Combined Method Structure of Stuxnet

Stuxnet is a sophisticated and highly specialized piece of malware that was first discovered in *June 2010*. It is notable for being the first known malware to target **industrial control systems (ICS)**, specifically those using Siemens software. The purpose of **Stuxnet** was to sabotage *Iran*'s nuclear enrichment capabilities. So this is a matter of both military cyber espionage and a government-sponsored cyber-attack.

Stuxnet is a multi-component malware, which includes several modules that work together. In the following years, this attack method was taken to even more advanced levels by many hacker groups.

Propagation Mechanisms

Stuxnet uses multiple zero-day exploits to propagate itself. The known exploits include:

- **CVE-2010–2568**: A vulnerability in Windows' shortcut (LNK) handling, allowing execution of malicious code when a user views a specially crafted shortcut.

- **CVE-2010–2729**: A vulnerability in the Windows print spooler service, allowing remote code execution.

- **MS08–067**: A vulnerability in the Windows Server service that allows remote code execution.

Rootkit*: **Stuxnet** includes rootkit functionalities that hide its presence on infected systems. This makes detection and analysis significantly more difficult.*

Command and Control (C2) Servers*: **Stuxnet** communicates with external servers to receive updates and instructions. These C2 servers allow the attackers to control the malware post-infection.*

Payload*: The most critical component of **Stuxnet** is its payload, designed to target specific Siemens PLC (Programmable Logic Controller) systems.*

- **Step 7 PLC Targeting**: Stuxnet specifically targets Siemens Step 7 software, which is used to program PLCs.
- **PLC Infection**: Once Stuxnet identifies the presence of Step 7 software, it injects its own code into the PLCs. This code is designed to subtly alter the operation of industrial machinery controlled by the PLCs.

This composite attack method was designed by considering many composite structures and exposing many systems to vulnerability. This is an example of a complex cyber military operation in many ways.

Stuxnet *also uses several advanced techniques to spread across networks.*

- **USB Propagation**: Stuxnet can spread via infected USB drives. When an infected USB is inserted into a computer, Stuxnet exploits the LNK vulnerability to execute automatically.
- **Network Propagation**: Using the Windows print spooler and MS08–067 vulnerabilities, Stuxnet can propagate over a network without user intervention.
- **Peer-to-Peer Updates**: Stuxnet can update itself via a peer-to-peer mechanism, ensuring that the latest version of the malware spreads through infected networks.

Upon gaining a foothold in a system, ***Stuxnet*** *performs several actions.*

- **System Fingerprinting**: Stuxnet first checks if the infected system matches its target environment, specifically looking for Siemens Step 7 software.
- **Privilege Escalation**: The malware exploits zero-day vulnerabilities to gain higher privileges on the system.
- **Installation of Rootkit**: Stuxnet installs a rootkit to hide its files and processes. This rootkit modifies system files and kernel drivers to prevent detection by antivirus software.

*The payload execution method is perhaps the part of **Stuxnet** that has garnered the most attention.*

- **PLC Code Injection**: Once the malware identifies the correct PLC configuration, it injects its own code into the PLC. This code is designed to manipulate the frequency converters in the centrifuges used for uranium enrichment.
- **Subtle Sabotage**: The injected code subtly alters the operation of the centrifuges, causing them to spin at incorrect speeds, thereby damaging them over time. The modifications are carefully timed to avoid immediate detection by operators.
- **Masking Operations**: Stuxnet ensures that the operators see normal readings on their monitoring equipment, even while the centrifuges are being damaged. This is achieved by intercepting and modifying the signals sent from the PLC to the monitoring systems.

Understanding exploited CVE's by Stuxnet

CVE-2010-2568

CVE-2010-2568 is a vulnerability in the Windows operating system's handling of shortcut files (**.LNK files**). This vulnerability was one of the key components leveraged by **Stuxnet** to propagate and execute malicious code without user interaction.

CVE-2010–2568 refers to a vulnerability in how Windows parses shortcut files. Normally, when a user views a folder containing a shortcut (a **.LNK file**), Windows automatically parses the file to display the appropriate icon. However, this vulnerability allows a specially crafted **.LNK file** to specify an arbitrary executable as the icon source, which Windows will load and execute without any further user action.

Shortcut files (**.LNK**) contain a variety of metadata, including the path to the file or executable they point to. In a typical **LNK** file, this path would point to a local or network file. **Stuxnet's LNK** files were crafted such that the path pointed to a malicious **DLL** file stored on the *USB drive*. Windows' **shell32.dll** library is responsible for handling shell operations, including displaying shortcut icons. The vulnerability in **shell32.dll** allowed the **LNK** file to specify a malicious icon location, leading to the automatic execution of the malicious **DLL**.

Stuxnet was engineered to generate specifically crafted shortcut files that contained malicious code. When a user opened a folder containing one of these shortcut files or merely viewed the folder in **Windows Explorer**, the malicious code would automatically execute without requiring any further action from the user. One of the critical aspects of **CVE-2010-2568** is that it did not require user interaction beyond plugging in the *USB drive*. As soon as **Windows Explorer** attempted to display the icon for the **LNK** file, the malicious code was executed.

Here's how it worked step by step:

- **Stuxnet** created **.LNK** files that contained specially crafted parameters pointing to a malicious **DLL** file.
- Alongside the **.LNK** files, **Stuxnet** included the malicious **DLL** (dynamic-link library) on removable media (like *USB flash drives*). This **DLL** contained the actual malicious code to be executed.

- When an infected *USB drive* was plugged into a computer, Windows would automatically parse the **.LNK** files to display their icons. Due to **CVE-2010-2568**, this parsing process would lead to the loading and execution of the malicious **DLL**.
- The malicious **DLL** executed by exploiting **CVE-2010-2568** would then drop the main **Stuxnet** payload onto the system.
- This payload would install itself and begin its various routines, including propagation, communication with command-and-control servers, and the targeted sabotage of industrial control systems.

CVE-2010-2729

CVE-2010–2729 is a vulnerability in the **Windows Print Spooler** service that Stuxnet exploited to spread across networks and infect additional machines. **CVE-2010–2729** is a remote code execution vulnerability in the **Windows Print Spooler** service. This service is responsible for managing all print jobs sent to the computer printer or print server. The vulnerability arises from the way the **Print Spooler** service processes print requests and handles specially crafted print jobs. An attacker who successfully exploits this vulnerability can execute arbitrary code with elevated privileges on the targeted system.

The vulnerability in the **Print Spooler** service involved improper handling of printer spool files. Specially crafted files could cause the service to execute arbitrary code instead of just handling the print job data. The **Windows Print Spooler** service typically listens for print jobs from both local and network sources. **Stuxnet** exploited this by sending crafted requests that triggered the vulnerability.

CVE-2010-2729 allowed **Stuxnet** to execute code with the same privileges as the **Print Spooler** service, which typically runs with elevated system privileges. This high level of access was critical for **Stuxnet** to perform its operations and spread further.

Stuxnet leveraged **CVE-2010-2729** as part of its network propagation strategy. The ability to exploit a service that listens for network traffic allowed **Stuxnet** to move laterally within corporate or industrial networks, increasing its infection rate and reach. By executing code with system-level privileges, **Stuxnet** could make significant changes to the system configuration and ensure persistence while avoiding detection and removal. At the

time **Stuxnet** was active, **CVE-2010-2729** was a zero-day vulnerability, meaning no patches were available, and systems were widely vulnerable.

Here's how it was used step by step:

- **Stuxnet** would first perform network discovery to identify other computers on the same network, focusing on those with the **Windows Print Spooler** service running.
- **Stuxnet** crafted special print jobs that exploited the vulnerability in the **Print Spooler** service. These print jobs included embedded code designed to execute on the targeted system.
- When the **Print Spooler** service on a target machine processed the malicious print job, the embedded code would be executed. Due to the nature of the vulnerability, this execution occurred with system-level privileges, granting Stuxnet high-level access to the infected machine.
- Once the malicious code executed on the target system, it would install the Stuxnet payload. This included dropping the necessary files and making registry changes to ensure persistence.
- The payload would then continue to propagate by seeking out new targets within the network, using the same and other exploits.

MS08-067

MS08-067 is a critical vulnerability in the **Windows Server** service that allows remote code execution. This vulnerability was another key component used by **Stuxnet** for propagation. **MS08-067** refers to a vulnerability in the **Windows Server** service, which is responsible for handling network requests via the **Server Message Block (SMB)** protocol. The vulnerability arises from the incorrect handling of specially crafted **remote procedure call (RPC)** requests. An attacker could exploit this vulnerability to execute arbitrary code on the affected system.

The vulnerability involved a buffer overflow in the Server service's handling of **SMB** packets. By sending an **RPC** request with a specially crafted payload, an attacker could cause the service to overwrite memory with arbitrary code. Upon receiving the malicious **RPC** request, the Server service would process it, leading to the execution of the

embedded payload with the same privileges as the service, typically system-level privileges. Because the Server service often runs with high privileges, exploiting **MS08–067** allowed **Stuxnet** to execute its code with elevated privileges, facilitating further system modifications and persistence mechanisms. The ability to exploit this vulnerability remotely without user interaction made it particularly dangerous and effective for spreading malware.

Stuxnet leveraged **MS08-067** as part of its network propagation strategy, exploiting the vulnerability to infect additional machines without requiring user interaction.

Here's how **Stuxnet** used this vulnerability:

- **Stuxnet** first scanned the local network to identify other computers that had the Server service running and were potentially vulnerable to **MS08–067**.
- Once potential targets were identified, **Stuxnet** sent specially crafted **RPC** requests to the **SMB** service on these machines. These requests exploited the vulnerability in the way the Server service handled these requests.
- The crafted **RPC** request caused the Server service on the targeted machine to execute arbitrary code embedded in the request. This allowed **Stuxnet** to execute its payload on the remote machine without needing any user interaction or additional exploitation.
- After successfully exploiting **MS08–067**, **Stuxnet** would install its payload on the targeted machine. This involved dropping its main components and making necessary changes to ensure persistence.
- The newly infected machine would then become part of **Stuxnet's** propagation network, continuing to scan for other vulnerable systems and repeating the exploitation process.

A Look at Siemens Step 7 Software

Siemens Step 7 is a comprehensive suite of software tools used for programming and configuring Siemens programmable logic controllers (**PLCs**). Siemens **PLCs** are widely used in industrial automation for controlling machinery and processes in manufacturing, power plants, and other industrial applications. **Step 7** is part of the **Siemens SIMATIC** product family, which includes various hardware and software components for automation.

Key features:

- **Ladder Logic (LAD)**: A graphical programming language resembling electrical relay logic diagrams.
- **Function Block Diagram (FBD)**: Another graphical language that represents functions and their connections.
- **Statement List (STL)**: A low-level, textual programming language similar to assembly language.
- **Structured Control Language (SCL)**: A high-level, text-based language similar to **Pascal** or **C**.
- Allows users to configure hardware components, such as **CPUs**, *input/output* modules, and communication interfaces.
- Supports configuration of network settings and communication protocols.
- Includes tools for simulating **PLC** programs to test logic without physical hardware.
- Provides debugging features such as breakpoints, watch tables, and diagnostic functions.
- **Step 7** integrates with **Siemens HMI (Human-Machine Interface)** products, allowing for the creation of user interfaces for monitoring and controlling industrial processes.
- Supports various industrial communication protocols such as *PROFINET*, *PROFIBUS*, and *Ethernet/IP*.
- Allows for the integration of **PLCs** into larger automation networks.
- Offers tools for monitoring and diagnosing issues in **PLC** systems.

- Provides maintenance features such as firmware updates and system backups.

Stuxnet specifically targeted **Siemens Step 7** software to sabotage industrial control systems. Here's how:

- **Detection**: **Stuxnet** detected the presence of **Siemens Step 7** software on an infected system to determine if it was in the right environment to activate its payload.
- **Code Injection**: Once it identified **Step 7**, **Stuxnet** injected malicious code into the **PLCs** programmed by **Step 7**. This code altered the operation of machinery, such as centrifuges in Iran's nuclear facilities, causing them to malfunction.
- **Stealth**: **Stuxnet** was able to hide its modifications from operators by intercepting and falsifying the data reported back to the **Step 7** software, making the system appear to be functioning normally while it was actually being sabotaged.

Analysis of a Malware-Infected Image

We will analyze the *stuxnet-vmem* file, which is a sample image file

File Hash Analysis

```
FLARE-VM 11/24/2025 18:42:23
PS C:\Users\hrp\Desktop\Stux > Get-FileHash .\stuxnet.vmem -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            6783D95883A32762042CAE731887AE3693B030C1                               C:\Users\hrp\Desktop\Stux\stu...


FLARE-VM 11/24/2025 18:42:37
PS C:\Users\hrp\Desktop\Stux > Get-FileHash .\stuxnet.vmem -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          5F19FF1333FC3901FBF3FAFB50D2ADB0C495CF6D33789E5A959499A92AEEFE77       C:\Users\hrp\Desktop\Stux\stu...


FLARE-VM 11/24/2025 18:42:42
PS C:\Users\hrp\Desktop\Stux > Get-FileHash .\stuxnet.vmem -Algorithm MD5

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             9FB971822DCB393F930227C8854F7679                                       C:\Users\hrp\Desktop\Stux\stu...
```

Image File Info

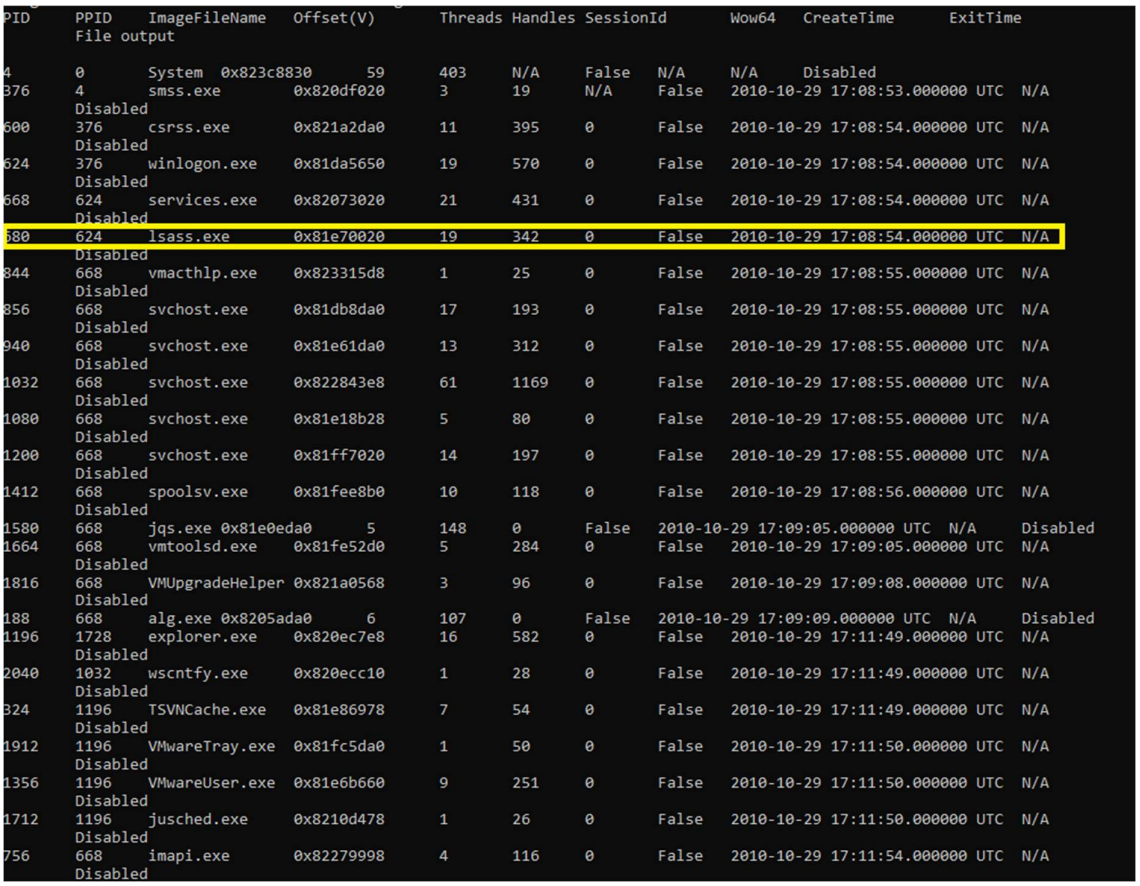
Now we can get information about the image file we have

```
Kernel Base     0x804d7000
DTB     0x319000
Symbols jar:file:C:\Users\hrp\Desktop\volatility3\volatility3\symbols\windows\windows.zip!windows/ntkrnlpa.pdb/30B5FB31A
E7E4ACAABA750AA241FF331-1.json.xz
Is64Bit False
IsPAE   True
layer_name      0 WindowsIntelPAE
memory_layer    1 FileLayer
KdDebuggerDataBlock     0x80545ae0
NTBuildLab      2600.xpsp.080413-2111
CSDVersion      3
KdVersionBlock  0x80545ab8
Major/Minor     15.2600
MachineType     332
KeNumberProcessors      1
SystemTime      2011-06-03 04:31:36+00:00
NtSystemRoot    C:\WINDOWS
NtProductType   NtProductWinNt
NtMajorVersion  5
NtMinorVersion  1
PE MajorOperatingSystemVersion  5
PE MinorOperatingSystemVersion  1
PE Machine      332
PE TimeDateStamp        Sun Apr 13 18:31:06 2008
```

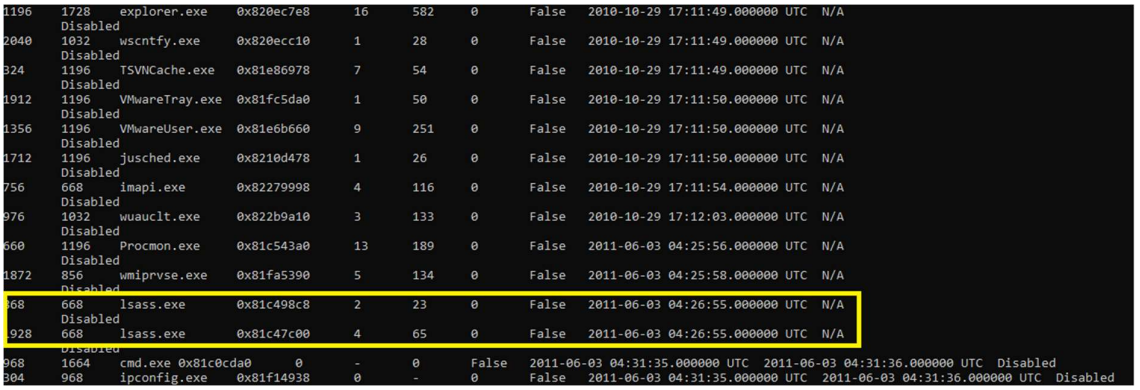
We can see the suggested profile is WinXPSP2x86 and WinXPSP3x86. WinXPSP2x86 and WinXPSP3x86 refer to specific versions of the Microsoft Windows XP operating system, with “x86” indicating that they are for 32-bit processors, and “SP2” and “SP3” standing for Service Pack 2 and Service Pack 3, respectively.

For Volatility we will use the WinXPSP3x86 profile definition.

Process Analysis



```
PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime
        File output

4       0       System  0x823c8830      59      403     N/A     False   N/A     N/A     Disabled
376     4       smss.exe        0x820df020      3       19      N/A     False   2010-10-29 17:08:53.000000 UTC  N/A
        Disabled
600     376     csrss.exe       0x821a2da0      11      395     0       False   2010-10-29 17:08:54.000000 UTC  N/A
        Disabled
624     376     winlogon.exe    0x81da5650      19      570     0       False   2010-10-29 17:08:54.000000 UTC  N/A
        Disabled
668     624     services.exe    0x82073020      21      431     0       False   2010-10-29 17:08:54.000000 UTC  N/A
        Disabled
680     624     lsass.exe       0x81e70020      19      342     0       False   2010-10-29 17:08:54.000000 UTC  N/A
        Disabled
844     668     vmacthlp.exe    0x823315d8      1       25      0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
856     668     svchost.exe     0x81db8da0      17      193     0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
940     668     svchost.exe     0x81e61da0      13      312     0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
1032    668     svchost.exe     0x822843e8      61      1169    0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
1080    668     svchost.exe     0x81e18b28      5       80      0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
1200    668     svchost.exe     0x81ff7020      14      197     0       False   2010-10-29 17:08:55.000000 UTC  N/A
        Disabled
1412    668     spoolsv.exe     0x81fee8b0      10      118     0       False   2010-10-29 17:08:56.000000 UTC  N/A
        Disabled
1580    668     jqs.exe 0x81e0eda0      5       148     0       False   2010-10-29 17:09:05.000000 UTC  N/A     Disabled
1664    668     vmtoolsd.exe    0x81fe52d0      5       284     0       False   2010-10-29 17:09:05.000000 UTC  N/A
        Disabled
1816    668     VMUpgradeHelper 0x821a0568      3       96      0       False   2010-10-29 17:09:08.000000 UTC  N/A
        Disabled
188     668     alg.exe 0x8205ada0      6       107     0       False   2010-10-29 17:09:09.000000 UTC  N/A     Disabled
1196    1728    explorer.exe    0x820ec7e8      16      582     0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
2040    1032    wscntfy.exe     0x820ecc10      1       28      0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
324     1196    TSVNCache.exe   0x81e86978      7       54      0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
1912    1196    VMwareTray.exe  0x81fc5da0      1       50      0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
1356    1196    VMwareUser.exe  0x81e6b660      9       251     0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
1712    1196    jusched.exe     0x8210d478      1       26      0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
756     668     imapi.exe       0x82279998      4       116     0       False   2010-10-29 17:11:54.000000 UTC  N/A
        Disabled
```



```
1196    1728    explorer.exe    0x820ec7e8      16      582     0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
2040    1032    wscntfy.exe     0x820ecc10      1       28      0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
324     1196    TSVNCache.exe   0x81e86978      7       54      0       False   2010-10-29 17:11:49.000000 UTC  N/A
        Disabled
1912    1196    VMwareTray.exe  0x81fc5da0      1       50      0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
1356    1196    VMwareUser.exe  0x81e6b660      9       251     0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
1712    1196    jusched.exe     0x8210d478      1       26      0       False   2010-10-29 17:11:50.000000 UTC  N/A
        Disabled
756     668     imapi.exe       0x82279998      4       116     0       False   2010-10-29 17:11:54.000000 UTC  N/A
        Disabled
976     1032    wuauclt.exe     0x822b9a10      3       133     0       False   2010-10-29 17:12:03.000000 UTC  N/A
        Disabled
660     1196    Procmon.exe     0x81c543a0      13      189     0       False   2011-06-03 04:25:56.000000 UTC  N/A
        Disabled
1872    856     wmiprvse.exe    0x81fa5390      5       134     0       False   2011-06-03 04:25:58.000000 UTC  N/A
        Disabled
668     668     lsass.exe       0x81c498c8      2       23      0       False   2011-06-03 04:26:55.000000 UTC  N/A
        Disabled
1928    668     lsass.exe       0x81c47c00      4       65      0       False   2011-06-03 04:26:55.000000 UTC  N/A
        Disabled
968     1664    cmd.exe 0x81c0cda0      0       -       0       False   2011-06-03 04:31:35.000000 UTC  2011-06-03 04:31:36.000000 UTC  Disabled
804     968     ipconfig.exe    0x81f14938      0       -       0       False   2011-06-03 04:31:35.000000 UTC  2011-06-03 04:31:36.000000 UTC  Disabled
```

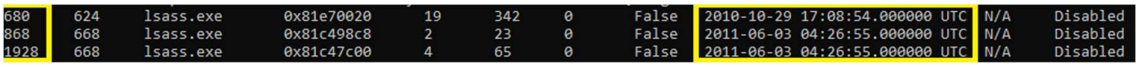
"**lsass.exe**" looks suspicious.

***lsass.exe**: **The Local Security Authority Subsystem Service** process is responsible for enforcing the security policy on Windows systems. It handles the authentication of users*

*logging into the Windows system. It verifies user credentials against the stored security database. The **lsass.exe** file is typically located in the **C:\Windows\System32** directory. It is a vital system process, and terminating it can cause system instability and may result in a system shutdown or restart.*

Malware may disguise itself as lsass.exe or use a similarly named process to avoid detection. Ensuring the file is located in the correct directory (**C:\Windows\System32**) can help verify its authenticity.

Process ID analysis



680	624	lsass.exe	0x81e70020	19	342	0	False	2010-10-29 17:08:54.000000 UTC	N/A	Disabled
868	668	lsass.exe	0x81c498c8	2	23	0	False	2011-06-03 04:26:55.000000 UTC	N/A	Disabled
1928	668	lsass.exe	0x81c47c00	4	65	0	False	2011-06-03 04:26:55.000000 UTC	N/A	Disabled

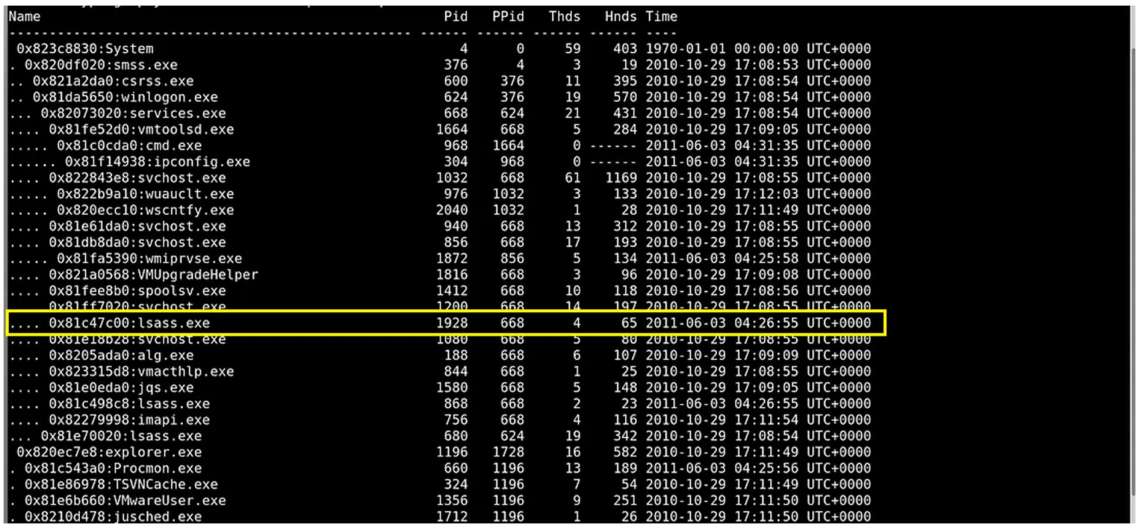
- **PID 680**
- **PID 868**
- **PID 1928**

Also the startup times may be questionable.

- **PID 680** at *2010–10–29 17:08:54*
- **PID 868** at *2011–06–03 04:26:55*
- **PID 1928** at *2011–06–03 04:26:55*

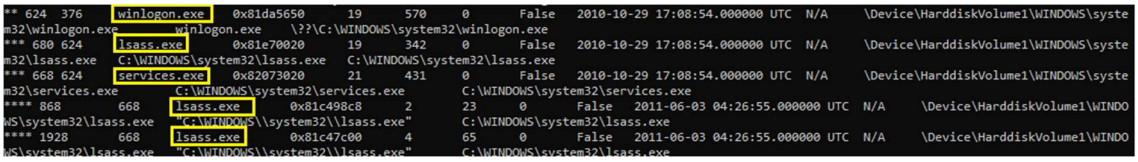
Process Tree Analysis

We saw two separate PIDs at the same time. We need to examine the process tree for a detailed analysis:



```
Name                                                 Pid   PPid   Thds   Hnds Time
---------------------------------------------------- ------ ------ ------ ------ ----
 0x823c8830:System                                      4      0     59    403 1970-01-01 00:00:00 UTC+0000
. 0x820df020:smss.exe                                 376      4      3     19 2010-10-29 17:08:53 UTC+0000
.. 0x821a2da0:csrss.exe                               600    376     11    395 2010-10-29 17:08:54 UTC+0000
.. 0x81da5650:winlogon.exe                            624    376     19    570 2010-10-29 17:08:54 UTC+0000
... 0x82073020:services.exe                           668    624     21    431 2010-10-29 17:08:54 UTC+0000
.... 0x81fe52d0:vmtoolsd.exe                         1664    668      5    284 2010-10-29 17:09:05 UTC+0000
..... 0x81c0cda0:cmd.exe                              968   1664      0 ------ 2011-06-03 04:31:35 UTC+0000
...... 0x81f14938:ipconfig.exe                        304    968      0 ------ 2011-06-03 04:31:35 UTC+0000
.... 0x822843e8:svchost.exe                          1032    668     61   1169 2010-10-29 17:08:55 UTC+0000
..... 0x822b9a10:wuauclt.exe                          976   1032      3    133 2010-10-29 17:12:03 UTC+0000
..... 0x820ecc10:wscntfy.exe                         2040   1032      1     28 2010-10-29 17:11:49 UTC+0000
.... 0x81e61da0:svchost.exe                           940    668     13    312 2010-10-29 17:08:55 UTC+0000
.... 0x81db8da0:svchost.exe                           856    668     17    193 2010-10-29 17:08:55 UTC+0000
..... 0x81fa5390:wmiprvse.exe                        1872    856      5    134 2011-06-03 04:25:58 UTC+0000
.... 0x821a0568:VMUpgradeHelper                      1816    668      3     96 2010-10-29 17:09:08 UTC+0000
.... 0x81fee8b0:spoolsv.exe                          1412    668     10    118 2010-10-29 17:08:56 UTC+0000
.... 0x81ff7020:svchost.exe                          1200    668     14    197 2010-10-29 17:08:55 UTC+0000
.... 0x81c47c00:lsass.exe                            1928    668      4     65 2011-06-03 04:26:55 UTC+0000
.... 0x81e18b28:svchost.exe                          1080    668      5     80 2010-10-29 17:08:55 UTC+0000
.... 0x8205ada0:alg.exe                               188    668      6    107 2010-10-29 17:09:09 UTC+0000
.... 0x823315d8:vmacthlp.exe                          844    668      1     25 2010-10-29 17:08:55 UTC+0000
.... 0x81e0eda0:jqs.exe                              1580    668      5    148 2010-10-29 17:09:05 UTC+0000
.... 0x81c498c8:lsass.exe                             868    668      2     23 2011-06-03 04:26:55 UTC+0000
.... 0x82279998:imapi.exe                             756    668      4    116 2010-10-29 17:11:54 UTC+0000
... 0x81e70020:lsass.exe                              680    624     19    342 2010-10-29 17:08:54 UTC+0000
 0x820ec7e8:explorer.exe                             1196   1728     16    582 2010-10-29 17:11:49 UTC+0000
. 0x81c543a0:Procmon.exe                             660   1196     13    189 2011-06-03 04:25:56 UTC+0000
. 0x81e86978:TSVNCache.exe                           324   1196      7     54 2010-10-29 17:11:49 UTC+0000
. 0x81e6b660:VMwareUser.exe                         1356   1196      9    251 2010-10-29 17:11:50 UTC+0000
. 0x8210d478:jusched.exe                            1712   1196      1     26 2010-10-29 17:11:50 UTC+0000
```

On analyzing the process tree with pid = 868, 1928 and 680 we get the following information:



```
** 624  376     winlogon.exe     0x81da5650      19      570     0       False   2010-10-29 17:08:54.000000 UTC  N/A     \Device\HarddiskVolume1\WINDOWS\syste
m32\winlogon.exe        winlogon.exe    \??\C:\WINDOWS\system32\winlogon.exe
*** 680 624     lsass.exe       0x81e70020      19      342     0       False   2010-10-29 17:08:54.000000 UTC  N/A     \Device\HarddiskVolume1\WINDOWS\syste
m32\lsass.exe   C:\WINDOWS\system32\lsass.exe   C:\WINDOWS\system32\lsass.exe
*** 668 624     services.exe    0x82073020      21      431     0       False   2010-10-29 17:08:54.000000 UTC  N/A     \Device\HarddiskVolume1\WINDOWS\syste
m32\services.exe        C:\WINDOWS\system32\services.exe        C:\WINDOWS\system32\services.exe
**** 868        668     lsass.exe       0x81c498c8      2       23      0       False   2011-06-03 04:26:55.000000 UTC  N/A     \Device\HarddiskVolume1\WINDO
WS\system32\lsass.exe   "C:\WINDOWS\\system32\\lsass.exe"        C:\WINDOWS\system32\lsass.exe
**** 1928       668     lsass.exe       0x81c47c00      4       65      0       False   2011-06-03 04:26:55.000000 UTC  N/A     \Device\HarddiskVolume1\WINDO
WS\system32\lsass.exe   "C:\WINDOWS\\system32\\lsass.exe"        C:\WINDOWS\system32\lsass.exe
```

We can notice that Winlogon.exe (**PID 624**) started one of the "**lsass.exe**" process (**PID 680**). "**Winlogon.exe**" always starts the real "**lsass.exe**". Therefore, the "**lsass.exe**" included in this PID appears innocent. The legitimate lsass.exe process is usually started by winlogon.exe during system boot-up. This is a standard part of the Windows initialization process.

The "**lsass.exe**" with **PID 868** and **PID 1928** was started by the "**services.exe**" process. It isn't a normal behavior.

Services.exe*: This process is responsible for starting, stopping, and interacting with system services. It should not be responsible for starting lsass.exe.*

The additional lsass.exe processes started by services.exe could indicate malware or unauthorized software attempting to masquerade as lsass.exe to avoid detection.

Process Privileges Used

PID	Process Value	Privilege	Attributes	Description
868	lsass.exe	7	SeTcbPrivilege	Present,Enabled,Default Act as part of the operating system
868	lsass.exe	2	SeCreateTokenPrivilege	Present Create a token object
868	lsass.exe	9	SeTakeOwnershipPrivilege	Present Take ownership of files/objects
868	lsass.exe	15	SeCreatePagefilePrivilege	Present,Enabled,Default Create a pagefile
868	lsass.exe	4	SeLockMemoryPrivilege	Present,Enabled,Default Lock pages in memory
868	lsass.exe	3	SeAssignPrimaryTokenPrivilege	Present Replace a process-level token
868	lsass.exe	5	SeIncreaseQuotaPrivilege	Present Increase quotas
868	lsass.exe	14	SeIncreaseBasePriorityPrivilege	Present,Enabled,Default Increase scheduling priority
868	lsass.exe	16	SeCreatePermanentPrivilege	Present,Enabled,Default Create permanent shared objects
868	lsass.exe	20	SeDebugPrivilege	Present,Enabled,Default Debug programs
868	lsass.exe	21	SeAuditPrivilege	Present,Enabled,Default Generate security audits
868	lsass.exe	8	SeSecurityPrivilege	Present Manage auditing and security log
868	lsass.exe	22	SeSystemEnvironmentPrivilege	Present Edit firmware environment values
868	lsass.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default Receive notifications of changes to files or directories
868	lsass.exe	17	SeBackupPrivilege	Present Backup files and directories
868	lsass.exe	18	SeRestorePrivilege	Present Restore files and directories
868	lsass.exe	19	SeShutdownPrivilege	Present Shut down the system
868	lsass.exe	10	SeLoadDriverPrivilege	Present,Enabled Load and unload device drivers
868	lsass.exe	13	SeProfileSingleProcessPrivilege	Present,Enabled,Default Profile a single process
868	lsass.exe	12	SeSystemtimePrivilege	Present Change the system time
868	lsass.exe	25	SeUndockPrivilege	Present,Enabled Remove computer from docking station
868	lsass.exe	28	SeManageVolumePrivilege	Present Manage the files on a volume
868	lsass.exe	29	SeImpersonatePrivilege	Present,Enabled,Default Impersonate a client after authentication
868	lsass.exe	30	SeCreateGlobalPrivilege	Present,Enabled,Default Create global objects

PID	Process Value	Privilege	Attributes	Description
1928	lsass.exe	7	SeTcbPrivilege	Present,Enabled,Default Act as part of the operating system
1928	lsass.exe	2	SeCreateTokenPrivilege	Present Create a token object
1928	lsass.exe	9	SeTakeOwnershipPrivilege	Present Take ownership of files/objects
1928	lsass.exe	15	SeCreatePagefilePrivilege	Present,Enabled,Default Create a pagefile
1928	lsass.exe	4	SeLockMemoryPrivilege	Present,Enabled,Default Lock pages in memory
1928	lsass.exe	3	SeAssignPrimaryTokenPrivilege	Present Replace a process-level token
1928	lsass.exe	5	SeIncreaseQuotaPrivilege	Present Increase quotas
1928	lsass.exe	14	SeIncreaseBasePriorityPrivilege	Present,Enabled,Default Increase scheduling priority
1928	lsass.exe	16	SeCreatePermanentPrivilege	Present,Enabled,Default Create permanent shared objects
1928	lsass.exe	20	SeDebugPrivilege	Present,Enabled,Default Debug programs
1928	lsass.exe	21	SeAuditPrivilege	Present,Enabled,Default Generate security audits
1928	lsass.exe	8	SeSecurityPrivilege	Present Manage auditing and security log
1928	lsass.exe	22	SeSystemEnvironmentPrivilege	Present Edit firmware environment values
1928	lsass.exe	23	SeChangeNotifyPrivilege	Present,Enabled,Default Receive notifications of changes to files or directories
1928	lsass.exe	17	SeBackupPrivilege	Present Backup files and directories
1928	lsass.exe	18	SeRestorePrivilege	Present Restore files and directories
1928	lsass.exe	19	SeShutdownPrivilege	Present Shut down the system
1928	lsass.exe	10	SeLoadDriverPrivilege	Present,Enabled Load and unload device drivers
1928	lsass.exe	13	SeProfileSingleProcessPrivilege	Present,Enabled,Default Profile a single process
1928	lsass.exe	12	SeSystemtimePrivilege	Present Change the system time
1928	lsass.exe	25	SeUndockPrivilege	Present,Enabled Remove computer from docking station
1928	lsass.exe	28	SeManageVolumePrivilege	Present Manage the files on a volume
1928	lsass.exe	29	SeImpersonatePrivilege	Present,Enabled,Default Impersonate a client after authentication
1928	lsass.exe	30	SeCreateGlobalPrivilege	Present,Enabled,Default Create global objects

Network Communications

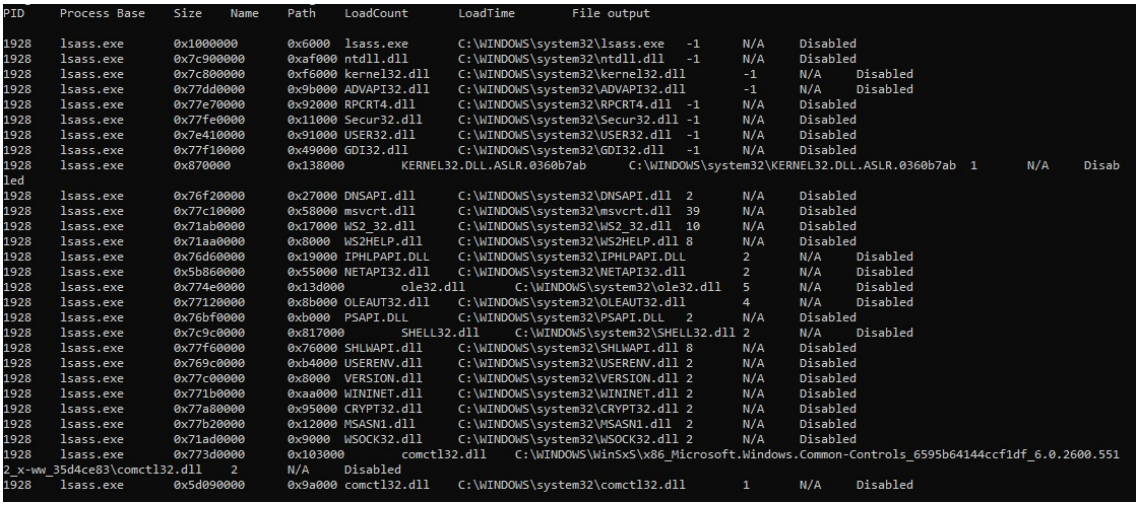
Offset(V)	PID	Port	Proto	Protocol	Address	Create Time
0x81dc2008	680	500	17	UDP	0.0.0.0	2010-10-29 17:09:05 UTC+0000
0x82061c08	4	445	6	TCP	0.0.0.0	2010-10-29 17:08:53 UTC+0000
0x82294aa8	940	135	6	TCP	0.0.0.0	2010-10-29 17:08:55 UTC+0000
0x821a5008	188	1025	6	TCP	127.0.0.1	2010-10-29 17:09:09 UTC+0000
0x81cb3d70	1080	1141	17	UDP	0.0.0.0	2010-10-31 16:36:16 UTC+0000
0x81da4d18	680	0	255	Reserved	0.0.0.0	2010-10-29 17:09:05 UTC+0000
0x81fdbe98	1032	123	17	UDP	127.0.0.1	2011-06-03 04:25:47 UTC+0000
0x81c79778	1080	1142	17	UDP	0.0.0.0	2010-10-31 16:36:16 UTC+0000
0x81c20898	1200	1900	17	UDP	127.0.0.1	2011-06-03 04:25:47 UTC+0000
0x82060008	680	4500	17	UDP	0.0.0.0	2010-10-29 17:09:05 UTC+0000
0x81cb9e98	1580	5152	6	TCP	127.0.0.1	2010-10-29 17:09:05 UTC+0000
0x81da54b0	4	445	17	UDP	0.0.0.0	2010-10-29 17:08:53 UTC+0000

We observe that the **PID 680** process is listening on ports **500** and **4500**. These ports are commonly associated with **IPsec (Internet Protocol Security) VPN (Virtual Private**

Network) communication, specifically for **IKE (Internet Key Exchange)** and **IPsec NAT-T (Network Address Translation Traversal)** protocols. It has a normal behavior.

But the other processes **PID 868 and PID 1928** are not binding to these ports suggests they might not be performing the legitimate functions of the lsass.exe process. It could indicate potential malicious activity or unauthorized processes masquerading as lsass.exe.

.dll files Used



```
PID    Process Base    Size    Name    Path    LoadCount       LoadTime        File output

1928    lsass.exe       0x1000000       0x6000  lsass.exe       C:\WINDOWS\system32\lsass.exe   -1      N/A     Disabled
1928    lsass.exe       0x7c900000      0xaf000 ntdll.dll       C:\WINDOWS\system32\ntdll.dll   -1      N/A     Disabled
1928    lsass.exe       0x7c800000      0xf6000 kernel32.dll    C:\WINDOWS\system32\kernel32.dll        -1      N/A     Disabled
1928    lsass.exe       0x77dd0000      0x9b000 ADVAPI32.dll    C:\WINDOWS\system32\ADVAPI32.dll        -1      N/A     Disabled
1928    lsass.exe       0x77e70000      0x92000 RPCRT4.dll      C:\WINDOWS\system32\RPCRT4.dll  -1      N/A     Disabled
1928    lsass.exe       0x77fe0000      0x11000 Secur32.dll     C:\WINDOWS\system32\Secur32.dll -1      N/A     Disabled
1928    lsass.exe       0x7e410000      0x91000 USER32.dll      C:\WINDOWS\system32\USER32.dll  -1      N/A     Disabled
1928    lsass.exe       0x77f10000      0x49000 GDI32.dll       C:\WINDOWS\system32\GDI32.dll   -1      N/A     Disabled
1928    lsass.exe       0x870000        0x138000        KERNEL32.DLL.ASLR.0360b7ab      C:\WINDOWS\system32\KERNEL32.DLL.ASLR.0360b7ab  1       N/A     Disab
led
1928    lsass.exe       0x76f20000      0x27000 DNSAPI.dll      C:\WINDOWS\system32\DNSAPI.dll  2       N/A     Disabled
1928    lsass.exe       0x77c10000      0x58000 msvcrt.dll      C:\WINDOWS\system32\msvcrt.dll  39      N/A     Disabled
1928    lsass.exe       0x71ab0000      0x17000 WS2_32.dll      C:\WINDOWS\system32\WS2_32.dll  10      N/A     Disabled
1928    lsass.exe       0x71aa0000      0x8000  WS2HELP.dll     C:\WINDOWS\system32\WS2HELP.dll 8       N/A     Disabled
1928    lsass.exe       0x76d60000      0x19000 IPHLPAPI.DLL    C:\WINDOWS\system32\IPHLPAPI.DLL        2       N/A     Disabled
1928    lsass.exe       0x5b860000      0x55000 NETAPI32.dll    C:\WINDOWS\system32\NETAPI32.dll        2       N/A     Disabled
1928    lsass.exe       0x774e0000      0x13d000        ole32.dll       C:\WINDOWS\system32\ole32.dll   5       N/A     Disabled
1928    lsass.exe       0x77120000      0x8b000 OLEAUT32.dll    C:\WINDOWS\system32\OLEAUT32.dll        4       N/A     Disabled
1928    lsass.exe       0x76bf0000      0xb000  PSAPI.DLL       C:\WINDOWS\system32\PSAPI.DLL   2       N/A     Disabled
1928    lsass.exe       0x7c9c0000      0x817000        SHELL32.dll     C:\WINDOWS\system32\SHELL32.dll 2       N/A     Disabled
1928    lsass.exe       0x77f60000      0x76000 SHLWAPI.dll     C:\WINDOWS\system32\SHLWAPI.dll 8       N/A     Disabled
1928    lsass.exe       0x769c0000      0xb4000 USERENV.dll     C:\WINDOWS\system32\USERENV.dll 2       N/A     Disabled
1928    lsass.exe       0x77c00000      0x8000  VERSION.dll     C:\WINDOWS\system32\VERSION.dll 2       N/A     Disabled
1928    lsass.exe       0x771b0000      0xaa000 WININET.dll     C:\WINDOWS\system32\WININET.dll 2       N/A     Disabled
1928    lsass.exe       0x77a80000      0x95000 CRYPT32.dll     C:\WINDOWS\system32\CRYPT32.dll 2       N/A     Disabled
1928    lsass.exe       0x77b20000      0x12000 MSASN1.dll      C:\WINDOWS\system32\MSASN1.dll  2       N/A     Disabled
1928    lsass.exe       0x71ad0000      0x9000  WSOCK32.dll     C:\WINDOWS\system32\WSOCK32.dll 2       N/A     Disabled
1928    lsass.exe       0x773d0000      0x103000        comctl32.dll    C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.551
2_x-ww_35d4ce83\comctl32.dll    2       N/A     Disabled
1928    lsass.exe       0x5d090000      0x9a000 comctl32.dll    C:\WINDOWS\system32\comctl32.dll        1       N/A     Disabled
```

```
PID     Process Base    Size    Name    Path    LoadCount       LoadTime        File output

868     lsass.exe       0x1000000       0x6000  lsass.exe       C:\WINDOWS\system32\lsass.exe   -1      N/A     Disabled
868     lsass.exe       0x7c900000      0xaf000 ntdll.dll       C:\WINDOWS\system32\ntdll.dll   -1      N/A     Disabled
868     lsass.exe       0x7c800000      0xf6000 kernel32.dll    C:\WINDOWS\system32\kernel32.dll        -1      N/A     Disabled
868     lsass.exe       0x77dd0000      0x9b000 ADVAPI32.dll    C:\WINDOWS\system32\ADVAPI32.dll        -1      N/A     Disabled
868     lsass.exe       0x77e70000      0x92000 RPCRT4.dll      C:\WINDOWS\system32\RPCRT4.dll  -1      N/A     Disabled
868     lsass.exe       0x77fe0000      0x11000 Secur32.dll     C:\WINDOWS\system32\Secur32.dll -1      N/A     Disabled
868     lsass.exe       0x7e410000      0x91000 USER32.dll      C:\WINDOWS\system32\USER32.dll  -1      N/A     Disabled
868     lsass.exe       0x77f10000      0x49000 GDI32.dll       C:\WINDOWS\system32\GDI32.dll   -1      N/A     Disabled
```

Some suspicious downloads on **PID 1928** attract attention. We need to detect **unlinked DLLs**

Unlinked DLLs

Pid	Process	Base	InLoad	InInit	InMem	MappedPath
1928	lsass.exe	0x80000	False	False	False	N/A
1928	lsass.exe	0x7c900000	True	True	True	\WINDOWS\system32\ntdll.dll
1928	lsass.exe	0x1000000	True	False	True	N/A
1928	lsass.exe	0x870000	True	True	True	N/A
1928	lsass.exe	0x7c800000	True	True	True	\WINDOWS\system32\kernel32.dll
1928	lsass.exe	0x77dd0000	True	True	True	\WINDOWS\system32\advapi32.dll
1928	lsass.exe	0x76f20000	True	True	True	\WINDOWS\system32\dnsapi.dll
1928	lsass.exe	0x71ab0000	True	True	True	\WINDOWS\system32\ws2_32.dll
1928	lsass.exe	0x71aa0000	True	True	True	\WINDOWS\system32\ws2help.dll
1928	lsass.exe	0x5b860000	True	True	True	\WINDOWS\system32\netapi32.dll
1928	lsass.exe	0x5d090000	True	True	True	\WINDOWS\system32\comctl32.dll
1928	lsass.exe	0x76d60000	True	True	True	\WINDOWS\system32\iphlpapi.dll
1928	lsass.exe	0x76bf0000	True	True	True	\WINDOWS\system32\psapi.dll
1928	lsass.exe	0x769c0000	True	True	True	\WINDOWS\system32\userenv.dll
1928	lsass.exe	0x71ad0000	True	True	True	\WINDOWS\system32\wsock32.dll
1928	lsass.exe	0x77c10000	True	True	True	\WINDOWS\system32\msvcrt.dll
1928	lsass.exe	0x774e0000	True	True	True	\WINDOWS\system32\ole32.dll
1928	lsass.exe	0x77120000	True	True	True	\WINDOWS\system32\oleaut32.dll
1928	lsass.exe	0x771b0000	True	True	True	\WINDOWS\system32\wininet.dll
1928	lsass.exe	0x773d0000	True	True	True	\WINDOWS\WinSxS\x86_Microsoft.Windows.4ce83\comctl32.dll
1928	lsass.exe	0x77c00000	True	True	True	\WINDOWS\system32\version.dll
1928	lsass.exe	0x77a80000	True	True	True	\WINDOWS\system32\crypt32.dll
1928	lsass.exe	0x77b20000	True	True	True	\WINDOWS\system32\msasn1.dll
1928	lsass.exe	0x77e70000	True	True	True	\WINDOWS\system32\rpcrt4.dll
1928	lsass.exe	0x77fe0000	True	True	True	\WINDOWS\system32\secur32.dll
1928	lsass.exe	0x77f10000	True	True	True	\WINDOWS\system32\gdi32.dll
1928	lsass.exe	0x77f60000	True	True	True	\WINDOWS\system32\shlwapi.dll
1928	lsass.exe	0x7e410000	True	True	True	\WINDOWS\system32\user32.dll
1928	lsass.exe	0x7c9c0000	True	True	True	\WINDOWS\system32\shell32.dll

Process Analysis to Identify Hidden Processes

Offset(Virtual)	Name	PID	pslist	psscan	thrdscan	csrss	Exit Time
0x81c47c00	lsass.exe	1928	True	False	True	True	
0x1e47c00	lsass.exe	1928	False	True	False	False	
0x21f7020	svchost.exe	1200	False	True	False	False	
0x230d478	jusched.exe	1712	False	True	False	False	
0x24b9a10	wuauclt.exe	976	False	True	False	False	
0x81fc5da0	VMwareTray.exe	1912	True	False	True	True	
0x81c0cda0	cmd.exe 968	True	False	True	False	2011-06-03 04:31:36+00:00	
0x820ecc10	wscntfy.exe	2040	True	False	True	True	
0x822b9a10	wuauclt.exe	976	True	False	True	True	
0x81fa5390	wmiprvse.exe	1872	True	False	True	True	
0x1e0cda0	cmd.exe 968	False	True	False	False	2011-06-03 04:31:36+00:00	
0x1fb8da0	svchost.exe	856	False	True	False	False	
0x200eda0	jqs.exe 1580	False	True	False	False		
0x2061da0	svchost.exe	940	False	True	False	False	
0x21a5390	wmiprvse.exe	1872	False	True	False	False	
0x82279998	imapi.exe	756	True	False	True	True	
0x22ecc10	wscntfy.exe	2040	False	True	False	False	
0x2479998	imapi.exe	756	False	True	False	False	
0x820df020	smss.exe	376	True	False	True	False	
0x821a2da0	csrss.exe	600	True	False	True	False	
0x82073020	services.exe	668	True	False	True	True	
0x81e70020	lsass.exe	680	True	False	True	True	
0x81db8da0	svchost.exe	856	True	False	True	True	
0x81e61da0	svchost.exe	940	True	False	True	True	
0x81ff7020	svchost.exe	1200	True	False	True	True	
0x81e0eda0	jqs.exe 1580	True	False	True	True		
0x81e18b28	svchost.exe	1080	True	False	True	True	
0x8205ada0	alg.exe 188	True	False	True	True		
0x2018b28	svchost.exe	1080	False	True	False	False	
0x2070020	lsass.exe	680	False	True	False	False	
0x2086978	TSVNCache.exe	324	False	True	False	False	
0x22df020	smss.exe	376	False	True	False	False	
0x823c8830	System 4	True	False	True	False		
0x81fee8b0	spoolsv.exe	1412	True	False	True	True	
0x21ee8b0	spoolsv.exe	1412	False	True	False	False	

When we dump the target processes and query them with their hash information, we get some file information like:

```
PID    Process Start VPN      End VPN Tag    Protection     CommitCharge    PrivateMemory   File output    Notes   Hexdump Disasm

1928   lsass.exe      0x80000 0xf9fff Vad    PAGE_EXECUTE_READWRITE  0      0       pid.1928.vad.0x80000-0xf9fff.dmp       MZ header
4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00 ................
0x80000:       dec     ebp
0x80001:       pop     edx
0x80002:       nop
0x80003:       add     byte ptr [ebx], al
0x80005:       add     byte ptr [eax], al
0x80007:       add     byte ptr [eax + eax], al
0x8000a:       add     byte ptr [eax], al
1928   lsass.exe      0x1000000       0x1005fff      Vad    PAGE_EXECUTE_READWRITE  2      0       pid.1928.vad.0x1000000-0x1005fff.dmp    MZ header

4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 00 ................
0x1000000:     dec     ebp
0x1000001:     pop     edx
0x1000002:     nop
0x1000003:     add     byte ptr [ebx], al
0x1000005:     add     byte ptr [eax], al
0x1000007:     add     byte ptr [eax + eax], al
0x100000a:     add     byte ptr [eax], al
1928   lsass.exe      0x6f0000        0x769fff       Vad    PAGE_EXECUTE_READWRITE  0      0       pid.1928.vad.0x6f0000-0x769fff.dmp      N/A
29 87 7f ae 00 00 00 00 ff ff ff ff 77 35 00 01 )...........w5..
4b 00 45 00 52 00 4e 00 45 00 4c 00 33 00 32 00 K.E.R.N.E.L.3.2.
2e 00 44 00 4c 00 4c 00 2e 00 41 00 53 00 4c 00 ..D.L.L...A.S.L.
52 00 2e 00 30 00 33 00 36 00 30 00 62 00 37 00 R...0.3.6.0.b.7.
0x6f0000:      sub     dword ptr [edi + 0xae7f], eax
0x6f0006:      add     byte ptr [eax], al
1928   lsass.exe      0x680000        0x680fff       Vad    PAGE_EXECUTE_READWRITE  0      0       pid.1928.vad.0x680000-0x680fff.dmp      N/A
90 06 68 00 c6 07 68 00 24 00 68 00 a5 04 00 00 ..h...h.$.h.....
f2 04 68 00 48 06 00 00 c9 04 68 00 29 00 00 00 ..h.H.....h.)...
00 00 6f 00 e8 13 00 00 00 5a 77 4d 61 70 56 69 ..o......ZwMapVi
65 77 4f 66 53 65 63 74 69 6f 6e 00 5a 51 81 c1 ewOfSection.ZQ..
```

```
0x680000:      nop
0x680001:      push    es
0x680002:      push    0x6807c600
0x680007:      add     byte ptr [eax + eax], ah
0x68000a:      push    0x4a500
0x68000f:      add     dl, dh
0x680011:      add     al, 0x68
0x680013:      add     byte ptr [eax + 6], cl
0x680016:      add     byte ptr [eax], al
0x680018:      leave
0x680019:      add     al, 0x68
0x68001b:      add     byte ptr [ecx], ch
0x68001d:      add     byte ptr [eax], al
0x68001f:      add     byte ptr [eax], al
0x680021:      add     byte ptr [edi], ch
0x680024:      call    0x68003c
0x680029:      pop     edx
0x68002a:      ja      0x680079
0x68002c:      popal
0x68002d:      jo      0x680085
0x68002f:      imul    esp, dword ptr [ebp + 0x77], 0x6553664f
0x680036:      arpl    word ptr [ecx + ebp*2 + 0x6f], si
0x68003a:      outsb   dx, byte ptr [esi]
0x68003b:      add     byte ptr [edx + 0x51], bl
1928   lsass.exe      0x870000        0x9a7fff       Vad    PAGE_EXECUTE_READWRITE  0      0       pid.1928.vad.0x870000-0x9a7fff.dmp      MZ header

4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 08 01 00 00 ................
0x870000:      dec     ebp
0x870001:      pop     edx
0x870002:      nop
0x870003:      add     byte ptr [ebx], al
0x870005:      add     byte ptr [eax], al
0x870007:      add     byte ptr [eax + eax], al
0x87000a:      add     byte ptr [eax], al
```

And the dump files are created in memory

pid.1928.vad.0x6f0000-0x769fff.dmp	26-11-2025 21:17	DMP File	488 KB
pid.1928.vad.0x80000-0xf9fff.dmp	26-11-2025 21:17	DMP File	488 KB
pid.1928.vad.0x680000-0x680fff.dmp	26-11-2025 21:17	DMP File	4 KB
pid.1928.vad.0x870000-0x9a7fff.dmp	26-11-2025 21:17	DMP File	1,248 KB
pid.1928.vad.0x1000000-0x1005fff.dmp	26-11-2025 21:17	DMP File	24 KB

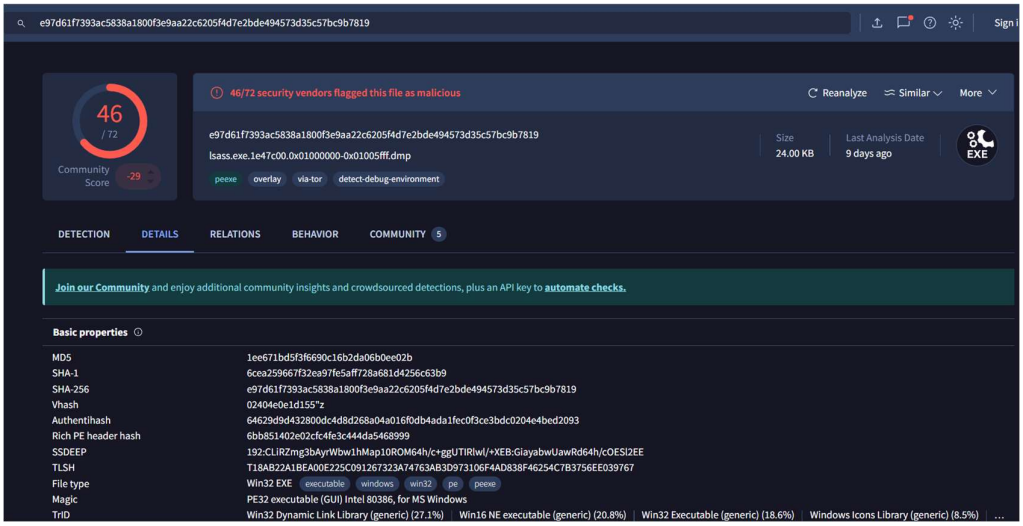
For further analysis, we need to get the file hash of the dump files

```
pid.1928.vad.0x1000000-0x1005fff.dmp  E97D61F7393AC5838A1800F3E9AA22C6205F4D7E2BDE494573D35C57BC9B7819
pid.1928.vad.0x680000-0x680fff.dmp  163B7DA37DF4AE6DAFBFB5BF88B319DABF7846CEE73D4192C6A7593E835857A8
pid.1928.vad.0x6f0000-0x769fff.dmp  ABCE3E79E26B5116FE7F3D40D21EAA4C8563E433B6086F9EC07C2925593F69DC
pid.1928.vad.0x80000-0xf9fff.dmp  2B2945F7CC7CF5B30CCDF37E2ADBB236594208E409133BCD56F57F7C009FFE6D
pid.1928.vad.0x870000-0x9a7fff.dmp  10F07B9FBBC6A8C6DC4ABF7A3D31A01E478ACCD115B33EC94FE885CB296A3586
```

File Hash Analysis

On verifying the file hash with the VirusTotal database, we get:

We can observe that almost all of the vendors have marked this hash as malicious and is marked as **“harmful”**



On analyzing the other file hases on VirusTotal we can observe similar results, with file hash being marked malicious.

Analysing using CAPA

Capa is an open-source static analysis tool developed by Mandiant's FLARE team for identifying capabilities within executable files. It is a valuable tool for malware analysis and reverse engineering, as it helps us to quickly understand what a program can do without requiring dynamic execution.

Before analyzing, we convert the process dump to into .exe format

```
Volatility Foundation Volatility Framework 2.6.1
/usr/local/lib/python2.7/dist-packages/volatility/plugins/community/YingLi/ssh_agent_key.py:12: CryptographyDeprecationWarning: Python 2 is
 no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.
  from cryptography.hazmat.backends.openssl import backend
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x81c47c00 0x01000000 lsass.exe            OK: executable.1928.exe
```

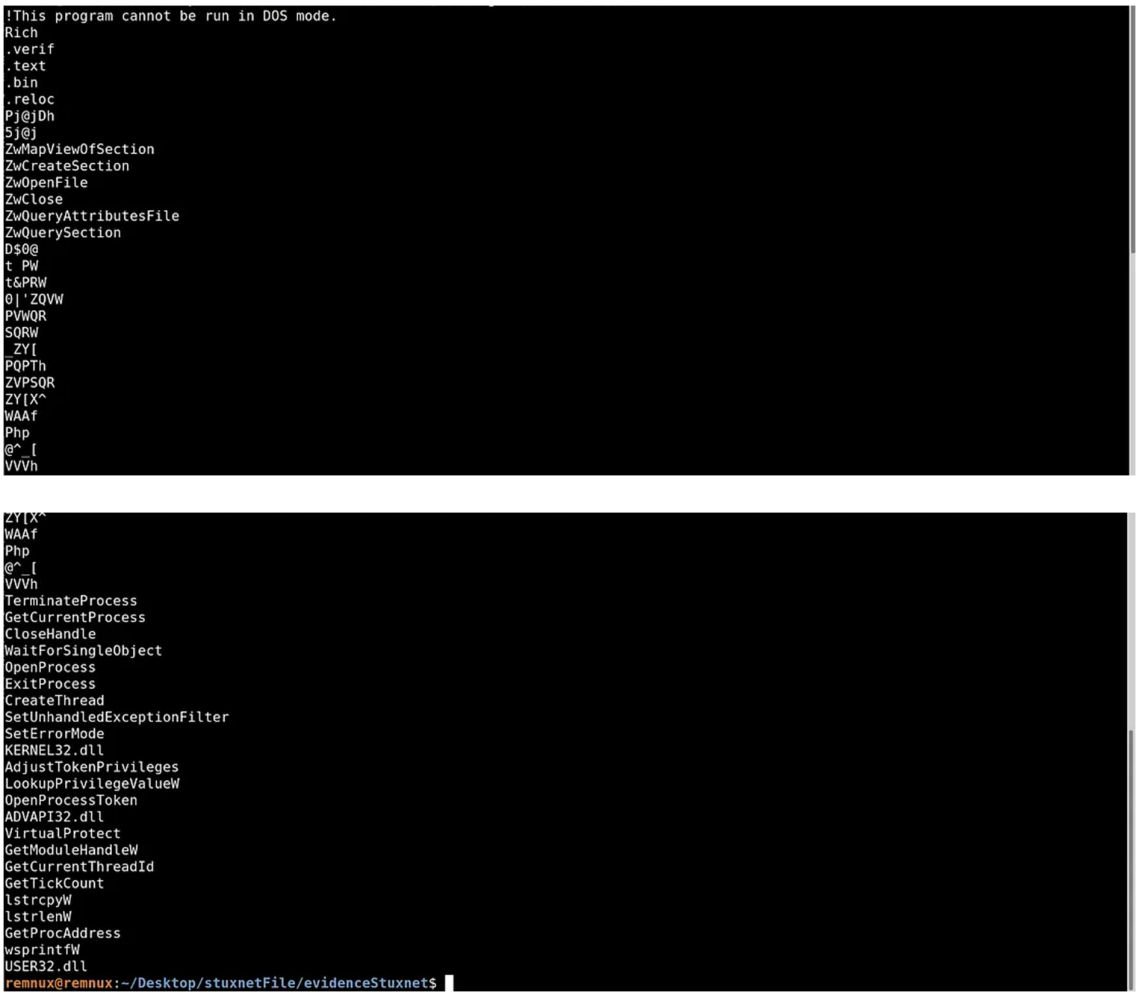
Analyzing executable file

```
loading : 100%|                                                                  | 661/661 [00:00<00:00, 5193.66 rules/s]
matching: 100%|          | 31/31 [00:00<00:00, 111.65 functions/s, skipped 1 library functions (3%)]
+-------------------------+--------------------------------------------------------------------------------------+
| md5                     | e1e00c2d5815e4129d8ac503f6fac095                                                     |
| sha1                    | 558f3dac7711b9b6209431916f1c455220df40a7                                             |
| sha256                  | 20a3c5f02b6b79bcac9adaef7ee138763054bbedc298fb2710b5adaf9b74a47d                     |
| os                      | windows                                                                              |
| format                  | pe                                                                                   |
| arch                    | i386                                                                                 |
| path                    | executable.1928.exe                                                                  |
+-------------------------+--------------------------------------------------------------------------------------+

+-------------------------+--------------------------------------------------------------------------------------+
| ATT&CK Tactic           | ATT&CK Technique                                                                     |
|-------------------------+--------------------------------------------------------------------------------------|
| DEFENSE EVASION         | Obfuscated Files or Information:: T1027                                              |
| EXECUTION               | Shared Modules:: T1129                                                               |
| PRIVILEGE ESCALATION    | Access Token Manipulation:: T1134                                                    |
+-------------------------+--------------------------------------------------------------------------------------+

+---------------------------+------------------------------------------------------------------------------+
| MBC Objective             | MBC Behavior                                                                 |
|---------------------------+------------------------------------------------------------------------------|
| DATA                      | Encode Data::XOR [C0026.002]                                                 |
| DEFENSE EVASION           | Obfuscated Files or Information::Encoding-Standard Algorithm [E1027.m02]     |
| DISCOVERY                 | Code Discovery::Enumerate PE Sections [B0046.001]                            |
| MEMORY                    | Allocate Memory:: [C0007]                                                    |
+---------------------------+------------------------------------------------------------------------------+
```

Now we have the ATT&CK technique numbers. They are T1027, T1129 and T1134. There are many sub-techniques and direct techniques here as this is a complex attack scenario.

Strings Extraction

The strings extraction from the .exe file would allow us to read the human understandable strings used in this file revealing as harmful or malicious API or suspicious Function Calls.



These are Windows API functions that the dumped process was either calling directly or referencing:

- TerminateProcess, GetCurrentProcess, CloseHandle, WaitForSingleObject, OpenProcess, ExitProcess, CreateThread, SetUnhandledExceptionFilter, SetErrorMode, AdjustTokenPrivileges, LookupPrivilegeValueW, OpenProcessToken, VirtualProtect, GetModuleHandleW, GetCurrentThreadId, GetTickCount, lstrcpyW, lstrlenW, GetProcAddress, wsprintfW.

Purpose of functions**:**

- TerminateProcess, GetCurrentProcess, CloseHandle, WaitForSingleObject, OpenProcess, ExitProcess, CreateThread: These functions are commonly used for process and thread management.
- SetUnhandledExceptionFilter, SetErrorMode: These functions are used for handling and controlling exceptions and errors in the application.
- AdjustTokenPrivileges, LookupPrivilegeValueW, OpenProcessToken: These functions are related to handling security tokens and privileges.
- VirtualProtect, GetModuleHandleW: These functions deal with memory protection and module handling.
- GetTickCount, lstrcpyW, lstrlenW, GetProcAddress, wsprintfW: These functions are related to various utility tasks such as string manipulation, getting system information, and dynamic function loading.

CreateThread:

- **Function**: Creates a new thread for concurrent execution within the process.
- **Relevance to Stuxnet**: Stuxnet may use CreateThread to execute its payload concurrently with other operations, allowing it to perform multiple tasks simultaneously, such as spreading across networks, gathering information, or performing sabotage actions.

VirtualProtect:

- **Function**: Changes the protection attributes of a region of virtual memory.
- **Relevance to Stuxnet**: Stuxnet might use VirtualProtect to modify memory permissions, potentially to inject code into processes or to evade detection by manipulating memory protection.

GetModuleHandleW and **GetProcAddress**:

- **Functions**: GetModuleHandleW retrieves the handle of a loaded module (DLL) specified by its name. GetProcAddress retrieves the address of an exported function or variable from a specified dynamic-link library (DLL).

- **Relevance to Stuxnet**: Stuxnet could use these functions to dynamically load additional modules or libraries, allowing it to extend its capabilities or to evade static analysis by hiding the import of malicious functions until runtime.

OpenProcess and **TerminateProcess**:

- **Functions**: OpenProcess opens an existing process object for manipulation, while TerminateProcess terminates a specified process.
- **Relevance to Stuxnet**: Stuxnet might use these functions to manipulate and terminate processes that interfere with its operation or to terminate security software that could detect or block its activities.

SetUnhandledExceptionFilter and **SetErrorMode**:

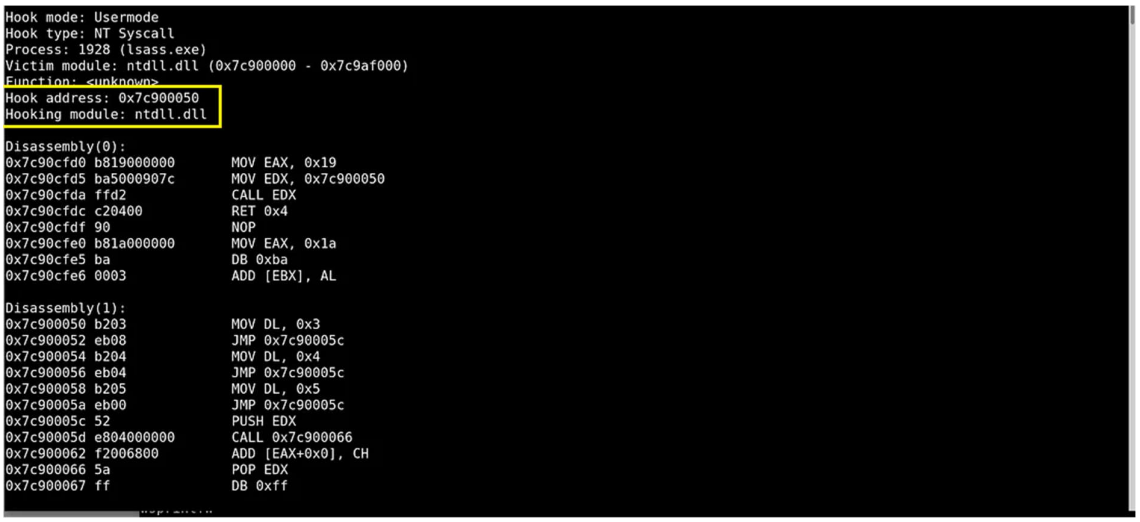
- **Functions**: SetUnhandledExceptionFilter sets a custom handler for unhandled exceptions, while SetErrorMode sets the error mode for the process, controlling how the system responds to serious errors.
- **Relevance to Stuxnet**: Stuxnet could use these functions to handle exceptions or errors gracefully to avoid crashing and revealing its presence, or to suppress error messages that could alert users or administrators.

AdjustTokenPrivileges and **OpenProcessToken**:

- **Functions**: AdjustTokenPrivileges adjusts the privileges of an access token, while OpenProcessToken opens the access token of a process.
- **Relevance to Stuxnet**: Stuxnet might use these functions to escalate its privileges, gain additional permissions, or perform actions that require elevated privileges, such as modifying system settings or accessing protected resources.

API hooks Analysis

API hooking is a technique used in software development and system-level programming to intercept and modify the behaviour of Application Programming Interfaces (APIs) provided by operating systems or third-party libraries.



ntdll.dll is a core system DLL that contains numerous low-level functions and system call implementations for the **Windows NT** kernel.

- **NT Syscall Hook**: This type of hook intercepts system calls (syscalls) made by user-mode applications before they reach the Windows kernel. It allows monitoring and potentially modifying the behavior of these calls.
- **Usermode Hook Mode**: Indicates that the hook operates within the user-mode context of the process (lsass.exe in this case), as opposed to kernel-mode. User-mode hooks are typically used for interception and manipulation of API calls and syscalls made by user-mode applications.

Stuxnet could hook into ntdll.dll to intercept system calls (syscalls) made by applications and processes. This interception could allow Stuxnet to monitor and potentially modify the behavior of system-level operations without directly accessing kernel-mode code. **Stuxnet** might intercept ntdll.dll functions related to file system access (NtCreateFile, NtReadFile, NtWriteFile) or process management (NtOpenProcess, NtTerminateProcess) to manipulate files, terminate processes, or evade detection.