Challenge: DarkCrystal Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: EvtxECmd, Timeline Explorer, Volatility 3

Summary: This lab involved investigating a compromised machine through analysing Windows event logs and a memory dump. The primary tools used were EvtxECmd, Volatility 3, and timeline explorer. I found it relatively enjoyable, and learnt some new techniques, including parsing the entire event logs directory at once (I originally thought EvtxECmd could only parse one log file at a time). For those of you who enjoy memory forensics and analysing event logs, you should give this a go.

Scenario: As a member of the DFIR team, you are assigned to investigate a security incident involving suspicious remote activity. Alerts have flagged unauthorized access attempts to internal systems and unusual outbound connections to unknown servers.
Your mission is to identify the source of the activity, determine any malicious processes involved, assess the scope of the impact, and provide recommendations to guide the response strategy.

Malware frequently disguises itself by using names similar to legitimate executables to avoid detection. What is the filename of the malicious executable that serves as the origin and initiates the chain of malicious activities?

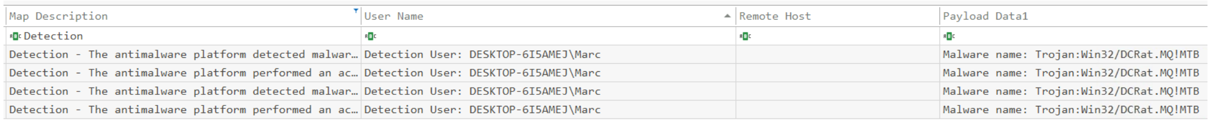
To determine if Windows Defender detected any malware on the system, we can start by parsing the Windows Defender evtx file located at:

- `C:\Users\Administrator\Desktop\Start Here\Artifacts\Logs\ Microsoft-Windows-Windows Defender_4Operational.evtx`

I like to use EvtxECmd to parse event logs, and view the output using Timeline Explorer:

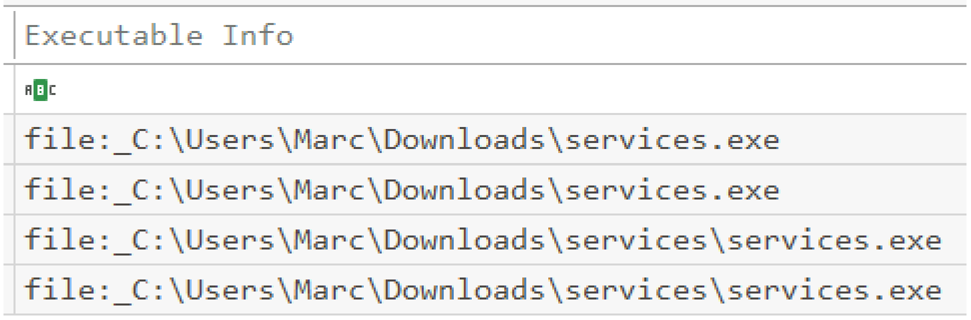
- `.\EvtxECmd.exe -f ".\Microsoft-Windows-Windows Defender_4Operational.evtx" --csv . --csvf defender_out.csv`

If you filter for event ID 1116 (malware detected) or the keyword "Detection" within the Map Description field, you can see that Windows Defender detected a trojan tagged as DCRat on DESKTOP-6I5AMEJ\Marc:



Map Description	User Name	Remote Host	Payload Data1
Detection			
Detection - The antimalware platform detected malwar…	Detection User: DESKTOP-6I5AMEJ\Marc		Malware name: Trojan:Win32/DCRat.MQ!MTB
Detection - The antimalware platform performed an ac…	Detection User: DESKTOP-6I5AMEJ\Marc		Malware name: Trojan:Win32/DCRat.MQ!MTB
Detection - The antimalware platform detected malwar…	Detection User: DESKTOP-6I5AMEJ\Marc		Malware name: Trojan:Win32/DCRat.MQ!MTB
Detection - The antimalware platform performed an ac…	Detection User: DESKTOP-6I5AMEJ\Marc		Malware name: Trojan:Win32/DCRat.MQ!MTB

If you scroll over to the Executable Info column, we can see the file that was detected as malware:



Executable Info
file:_C:\Users\Marc\Downloads\services.exe
file:_C:\Users\Marc\Downloads\services.exe
file:_C:\Users\Marc\Downloads\services\services.exe
file:_C:\Users\Marc\Downloads\services\services.exe

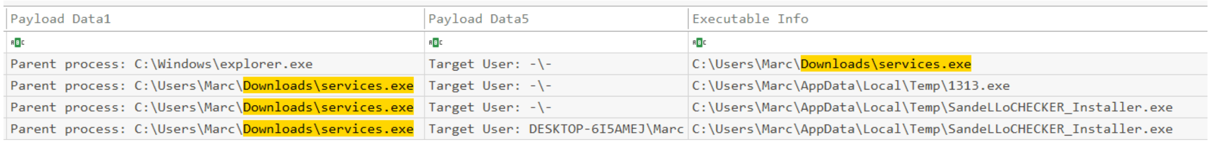
Answer: services.exe

Identifying the initial file dropped by the malware is essential for understanding its execution flow. What is the name of the first file created by the malicious executable identified in the previous question?

Unfortunately, Sysmon logs aren't available, so we won't be able to look for event ID 11 (file create) logs. We can use EvtxECmd to parse the entire logs directory and look for events related to services.exe:

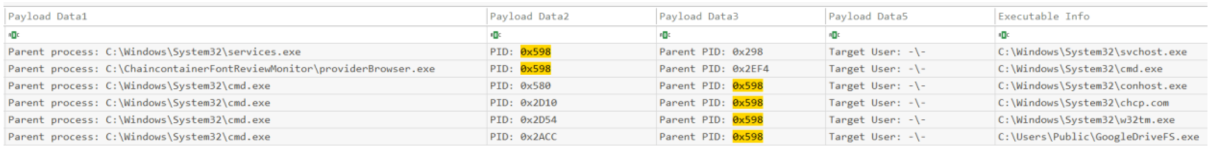
- `.\EvtxECmd.exe -d "C:\Users\Administrator\Desktop\Start Here\Artifacts\Logs" --csv . --csvf all_out.csv`

Let's now filter for Downloads\services.exe within Timeline Explorer. There are four results, all for event ID 4688 (process create):



Payload Data1	Payload Data5	Executable Info
Parent process: C:\Windows\explorer.exe	Target User: -\-	C:\Users\Marc\Downloads\services.exe
Parent process: C:\Users\Marc\Downloads\services.exe	Target User: -\-	C:\Users\Marc\AppData\Local\Temp\1313.exe
Parent process: C:\Users\Marc\Downloads\services.exe	Target User: -\-	C:\Users\Marc\AppData\Local\Temp\SandeLLoCHECKER_Installer.exe
Parent process: C:\Users\Marc\Downloads\services.exe	Target User: DESKTOP-6I5AMEJ\Marc	C:\Users\Marc\AppData\Local\Temp\SandeLLoCHECKER_Installer.exe

From this, we can determine that services.exe was executed from the file explorer, suggesting that the user manually executed it. services.exe then proceeded to execute 1313.exe and SandeLLoCHECKER_Installer.exe. The presence of both these executables being in the Temp directory, along with them being executed by services.exe is extremely suspicious. To find all processes spawned by services.exe, we can filter for the PID of services.exe like as follows:

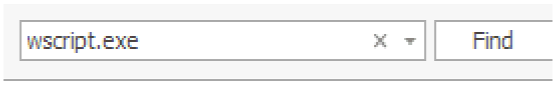


Payload Data1	Payload Data2	Payload Data3	Payload Data5	Executable Info
Parent process: C:\Windows\System32\services.exe	PID: 0x598	Parent PID: 0x298	Target User: -\-	C:\Windows\System32\svchost.exe
Parent process: C:\ChaincontainerFontReviewMonitor\providerBrowser.exe	PID: 0x598	Parent PID: 0x2EF4	Target User: -\-	C:\Windows\System32\cmd.exe
Parent process: C:\Windows\System32\cmd.exe	PID: 0x580	Parent PID: 0x598	Target User: -\-	C:\Windows\System32\conhost.exe
Parent process: C:\Windows\System32\cmd.exe	PID: 0x2D10	Parent PID: 0x598	Target User: -\-	C:\Windows\System32\chcp.com
Parent process: C:\Windows\System32\cmd.exe	PID: 0x2D54	Parent PID: 0x598	Target User: -\-	C:\Windows\System32\w32tm.exe
Parent process: C:\Windows\System32\cmd.exe	PID: 0x2ACC	Parent PID: 0x598	Target User: -\-	C:\Users\Public\GoogleDriveFS.exe

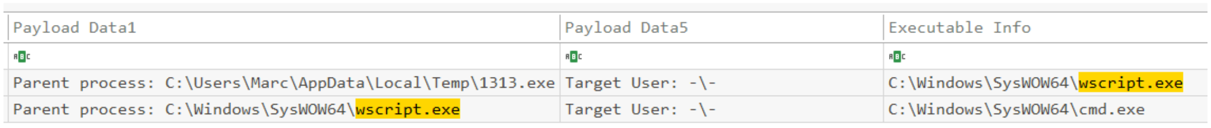
Answer: 1313.exe

Understanding how malware executes malicious scripts is critical for analyzing its workflow. If the malware leverages wscript to execute scripts, what executable file was launched as a result of their execution?

Start by filtering for wscript.exe:

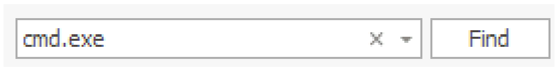


For context, wscript.exe is a legitimate Windows binary that enables you to execute scripts written in languages like VBScript and JScript. There are two results:

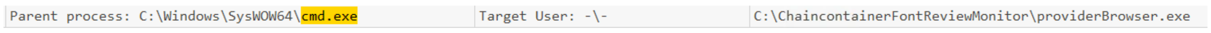


Payload Data1	Payload Data5	Executable Info
Parent process: C:\Users\Marc\AppData\Local\Temp\1313.exe	Target User: -\-	C:\Windows\SysWOW64\wscript.exe
Parent process: C:\Windows\SysWOW64\wscript.exe	Target User: -\-	C:\Windows\SysWOW64\cmd.exe

We can see that at 2024-12-05 15:45:14 1313.exe executed wscript.exe, and at 2024-12-05 15:45:32 wscript then spawned cmd.exe. Let's pivot from cmd.exe to see what it executes:



At 2024-12-05 15:45:32, cmd.exe was observed executing providerBrowser.exe:



Parent process: C:\Windows\SysWOW64\cmd.exe	Target User: -\-	C:\ChaincontainerFontReviewMonitor\providerBrowser.exe

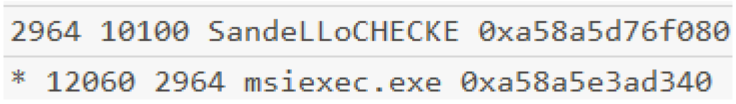
Answer: providerBrowser.exe

Malware often exploits legitimate system processes, known as Living Off the Land Binaries (LOLBins), to evade detection. What is the name of the first file executed by the malware using a LOLBin?

To find what LOLBins are being used by the threat actor, we can use Volatility along with the pstree plugin to visualise process lineage (you can find the memory dump within the Artifacts folder):

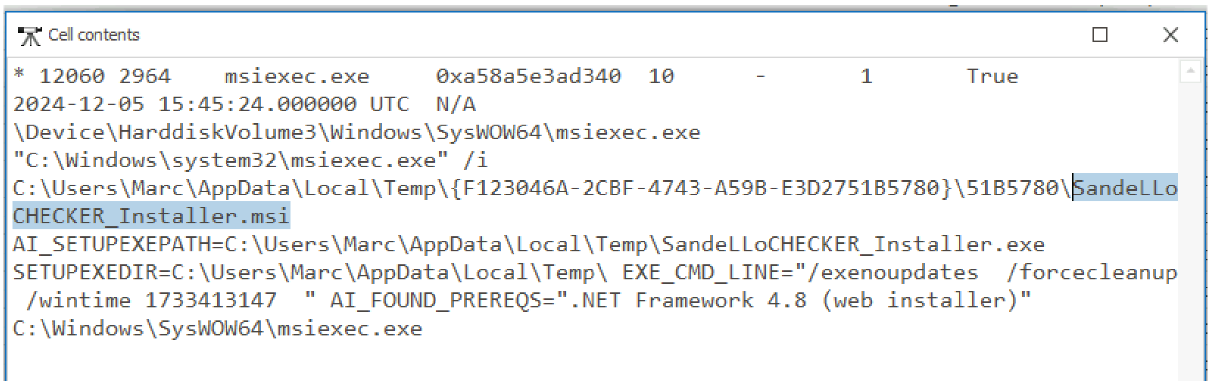
- `python .\vol.py -f .\memory.dmp windows.pstree > pstree_out.csv`

If you explore the output in Timeline Explorer, you can see that SandeLLoCHECKER.exe (we identified this binary being executed by 1313.exe earlier) spawning msiexec.exe:



```
2964 10100 SandeLLoCHECKE 0xa58a5d76f080
* 12060 2964 msiexec.exe 0xa58a5e3ad340
```

Msiexec.exe is used by Windows to execute msi files. In this case, it was used to execute an msi file called SandeLLoCHECKER_Installer.msi:

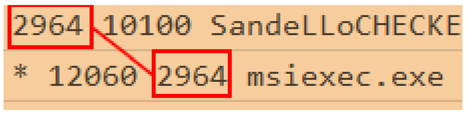

Cell contents

```
* 12060 2964    msiexec.exe     0xa58a5e3ad340  10      -       1       True
2024-12-05 15:45:24.000000 UTC  N/A
\Device\HarddiskVolume3\Windows\SysWOW64\msiexec.exe
"C:\Windows\system32\msiexec.exe" /i
C:\Users\Marc\AppData\Local\Temp\{F123046A-2CBF-4743-A59B-E3D2751B5780}\51B5780\SandeLLo
CHECKER_Installer.msi
AI_SETUPEXEPATH=C:\Users\Marc\AppData\Local\Temp\SandeLLoCHECKER_Installer.exe
SETUPEXEDIR=C:\Users\Marc\AppData\Local\Temp\ EXE_CMD_LINE="/exenoupdates  /forcecleanup
 /wintime 1733413147  " AI_FOUND_PREREQS=".NET Framework 4.8 (web installer)"
C:\Windows\SysWOW64\msiexec.exe
```

Answer: SandeLLoCHECKER_Installer.msi

Malware frequently renames system utilities to bypass security defenses. What is the PID of the parent process linked to the legitimate LOLBin mentioned in the previous question?

The PID of the parent process is 2964:



```
2964 10100 SandeLLoCHECKE
* 12060 2964 msiexec.exe
```

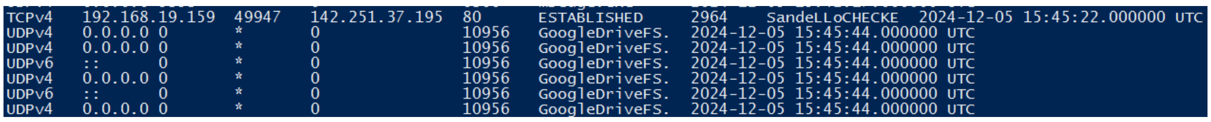
Answer: 2964

Remote Access Trojans (RATs) depend on command-and-control (C&C) channels to establish communication with attackers, enabling them to issue commands and carry out malicious operations. Which executable is responsible for facilitating the malware's C&C activities?

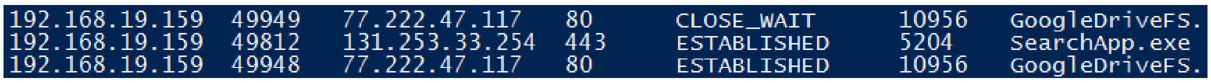
We can use the netscan plugin to identify any network communications that were made by a process:

- `python .\vol.py -f .\memory.dmp windows.netscan`

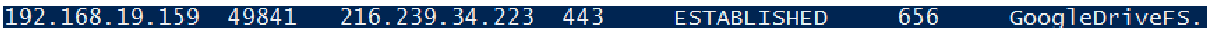
Up until this point we know there are multiple malicious binaries, including services.exe, 1313.exe, SandeLLoCHECKER_Installer.exe, msiexec.exe, providerBrowser.exe, and GoogleDriveFS.exe.



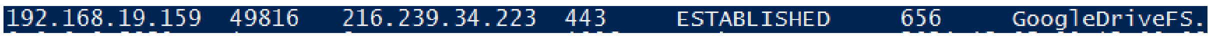
```
TCPv4   192.168.19.159  49947   142.251.37.195  80      ESTABLISHED     2964    SandeLLoCHECKE  2024-12-05 15:45:22.000000 UTC
UDPv4   0.0.0.0 0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
UDPv4   0.0.0.0 0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
UDPv6   ::      0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
UDPv4   0.0.0.0 0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
UDPv6   ::      0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
UDPv4   0.0.0.0 0       *       0               10956   GoogleDriveFS.  2024-12-05 15:45:44.000000 UTC
```



```
192.168.19.159  49949   77.222.47.117   80      CLOSE_WAIT      10956   GoogleDriveFS.
192.168.19.159  49812   131.253.33.254  443     ESTABLISHED     5204    SearchApp.exe
192.168.19.159  49948   77.222.47.117   80      ESTABLISHED     10956   GoogleDriveFS.
```



```
192.168.19.159  49841   216.239.34.223  443     ESTABLISHED     656     GoogleDriveFS.
```



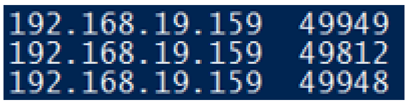
```
192.168.19.159  49816   216.239.34.223  443     ESTABLISHED     656     GoogleDriveFS.
```

We can see that SandeLLoCHECKER_Installer.exe established a connection to 142.251.37.195 on port 80. GoogleDriveFS.exe has multiple connections to 77.222.47.117 and 216.239.34.224. The filename suggests that this executable is trying to impersonate some sort of legitimate Google Drive binary, likely to try and avoid detection.

Answer: GoogleDriveFS.exe

Identifying the network communication used by malware is crucial for detecting its presence and understanding its command-and-control (C&C) infrastructure. What are the two local ports the malware uses to connect to its C&C servers?

In the output of the netscan plugin, we determined that the GoogleDriveFS.exe binary makes connections to the C2 infrastructure from local port 49948 and 49949:



```
192.168.19.159  49949
192.168.19.159  49812
192.168.19.159  49948
```

Answer: 49948, 49949

The malware likely retrieves a script from a remote server to execute commands or exfiltrate data. What is the name of the script retrieved in this case?

Let’s start by dumping the process memory associated with GoogleDriveFS.exe (PID 10956):

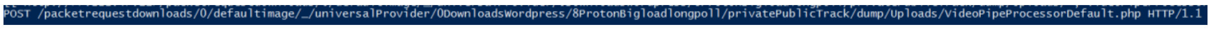
- `python .\vol.py -f .\memory.dmp windows.memmap --pid 10956 --dump`

We can then execute strings against this process dump and look for any requests being made to the C2 server 77.222.47.117:

- `strings .\pid.10956.dmp | Select-String "77.222.47.117"`

In the output, I can see something being uploaded, the file appears to be called VideoPipeProcessorDefault, let’s search for this:

- `strings .\pid.10956.dmp | Select-String "VideoPipeProcessorDefault"`



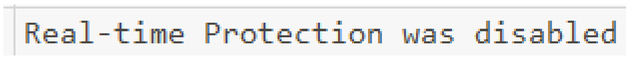
```
POST /packetrequestdownloads/0/defaultimage/_/universalProvider/0DownloadsWordpress/8ProtonBigloadlongpoll/privatePublicTrack/dump/Uploads/VideoPipeProcessorDefault.php HTTP/1.1
```

We can see a post request being made that uploads VideoPipeProcessorDefault.php to the C2 server.

Answer: VideoPipeProcessorDefault.php

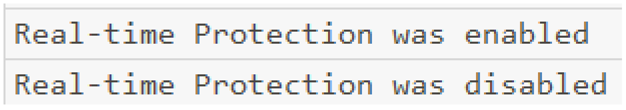
Windows Defender initially detected the malware but was later disabled, allowing the malware to run again the next day. What detection name did Windows Defender assign to this malware sample?

Going back to the Defender logs we parsed earlier, at 2024-12-04 16:00:16 Windows Defender real time monitoring was disabled:



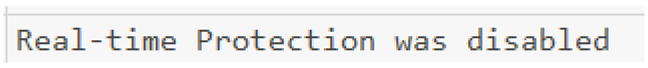
```
Real-time Protection was disabled
```

At 2024-12-04 16:02:04 it was then reenabled, followed by being disabled at 2024-12-04 16:02:43:



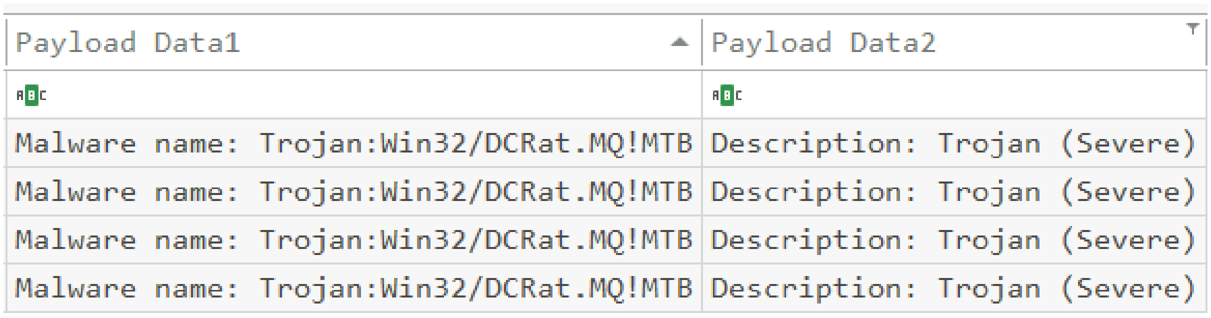
```
Real-time Protection was enabled
Real-time Protection was disabled
```

This pattern continued until 2024-12-05 09:13:39 where it was disabled and not reenabled:



```
Real-time Protection was disabled
```

If you focus on detection events, we can see that the malware was given the detection name Trojan:Win32/DCRat.MQ!MTB.



Payload Data1	Payload Data2
Malware name: Trojan:Win32/DCRat.MQ!MTB	Description: Trojan (Severe)
Malware name: Trojan:Win32/DCRat.MQ!MTB	Description: Trojan (Severe)
Malware name: Trojan:Win32/DCRat.MQ!MTB	Description: Trojan (Severe)
Malware name: Trojan:Win32/DCRat.MQ!MTB	Description: Trojan (Severe)

Answer: Trojan:Win32/DCRat.MQ!MTB