

# Hunt Evil

DFPS\_FOR508\_v4.6\_4-19 Poster was created by Rob Lee and Mike Pilkington with support of the SANS DFIR Faculty ©2019 Rob Lee and Mike Pilkington. All Rights Reserved.

dfir.sans.org



## Find Evil – Know Normal

Knowing what's normal on a Windows host helps cut through the noise to quickly locate potential malware. Use the information below as a reference to know what's normal in Windows and to focus your attention on the outliers.



Image Path: N/A for system.exe - Not generated from an executable image **Parent Process:** None

**Number of Instances:** One **User Account:** Local System

**Start Time:** At boot time **Description:** The **System** process is responsible for most kernel-mode threads. Modules run under **System** are primarily drivers (.sys files), but also include several important DLLs as well as the kernel executable, ntoskrnl.exe.

#### smss.exe

Image Path: %SystemRoot%\System32\smss.exe

**Parent Process:** System

**Number of Instances:** One master instance and another child instance per

session. Children exit after creating their session.

**User Account:** Local System

**Start Time:** Within seconds of boot time for the master instance

**Description:** The Session Manager process is responsible for creating new sessions. The first instance creates a child instance for each new session. Once the child instance initializes the new session by starting the Windows subsystem (csrss.exe) and wininit.exe for Session 0 or winlogon.exe for Session 1 and higher, the child instance exits.

#### wininit.exe

Image Path: %SystemRoot%\System32\wininit.exe

Parent Process: Created by an instance of smss.exe that exits, so tools

usually do not provide the parent process name. **Number of Instances:** One

**User Account:** Local System

**Start Time:** Within seconds of boot time

**Description:** Wininit.exe starts key background processes within Session 0. It starts the Service Control Manager (services.exe), the Local Security Authority process (lsass.exe), and lsaiso.exe for systems with Credential Guard enabled. Note that prior to Windows 10, the Local Session Manager process (1sm.exe) was also started by wininit.exe. As of Windows 10, that functionality has moved to a service DLL (Ism.dll) hosted by svchost.exe.

#### RuntimeBroker.exe

Image Path: %SystemRoot%\System32\RuntimeBroker.exe

Parent Process: svchost.exe **Number of Instances:** One or more

**User Account:** Typically the logged-on user(s)

**Start Time:** Start times vary greatly

**Description:** RuntimeBroker.exe acts as a proxy between the constrained Universal Windows Platform (UWP) apps (formerly called Metro apps) and the full Windows API. UWP apps have limited capability to interface with hardware and the file system. Broker processes such as RuntimeBroker.exe are therefore used to provide the necessary level of access for UWP apps. Generally, there will be one RuntimeBroker.exe for each UWP app. For example, starting Calculator.exe will cause a corresponding RuntimeBroker.exe process to initiate.

#### taskhostw.exe

Image Path: %SystemRoot%\System32\taskhostw.exe

Parent Process: svchost.exe **Number of Instances:** One or more

**User Account:** Multiple taskhostw.exe processes are normal. One or more may be owned by logged-on users and/or by local service accounts.

**Start Time:** Start times vary greatly

**Description:** The generic host process for Windows Tasks. Upon initialization, taskhostw.exe runs a continuous loop listening for trigger events. Example trigger events that can initiate a task include a defined schedule, user logon, system startup, idle CPU time, a Windows log event, workstation lock, or

There are more than 160 tasks preconfigured on a default installation of Windows 10 Enterprise (though many are disabled). All executable files (DLLs & EXEs) used by the default Windows 10 scheduled tasks are signed by Microsoft.

#### winlogon.exe

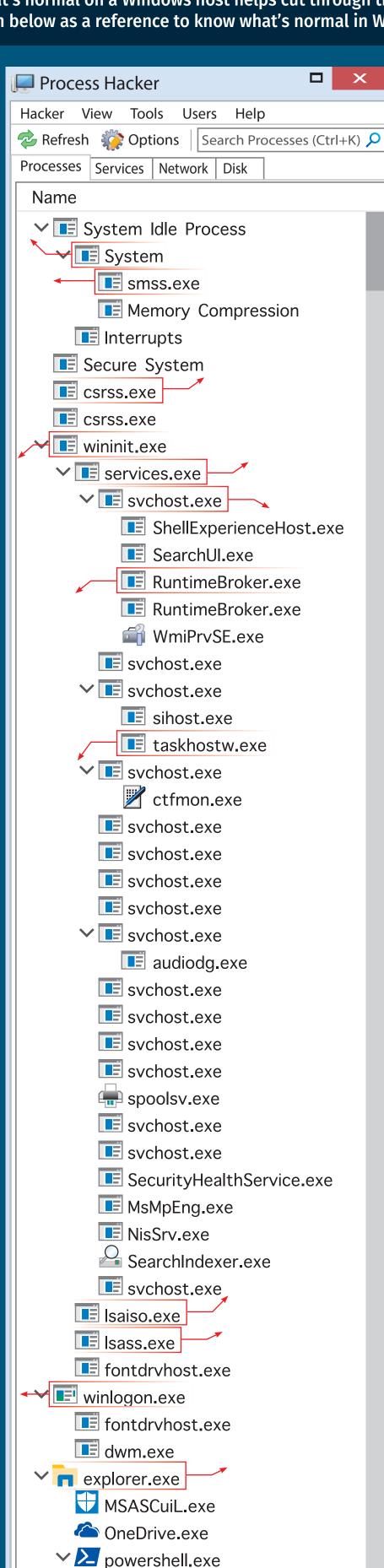
Image Path: %SystemRoot%\System32\winlogon.exe Parent Process: Created by an instance of smss.exe that exits, so analysis tools usually do not provide the parent process name.

**Number of Instances:** One or more

**User Account:** Local System

**Start Time:** Within seconds of boot time for the first instance (for Session 1). Start times for additional instances occur as new sessions are created, typically through Remote Desktop or Fast User Switching logons.

**Description:** Winlogon handles interactive user logons and logoffs. It launches **LogonUI.exe**, which uses a credential provider to gather credentials from the user, and then passes the credentials to **lsass.exe** for validation. Once the user is authenticated, Winlogon loads the user's **NTUSER.DAT** into **HKCU** and starts the user's shell (usually explorer.exe) via userinit.exe.



#### csrss.exe

Image Path: %SystemRoot%\System32\csrss.exe

Parent Process: Created by an instance of smss.exe that exits, so analysis tools usually do not provide the parent process name.

**Number of Instances:** Two or more

**User Account:** Local System

**Start Time:** Within seconds of boot time for the first two instances (for Session 0 and 1). Start times for additional instances occur as new sessions are created, although often only Sessions 0 and 1 are created.

**Description:** The Client/Server Run-Time Subsystem is the user-mode process for the Windows subsystem. Its duties include managing processes and threads, importing many of the DLLs that provide the Windows API, and facilitating shutdown of the GUI during system shutdown. An instance of csrss.exe will run for each session. Session 0 is for services and Session 1 for the local console session. Additional sessions are created through the use of Remote Desktop and/or Fast User Switching. Each new session results in a new instance of csrss.exe.

#### services.exe

Image Path: %SystemRoot%\System32\services.exe

Parent Process: wininit.exe **Number of Instances: One** 

**User Account:** Local System

**Start Time:** Within seconds of boot time

**Description:** Implements the Unified Background Process Manager (UBPM), which is responsible for background activities such as services and scheduled tasks. Services.exe also implements the Service Control Manager (SCM), which specifically handles the loading of services and device drivers marked for auto-start. In addition, once a user has successfully logged on interactively, the SCM (services.exe) considers the boot successful and sets the Last Known Good control set (HKLM\SYSTEM\Select\LastKnownGood) to the value of the CurrentControlSet.

#### svchost.exe

Image Path: %SystemRoot%\system32\svchost.exe

Parent Process: services.exe (most often)

**Number of Instances:** Many (generally at least 10)

User Account: Varies depending on sychost instance, though it typically will be Local System, Network Service, or Local Service accounts. Windows 10 also has some instances running as logged-on users.

**Start Time:** Typically within seconds of boot time. However, services can be started after boot (e.g., at logon), which results in new instances of **svchost.exe** after boot time.

**Description:** Generic host process for Windows services. It is used for running service DLLs. Windows will run multiple instances of svchost.exe, each using a unique "-k" parameter for grouping similar services. Typical "-k" parameters include DcomLaunch, RPCSS, LocalServiceNetworkRestricted, LocalServiceNoNetwork,

LocalServiceAndNoImpersonation, netsvcs, NetworkService, and more. Malware authors often take advantage of the ubiquitous nature of svchost.exe and use it either to host a malicious DLL as a service, or run a malicious process named svchost.exe or similar spelling. Beginning in Windows 10 version 1703, Microsoft changed the default grouping of similar services if the system has more than 3.5 GB of RAM. In such cases, most services will run under their own instance of svchost.exe. On systems with more than 3.5 GB RAM, expect to see more than 50 instances of svchost.exe (the screenshot in the poster is a Windows 10 VM with 3 GB RAM).

#### Isaiso.exe

lmage Path: %SystemRoot%\System32\lsaiso.exe Parent Process: wininit.exe

Number of Instances: Zero or one

**User Account:** Local System

**Start Time:** Within seconds of boot time

**Description:** When Credential Guard is enabled, the functionality of **lsass.exe** is split between two processes – itself and lsaiso.exe. Most of the functionality stays within lsass.exe, but the important role of safely storing account credentials moves to lsaiso.exe. It provides safe storage by running in a context that is isolated from other processes through hardware virtualization technology. When remote authentication is required, lsass.exe proxies the requests using an RPC channel with <code>lsaiso.exe</code> in order to authenticate the user to the remote service. Note that if Credential Guard is not enabled, lsaiso.exe should not be running on the system.

#### Isass.exe

Image Path: %SystemRoot%\System32\lsass.exe Parent Process: wininit.exe

**Number of Instances: One** 

**User Account:** Local System

**Start Time:** Within seconds of boot time

**Description:** The Local Security Authentication Subsystem Service process is responsible for authenticating users by calling an appropriate authentication package specified in HKLM\SYSTEM\CurrentControlSet\Control\Lsa. Typically, this will be Kerberos for domain accounts or MSV1\_0 for local accounts. In addition to authenticating users, **1sass.exe** is also responsible for implementing the local security policy (such as password policies and audit policies) and for writing events to the security event log. Only one instance of this process should occur and it should rarely have child processes (EFS is a known exception).

### explorer.exe

Image Path: %SystemRoot%\explorer.exe

Parent Process: Created by an instance of userinit.exe that exits, so analysis tools usually do not provide the parent process name.

Number of Instances: One or more per interactively logged-on user

**User Account:** <logged-on user(s)>

**Start Time:** First instance starts when the owner's interactive logon begins

**Description:** At its core, Explorer provides users access to files. Functionally, though, it is both a file browser via Windows Explorer (though still explorer.exe) and a user interface providing features such as the user's Desktop, the Start Menu, the Taskbar, the Control Panel, and application launching via file extension associations and shortcut files. Explorer.exe is the default user interface specified in the Registry value HKLM\SOFTWARE\ Microsoft\Windows NT\CurrentVersion\Winlogon\Shell, though Windows can alternatively function with another interface such as cmd.exe or powershell.exe. Notice that the legitimate explorer.exe resides in the **%SystemRoot**% directory rather than **%SystemRoot**%\System32. Multiple instances per user can occur, such as when the option "Launch folder windows in a separate process" is enabled.

conhost.exe

CPU Usage: 4.50% | Physical Memory: 20.67% | Processes: 125

## Hunt Evil: Lateral Movement

During incident response and threat hunting, it is critical to understand how attackers move around your network. Lateral movement is an inescapable requirement for attackers to stealthily move from system to system and accomplish their objectives. Every adversary, including the most skilled, will use some form of lateral movement technique described here during a breach. Understanding lateral movement tools and techniques allows responders to hunt more efficiently, quickly perform incident response scoping, and better anticipate future attacker activity. Tools and techniques to hunt the artifacts described below are detailed in the SANS DFIR course FOR508: Advanced Digital Forensics, Incident Response, and Threat Hunting

#### **Additional Event Logs**

destination logon (e.g. bad

password)

•net.exe

•net1.exe

stem and last execution date/time

Process-tracking events, Sysmon, and similar logging capabilities are not listed here for the sake of brevity. However, this type of enhanced logging can provide significant visibility of an intruder's lateral movement, given that the logs are not overwritten or otherwise deleted.

#### **Additional FileSystem Artifacts**

Deep-dive analysis techniques such as file carving, volume shadow analysis, and NTFS log file analysis can be instrumental in recovering many of these artifacts (including the recovery of registry and event log files and records).

#### **Additional References**

SANS DFIR FOR508 course: http://sans.org/FOR508 ATT&CK Lateral Movement: http://for508.com/attck-lm JPCERT Lateral Movement: http://for508.com/jpcert-lm

#### **Artifacts in Memory Analysis**

Artifacts in memory analysis will allow for additional tracking of potential evidence of execution and command line history. We recommend auditing and dumping the "conhost" processes on the various systems. Example: vol.py -f memory.img --profile=Frofile> memdump -n conhost --dump-dir=. strings -t d -e 1 \*.dmp >> conhost.uni

Perform searches for executable keywords using grep. Also check running processes (mstsc, rdpclip, etc.).

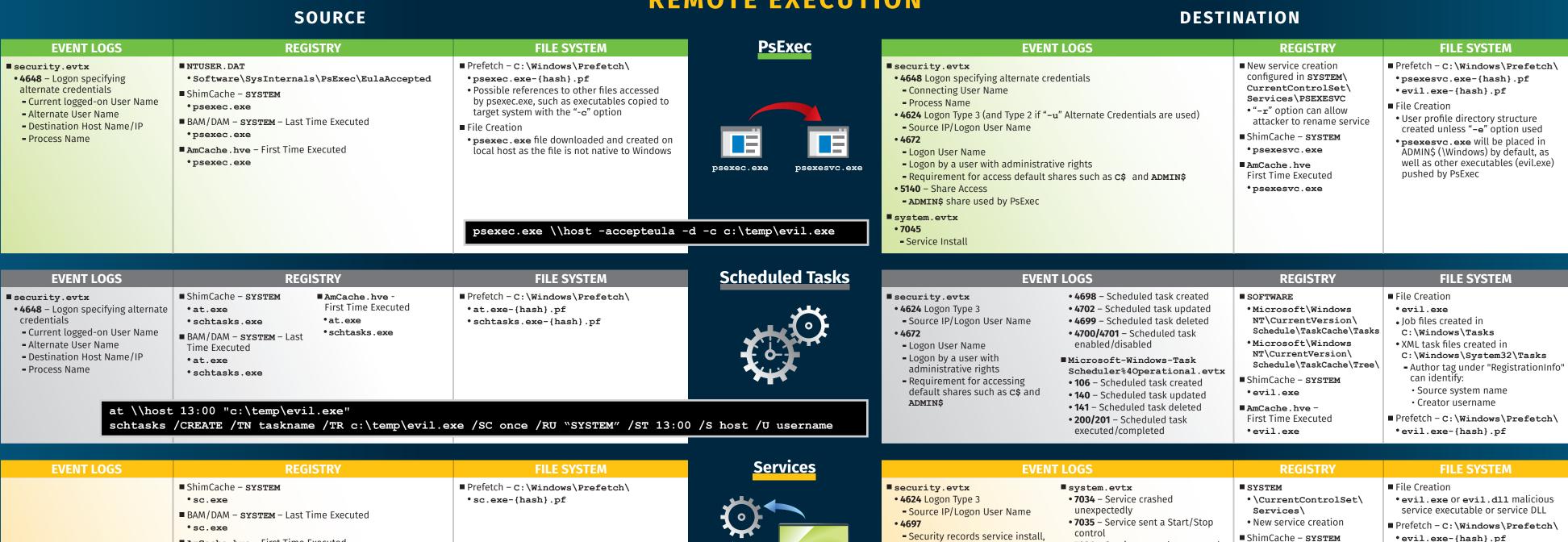
**DESTINATION** 

#### **REMOTE ACCESS** SOURCE

#### Remote Desktop **EVENT LOGS REGISTRY FILE SYSTEM EVENT LOGS** REGISTRY **FILE SYSTEM** ■ Remote desktop destinations ■ UserAssist - NTUSER.DAT ■ Jumplists - C: \Users\<Username>\ ■ Security Event Log -■ ShimCache - **SYSTEM** ■ Prefetch - C:\Windows\Prefetch\ security.evtx ■Microsoft-Windows-Terminal are tracked per-user AppData\Roaming\Microsoft\Windows\ security.evtx Services-RemoteConnection • 4648 - Logon specifying alternate •mstsc.exe Remote •rdpclip.exe •rdpclip.exe-{hash}.pf Recent\AutomaticDestinations\ • 4624 Logon Type 10 Manager%4Operational.evtx credentials - if NLA enabled on • NTUSER\Software\ Desktop Client execution •tstheme.exe •tstheme.exe-{hash}.pf • Last Time Executed destination Microsoft\Terminal • {MSTSC-APPID}-- Source IP/Logon User Name ■ AmCache.hve -- Current logged-on User Name Server Client\Servers automaticDestinations-ms - Source IP/Logon User Name Number of Times Executed • 4778/4779 First Time Executed Tracks remote desktop connection - IP Address of Source/Source Blank user name may indicate Alternate User Name ■ ShimCache - **SYSTEM** ■ RecentApps - **NTUSER.DAT** •rdpclip.exe destination and times System Name use of Sticky Keys - Destination Host Name/IP • mstsc. exe Remote •mstsc.exe Remote • tstheme.exe - Logon User Name - Process Name Desktop Client ■ Prefetch - C:\Windows\Prefetch\ ■Microsoft-Windows-Terminal Desktop Client execution •mstsc.exe-{hash}.pf Services-LocalSession • Last Time Executed Microsoft-Windows-Microsoft-Windows-■ BAM/DAM - SYSTEM - Last Manager%4Operational.evtx TerminalServices-Time Executed • Number of Times Executed RemoteDesktopServices-■ Bitmap Cache - C:\USERS\<USERNAME>\ • 21, 22, 25 RdpCoreTS%4Operational.evtx RDPClient%4Operational.evtx •mstsc.exe Remote AppData\Local\Microsoft\Terminal RecentItems subkey tracks Source IP/Logon User Name Desktop Client • 1024 • 131 – Connection Attempts connection destinations and Server Client\Cache • 41 •bcache##.bmc Source IP Destination Host Name ■ AmCache.hve - First Time - Logon User Name • 98 – Successful Connections • 1102 •cache###.bin Executed - Destination IP Address • mstsc.exe **Map Network Shares EVENT LOGS REGISTRY FILE SYSTEM EVENT LOGS REGISTRY FILE SYSTEM** (net.exe) ■ MountPoints2 – Remotely mapped shares ■ Prefetch - C:\Windows\Prefetch\ Security Event Log – • **4768** – TGT Granted ■ File Creation security.evtx security.evtx - Source Host Name/Logon User • NTUSER\Software\Microsoft\Windows\ • Attacker's files (malware) copied to • 4648 - Logon specifying •net.exe-{hash}.pf to C\$ or Admin\$ • **4624** Logon Type 3 destination system CurrentVersion\Explorer\MountPoints2 alternate credentials •net1.exe-{hash}.pf Source IP/Logon User Name - Available only on domain controller · Look for Modified Time before - Current logged-on User Name ■ Shellbags - **USRCLASS.DAT** ■ User Profile Artifacts Creation Time 4672 • 4769 – Service Ticket Granted if - Alternate User Name • Review shortcut files and jumplists for Remote folders accessed inside an interactive session via authenticating to Domain Controller • Creation Time is time of file copy Destination Host Name/IP Logon User Name Explorer by attackers remote files accessed by attackers, if they - Destination Host Name/Logon User Logon by user with Process Name had interactive access (RDP) ShimCache - SYSTEM administrative rights Name Microsoft-Windows-•net.exe - Source IP Requirement for accessing SmbClient%4Security.evtx default shares such as c\$ - Available only on domain controller • **31001** – Failed logon to ■ BAM/DAM - NTUSER.DAT - Last Time Executed and ADMINS • 5140 4776 - NTLM if authenticating •net.exe Share Access Destination Host Name to Local System •net1.exe • 5145 User Name for failed logon - Source Host Name/Logon - Auditing of shared files - NOISY! ■ AmCache.hve - First Time Executed Reason code for failed **User Name**

#### REMOTE EXECUTION

net use z: \\host\c\$ /user:domain\username <password>





	■ AmCache.hve - First Time Executed • wmic.exe  wmic /node:host process call create "C:\		<pre>\temp\evil.exe" Win32_Process -Name create -Argument "c:\temp\evil.exe"</pre>		• 4672  • Logon User Name  • Logon by an a user with administrative rights	<ul> <li>Indicates time of wmiprvse execution and path to provider DLL – attackers sometimes install malicious WMI provider DLLs</li> <li>5860, 5861</li> <li>Registration of Temporary (5860) and Permanent (5861) Event Consumers. Typically used for persistence, but can be used for remote execution.</li> </ul>	• wmiprvse.exe • evil.exe  • AmCache.hve - First Time Executed • scrcons.exe • mofcomp.exe • wmiprvse.exe • evil.exe	to manage the WMI Repository  Prefetch - C:\Windows\Prefetch\ •scrcons.exe-{hash}.pf •mofcomp.exe-{hash}.pf •wmiprvse.exe-{hash}.pf •evil.exe-{hash}.pf  Unauthorized changes to the WMI Repository in C:\Windows\ System32\wbem\Repository
EVENT	LOGS	REGISTRY	FILE SYSTEM	PowerShell Remoting	EVEN	IT LOGS	REGISTRY	FILE SYSTEM
■ security.evtx  • 4648 - Logon specifying alternate credentials  • Current logged-on User Name  • Alternate User Name  • Destination Host Name/IP  • Process Name  ■ Microsoft-Windows- WinRM%4Operational.evtx  • 6 - WSMan Session initialize  • Session created  • Destination Host Name or IP  • Current logged-on User Name	= Soccion closed	• powershell.exe  ■ AmCache.hve - First Time Executed • powershell.exe  - PSSession - Compute	■ Prefetch - C:\Windows\Prefetch\         •powershell.exe-{hash}.pf         •PowerShell scripts (.ps1 files) that run within 10 seconds of powershell.exe launching will be tracked in powershell.exe prefetch file         ■ Command history         C:\USERS\ <username>\AppData\Roaming\         Microsoft\Windows\PowerShell\         PSReadline\ConsoleHost_history.txt         •With PS v5+, a history file with previous 4096 commands is maintained per user  CENAME host  PSRing host —ScriptBlock {Start-Products of the start of t</username>	powershell.exe wsmprovhost.exe	■ security.evtx  • 4624 Logon Type 3  • Source IP/Logon User Name  • 4672  • Logon User Name  • Logon by an a user with administrative rights  ■ Microsoft-Windows- PowerShell%4Operational.evtx  • 4103, 4104 - Script Block logging  • Logs suspicious scripts by default in PS v5  • Logs all scripts if configured  • 53504 Records the authenticating user	user	■ ShimCache - SYSTEM  • wsmprovhost.exe • evil.exe  ■ SOFTWARE  • Microsoft\PowerShell\1 \ShellIds\Microsoft. PowerShell\ ExecutionPolicy  • Attacker may change execution policy to a less restrictive setting, such as "bypass"  ■ AmCache.hve - First Time Executed • wsmprovhost.exe • evil.exe	■ File Creation         • evil.exe         • With Enter-PSSession, a user profile directory may be created         ■ Prefetch - C:\Windows\Prefetch\         • evil.exe-{hash].pf         • wsmprovhost.exe-{hash].pf

