Investigating Memory Image

Memory Forensics: Volatility Analysis

[Sunday, July 6 2025]

🧠 Context

We examined a memory image (`Challenge.raw`) using Volatility 2 to identify suspicious processes, DLL injections, or unusual command history.

🧾 Memory Image Metadata

- **Filename:** Challenge.raw
- **Size:** ~1.6 GB
- **Source:** Volatility Foundation — CTF 2019
- **Download Link for *Challenge.raw* [zip file]:**

  https://drive.usercontent.google.com/download?id=1bER4wmHP_LAMgdB52LGkb8×2Mf8hG3V6&export=download&authuser=0
- **Use Case:** Windows 7 ×64 Live Memory Image Dump for Digital Forensics

The link to download the Memory image dump:

🔧 Environment

- Tool: Volatility 2.6.1
- OS: Kali Linux (VM)
- Image Profile: Win7SP1×64
- File: `Challenge.raw`

🔍 Plugins Used & Findings

▼ 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:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:

→ ***svchost.exe*** - its an interesting file to say the lease here's why:

- It seems normal at glance, as instantiated instead by services.exe process standard 5-15 svchost.exe simultaneous running processes. Pretty standard and normal. Here's where things get slightly grey; Most svchost.exe processes typically run **continuously** throughout your Windows session. But this ones here, 9 of them started at **14:40:11** and finished all their tasks by **14:40:13**, 2 seconds? That's extremely suspicious. There's a lot the can be unpacked here if we get more evidences and explore more tangents to this.

▼ Learnings [*unaware*]:

1. csrss.exe: its a system generate file, responsible for console windows and user interfaces
2. **wininit.exe**: Starts up important Windows services when your computer boots
3. **winlogon.exe**: Handles your Windows login screen and manages logging in/out and locking your computer
4. You can use **grep -E "(abc|acb|cba)"** to search for either of those one values.
5. **explorer.exe**: The Windows desktop shell that gives you your taskbar, Start menu, file browser, and desktop icons

6. **conhost.exe**: Hosts console (command prompt) windows in Windows - it's like the container that runs your cmd.exe, PowerShell, and other command-line programs.

▼ Insights:

1. ***'grep'*** command was extremely useful for searching of particular PID [Process IDs] or PPID [Parent Process IDs] or even time stamps. Learn it and having a proficiency in tools that help with text filtering and text manipulation is a valid and very foundational in nature [Must know in my opinion].

▼ 2. 🌳 pstree

→ Shows the hierarchical relationship between running processes, displaying which processes spawned which other processes in a tree-like format.

Command:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:

Nothing unique as such compared to ***pslist- findings*** except 2 browser anomalies [which can be or not be critical, which would be unveiled on further investigation and analysis]:

1. Firefox anomaly:

- firefox.exe(2080) has PPID=3060, but no process with PID 3060 exists in the dump
- This might suggest the parent process terminated or was hidden - potential process injection/hollowing

2. **Google anomaly**:

- GoogleCrashHan processes (1292, 924) have PPID=1928 - no PID 1928 visible
- GoogleUpdate.exe(2256) has PPID=2396 - no PID 2396 visible
- Could indicate terminated parent processes or malicious activity

→ I am not so confident in declaring them CRITICAL yet, is because this is a memory dump essentially a screenshot of the memory at one point, and it is fairly plausible to assume that some processes or parent processes might not have been registered in that moment itself.

▼ Learnings [*unaware*]:

1. dwm.exe: Desktop Window Manager - handles Windows' visual effects like transparency, window animations, and compositing the desktop display.
2. taskhost.exe: Task Scheduler host that runs scheduled tasks, commonly spawned by services.exe or svchost.exe.
3. spoolsv.exe: Print Spooler service that manages printing, typically spawned by services.exe.

▼ Insights:

→ N/A for the session.

▼ 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:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:

1. I did not get any suspicious or tacky behavior that would raise alarms or eyebrows, [I tried my humor], I found a tad bit interesting thing here (maybe just to me):

> WinRAR.exe PID: 3716
> Command line : "C:\Program Files\WinRAR\WinRAR.exe" "C:\Users\Jaffa\Desktop\pr0t3ct3d\flag.rar"

Essentially a user namely- "***Jaffa***", is trying to extract a file named flag.rar from a folder 'pr0t3ct3d', using ***WinRAR.exe***.

Now, the thing here is it seems it was a part of some CTF challenge and the memory dump was captured while the challenge was going on.

2. Follow up [Jaffa]:
   In seconds I also found out that Jaffa was also the user that created and took this Memory dump image [snapshot].

   > C:\Users\Jaffa\Desktop\DumpIt.exe

   Making it credible that it indeed was some kind of CTF challenge snapshot, bu a legitimate user Jaffa.

▼ Learnings [*unaware*]:

1. **sppsvc.exe**: Software Protection Platform Service - handles Windows license activation and validation to make sure your copy of Windows is genuine.RetryX
   → it's normal for
   sppsvc.exe to have **svchost.exe** as its parent, services can spawn it directly or through svchost.exe depending on the system configuration.
2. **-k RPCSS**:

   ```
   "C:\Windows\system32\svchost.exe -k RPCSS"
   ```

   → The
   `-k` flag tells svchost.exe to load a specific group of related services, and "RPCSS" is the service group name that contains RPC (Remote Procedure Call) related services. [standard behavior]

▼ Insights:

→ Understanding Linux isn't gonna make the cut, as somebody who aspires to create genuine impact of their understanding and depth in the world of Cybersecurity and networks, I also need a solid muscle memory of various OS such as Windows, MacOS, Android etc., thoroughly understand their

system processes how do they work what are their anomalies and so on and so forth, you get the idea.

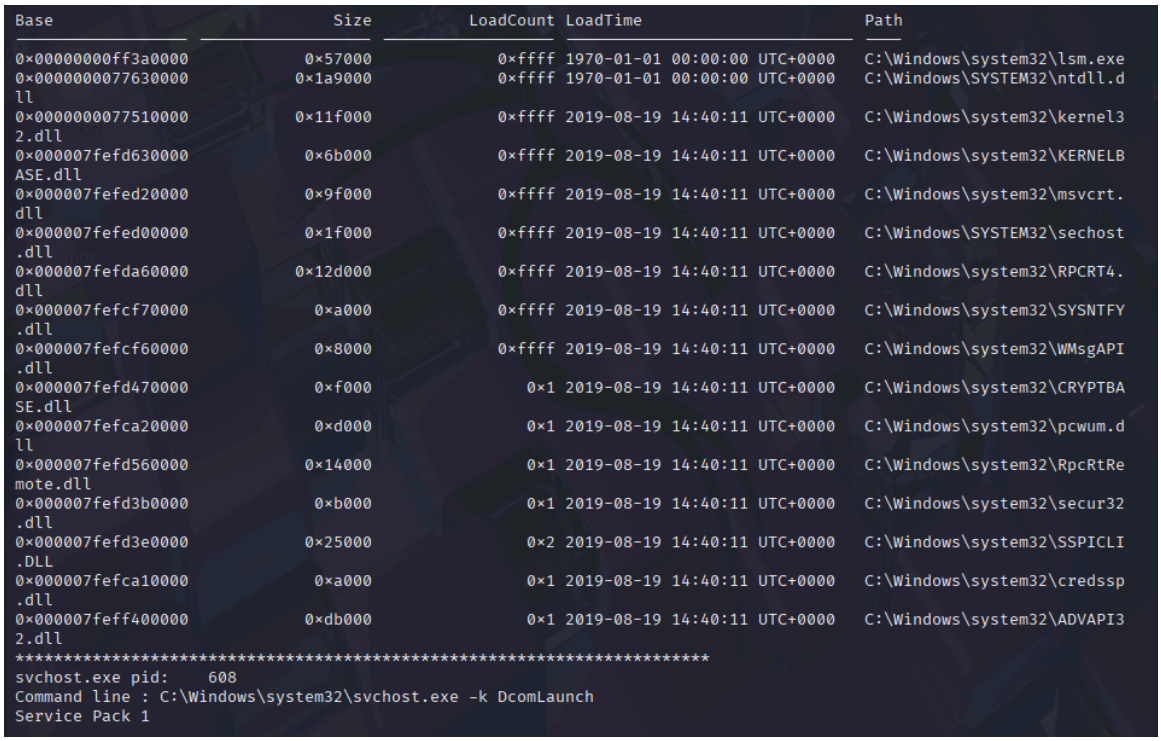
▼ 4. 🧬 dlllist

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

Command:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:



```
Base                  Size          LoadCount LoadTime                   Path
0×00000000ff3a0000    0×57000           0×ffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\system32\lsm.exe
0×0000000077630000    0×1a9000          0×ffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\SYSTEM32\ntdll.d
ll
0×0000000077510000    0×11f000          0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\kernel3
2.dll
0×000007fefd630000    0×6b000           0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\KERNELB
ASE.dll
0×000007fefed20000    0×9f000           0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\msvcrt.
dll
0×000007fefed00000    0×1f000           0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\SYSTEM32\sechost
.dll
0×000007fefda60000    0×12d000          0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\RPCRT4.
dll
0×000007fefcf70000    0×a000            0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\SYSNTFY
.dll
0×000007fefcf60000    0×8000            0×ffff 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\WMsgAPI
.dll
0×000007fefd470000    0×f000               0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\CRYPTBA
SE.dll
0×000007fefca20000    0×d000               0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\pcwum.d
ll
0×000007fefd560000    0×14000              0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\RpcRtRe
mote.dll
0×000007fefd3b0000    0×b000               0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\secur32
.dll
0×000007fefd3e0000    0×25000              0×2 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\SSPICLI
.DLL
0×000007fefca10000    0×a000               0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\credssp
.dll
0×000007feff400000    0×db000              0×1 2019-08-19 14:40:11 UTC+0000   C:\Windows\system32\ADVAPI3
2.dll
**************************************************************************
svchost.exe pid:    608
Command line : C:\Windows\system32\svchost.exe -k DcomLaunch
Service Pack 1
```

1. **Epoch timestamp anomaly**: The first two entries show `1970-01-01 00:00:00 UTC+0000` timestamps.
2. **Load count of 0xffff,** Multiple DLLs show this maximum value, which is unusual and could indicate; Memory corruption or Anti-forensics techniques.

```
Base                 Size         LoadCount LoadTime                  Path
0x00000000ff120000   0x74000          0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\system32\taskeng
.exe
0x0000000077630000   0x1a9000         0xffff 1970-01-01 00:00:00 UTC+0000   C:\Windows\SYSTEM32\ntdll.d
ll
0x0000000077510000   0x11f000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\kernel3
2.dll
0x000007fefd630000   0x6b000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\KERNELB
ASE.dll
0x0000000077410000   0xfa000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\USER32.
dll
0x000007fefec90000   0x67000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\GDI32.d
ll
0x000007fefdb90000    0xe000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\LPK.dll
0x000007fefee10000    0xc9000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\USP10.d
ll
0x000007fefed20000    0x9f000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\msvcrt.
dll
0x000007feff690000   0x203000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\ole32.d
ll
0x000007fefda60000   0x12d000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\RPCRT4.
dll
0x000007fefea00000    0xd7000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\OLEAUT3
2.dll
0x000007fefa5d0000    0xa000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\ktmw32.
dll
0x000007fefd040000    0x6d000         0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\wevtapi
.dll
0x000007feff660000    0x2e000             0x2 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\IMM32.D
LL
0x000007fefd950000   0x109000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\MSCTF.d
ll
0x000007fefd470000    0xf000              0x2 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\CRYPTBA
SE.dll
0x000007fefed00000    0x1f000            0x12 2019-08-19 14:40:18 UTC+0000   C:\Windows\SYSTEM32\sechost
.dll
0x000007feff400000    0xdb000             0x4 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\ADVAPI3
2.dll
0x000007fefce10000    0x17000             0x2 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\CRYPTSP
.dll
0x000007fefcb10000    0x47000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\rsaenh.
dll
0x000007fefec10000    0x71000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\SHLWAPI
.dll
0x000007fefd3e0000    0x25000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\SspiCli
.dll
0x000007fefd560000    0x14000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\RpcRtRe
mote.dll
0x000007fefdbb0000    0x99000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\CLBCatQ
.DLL
0x000007fef86c0000     0x9000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\tschann
el.dll
0x000007fefb690000    0x35000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\XmlLite
```

```
0x000007fef86c0000     0x9000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\tschann
el.dll
0x000007fefb690000    0x35000             0x1 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\XmlLite
.dll
0x000007fefd410000    0x57000          0xffff 2019-08-19 14:40:18 UTC+0000   C:\Windows\system32\apphelp
.dll
**********************************************************************
explorer.exe pid:   1944
Command line : C:\Windows\Explorer.EXE
Service Pack 1
```

Looking at this process dump from `taskeng.exe` :

1. Unusual Load Timestamps: `1970-01-01 00:00:00 UTC+0000` [same as that in svchost.exe]
2. Inconsistent Load Patterns**:** Some modules have normal load counts (0x1, 0x2, 0x4, 0x12) with proper 2019 timestamps, Others show `0xffff` (65535) load count, which is unusual. This mixed pattern suggests potential process manipulation or injection.

▼ Learnings [*unaware*]:

▼ ntdll.dll

ntdll.dll (NT Dynamic Link Library) is one of the most critical system files in Windows:

Purpose & Function:

- **Lowest-level user-mode interface** to the Windows kernel
- **Bridge between user-mode applications** and the Windows kernel (ntoskrnl.exe)
- Contains the **Native API** (also called NT API or Zw/Nt functions)
- **Every Windows process** must load ntdll.dll to function.

Critical Nature:

- **Cannot be unloaded** once loaded (hence why seeing timestamp issues is concerning)
- **Common target for malware** due to its privileged position
- **Present in every process** - if compromised, entire system is at risk
- **Rootkits often hook** ntdll.dll functions to hide their presence

▼ lsm.exe

lsm.exe (Local Session Manager) is a Windows system process:

Purpose & Function:

- Manages user sessions on the local machine
- Handles session state changes (logon, logoff, lock, unlock)
- Coordinates with other session management components

Normal Behavior:

- **Runs as a separate process** (not as a DLL within another process)
- **Typically runs as SYSTEM** account
- **Should have normal file timestamps** matching system installation

- **Low resource usage** under normal circumstances

Why It's Suspicious Here:

- **Appears as a loaded DLL** instead of running as its own process
- **Impossible timestamp** (1970-01-01) suggests manipulation or corruption
- **Could indicate process hollowing** or DLL injection techniques

▼ What is 0xffff?

0xffff is a hexadecimal value that equals 65,535 in decimal:

Technical Significance:

- Maximum value for 16-bit unsigned integer (2^16 - 1)
- All bits set to 1 in a 16-bit field (1111111111111111 in binary)
- Often used as a sentinel value meaning "maximum" or "invalid"

In This Context (Load Count):

- Load Count typically indicates how many times a DLL has been loaded/referenced
- Normal load counts are usually small numbers (1-10)
- 0xffff suggests:
  - Memory corruption - counter overflowed or was manipulated
  - Anti-forensics technique - malware setting artificial values
  - Rootkit activity - hiding true load counts
  - Analysis artifact - memory dump corruption or analysis tool error

▼ Process Hollowing

- The combination of epoch timestamps and maximum load counts (`0xffff`) is consistent with process hollowing techniques
- Process hollowing involves creating a legitimate process in suspended state, then replacing its memory content with malicious code

▼ Insights:

→ ‘dlllist’ output spanned over multiple lines, it was extremely length and tedious task at a glance, but often numbers can be overwhelming, take a breathe and and then try to understand something messy or tedious and it wouldn't be as hard as it seemed at first glance, was still very hard for me- but that’s just me again and not necessarily you.

▼ 5. 🌐 netscan

→ Scans memory for network artifacts - shows network connections, listening ports, and network activity including closed/terminated connections that might not appear in normal netstat output.

Command:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:

Foreign IP Address 117.18.237.29 - This IP appears in multiple connections and stands out as potentially suspicious (ports 80, 49178, 49179, 49181, 49182). The frequency of connections to this single external IP is unusual

▼ Learnings [*unaware*]:

1. **Localhost Tunneling Activity**

```
127.0.0.1:49171 ↔ 127.0.0.1:49170 (PID 2968)
127.0.0.1:49166 ↔ 127.0.0.1:49165 (PID 2080)
127.0.0.1:49167 ↔ 127.0.0.1:49168 (PID 3016)
127.0.0.1:49186 ↔ 127.0.0.1:49185 (PID 3316)
```

Multiple localhost-to-localhost connections suggest potential:

- Malware communication between processes
- Proxy/tunneling activity
- Data exfiltration preparation

High Volume of Connections

- Firefox PID 2080 has **30+ active connections**, which is excessive for normal browsing
- Many connections to Google/YouTube infrastructure mixed with suspicious IPs

▼ Insights:

- **Pattern Analysis**: Grouped connections by process and identified abnormal clustering
- **Process Behavior**: Multiple Firefox instances with different PIDs is atypical
- **Localhost Analysis**: Internal connections often indicate malware communication

▼ 6. ☠️ 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:

```
python2 vol.py -f ~/Desktop/forensics/vol3/Challenge.raw --profile=Win7SP1×
```

▼ Findings:

explorer.exe (PID 1944) at 0x4320000

1. **Shellcode in Explorer.exe (Address: 0x4320000)**

This is the most concerning finding:

```
41 ba 80 00 00 00    ; MOV R10D, 0x80
48 b8 38 a1 86 ff    ; MOV RAX, 0xff86a138
fe 07 00 00 48 ff 20 ; JMP [RAX]
```

Why this is malicious:

- **Repeating pattern**: The code repeats with incrementing values (0x80, 0x81, 0x82, 0x83).
- **Indirect jumps**: `JMP [RAX]` with calculated addresses is classic shellcode behavior.
- **Consistent target address**: `0xff86a138` suggests a function pointer table or API hooking.
- **Explorer.exe compromise**: This is a critical system process - injection here affects the entire desktop.

2. **Memory Protection Flags Are Suspicious**

   All flagged regions have `PAGE_EXECUTE_READWRITE` protection:

   - **Normal code**: Should be `PAGE_EXECUTE_READ` (no write access)
   - **RWX memory**: Allows writing and executing code - classic malware technique
   - **Private memory**: Indicates injected code, not loaded from legitimate files

3. **Hollow/Zeroed Memory Regions**

   - **Explorer.exe (0x3ce0000)**: Mostly null bytes with minimal data
   - **Chrome.exe (0x4830000)**: Almost entirely null bytes
   - **WmiPrvSE.exe (0x1bd0000)**: Contains suspicious data patterns

▼ Learnings [*unaware*]:

1. Assembly level code instructions.

   → It's important to realize why they are important and how they function if we are to a=work so closely an collaboratively with system hardware and memory address.

2. Understanding and reading between the lines for malfind utility's outputs, it was really bouncing off my head initially (had to take help from the web).

▼ Insights:

1. Location is Wrong

- Legitimate explorer.exe code should be in **C:\Windows\explorer.exe**
- This code is in **random memory space** (0x4320000)
- **Conclusion**: Someone injected code into explorer.exe

2. Code Pattern is Suspicious

- **Systematic API calls**: Normal programs don't have such repetitive patterns
- **Indirect jumps**: Classic malware technique to hide API calls
- **Counter-based enumeration**: Suggests automated API resolution

3. Memory Permissions

- **PAGE_EXECUTE_READWRITE**: Allows code modification at runtime
- **Normal processes**: Usually have separate read-only code and read-write data
- **Malware**: Needs writable+executable memory for dynamic code

What does it mean?

- **explorer.exe has been infected** with malware
- **Code injection attack** is executed
- **System is compromised**

▼ What I don't understand yet?

1. Depth and clarity in outputs of dlllist, netscan, malfind.
2. Assembly level basic instructions.
3. Poor pattern recognition, pretty expected looking at my humble backgrounds.
4. Proper system process and child process relation, understanding how computer processes and internal looping works.
5. classic textbook level malware analysis.

Timeline Reconstruction

→ **Image Capture**: Date: 2019-08-19 | **Time: 14:41:58 UTC**

Time (UTC)	Event	Plugin	Alarm
14:40:13	9 simultaneous svchost.exe processes begin and short lived- 2 seconds. [They are running till the system isn't shutdown.]	'pslist'	⚠️**Moderate**
14:40:06	No parent process ID [PPID] for firefox.exe.	'pstree'	ℹ️**Informational**
14:40:06	No parent process ID [PPID] for chrome.exe and google process crash witnessed.	'pstree'	ℹ️**Informational**
PID: 3716	'Jaffa' took the snapshot of memory dump, as a part of 2019-CTF challenge. He also extracted 'flag.rar' from a folder- 'pr0t3ct3d'.	'cmdline'	✅**Confirmed**
1970-01-01 00:00:00	**svchost.exe** - Unusual Load Timestamps, Inconsistent Load Patterns, Confirmed: combination of epoch timestamps and maximum load counts, Process Hollowing- **CONFIRM**.	'dlllist'	🚨 **Critical**
1970-01-01 00:00:00	explorer.exe - Unusual Load Timestamps, Inconsistent Load Patterns, Confirmed: combination of epoch timestamps and maximum load counts, Process Hollowing- **CONFIRM**.	'dlllist'	🚨 **Critical**

PID: 2968 PID: 2080 PID: 3016 PID: 3316	Local Host Tunneling Activity, Foreign IP Address **117.18.237.29** , Potential: Malware communication, proxy/tunneling activity, data exfiltration preparation.	‘netscan’	🚨 **Critical**
PID: 1944	Shellcode in Explorer.exe (Address: 0x4320000), **Repeating pattern, Consistent target address**: `0xff86a138` . explorer.exe is compromised- **CONFIRM**.	‘malfind’	🚨 **Critical**