Challenge: Volatility Traces Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Easy

Tools Used: Volatility 3

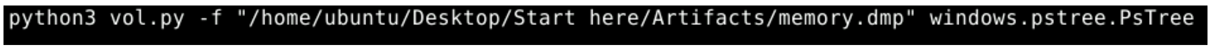
Summary: This lab involved using volatility 3 to analyse a Windows memory dump. It only required the use of a couple plugins, making it relatively beginner friendly. I really enjoyed the overall flow of this room, from identifying the malicious process, persistence mechanisms, and exploring defence evasion techniques. For those that enjoy doing memory forensics, I highly recommend giving this room a try.

Scenario: On May 2, 2024, a multinational corporation identified suspicious PowerShell processes on critical systems, indicating a potential malware infiltration. This activity poses a threat to sensitive data and operational integrity.

You have been provided with a memory dump (memory.dmp) from the affected system. Your task is to analyse the dump to trace the malware's actions, uncover its evasion techniques, and understand its persistence mechanisms.

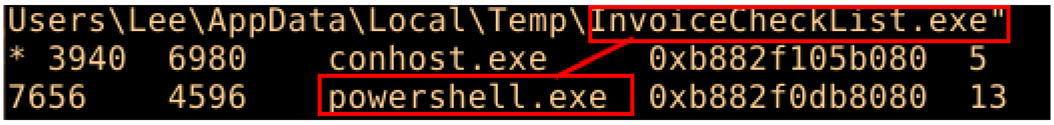
Identifying the parent process reveals the source and potential additional malicious activity. What is the name of the suspicious process that spawned two malicious PowerShell processes?

A useful volatility plugin that you should use for almost any situation, is pstree. As the name implies, pstree lists all the running processes and their parent-child relationships in a hierarchical fashion.



```
python3 vol.py -f "/home/ubuntu/Desktop/Start here/Artifacts/memory.dmp" windows.pstree.PsTree
```

If you look through the long output, you will eventually come across a process for “InvoiceCheckList.exe”:



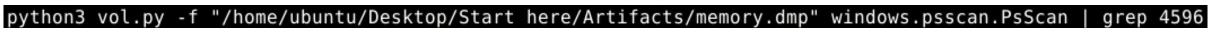
```
Users\Lee\AppData\Local\Temp\InvoiceCheckList.exe"
* 3940   6980     conhost.exe       0xb882f105b080   5
7656     4596     powershell.exe    0xb882f0db8080   13
```

Whilst a process spawning powershell.exe is not always malicious, based on the name of the executable that spawned PowerShell, we can assume to a high degree of certainty that this is malicious.

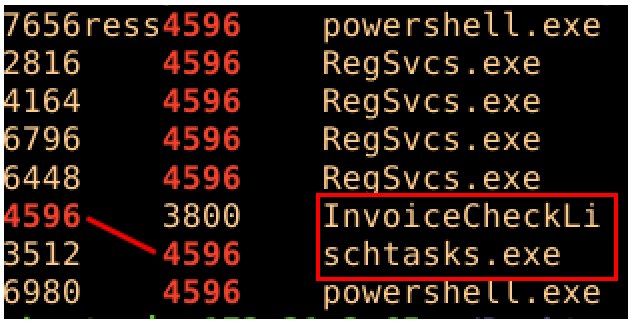
Answer: InvoiceCheckList.exe

By determining which executable is utilized by the malware to ensure its persistence, we can strategize for the eradication phase. Which executable is responsible for the malware's persistence?

We can utilise a plugin called psscan, which is capable of scanning for hidden processes. Recall, the PID of the malicious executable (InvoiceCheckList.exe) is 4596.



```
python3 vol.py -f "/home/ubuntu/Desktop/Start here/Artifacts/memory.dmp" windows.psscan.PsScan | grep 4596
```



```
7656ress4596    powershell.exe
2816    4596    RegSvcs.exe
4164    4596    RegSvcs.exe
6796    4596    RegSvcs.exe
6448    4596    RegSvcs.exe
4596    3800    InvoiceCheckLi
3512    4596    schtasks.exe
6980    4596    powershell.exe
```

The presence of schtasks.exe (Scheduled Tasks) suggests that a scheduled task was created. Scheduled Tasks are often abused by malware for persistence.

Answer: schtasks.exe

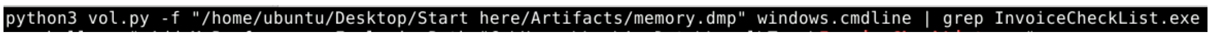
Understanding child processes reveals potential malicious behavior in incidents. Aside from the PowerShell processes, what other active suspicious process, originating from the same parent process, is identified?

In the output of the PsScan command, we can see that there are three processes originating from the same parent process 4596. Those being powershell.exe, schtasks.exe, and RegSvcs.exe. RegSvcs.exe is a command-line utility that is used to register .NET COM assemblies.

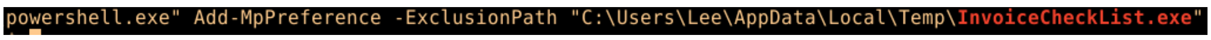
Answer: RegSvcs.exe

Analyzing malicious process parameters uncovers intentions like defense evasion for hidden, stealthy malware. What PowerShell cmdlet used by the malware for defense evasion?

Another helpful plugin is windows.cmdline, it lists the command line arguments of a process. These are parameters passed to a program when it gets executed.



```
python3 vol.py -f "/home/ubuntu/Desktop/Start here/Artifacts/memory.dmp" windows.cmdline | grep InvoiceCheckList.exe
```



```
powershell.exe" Add-MpPreference -ExclusionPath "C:\Users\Lee\AppData\Local\Temp\InvoiceCheckList.exe"
```

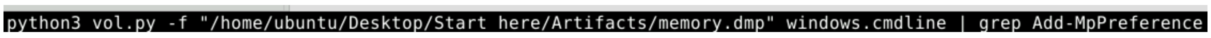
The command seen in the image above adds a defender exclusion for “InvoiceCheckList.exe”, meaning that Windows Defender will no longer scan or block that executable.

- **Add-MpPreference** is a PowerShell cmdlet used to modify settings for Microsoft Defender.
- **-ExclusionPath** ads a folder or file path to the list of paths excluded from scanning.

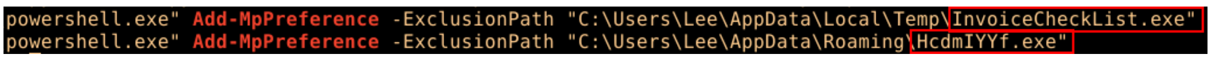
Answer: Add-MpPreference

Recognizing detection-evasive executables is crucial for monitoring their harmful and malicious system activities. Which two applications were excluded by the malware from the previously altered application's settings?

To find other programs that are added to the exclusion list, we can grep the output of the windows.cmdline plugin and search for the “Add-MpPreference” cmdlet:



```
python3 vol.py -f "/home/ubuntu/Desktop/Start here/Artifacts/memory.dmp" windows.cmdline | grep Add-MpPreference
```



```
powershell.exe" Add-MpPreference -ExclusionPath "C:\Users\Lee\AppData\Local\Temp\InvoiceCheckList.exe"
powershell.exe" Add-MpPreference -ExclusionPath "C:\Users\Lee\AppData\Roaming\HcdmIYYf.exe"
```

As you can see in the above image, two executables have been added to the exclusion list.

Answer: HcdmIYYf.exe, InvoiceCheckList.exe

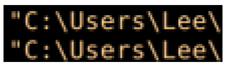
What is the specific MITRE sub-technique ID associated with PowerShell commands that aim to disable or modify antivirus settings to evade detection during incident analysis?

Recall, the PowerShell commands from earlier add the malicious executables to the Windows Defender exclusion list, ensuring that Windows Defender will not prevent it from executing. This is known as Impair Defenses: Disable or Modify Tools (T1562.001) as it modifies the Windows Defender exclusion list.

Answer: T1562.001

Determining the user account offers valuable information about its privileges, whether it is domain-based or local, and its potential involvement in malicious activities. Which user account is linked to the malicious processes?

If you look at the file path of the malicious executables, we can see that it is executed within the temp directory of user “Lee”:



```
"C:\Users\Lee\
"C:\Users\Lee\
```

Alternatively, you can use the windows.getsids plugin to identify the user associated with a specific process.

Answer: Lee