LAB 4: OBSESSION

▼ 🧾 General Info

- **Challenge Name:** **Obsession: Shadows of the Deleted**
- **Memory Dump File Name:** **MemLabs-Lab4.7z**
- **Source / URL:** *https://github.com/stuxnet999/MemLabs/tree/master/Lab 4*
- **Date Started:** Wednesday **July 30**, 2025
- **Date Completed: N/A**
- **System Profile (from `windows.info`):** Windows 7 SP1 ×64
- **Volatility Version Used:** Volatility 3 Framework - 2.26.0
- **Duration: N/A**
- **Challenge Description:**

  “My system was recently compromised. The Hacker stole a lot of information but he also deleted a very important file of mine. I have no idea on how to recover it. The only evidence we have, at this point of time is this memory dump. Please help me.”

  - **Note**: This challenge is composed of only **1** flag.
  - The flag format for this lab is: `inctf{s0me_l33t_Str1ng}`

▼ 🔍 Objective

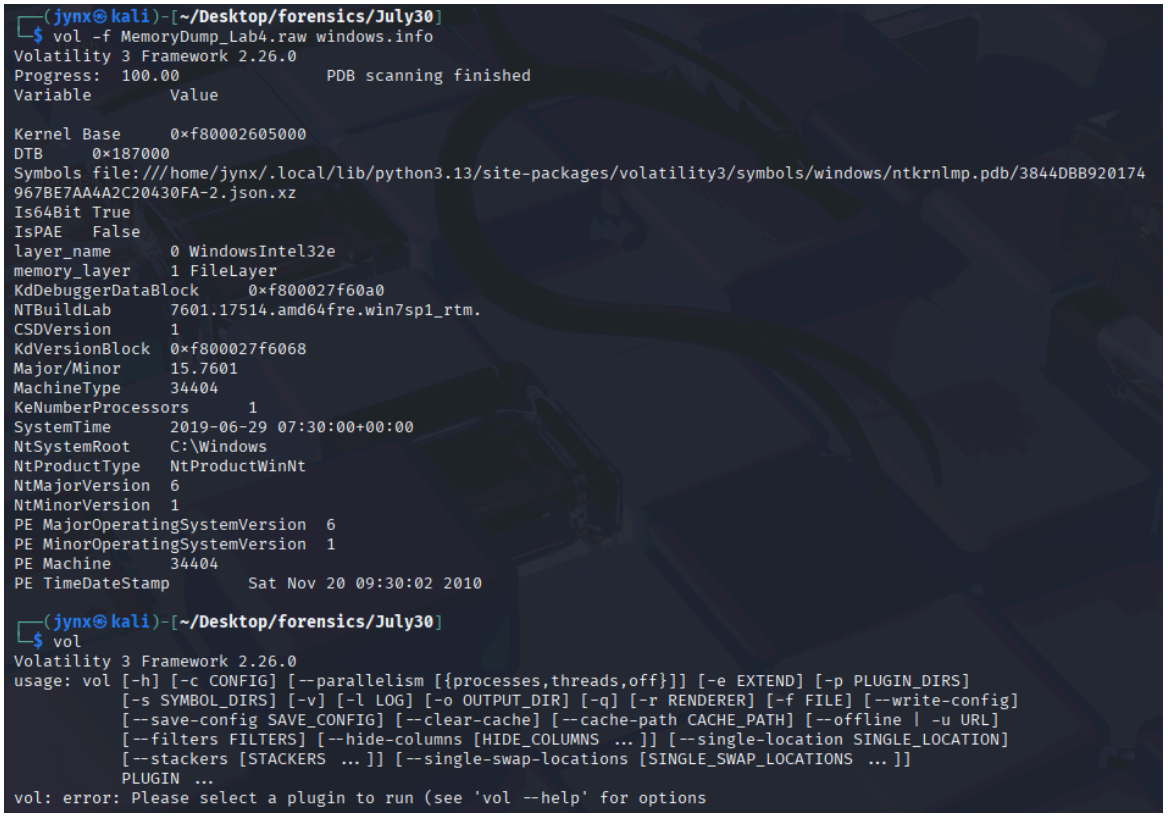
> Clearly describe the goal of the challenge.
>
> *Example: “Find a deleted file named Important.txt that contains the flag.”*

Lab 4 “Obsession” tests the analyst’s willingness to go beyond default outputs. It’s not about running a plugin — it’s about interpreting memory like a puzzle, obsessing over fragments of deleted data, and pushing every forensic tool to the limit. If you're not ready to dig deep, you won’t find the truth.

▼ 🛠️ Tools & Setup

Tool	Version/Details
Volatility	**Volatility 3** [2.26.0]
OS	**Kali Linux**

Other Tools	`Strings`
	`grep`
	`exiftool`
	`stehide`



▼ 🧩 Phase 1: Image Info + Process Mapping

▼ 💾 1. `pslist`

→ The `pslist` plugin is one of the **most fundamental commands in memory forensics** using Volatility.

`pslist` enumerates **active processes** by scanning the **EPROCESS** structures in the memory dump. It's a way of peeking into the **process list the OS kept in memory when the snapshot was taken**.

Command

```
vol -f MemoryDump_Lab4.raw windows.pslist
```

▼ Findings

▼ 1. High COUNT of Threads on `svchost.exe` [PID - 864]

```
┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ vol -f MemoryDump_Lab4.raw windows.pslist
Volatility 3 Framework 2.26.0
Progress:  100.00               PDB scanning finished
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        File output

4       0       System  0×fa8000ca0040  79      509     N/A     False   2019-06-29 07:28:07.000000 UTC  N/A     Disabled
256     4       smss.exe        0×fa80014af950  3       32      N/A     False   2019-06-29 07:28:07.000000 UTC  N/A     Disabled
328     320     csrss.exe       0×fa8001c57b30  11      385     0       False   2019-06-29 07:28:14.000000 UTC  N/A     Disabled
376     368     csrss.exe       0×fa8000ca8960  7       200     1       False   2019-06-29 07:28:15.000000 UTC  N/A     Disabled
384     320     wininit.exe     0×fa8001c6f760  3       75      0       False   2019-06-29 07:28:15.000000 UTC  N/A     Disabled
412     368     winlogon.exe    0×fa8001c751f0  6       119     1       False   2019-06-29 07:28:15.000000 UTC  N/A     Disabled
472     384     services.exe    0×fa8001bc1b30  13      193     0       False   2019-06-29 07:28:17.000000 UTC  N/A     Disabled
480     384     lsass.exe       0×fa8001cb5940  8       582     0       False   2019-06-29 07:28:17.000000 UTC  N/A     Disabled
488     384     lsm.exe 0×fa8001cc1b30  12      187     0       False   2019-06-29 07:28:17.000000 UTC  N/A     Disabled
580     472     svchost.exe     0×fa8001d02b30  11      358     0       False   2019-06-29 07:28:21.000000 UTC  N/A     Disabled
640     472     VBoxService.ex  0×fa8001d30b30  14      137     0       False   2019-06-29 07:28:21.000000 UTC  N/A     Disabled
708     472     svchost.exe     0×fa8001d43a70  7       260     0       False   2019-06-29 07:28:22.000000 UTC  N/A     Disabled
804     472     svchost.exe     0×fa8001dacb30  19      393     0       False   2019-06-29 07:28:23.000000 UTC  N/A     Disabled
840     472     svchost.exe     0×fa8001db9b30  21      431     0       False   2019-06-29 07:28:24.000000 UTC  N/A     Disabled
864     472     svchost.exe     0×fa8001dc6850  37      917     0       False   2019-06-29 07:28:24.000000 UTC  N/A     Disabled
952     804     audiodg.exe     0×fa8001df1060  7       131     0       False   2019-06-29 07:28:26.000000 UTC  N/A     Disabled
220     472     svchost.exe     0×fa8001e1b890  16      323     0       False   2019-06-29 07:28:27.000000 UTC  N/A     Disabled
484     472     svchost.exe     0×fa8001e45630  18      376     0       False   2019-06-29 07:28:29.000000 UTC  N/A     Disabled
1132    472     spoolsv.exe     0×fa8001eaab30  15      286     0       False   2019-06-29 07:28:32.000000 UTC  N/A     Disabled
1176    472     svchost.exe     0×fa8001ed7b30  21      307     0       False   2019-06-29 07:28:33.000000 UTC  N/A     Disabled
1276    472     svchost.exe     0×fa8001f452e0  14      220     0       False   2019-06-29 07:28:34.000000 UTC  N/A     Disabled
1804    472     taskhost.exe    0×fa8001f81b30  10      161     1       False   2019-06-29 07:28:42.000000 UTC  N/A     Disabled
```

- A typical `svchost.exe` instance runs **10–150 threads**, depending on the services it hosts.
- Having **900+ threads** in a single instance is not only unusual but likely points to:
  - **Malware injection**
  - **Process hollowing**
  - **Thread spraying** (evasion tactic)
  - **Rootkit or persistence technique**
  - Possibly **a memory exhaustion DoS vector**

▼ 2. Extremely Low `dwm.exe` handles

```
┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ vol -f MemoryDump_Lab4.raw windows.pslist | grep dwm
1908ress840100.0dwm.exe 0×fa80020bbb30  5       77      1       False   2019-06-29 07:28:43.000000 UTC  N/A     Disabled
3000    840     dwm.exe 0×fa8000e62b30  5       76      2       False   2019-06-29 07:29:36.000000 UTC  N/A     Disabled
```

- Normally, `dwm.exe` [Desktop Windows Manager] spawns **once**, and holds **several hundred handles** and dozens of threads.
- We have **two instances**, **both with only 1 and 2 handles respectively**, and very low thread counts:
  - PID `1908`: 77 threads
  - PID `3000`: 76 threads

This is **highly atypical**, especially since they're spawned under the same parent (PID 840 `svchost.exe`) — even more suspicious because `svchost` is already a candidate for malware injection.

SUSPICIOUS because;

1. **TWO dwm.exe processes** (should only be one!)
2. **Handle counts too low** (76-77 vs expected 200-800+)
3. **Different sessions** (1 and 2)

This Suggests:

- **Process hollowing** (legitimate process replaced with malware)
- **Process impersonation** (malware pretending to be dwm.exe)
- **Possible rootkit activity**

This is **highly atypical**, especially since they're spawned under the same parent (PID 840 `svchost.exe`) — even more suspicious because `svchost` is already a candidate for malware injection.

▼ 2. ☠️ malfind

→ Volatility plugin that detects signs of code injection and malware in process memory by looking for suspicious memory regions with executable code.

What it finds:

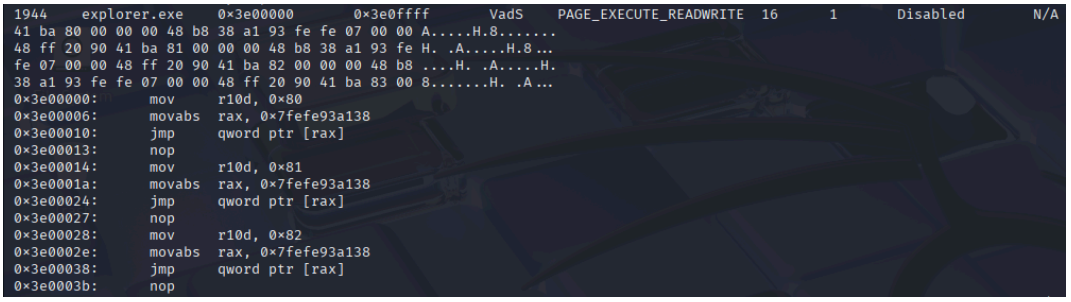
- Injected code in processes
- Shellcode and malware
- Executable memory regions that seem suspicious etc.

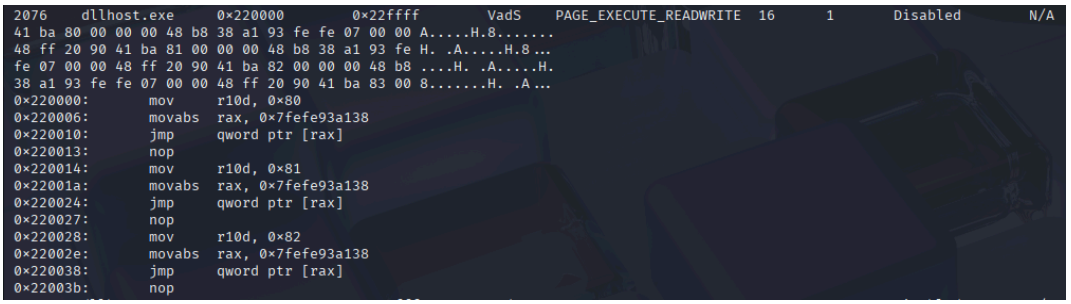
Command:

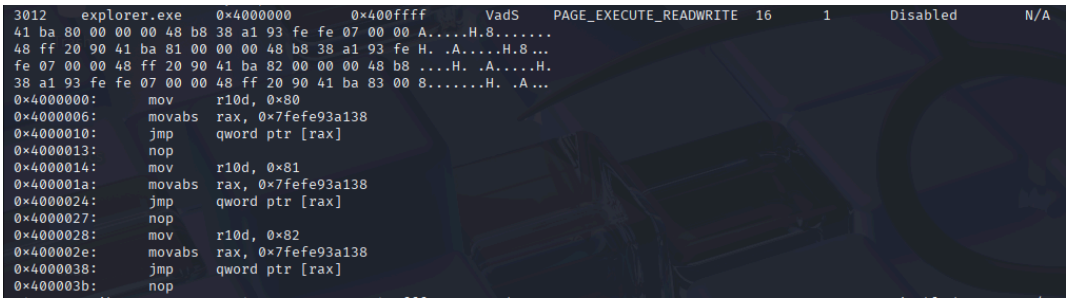
```
vol -f MemoryDump_Lab4.raw windows.malfind
```

▼ Findings:

TEXTBOOK case of **malware injection** and process hollowing







All three processes have the **EXACT SAME** malicious code:

assembly

```
mov r10d, 0x80/0x81/0x82    ← System call numbers
movabs rax, 0x7fefe93a138   ← SAME address in all processes
jmp qword ptr [rax]         ← Jump to malicious code
nop                         ← Padding
```

Classic Injection Indicators:

- **PAGE_EXECUTE_READWRITE** permissions (HUGE red flag!)
- **VadS** (Private memory allocation)
- **Same hex pattern:** `41 ba 80 00 00 00 48 b8 38 a1 93 fe fe 07 00 00`

Targeted Processes:

- `explorer.exe` (PID 1944, 3012) - Windows Shell

- `dllhost.exe` (PID 2076) - COM + surrogate

What the attacker probably did:

Step 1: **Process Hollowing**

- Injected malicious code into **legitimate processes**
- Replaced original code with **shellcode stubs**

Step 2: **System Call Hooking**

- `mov r10d, 0x80/0x81/0x82` → **System call numbers**
- `0x7fefe93a138` → **Shared malicious handler address**
- **Intercepts system calls** to hide activity

Step 3: **Persistence**

- Multiple processes → **redundancy**
- If one is killed, others continue

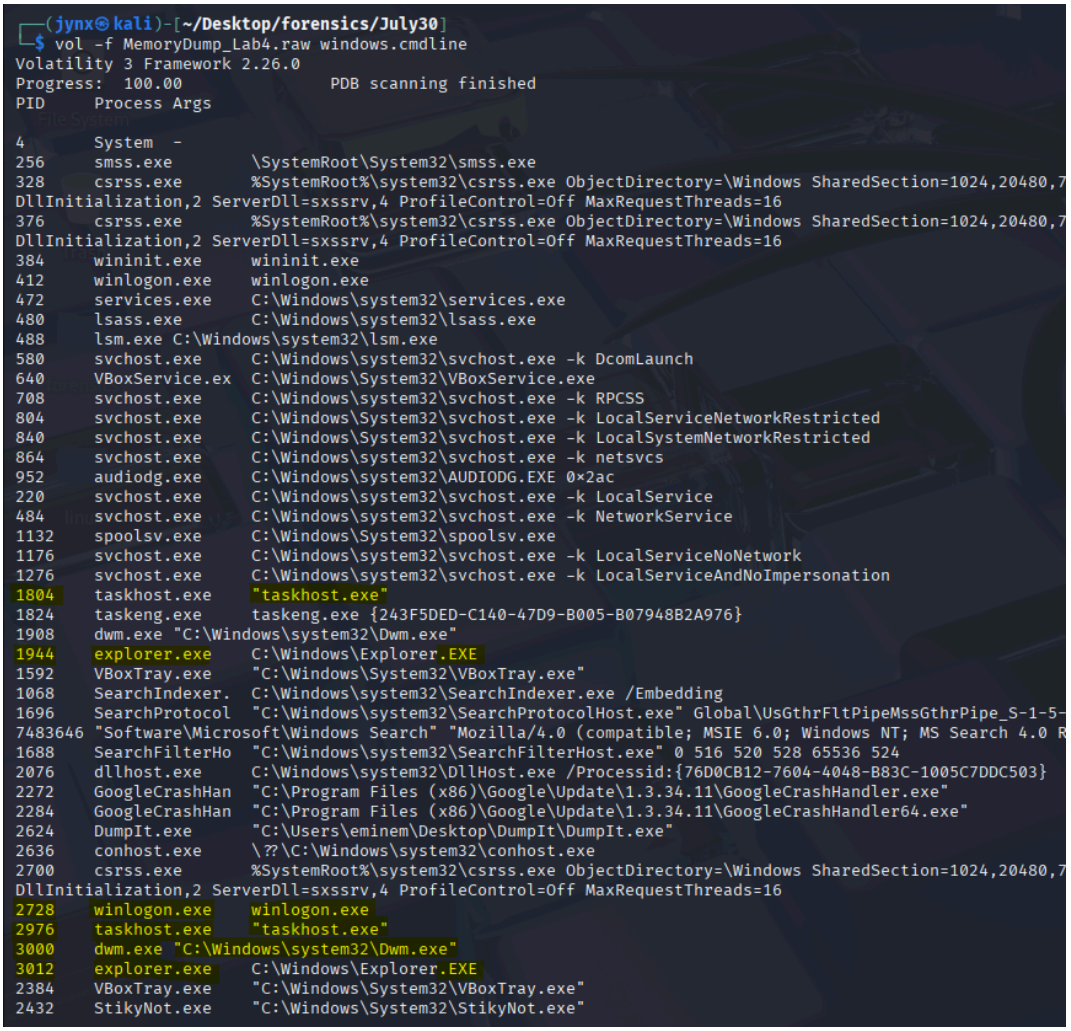
▼ 3. 🖨️ cmdline

Extracts and displays the command line arguments that were used to start each process, and exactly how each program was launched with what parameters and switches.

Command

```
vol -f MemoryDump_Lab4.raw windows.cmdline
```

▼ Findings




```
2976    taskhost.exe    "taskhost.exe"
1804    taskhost.exe    "taskhost.exe"

# vs normal taskhost should look like:
taskhost.exe {243F5DED-C140-47D9-B005-B07948B2A976}
```

→ `taskhost.exe` **(PID 1804 and 2976) - Missing the** `k` **parameter pattern.**

- **Real taskhost.exe** manages scheduled tasks with **specific task GUIDs**
- **This one** just has `"taskhost.exe"` in quotes - **no actual task management!**
- It's **masquerading** as a legitimate Windows service

Suspicions:

1. **Two taskhost.exe processes** (unusual)
2. **No task GUIDs or parameters** (fake)
3. **Simple quoted executable name** (lazy impersonation)
4. **Different sessions** (1 and 2) but same behavior

This suggests:

- **Process impersonation** - malware pretending to be taskhost.exe
- **Lazy coding** - attacker didn't bother with realistic parameters
- **Multi-session persistence** - running in both user sessions

```
1944    explorer.exe    C:\Windows\Explorer.EXE
2076    dllhost.exe     /Processid:{76D0CB12-7604-4048-B83C-10
05C7DDC503}
2728    winlogon.exe    winlogon.exe
2976    taskhost.exe    "taskhost.exe"
3000    dwm.exe         "C:\Windows\system32\Dwm.exe"
3012    explorer.exe    C:\Windows\Explorer.EXE
```

The "`.EXE`" vs "`.exe`" Case Anomaly:

Windows is case-insensitive BUT;

- **Real processes** typically show **consistent casing**
- **Registry entries** and **system calls** preserve original case
- **Malware** often has **different case patterns**

```
# Legitimate process creation preserves system casing
CreateProcess("C:\Windows\explorer.exe", ...)

# Malware might use different casing
CreateProcess("C:\Windows\Explorer.EXE", ...)  ← Hardcoded by attacker
```

▼ INSIGHTs:

The attacker got **lazy** or used an **automated tool** that doesn't properly mimic Windows process startup behavior. **Real Windows processes** have **rich command-line parameters**, while **malware processes** just have **basic executable paths**.

- **Real Windows processes** have **complex command-line arguments**
- **Malware** just uses **simple quoted paths** or **bare executable names**
- **No effort** to mimic legitimate process behavior

The attacker created **fake system processes** but didn't research how they **actually start**. They just copied the **executable names** without the **proper command-line arguments**.

This is why the `-k` parameter is like a "digital fingerprint" - it separates **real Windows processes** from **malware impersonation.**

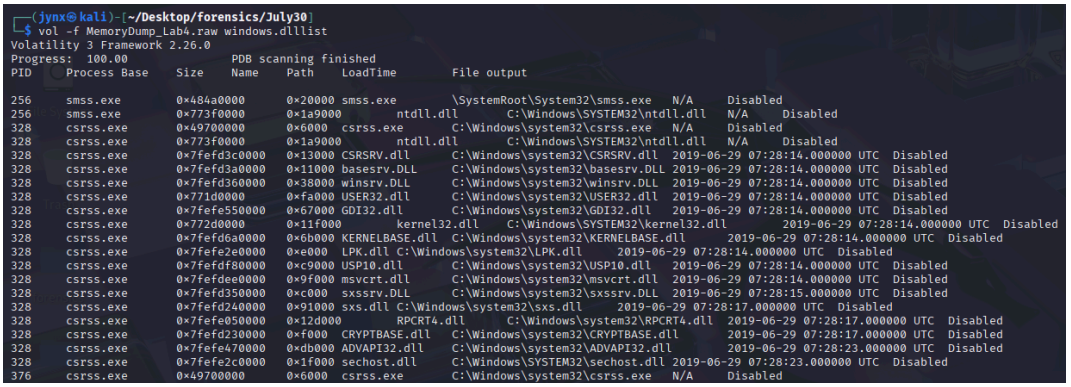
It's a **fundamental Windows internals concept** that many attackers overlook.

▼ 4. 🧬 dlllist

→ Shows all the Dynamic Link Libraries (DLLs) loaded into each process's memory space - reveals what code libraries and functions each program is using.

```
vol -f MemoryDump_Lab4.raw windows.dlllist
```

▼ Findings



```
┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ vol -f MemoryDump_Lab4.raw windows.dlllist
Volatility 3 Framework 2.26.0
Progress:  100.00               PDB scanning finished
PID     Process Base    Size    Name    Path    LoadTime        File output

256     smss.exe        0x484a0000      0x20000 smss.exe        \SystemRoot\System32\smss.exe   N/A     Disabled
256     smss.exe        0x773f0000      0x1a9000        ntdll.dll       C:\Windows\SYSTEM32\ntdll.dll   N/A     Disabled
328     csrss.exe       0x49700000      0x6000  csrss.exe       C:\Windows\system32\csrss.exe   N/A     Disabled
328     csrss.exe       0x773f0000      0x1a9000        ntdll.dll       C:\Windows\SYSTEM32\ntdll.dll   N/A     Disabled
328     csrss.exe       0x7fefd3c0000   0x13000 CSRSRV.dll      C:\Windows\system32\CSRSRV.dll  2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefd3a0000   0x11000 basesrv.DLL     C:\Windows\system32\basesrv.DLL 2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefd360000   0x38000 winsrv.DLL      C:\Windows\system32\winsrv.DLL  2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x771d0000      0xfa000 USER32.dll      C:\Windows\system32\USER32.dll  2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefe550000   0x67000 GDI32.dll       C:\Windows\system32\GDI32.dll   2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x772d0000      0x11f000        kernel32.dll    C:\Windows\SYSTEM32\kernel32.dll        2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefd6a0000   0x6b000 KERNELBASE.dll  C:\Windows\system32\KERNELBASE.dll      2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefe2e0000   0xe000  LPK.dll C:\Windows\system32\LPK.dll     2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefdf80000   0xc9000 USP10.dll       C:\Windows\system32\USP10.dll   2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefdee0000   0x9f000 msvcrt.dll      C:\Windows\system32\msvcrt.dll  2019-06-29 07:28:14.000000 UTC  Disabled
328     csrss.exe       0x7fefd350000   0xc000  sxssrv.DLL      C:\Windows\system32\sxssrv.DLL  2019-06-29 07:28:15.000000 UTC  Disabled
328     csrss.exe       0x7fefd240000   0x91000 sxs.dll C:\Windows\system32\sxs.dll      2019-06-29 07:28:17.000000 UTC  Disabled
328     csrss.exe       0x7fefe050000   0x12d000        RPCRT4.dll      C:\Windows\system32\RPCRT4.dll  2019-06-29 07:28:17.000000 UTC  Disabled
328     csrss.exe       0x7fefd230000   0xf000  CRYPTBASE.dll   C:\Windows\system32\CRYPTBASE.dll       2019-06-29 07:28:17.000000 UTC  Disabled
328     csrss.exe       0x7fefe470000   0xdb000 ADVAPI32.dll    C:\Windows\system32\ADVAPI32.dll        2019-06-29 07:28:23.000000 UTC  Disabled
328     csrss.exe       0x7fefe2c0000   0x1f000 sechost.dll     C:\Windows\SYSTEM32\sechost.dll 2019-06-29 07:28:23.000000 UTC  Disabled
376     csrss.exe       0x49700000      0x6000  csrss.exe       C:\Windows\system32\csrss.exe   N/A     Disabled
```

Duplicate Explorer Process

- One is **3012**, and the other explorer.exe was PID **1944**
- **Having multiple explorer.exe processes is highly unusual** and often indicates:
  - Process replacement/hollowing
  - Malware masquerading as explorer.exe
  - System instability or compromise

Suspicious Timing Pattern

- All DLLs load at exactly `2019-06-29 07:29:36.000000 UTC` initially
- This is **8 seconds after the dwm.exe** process we analyzed earlier
- Uniform timestamps across all initial DLLs suggest **batch loading** rather than organic process startup

Missing VirtualBox Components

- **Notable absence**: No `VBoxMRXNP.dll` like in the first explorer.exe (PID 1944)
- This suggests different execution contexts or potential masquerading

Extensive Network/HTTP Components
Additional network components not seen in first explorer:

- `WINHTTP.dll` - HTTP client functionality
- `webio.dll` - Web I/O operations
- Various wireless and network APIs loaded

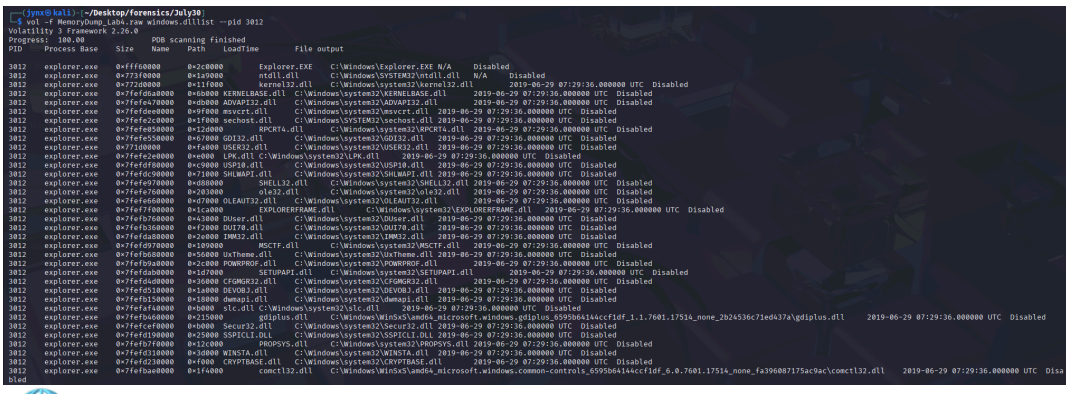
Delayed Fax Services Loading

- Fax-related DLLs (`fxsst.dll`, `FXSAPI.dll`, `FXSRESM.DLL`) load at `07:29:47`
- **11-second delay** after initial loading - unusual for standard explorer

HIGHLY SUSPICIOUS: Multiple explorer.exe processes running simultaneously is a **major red flag**. This pattern commonly indicates:

1. **Process Hollowing**: Malware replacing legitimate explorer.exe
2. **Persistence Mechanism**: Malware maintaining presence via fake explorer

3. **System Compromise**: Unauthorized process execution



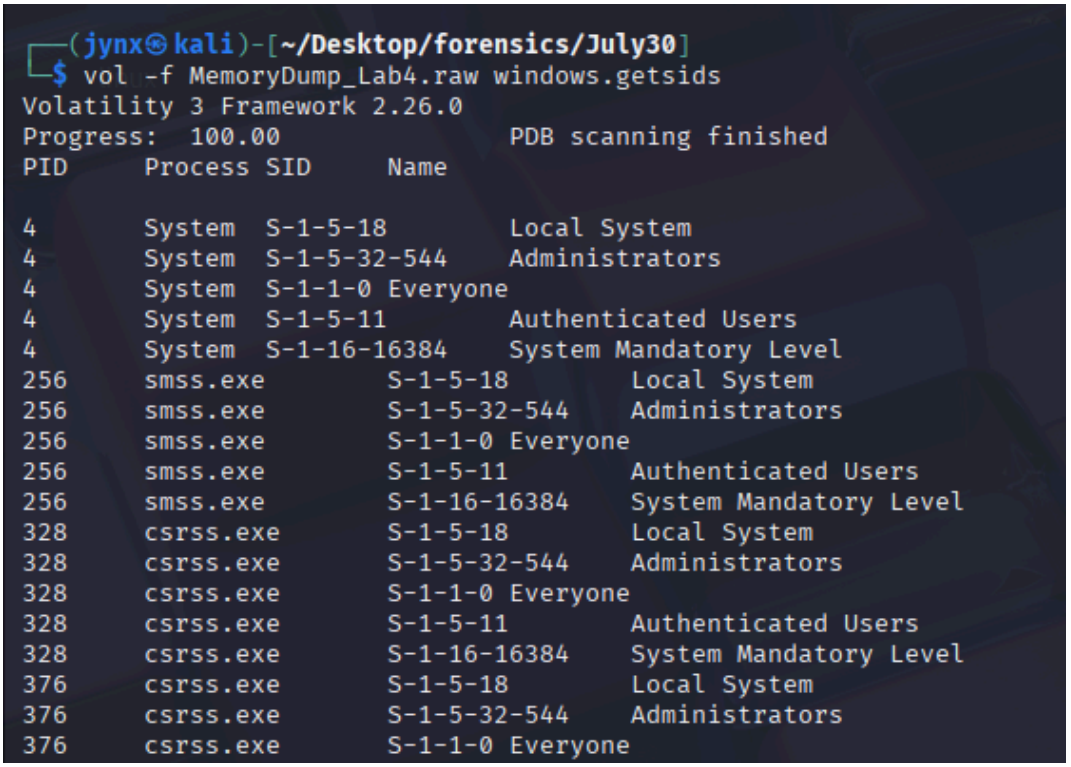
▼ 5. 🌐 getsids

→ Its **a plugin used in the Volatility framework for memory analysis, which prints the SIDs (Security Identifiers) owning each process.** SIDs are unique values used to identify a trustee, such as a user or group, and are crucial for understanding the privileges and identities of processes in a system's memory.

Command:

```
vol -f MemoryDump_Lab4.raw windows.getsids
```

▼ Findings:



User Account Names

- **"eminem" (RID 1000)** - This appears to be a non-standard administrative username, which is unusual for a typical corporate environment
- **"SlimShady" (RID 1001)** - This is clearly suspicious as it references Eminem's alter ego and suggests unauthorized account creation

Privilege Escalation Indicators

- The "eminem" user has **Administrator privileges** (S-1-5-32-544), which may indicate privilege escalation
- Both users are running multiple processes with interactive logon sessions

Unusual Process Execution

- **DumpIt.exe (PID 2624)** running under the "eminem" account with **High Mandatory Level** (S-1-16-12288) - This is a memory dumping tool, which could indicate:
  - Legitimate forensic analysis
  - Malicious memory extraction for credential harvesting

- Anti-forensics attempts

Multiple Active Sessions

- Two separate user sessions are active simultaneously (eminem and SlimShady)
- This could indicate:
  - Session hijacking
  - Unauthorized concurrent access
  - Lateral movement between accounts

Timing Concerns

- The presence of GoogleCrashHandler processes suggests recent system activity
- StikyNot.exe (Sticky Notes) running under SlimShady account indicates active user interaction

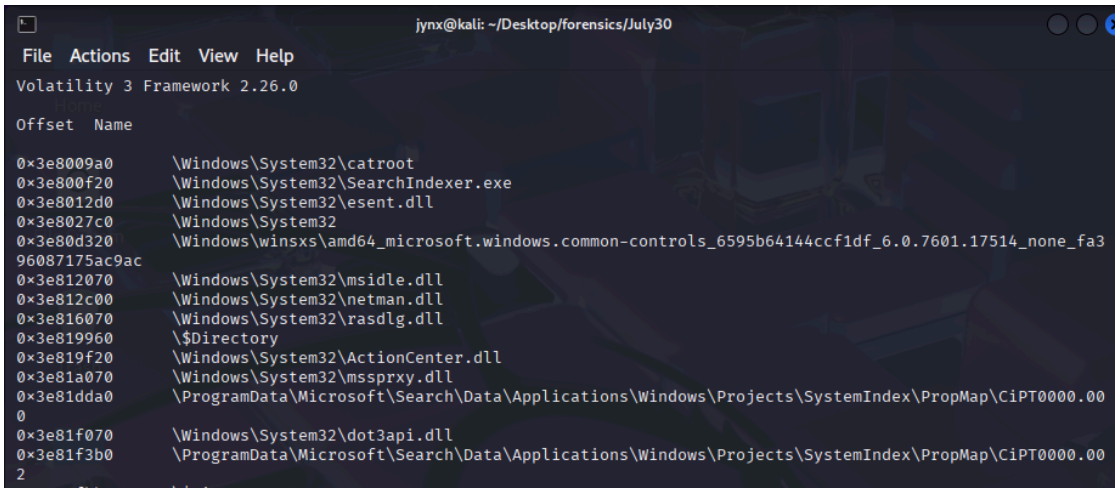
---

▼ 🧩 Phase 2: FLAG FINDING

Command:

```
vol -f MemoryDump_Lab4.raw windows.filescan > filescan.txt
```

▼ Findings:



The contents of the file `filescan.txt`

```
┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ less filescan.txt | grep png
0×3e8d19e0      \Users\eminem\Desktop\Screenshot1.png

┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ less filescan.txt | grep jpg
0×3ebc0690      \Users\eminem\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.jpg
0×3fd1bd50      \Users\SlimShady\AppData\Roaming\Microsoft\Windows\Themes\TranscodedWallpaper.jpg

┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ less filescan.txt | grep jpeg
0×3e8ad250      \Users\eminem\Desktop\galf.jpeg
```

Found a couple of images since the challenge mentions on how a file was deleted and maybe we can find clues on it or plain decoys depending on further look here.

→ I failed to extract the contents of the actual file. Ill see what I can find further on from here.

```
0×3e89b070      \Users\eminem\AppData\Local\GDIPFONTCACHEV1.DAT
0×3e8a4690      \Program Files\Windows Photo Viewer\ImagingEngine.dll
0×3e8a5880      \Windows\Fonts\segoeuiz.ttf
0×3e8a5cd0      \Windows\System32\UIAnimation.dll
0×3e8a6d00      \Windows\Fonts\segoeui.ttf
0×3e8a7a70      \Windows\winsxs\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.7601.17514_none_fa3
96087175ac9ac
0×3e8a85b0      \Users\eminem\AppData\Roaming\Microsoft\Windows\Recent\Flag not here.lnk
0×3e8a9370      \$Directory
0×3e8a9610      \Users\eminem\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\1b4dd67f29cb1962.au
tomaticDestinations-ms
0×3e8a9a70      \$Directory
```

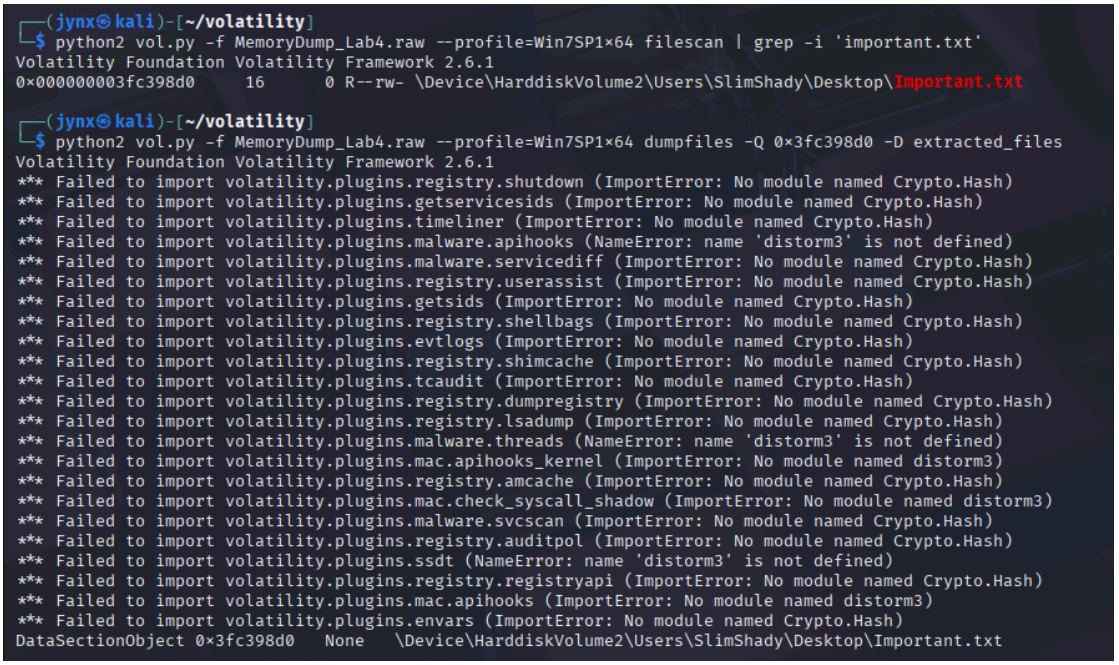
sure lol,

```
┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ grep imp filescan.txt

┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ grep "imp" filescan.txt

┌──(jynx㉿kali)-[~/Desktop/forensics/July30]
└─$ grep -i "imp" filescan.txt
0×3ea14f20      \Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Consolidator
0×3ea79750      \Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\KernelCeipTask
0×3ed13390      \Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\UsbCeip
0×3edee4f0      \Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Uploader
0×3f939720      \Users\SlimShady\AppData\Roaming\Microsoft\Windows\Recent\Important.lnk
0×3fc398d0      \Users\SlimShady\Desktop\Important.txt
```

Lets see if I can retrieve the contents of this Important.txt file or not- I hope I do lol its been hours now 🥲



Unfortunately, I cannot retrieve nor extract the Important.txt file- my challenge ends here for that matter why? Well try for yourself or watch the creator's solution to know, I'd recommend tying it first but sure you wise enough to make your own decisions.

▼ Creator's Solution Video:

https://www.youtube.com/watch?v=R4ogPvN63Xg&list=PLwHfQPh43gyqJmZxGP2Pbys1g9d0_DHiy