SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

Hunt Evil

POSTER

$25.00
Poster_FOR508_v4.2_12-18

Poster was created by Rob Lee and Mike Pilkington
with support of the SANS DFIR Faculty

dfir.sans.org

SANS DFIR

DIGITAL FORENSICS & INCIDENT RESPONSE

FOR500
Windows Forensics
GCFE

FOR518
Mac and iOS Forensic Analysis and Incident Response

FOR526
Advanced Memory Forensics & Threat Detection

FOR585
Advanced Smartphone Forensics GASF

OPERATING SYSTEM & DEVICE IN-DEPTH

INCIDENT RESPONSE & THREAT HUNTING

FOR508
Advanced Incident Response and Threat Hunting GCFA

FOR572
Advanced Network Forensics: Threat Hunting, Analysis, and Incident Response GNFA

FOR578
Cyber Threat Intelligence
GCTI

FOR610
REM: Malware Analysis
GREM

SEC504
Hacker Tools, Techniques, Exploits, and Incident Handling GCIH

@sansforensics | sansforensics | dfir.to/DFIRCast | dfir.to/gplus-sansforensics | dfir.to/MAIL-LIST

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.**

System

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 (**lsm.exe**) was also started by wininit.exe. As of Windows 10, that functionality has moved to a service DLL (lsm.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 workstation unlock.

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

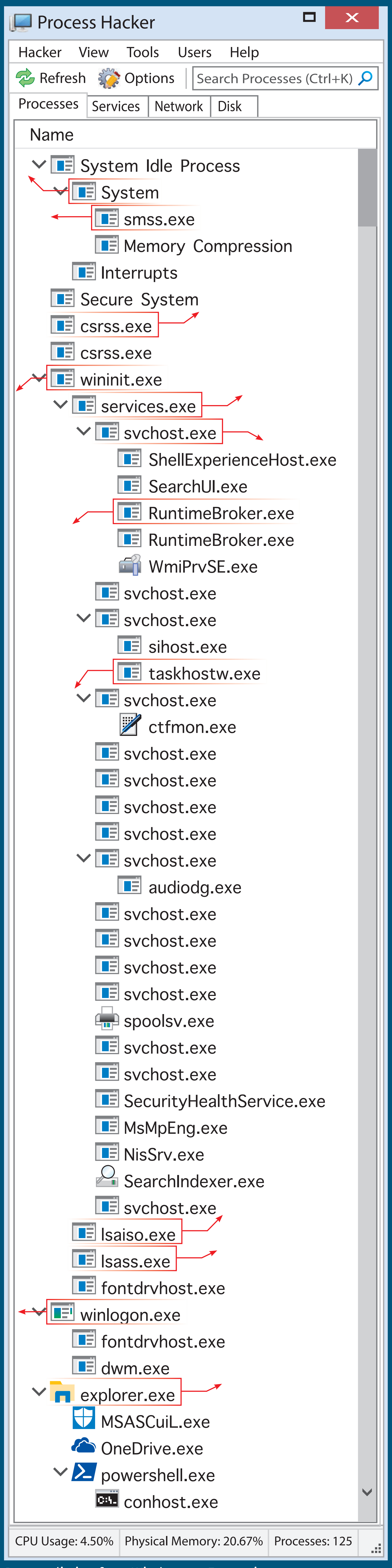
Parent Process: Created by an instance of **smss.exe** that exits, so analysis tools usually do not provide the parent process name.

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**.



Process Hacker

Hacker View Tools Users Help
Refresh Options Search Processes (Ctrl+K)
Processes Services Network Disk

```
Name
System Idle Process
  System
    smss.exe
    Memory Compression
    Interrupts
  Secure System
  csrss.exe
  csrss.exe
  wininit.exe
    services.exe
      svchost.exe
        ShellExperienceHost.exe
        SearchUI.exe
        RuntimeBroker.exe
        RuntimeBroker.exe
        WmiPrvSE.exe
      svchost.exe
      svchost.exe
        sihost.exe
        taskhostw.exe
      svchost.exe
        ctfmon.exe
      svchost.exe
      svchost.exe
      svchost.exe
      svchost.exe
      svchost.exe
        audiodg.exe
      svchost.exe
      svchost.exe
      svchost.exe
      svchost.exe
      spoolsv.exe
      svchost.exe
      svchost.exe
      SecurityHealthService.exe
      MsMpEng.exe
      NisSrv.exe
      SearchIndexer.exe
      svchost.exe
    lsaiso.exe
    lsass.exe
    fontdrvhost.exe
  winlogon.exe
    fontdrvhost.exe
    dwm.exe
  explorer.exe
    MSASCuiL.exe
    OneDrive.exe
    powershell.exe
      conhost.exe
```

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

Process listing from Windows 10 Enterprise

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 **svchost** 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).

lsaiso.exe

Image 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 **lsaiso.exe** 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.

lsass.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, **lsass.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 not have child processes.

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.

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

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

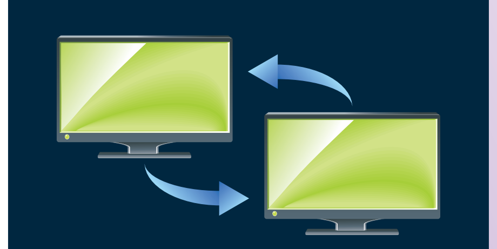
Artifacts in memory analysis will allow for additional tracking of potential evidence of execution and command line history. We recommend auditing and dumping the "conhost" processes on the various systems. Example:

```
vol.py -f memory.img --profile=<profile> memdump -n conhost --dump-dir=.
strings -t d -e l *.dmp >> conhost.uni
```

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

REMOTE ACCESS

Remote Desktop



SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials - if NLA enabled on destination
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name
- **Microsoft-Windows-TerminalServices-RDPClient%4Operational.evtx**
  - **1024**
    - Destination Host Name
  - **1102**
    - Destination IP Address

REGISTRY

- Remote desktop destinations are tracked per-user
  - **NTUSER\Software\Microsoft\Terminal Server Client\Servers**
- ShimCache – **SYSTEM**
  - **mstsc.exe** Remote Desktop Client
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **mstsc.exe** Remote Desktop Client
- **AmCache.hve** – First Time Executed
  - **mstsc.exe**
- UserAssist – **NTUSER.DAT**
  - **mstsc.exe** Remote Desktop Client execution
  - Last Time Executed
  - Number of Times Executed
- RecentApps – **NTUSER.DAT**
  - **mstsc.exe** Remote Desktop Client execution
  - Last Time Executed
  - Number of Times Executed
  - RecentItems subkey tracks connection destinations and times

FILE SYSTEM

- Jumplists – **C:\Users\<Username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\**
  - **{MSTSC-APPID}-automaticDestinations-ms**
  - Tracks remote desktop connection destination and times
- Prefetch – **C:\Windows\Prefetch\**
  - **mstsc.exe-{hash}.pf**
- Bitmap Cache – **C:\USERS\<USERNAME>\AppData\Local\Microsoft\Terminal Server Client\Cache**
  - **bcache##.bmc**
  - **cache####.bin**

DESTINATION

EVENT LOGS

- **Security Event Log – security.evtx**
  - **4624** Logon Type 10
    - Source IP/Logon User Name
  - **4778/4779**
    - IP Address of Source/Source System Name
    - Logon User Name
- **Microsoft-Windows-RemoteDesktopServices-RdpCoreTS%4Operational.evtx**
  - **131** – Connection Attempts
    - Source IP
  - **98** – Successful Connections
- **Microsoft-Windows-Terminal Services-RemoteConnection Manager%4Operational.evtx**
  - **1149**
    - Source IP/Logon User Name
      - Blank user name may indicate use of Sticky Keys
- **Microsoft-Windows-Terminal Services-LocalSession Manager%4Operational.evtx**
  - **21, 22, 25**
    - Source IP/Logon User Name
  - **41**
    - Logon User Name

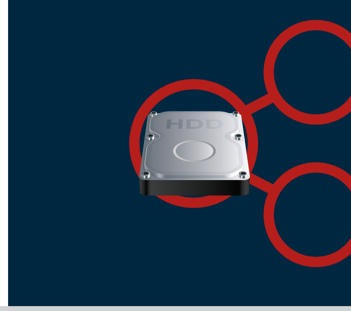
REGISTRY

- ShimCache – **SYSTEM**
  - **rdpclip.exe**
  - **tstheme.exe**
- **AmCache.hve** – First Time Executed
  - **rdpclip.exe**
  - **tstheme.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **rdpclip.exe-{hash}.pf**
  - **tstheme.exe-{hash}.pf**

Map Network Shares (net.exe) to C$ or Admin$



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

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name
- **Microsoft-Windows-SmbClient%4Security.evtx**
  - **31001** – Failed logon to destination
    - Destination Host Name
    - User Name for failed logon
    - Reason code for failed destination logon (e.g. bad password)

REGISTRY

- MountPoints2 – Remotely mapped shares
  - **NTUSER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2**
- Shellbags – **USRCLASS.DAT**
  - Remote folders accessed inside an interactive session via Explorer by attackers
- ShimCache – **SYSTEM**
  - **net.exe**
  - **net1.exe**
- BAM/DAM – **NTUSER.DAT** – Last Time Executed
  - **net.exe**
  - **net1.exe**
- **AmCache.hve** – First Time Executed
  - **net.exe**
  - **net1.exe**

FILE SYSTEM

- **Prefetch** – **C:\Windows\Prefetch\**
  - **net.exe-{hash}.pf**
  - **net1.exe-{hash}.pf**
- User Profile Artifacts
  - Review shortcut files and jumplists for remote files accessed by attackers, if they had interactive access (RDP)

DESTINATION

EVENT LOGS

- **Security Event Log – security.evtx**
  - **4624** Logon Type 3
    - Source IP/Logon User Name
  - **4672**
    - Logon User Name
    - Logon by user with administrative rights
    - Requirement for accessing default shares such as **C$** and **ADMIN$**
  - **4776** – NTLM if authenticating to Local System
    - Source Host Name/Logon User Name
  - **4768** – TGT Granted
    - Source Host Name/Logon User Name
    - Available only on domain controller
  - **4769** – Service Ticket Granted if authenticating to Domain Controller
    - Destination Host Name/Logon User Name
    - Source IP
    - Available only on domain controller
  - **5140**
    - Share Access
  - **5145**
    - Auditing of shared files – NOISY!

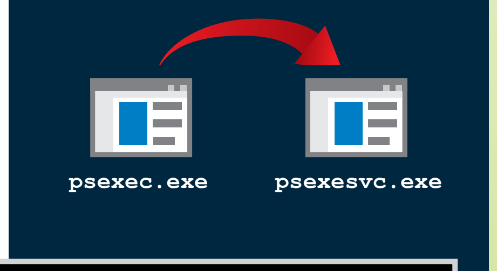
REGISTRY

FILE SYSTEM

- File Creation
  - Attacker's files (malware) copied to destination system
  - Look for Modified Time before Creation Time
  - Creation Time is time of file copy

REMOTE EXECUTION

PsExec



```
psexec.exe \\host -accepteula -d -c c:\temp\evil.exe
```

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name

REGISTRY

- **NTUSER.DAT**
  - **Software\SysInternals\PsExec\EulaAccepted**
- ShimCache – **SYSTEM**
  - **psexec.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **psexec.exe**
- **AmCache.hve** – First Time Executed
  - **psexec.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **psexec.exe-{hash}.pf**
  - Possible references to other files accessed by psexec.exe, such as executables copied to target system with the "**-c**" option
- File Creation
  - **psexec.exe** file downloaded and created on local host as the file is not native to Windows

DESTINATION

EVENT LOGS

- **security.evtx**
  - **4624** Logon Type 3 (and Type 2 if "**-u**" Alternate Credentials are used)
    - Source IP/Logon User Name
  - **4672**
    - Logon User Name
    - Logon by a user with administrative rights
    - Requirement for access default shares such as **C$** and **ADMIN$**
  - **5140** – Share Access
    - **ADMIN$** share used by PsExec
- **system.evtx**
  - **7045**
    - Service Install

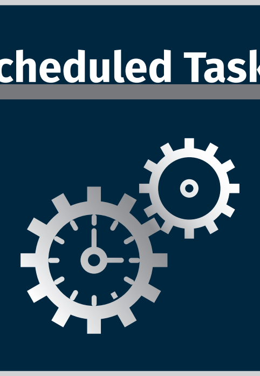
REGISTRY

- New service creation configured in **SYSTEM\CurrentControlSet\Services\PSEXESVC**
  - "**-r**" option can allow attacker to rename service
- ShimCache – **SYSTEM**
  - **psexesvc.exe**
- **AmCache.hve** First Time Executed
  - **psexesvc.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **psexesvc.exe-{hash}.pf**
  - **evil.exe-{hash}.pf**
- File Creation
  - User profile directory structure created unless "**-e**" option used
  - **psexesvc.exe** will be placed in ADMIN$ (\Windows) by default, as well as other executables (evil.exe) pushed by PsExec

Scheduled Tasks



```
at \\host 13:00 "c:\temp\evil.exe"
schtasks /CREATE /TN taskname /TR c:\temp\evil.exe /SC once /RU "SYSTEM" /ST 13:00 /S host /U username
```

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name

REGISTRY

- ShimCache – **SYSTEM**
  - **at.exe**
  - **schtasks.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **at.exe**
  - **schtasks.exe**
- **AmCache.hve** - First Time Executed
  - **at.exe**
  - **schtasks.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **at.exe-{hash}.pf**
  - **schtasks.exe-{hash}.pf**

DESTINATION

EVENT LOGS

- **security.evtx**
  - **4624** Logon Type 3
    - Source IP/Logon User Name
  - **4672**
    - Logon User Name
    - Logon by a user with administrative rights
    - Requirement for accessing default shares such as **C$** and **ADMIN$**
  - **4698** – Scheduled task created
  - **4702** – Scheduled task updated
  - **4699** – Scheduled task deleted
  - **4700/4701** – Scheduled task enabled/disabled
- **Microsoft-Windows-Task Scheduler%4Operational.evtx**
  - **106** – Scheduled task created
  - **140** – Scheduled task updated
  - **141** – Scheduled task deleted
  - **200/201** – Scheduled task executed/completed

REGISTRY

- **SOFTWARE**
  - **Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks**
  - **Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tree\**
- ShimCache – **SYSTEM**
  - **evil.exe**
- **AmCache.hve** – First Time Executed
  - **evil.exe**

FILE SYSTEM

- File Creation
  - **evil.exe**
  - Job files created in **C:\Windows\Tasks**
  - XML task files created in **C:\Windows\System32\Tasks**
    - Author tag under "RegistrationInfo" can identify:
      - Source system name
      - Creator username
- Prefetch – **C:\Windows\Prefetch\**
  - **evil.exe-{hash}.pf**

Services

```
sc \\host create servicename binpath= "c:\temp\evil.exe"
sc \\host start servicename
```

SOURCE

EVENT LOGS

REGISTRY

- ShimCache – **SYSTEM**
  - **sc.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **sc.exe**
- **AmCache.hve** – First Time Executed
  - **sc.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **sc.exe-{hash}.pf**

DESTINATION

EVENT LOGS

- **security.evtx**
  - **4624** Logon Type 3
    - Source IP/Logon User Name
  - **4697**
    - Security records service install, if enabled
    - Enabling non-default Security events such as ID 4697 are particularly useful if only the Security logs are forwarded to a centralized log server
- **system.evtx**
  - **7034** – Service crashed unexpectedly
  - **7035** – Service sent a Start/Stop control
  - **7036** – Service started or stopped
  - **7040** – Start type changed (**Boot | On Request | Disabled**)
  - **7045** – A service was installed on the system

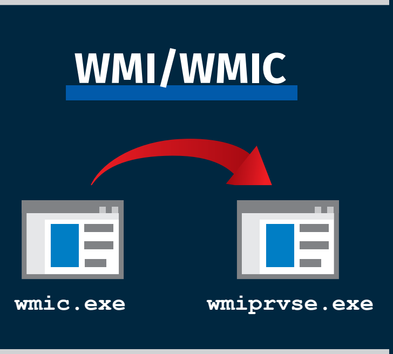
REGISTRY

- **SYSTEM**
  - **\CurrentControlSet\Services\**
  - New service creation
- ShimCache – **SYSTEM**
  - **evil.exe**
  - ShimCache records existence of malicious service executable, unless implemented as a service DLL
- **AmCache.hve** – First Time Executed
  - **evil.exe**

FILE SYSTEM

- File Creation
  - **evil.exe** or **evil.dll** malicious service executable or service DLL
- Prefetch – **C:\Windows\Prefetch\**
  - **evil.exe-{hash}.pf**

WMI/WMIC



```
wmic /node:host process call create "C:\temp\evil.exe"
Invoke-WmiMethod –Computer host –Class Win32_Process –Name create –Argument "c:\temp\evil.exe"
```

SOURCE

EVENT LOGS

- **security.evtx**
  - **4648** – Logon specifying alternate credentials
    - Current logged-on User Name
    - Alternate User Name
    - Destination Host Name/IP
    - Process Name

REGISTRY

- ShimCache – **SYSTEM**
  - **wmic.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **wmic.exe**
- **AmCache.hve** – First Time Executed
  - **wmic.exe**

FILE SYSTEM

- Prefetch – **C:\Windows\Prefetch\**
  - **wmic.exe-{hash}.pf**

DESTINATION

EVENT LOGS

- **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-WMI-Activity%4Operational.evtx**
  - **5857**
    - Indicates time of wmiprvse execution and path to provider DLL – attackers sometimes install malicious WMI provider DLLs
  - **5860, 5861**
    - Registration of Temporary (5860) and Permanent (5861) Event Consumers. Typically used for persistence, but can be used for remote execution.

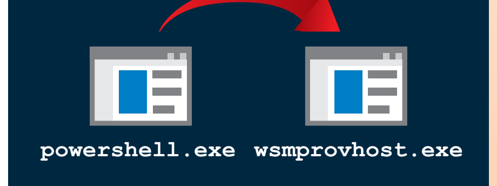
REGISTRY

- ShimCache – **SYSTEM**
  - **wmiprvse.exe**
  - **evil.exe**
  - **mofcomp.exe**
- **AmCache.hve** – First Time Executed
  - **wmiprvse.exe**
  - **evil.exe**
  - **mofcomp.exe**

FILE SYSTEM

- File Creation
  - **evil.exe**
  - **evil.mof** – .mof files can be used to manage the WMI Repository
- Prefetch – **C:\Windows\Prefetch\**
  - **evil.exe-{hash}.pf**
  - **wmiprvse.exe-{hash}.pf**
  - **mofcomp.exe-{hash}.pf**
- Unauthorized changes to the WMI Repository in **C:\Windows\System32\wbem\Repository**

PowerShell Remoting



```
Enter-PSSession –ComputerName host
Invoke-Command –ComputerName host –ScriptBlock {Start-Process c:\temp\evil.exe}
```

SOURCE

EVENT LOGS

- **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
  - **8, 15, 16, 33** – WSMan Session deinitialization
    - Closing of WSMan session
    - Current logged-on User Name
- **Microsoft-Windows-PowerShell%4Operational.evtx**
  - **40961, 40962**
    - Records the local initiation of powershell.exe and associated user account
  - **8193** & **8194**
    - Session created
  - **8197** - Connect
    - Session closed

REGISTRY

- ShimCache – **SYSTEM**
  - **powershell.exe**
- BAM/DAM – **SYSTEM** – Last Time Executed
  - **powershell.exe**
- **AmCache.hve** – First Time Executed
  - **powershell.exe**

FILE SYSTEM

- 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

DESTINATION

EVENT LOGS

- **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
- **Windows PowerShell.evtx**
  - **400/403** "ServerRemoteHost" indicates start/end of Remoting session
  - **800** Includes partial script code
- **Microsoft-Windows-WinRM%4Operational.evtx**
  - **91** Session creation
  - **168** Records the authenticating user

REGISTRY

- 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 SYSTEM

- 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**

Evidence of Program Execution

UserAssist

Description:
GUI-based programs launched from the desktop are tracked in the launcher on a Windows System.

Location:
NTUSER.DAT HIVE
NTUSER.DAT\Software\Microsoft\Windows\Currentversion\Explorer\UserAssist\{GUID}\Count

Interpretation:
All values are ROT-13 Encoded

- GUID for Win7/8/10
  - **CEBFF5CD** Executable File Execution
  - **F4E57C4B** Shortcut File Execution

BAM/DAM

Description:
Windows Background Activity Moderator (BAM)

Location:
Win10
SYSTEM\CurrentControlSet\Services\bam\UserSettings\{SID}
SYSTEM\CurrentControlSet\Services\dam\UserSettings\{SID}

Investigative Notes
Provides full path of the executable file that was run on the system and last execution date/time

RecentApps

Description:
Program execution launched on the Win10 system is tracked in the RecentApps key

Location:
Win10
NTUSER.DAT\Software\Microsoft\Windows\Current Version\Search\RecentApps

Interpretation:
Each GUID key points to a recent application.
AppID = Name of Application
LastAccessTime = Last execution time in UTC
LaunchCount = Number of times executed

ShimCache

Description:

- Windows Application Compatibility Database is used by Windows to identify possible application compatibility challenges with executables.
- Tracks the executables' file name, file size, last modified time

Location:
Win7/8/10
SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache

Interpretation:
Any executable run on the Windows system could be found in this key. You can use this key to identify systems that specific malware was executed on. In addition, based on the interpretation of the time-based data you might be able to determine the last time of execution or activity on the system.

- Windows 7/8/10 contains at most 1,024 entries
  - LastUpdateTime does not exist on Win7/8/10 systems

Jump Lists

Description:

- The Windows 7-10 task bar (Jump List) is engineered to allow users to "jump" or access items they have frequently or recently used quickly and easily. This functionality cannot only include recent media files; it must also include recent tasks.
- The data stored in the AutomaticDestinations folder will each have a unique file prepended with the AppID of the associated application.

Location:
Win7/8/10
C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations

Interpretation:

- First time of execution of application.
  - Creation Time = First time item added to the AppID file.
- Last time of execution of application with file open.
  - Modification Time = Last time item added to the AppID file.
- List of Jump List IDs -> **www.forensicswiki.org/wiki/List_of_Jump_List_IDs**

Prefetch

Description:

- Increases performance of a system by pre-loading code pages of commonly used applications. Cache Manager monitors all files and directories referenced for each application or process and maps them into a .pf file. Utilized to know an application was executed on a system.
- Limited to 128 files on Win7
- Limited to 1024 files on Win8-10
- (exename)-(hash).pf

Location:
Win7/8/10
C:\Windows\Prefetch

Interpretation:

- Each .pf will include last time of execution, number of times run, and device and file handles used by the program
- Date/Time file by that name and path was first executed
  - Creation Date of .pf file (-10 seconds)
- Date/Time file by that name and path was last executed
  - Embedded last execution time of .pf file
  - Last modification date of .pf file (-10 seconds)
  - Win8-10 will contain last 8 times of execution

Amcache.hve

Description:
ProgramDataUpdater (a task associated with the Application Experience Service) uses the registry file Amcache.hve to store data during process creation

Location:
Win7/8/10
C:\Windows\AppCompat\Programs\Amcache.hve (Windows 7/8/10)

Interpretation:

- Amcache.hve – Keys = **Amcache.hve\Root\File\{Volume GUID}\#######**
- Entry for every executable run, full path information, File's $StandardInfo Last Modification Time, and Disk volume the executable was run from
- First Run Time = Last Modification Time of Key
- SHA1 hash of executable also contained in the key