Core Windows Processes

These are my own notes from the Core Windows Processes room in TryHackMe: `https://tryhackme.com/room/btwindowsinternals`, which is created by tryhackme **and** ar33zy

---

```
~# whoami
https://www.linkedin.com/in/husamshbib
https://www.youtube.com/@Inforacle
https://github.com/0xirison
```

`This document provides my notes on the Core Windows Processes room, including their normal behavior and what to look for when investigating them. The covered processes are system, smss.exe, csrss.exe, wininit.exe, services.exe, svchost.exe, lsass.exe, winlogon.exe, and explorer.exe.`

General notes:

Windows processes are categorized as follows: `Apps` , `Background processes` and `Windows processes`.

Better to check processes in task manager with image path name and command columns enabled (helps in investigating processes)

Always check parent processes such as the parent for svchost.exe must be services.exe

Using command line tools to obtaining information about the running processes on Windows OS: `tasklist` `Get-Process` or `ps` (PowerShell), and `wmic` /turn

> 💡 **Core Windows Processes:**

System (home for kernel threads)> smss.exe (windows session manager) > csrss.exe (Client Server Runtime Subsystem)

💡 system process is *the home for a special kind of thread that runs only in kernel mode (PID 4)*

`what is normal:`

Image Path: C:\Windows\system32\ntoskrnl.exe (NT OS Kernel)

Parent Process: System Idle Process (0)

R**uns once in session 0**

system > smss.exe

💡 smss.exe (**Session Manager Subsystem**) is responsible for creating new sessions and creating environment variables. It is the first user-mode process started by the kernel.

> smss.exe > “csrss.exe > wininit.exe (0 session), csrss.exe > winlogon.exe (1 session)”

runs: One master instance and child instance per session. The child instance exits after creating the session.

💡 **csrss.exe** (**Client Server Runtime Process**) is the user-mode side of the Windows subsystem. This process is responsible for the Win32 console window and process thread creation and deletion and making the Windows API available to other processes

What is unusual? An actual parent process for csrss, because smss.exe calls this process and self-terminates

The **Windows Initialization Process** , **wininit.exe** , is responsible for launching services.exe (Service Control Manager), lsass.exe (Local Security Authority), and lsaiso.exe within Session 0

Note:

`lsaiso.exe is a process associated with` **`Credential Guard and KeyGuard`** `. You will only see this process if Credential Guard is enabled. If so, it will run as a child process to wininit.exe. This process workss with lsass.exe to enhance password protection on the endpoint.`

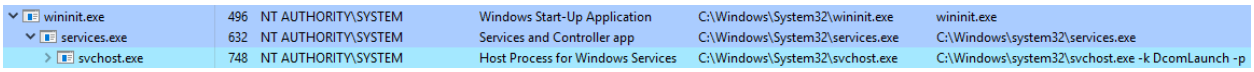
wininit.exe > services.exe

The next process is the **Service Control Manager** (SCM) or **services.exe.** Its primary responsibility is to handle **system services**: loading services, interacting with services and starting or ending services. This process also loads device drivers marked as auto-start into memory. It maintains a database that can be queried using a Windows built-in utility, `sc.exe`

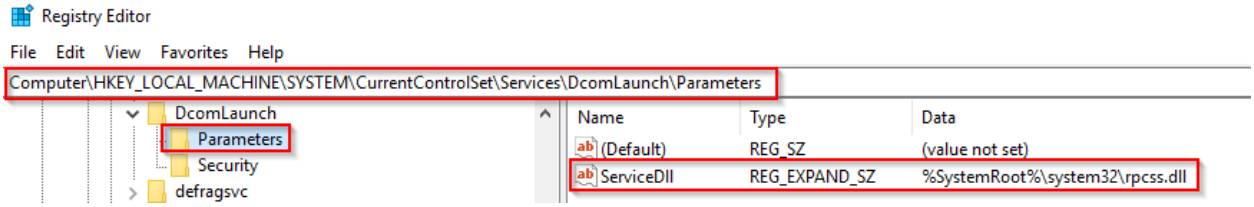
- This process is the parent to several other key processes: svchost.exe, spoolsv.exe, msmpeng.exe, and dllhost.exe
- Services.exe set last known good config `HKLM\System\Select\LastKnownGood`
- Information regarding services is stored in the registry, `HKLM\System\CurrentControlSet\Services`

wininit.exe > services.exe > svchost.exe

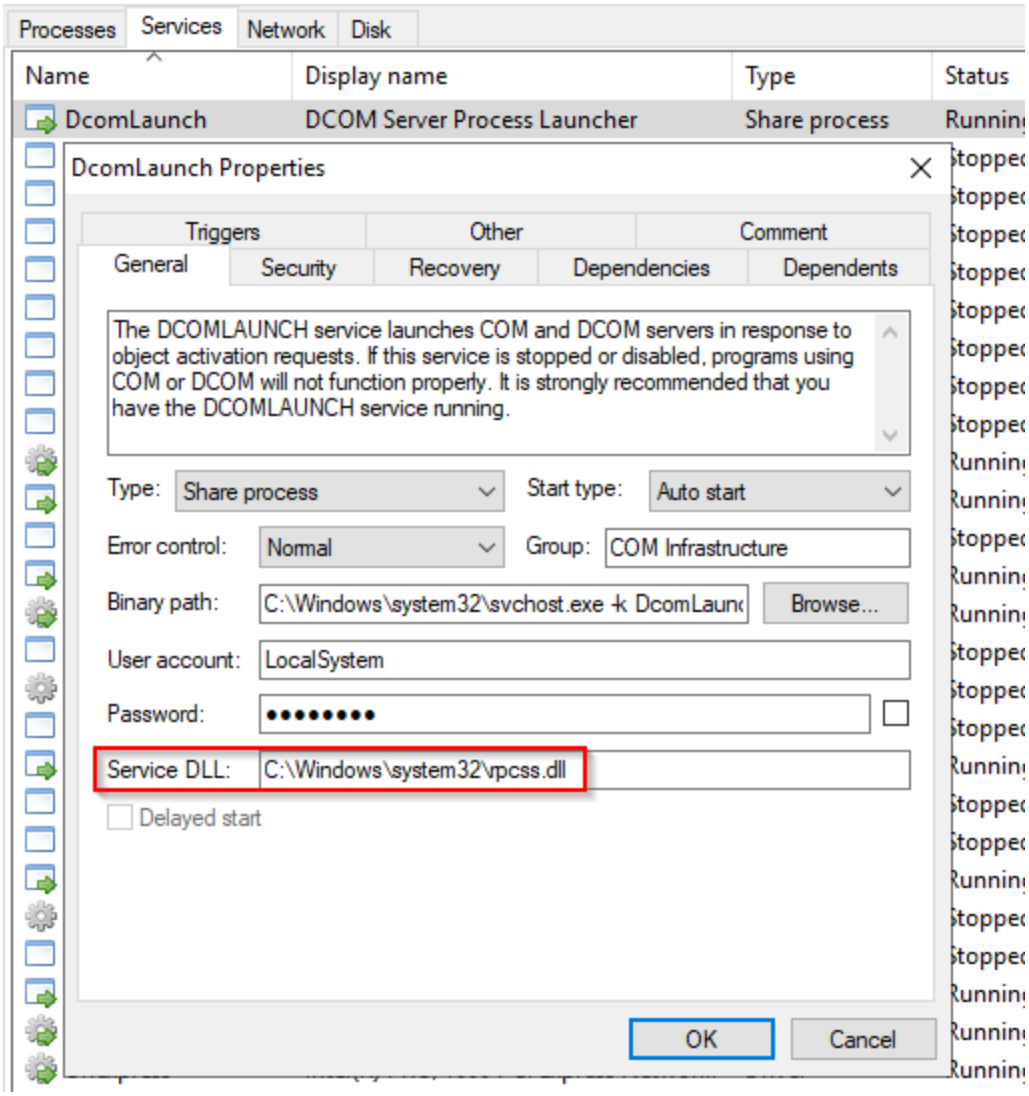
The **Service Host** (Host Process for Windows Services), or **svchost.exe**, is responsible for hosting and managing Windows services.



The services running in this process are implemented as DLLs. The DLL to implement is stored in the registry for the service under the `Parameters` subkey in `ServiceDLL`. The full path is `HKLM\SYSTEM\CurrentControlSet\Services\SERVICE NAME\Parameters`.



One of the services



Also, notice how it is structured. There is a key identifier in the binary path, and that identifier is

`-k` . This is how a **legitimate svchost.exe process is called**.

The -k parameter is for grouping similar services to share the same process.

This concept was based on the OS design and implemented to reduce resource consumption. Starting from **Windows 10 Version 1703,** services grouped into host processes changed. On machines running more than 3.5 GB of memory, each service will run its own process

Wininit.exe > lsass.exe

> 💡 Local Security Authority Subsystem Service (**LSASS**) is a process in Microsoft Windows operating systems that is responsible for enforcing the security policy on the system. It verifies users logging on to a Windows computer or server, handles password changes, and creates access tokens. It also writes to the Windows Security Log."

It creates security tokens for SAM (Security Account Manager), AD (Active Directory), and NETLOGON. It uses authentication packages specified in `HKLM\System\CurrentControlSet\Control\Lsa`

Lsass.exe is another process adversaries target like svchost.exe

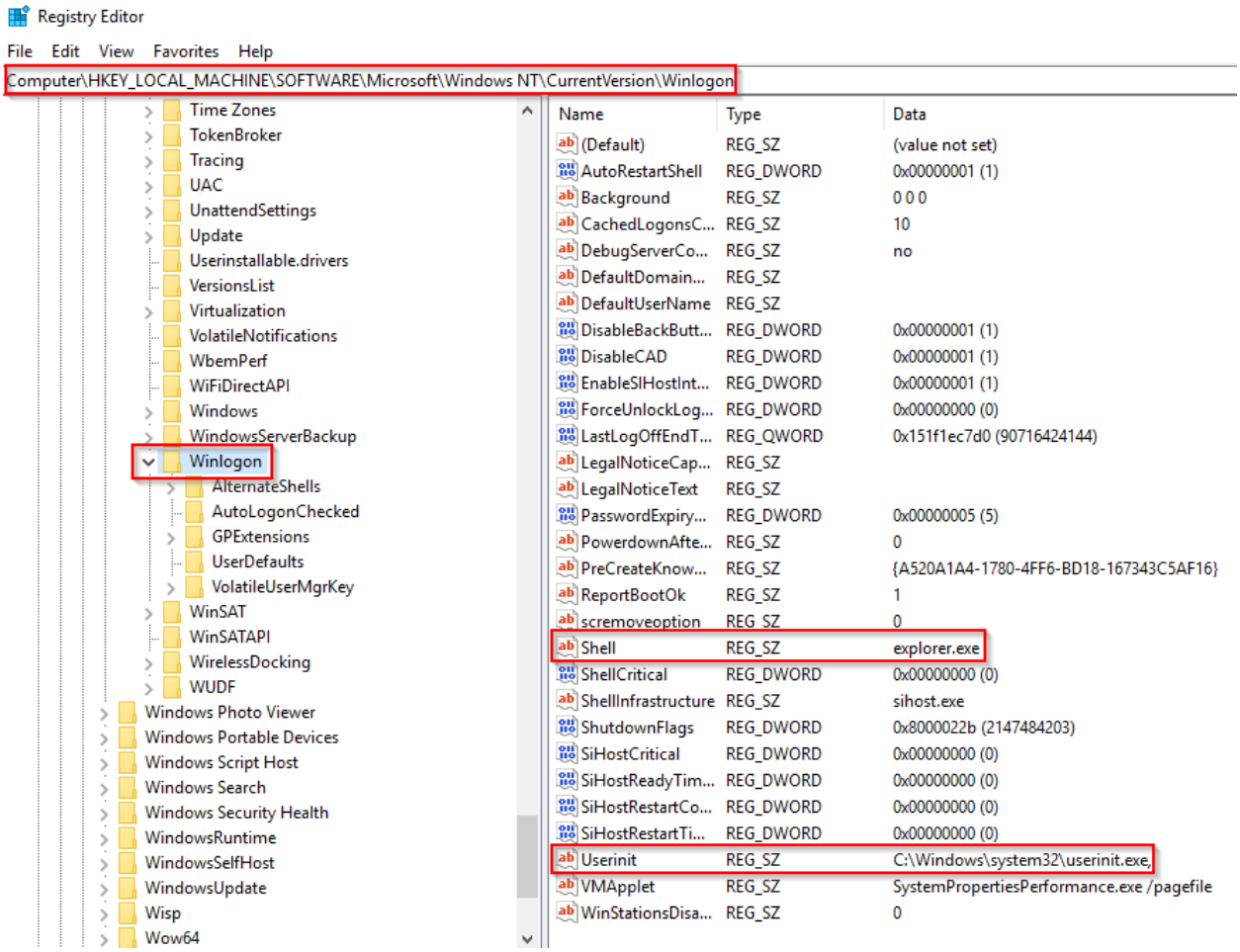
> 💡 The **Windows Logon**, **winlogon.exe**, is responsible for handling the **Secure Attention Sequence** (SAS). It is the ALT+CTRL+DELETE key combination users press to enter their username & password.

smss.exe > winlogon.exe , (then smss.exe terminate itself, so there is no existent parent for winlogon.exe)

This process is also responsible for loading the user profile. It loads the user's NTUSER.DAT into HKCU, and userinit.exe loads the user's shell. It is also responsible for locking the screen and running the user's screensaver, among other functions

`Winlogon.exe what is normal:`

- **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



`what is unusual`:

- An actual parent process. (smss.exe calls this process and self-terminates)
- Image file path other than C:\Windows\System32
- Subtle misspellings to hide rogue processes in plain sight
- Not running as SYSTEM
- Shell value in the registry other than explorer.exe

Windows Explorer, **explorer.exe**. This process gives the user access to their folders and files. It also provides functionality for other features, such as the Start Menu and Taskbar.

The Winlogon process runs userinit.exe, which launches the value in `HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell`. Userinit.exe exits after spawning explorer.exe. Because of this, the parent process is non-existent.

`What is normal?`

- **Image Path**: %SystemRoot%\explorer.exe
- **Parent Process**: Created by userinit.exe and exits
- **Number of Instances**: One or more per interactively logged-in user
- **User Account**: Logged-in user(s)
- **runs**: in session 1
- **Start Time**: First instance when the first interactive user logon session begins

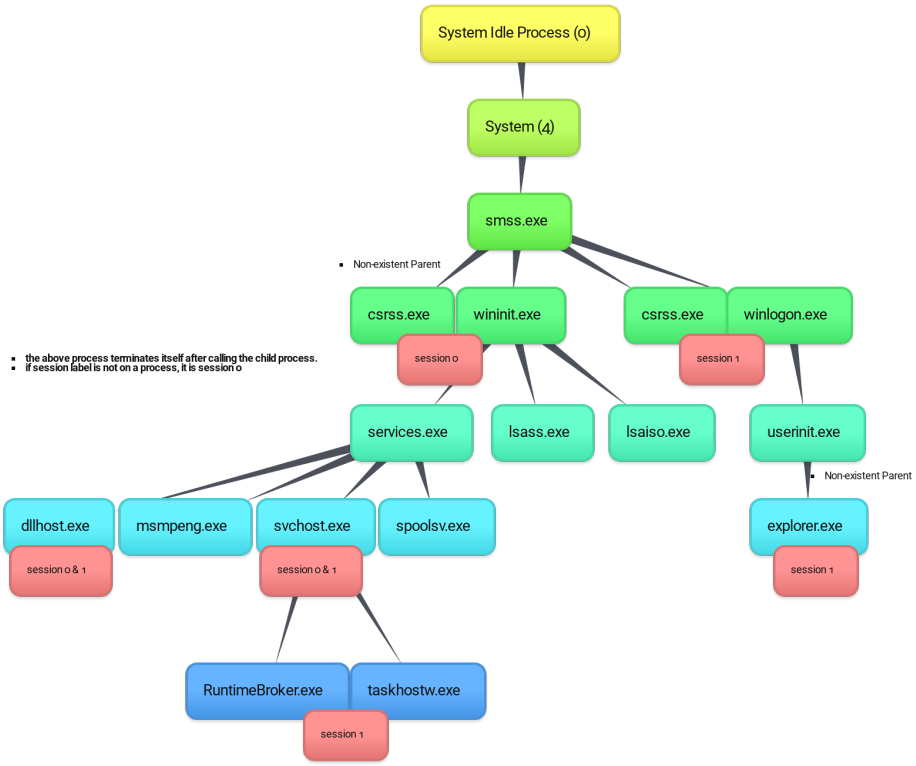
`What is unusual?`

- An actual parent process. (userinit.exe calls this process and exits)
- Image file path other than C:\Windows
- Running as an unknown user
- Subtle misspellings to hide rogue processes in plain sight
- Outbound TCP/IP connections

Note:

> 💡 Other core processes with Windows 10 are RuntimeBroker.exe and taskhostw.exe (formerly **taskhost.exe** and **taskhostex.exe**).

`(つ◔◡◔)つ ♥ Processes Hierarchy Diagram: ♥`



Resources:

- https://www.threathunting.se/tag/windows-process/
- https://www.sans.org/security-resources/posters/hunt-evil/165/download
- https://docs.microsoft.com/en-us/sysinternals/resources/windows-internals
- https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode

Hope you enjoyed reading my notes for this room! 🤩