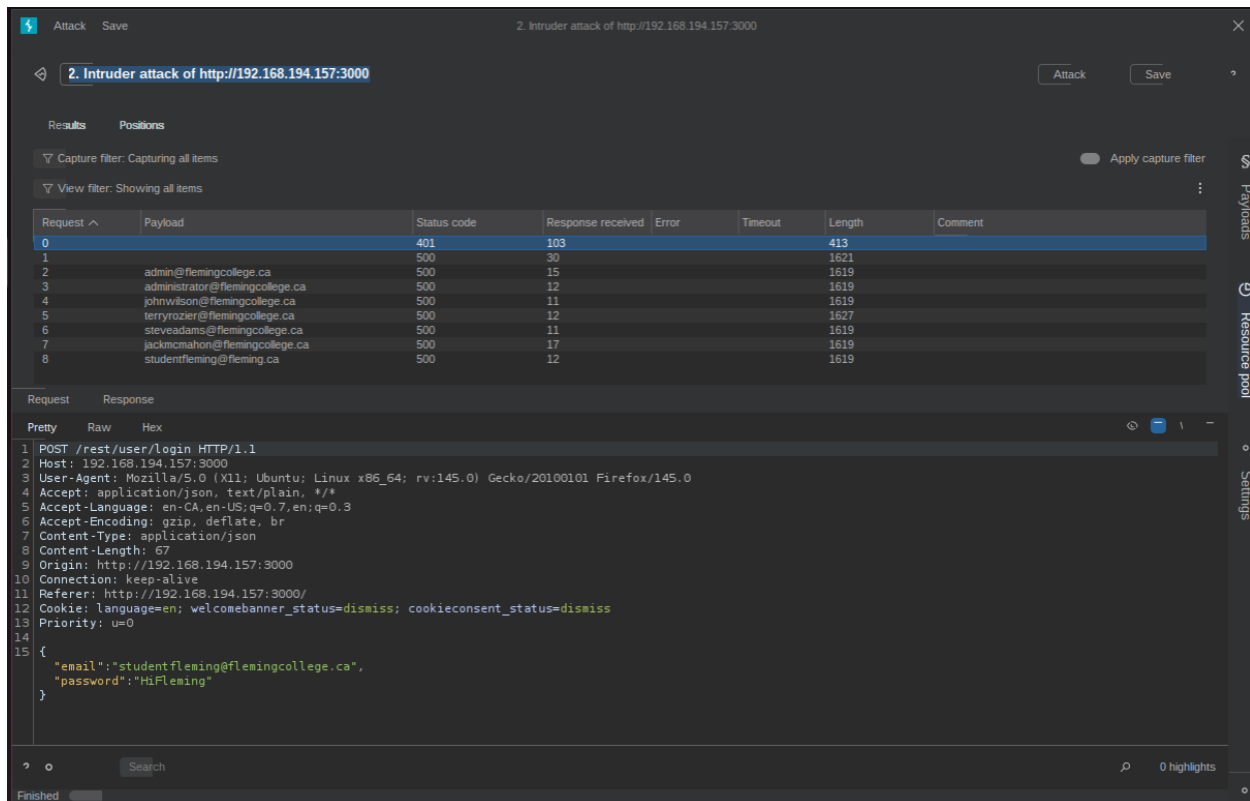
```
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.194.157:3000
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:145.0) Gecko/20100101 Firefox/145.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 67
9 Origin: http://192.168.194.157:3000
10 Connection: keep-alive
11 Referer: http://192.168.194.157:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
13 Priority: u=0
14
15 {
  "email": "studentfleming@flemingcollege.ca",
  "password": "HiFleming"
}
```

Therefore, we can confirm that the email address [studentfleming@flemingcollege.ca](mailto: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.



The screenshot shows the Burp Suite Intruder tool interface. The title bar indicates the target is "2. Intruder attack of http://192.168.194.157:3000". The main window displays a table of requests and responses. The first request (index 0) has a status code of 401 and a response received of 103. The subsequent requests (indices 1-8) all have a status code of 500. The request details for the first request are shown below the table.

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		401	103			413	
1		500	30			1621	
2	admin@flemingcollege.ca	500	15			1619	
3	administrator@flemingcollege.ca	500	12			1619	
4	johnwilson@flemingcollege.ca	500	11			1619	
5	terryrozier@flemingcollege.ca	500	12			1627	
6	steveadams@flemingcollege.ca	500	11			1619	
7	jackmahon@flemingcollege.ca	500	17			1619	
8	studentfleming@fleming.ca	500	12			1619	

Request details (Pretty view):

```
1 POST /rest/user/login HTTP/1.1
2 Host: 192.168.194.157:3000
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:145.0) Gecko/20100101 Firefox/145.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 67
9 Origin: http://192.168.194.157:3000
10 Connection: keep-alive
11 Referer: http://192.168.194.157:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
13 Priority: u=0
14
15 {
  "email": "studentfleming@flemingcollege.ca",
  "password": "HiFleming"
}
```

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.

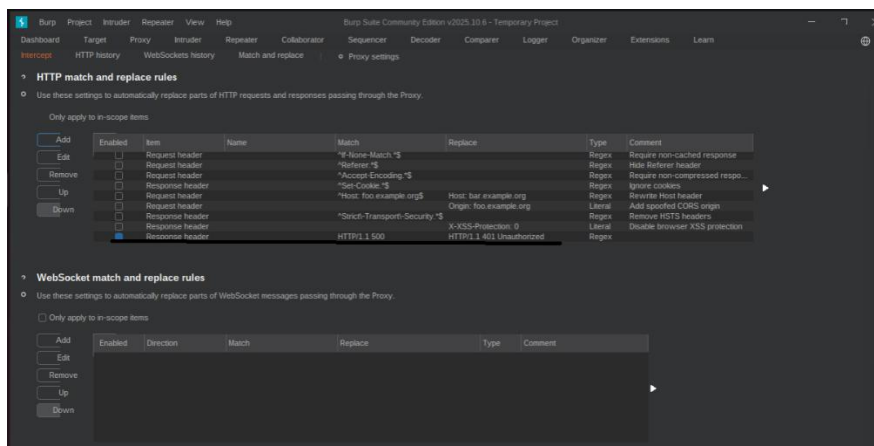
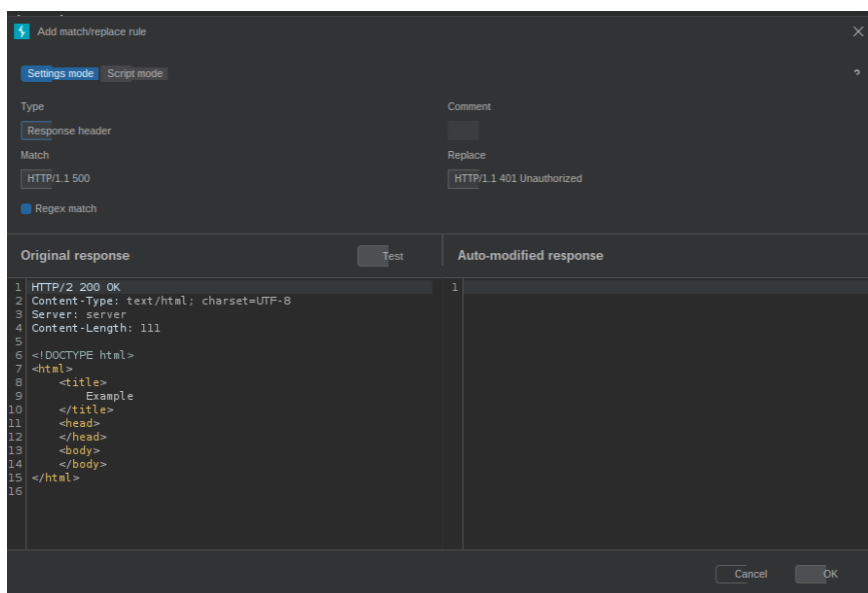
In order to achieve this, the login endpoint must always return the same type of response for every failed login, regardless of whether the email exists.

## 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>