Security Community Service

Detektion von Code-Injection Angriffen

Datum:
06.06.2017

Klassifikation:
TLP Grün

IOprotect GmbH
Dürstelenstrasse 136
8335 Hittnau
+41 (0)44 533 00 05
info@ioprotect.ch
www.ioprotect.ch

1. Inhaltsverzeichnis

Einführung

Indicator of Compromise (IOCs) sind derzeit in aller Munde. Mehr und mehr Firmen abonnieren IOC-Feeds, um mit der Menge an neu auftauchenden Malware-Samples Schritt halten und so infizierte Systeme rasch detektieren zu können. Aus Sicht von IOprotect haben IOCs durchaus ihren Wert, vor allem wenn es um Malware-Hunting im Unternehmen bei konkreten Fällen geht. Allerdings sind IOCs wenig nachhaltig. IOprotects bevorzugte Variante ist es vielmehr, den Modus Operandi der Angreifer zu verstehen und so nicht nach konkreter Malware zu suchen, sondern vielmehr deren auffälliges Verhalten zu detektieren. Das vorliegende Dokument soll genau ein solches Verhalten von Malware aufdecken: Code-Injection.

Im Dokument wird aufgezeigt, wie mit dieser Methode unterschiedlichste Formen von Angriffstools und Malware erkannt werden kann. Dies sind unter anderem folgende:

- PowerShell Empire
- Meterpreter via Migrate
- Process Hollowing
- WannaCry

1 Monitoring Use Case via Get-InjectedThread

Für die Umsetzung des Monitoring Use Cases wird ein PowerShell-Skript von Jared Atkinson (Get-InjectedThread.ps1) genutzt. Dieser Code stammt von der Präsentation "Taking Hunting to the Next Level: Hunting in Memory" gehalten von Jared Atkinson und Joe Desimone am SANS Threat Hunting Summit 2017. Der Quellcode ist unter folgendem Link zu finden: https://gist.github.com/jaredcatkinson/23905d34537ce4b5b1818c3e6405c1d2

Die Erklärung zum Skript[1]:

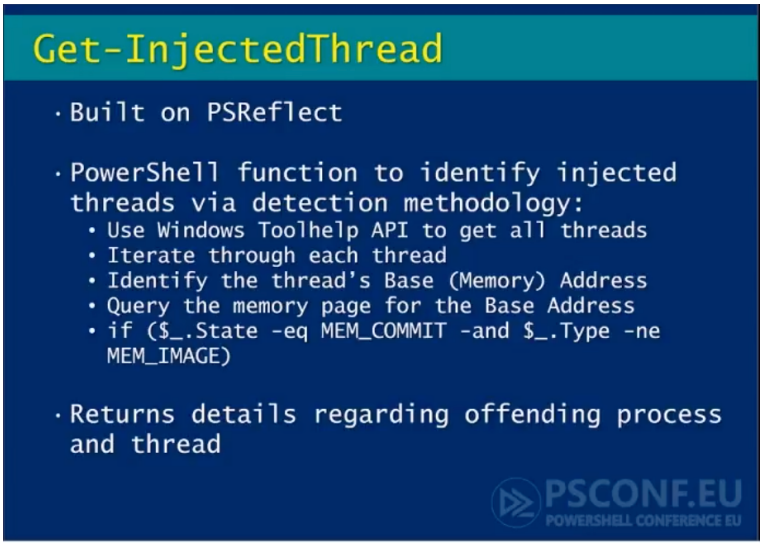

Abbildung 1: Erklärung zum PS-Skript

Mit dem Skript sollten damit klassische Code-Injection Techniken erkannt werden, auch wenn der Autor darauf hinweist, dass es keine Allheilmittel ist. Anhand von fünf Fallbeispielen wird die Effektivität des Monitoring Use Cases ermittelt:

1. WannaCry Ransomware
2. ReflectivePEInjection via PowerShell
3. PowerShell Empire
4. Metasploit Meterpreter
5. Ransomware mit EoP und Process Hollowing

[1] https://www.youtube.com/watch?v=Hhpi3Sp4W4k

2 Fallbeispiele

2.1 WannaCry-Infektion

In diesem Fallbeispiel wurden zwei VMware-Systeme genutzt:

- Ein Zielsystem, auf welchem WannaCry ausgeführt wurde.
- Ein Logsystem, auf welchem das Skript Get-InjectedThread.ps1 via PS-Remoting und mit lokalen Administratorenrechten auf dem Zielsystem ausgeführt wurde. Die Ausführung erfolgte manuell und direkt mehrfach hintereinander.

Anbei sind zwei Printscreens des Setups aufgeführt:

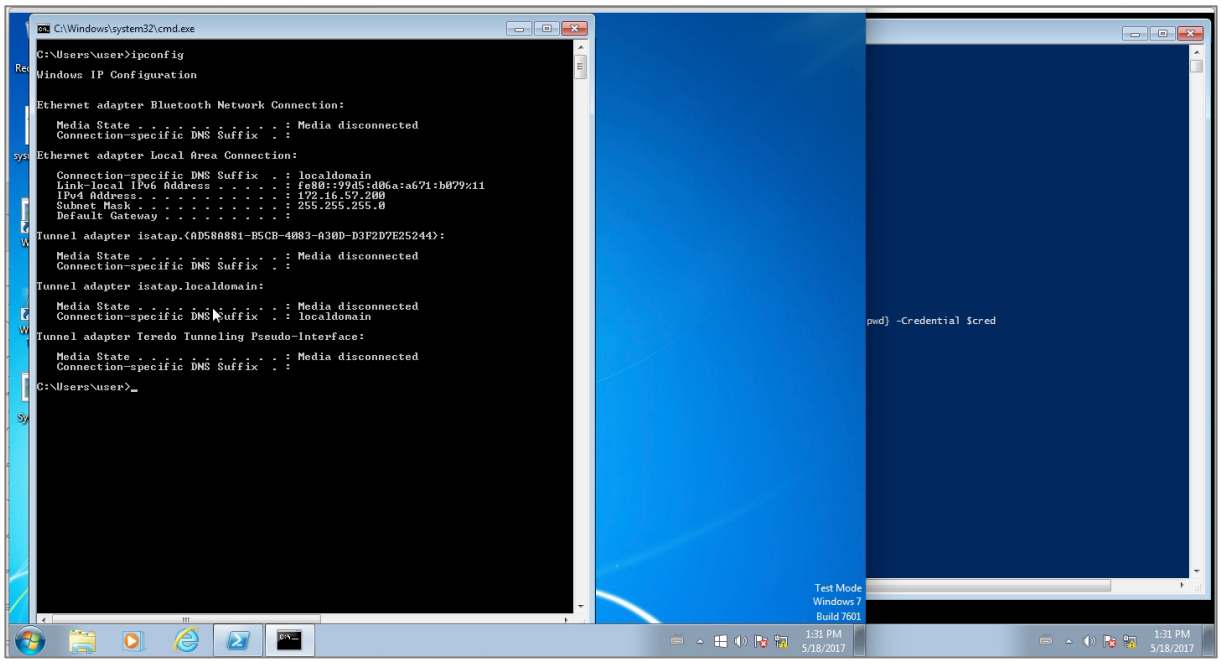

Abbildung 2: Links das Zielsystem, rechts im Hintergrund das Logsystem

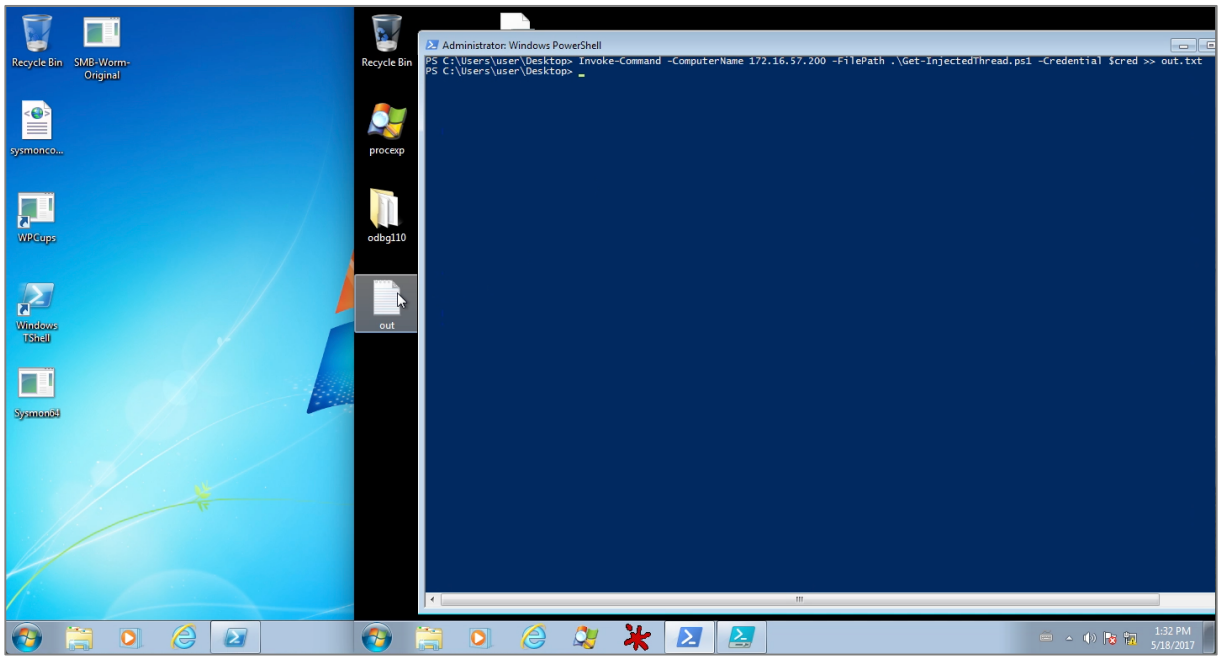

Abbildung 3: Das Logsystem ist im Vordergrund, mit dem auszuführenden PS-Remoting-Befehl

Unmittelbar nachdem die Malware auf dem Zielsystem lief, wurde der PS-Remoting-Befehle manuell und mehrfach hintereinander ausgeführt. Die Resultate sind automatisch in die Datei *out.txt* angefügt worden. Unten sind die Ergebnisse der ersten drei Läufe zu sehen:

```
ProcessName               : tasksche.exe
ProcessId                 : 2564
Path                      : C:\ProgramData\dqhtorqn403\tasksche.exe
KernelPath                : C:\ProgramData\dqhtorqn403\tasksche.exe
CommandLine               : C:\ProgramData\dqhtorqn403\tasksche.exe
PathMismatch              : False
ThreadId                  : 3360
AllocatedMemoryProtection : PAGE_READWRITE
MemoryProtection          : PAGE_EXECUTE_READ
MemoryState               : MEM_COMMIT
MemoryType                : MEM_PRIVATE
BasePriority              : 8
IsUniqueThreadToken       : False
Integrity                 :
Privilege                 :
LogonId                   :
SecurityIdentifier        :
UserName                  : \
LogonSessionStartTime     :
LogonType                 :
AuthenticationPackage     :
BaseAddress               : 268453776
Size                      : 12288
Bytes                     : {161, 144, 221, 0...}
PSComputerName            : 172.16.57.200
RunspaceId                : 9cb7ee61-5fcd-4c0a-b7d7-168b21f88370

ProcessName               : tasksche.exe
ProcessId                 : 2564
```

```
Path                        : C:\ProgramData\dqhtorqn403\tasksche.exe
KernelPath                  : C:\ProgramData\dqhtorqn403\tasksche.exe
CommandLine                 : C:\ProgramData\dqhtorqn403\tasksche.exe
PathMismatch                : False
ThreadId                    : 2516
AllocatedMemoryProtection   : PAGE_READWRITE
MemoryProtection            : PAGE_EXECUTE_READ
MemoryState                 : MEM_COMMIT
MemoryType                  : MEM_PRIVATE
BasePriority                : 8
IsUniqueThreadToken         : False
Integrity                   :
Privilege                   :
LogonId                     :
SecurityIdentifier          :
UserName                    : \
LogonSessionStartTime       :
LogonType                   :
AuthenticationPackage       :
BaseAddress                 : 268453312
Size                        : 12288
Bytes                       : {86, 139, 116, 36...}
PSComputerName              : 172.16.57.200
RunspaceId                  : 9cb7ee61-5fcd-4c0a-b7d7-168b21f88370

ProcessName                 : tasksche.exe
ProcessId                   : 2564
Path                        : C:\ProgramData\dqhtorqn403\tasksche.exe
KernelPath                  : C:\ProgramData\dqhtorqn403\tasksche.exe
CommandLine                 : C:\ProgramData\dqhtorqn403\tasksche.exe
PathMismatch                : False
ThreadId                    : 2176
AllocatedMemoryProtection   : PAGE_READWRITE
MemoryProtection            : PAGE_EXECUTE_READ
MemoryState                 : MEM_COMMIT
MemoryType                  : MEM_PRIVATE
BasePriority                : 8
IsUniqueThreadToken         : False
Integrity                   :
Privilege                   :
LogonId                     :
SecurityIdentifier          :
UserName                    : \
LogonSessionStartTime       :
LogonType                   :
AuthenticationPackage       :
BaseAddress                 : 268457776
Size                        : 8192
Bytes                       : {83, 85, 86, 87...}
PSComputerName              : 172.16.57.200
RunspaceId                  : 9cb7ee61-5fcd-4c0a-b7d7-168b21f88370
```

Es wird immer derselbe Prozess angezeigt, aber die Aktionen von WannaCry oder zumindest einige davon sind via diesem Use Case detektierbar.

2.2 ReflectivePEInjection via PowerShell

Für das Fallbeispiel wird das Skript Invoke-ReflectivePEInjection.ps1[2] genommen. Dieses ermöglicht es einem Angreifer, eine ausführbare Datei in den PowerShell-Prozess zu laden. Das Skript wurde wie folgt ausgeführt:

```
$PEBytes = [IO.File]::ReadAllBytes('C:\Users\user\Desktop\mw.exe')
Invoke-ReflectivePEInjection -PEBytes $PEBytes -ExeArgs "Arg1 Arg2 Arg3 Arg4" -
ForceASLR
```

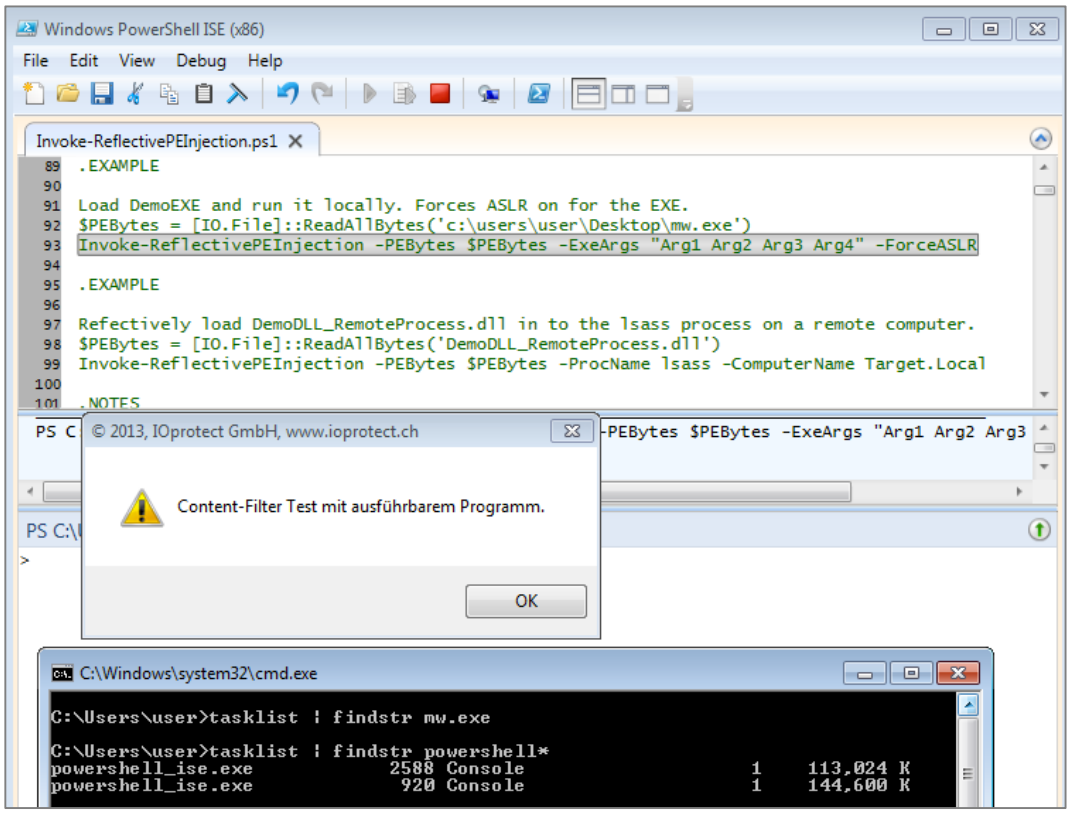

Abbildung 4: Die Datei mw.exe wird via Invoke-ReflectivePEInjection in den PowerShell-Prozess geladen

Ganz unten in der Abbildung ist zu sehen, dass kein Prozess mit dem Namen *mw.exe* existiert. Die PIDs der beiden laufenden PowerShell-Prozesse sind ebenfalls zu sehen.

[2] https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

Im zweiten PowerShell_ISE.exe wird Get-InjectedThread.ps1 ausgeführt. Dabei zeigt sich folgendes Bild:

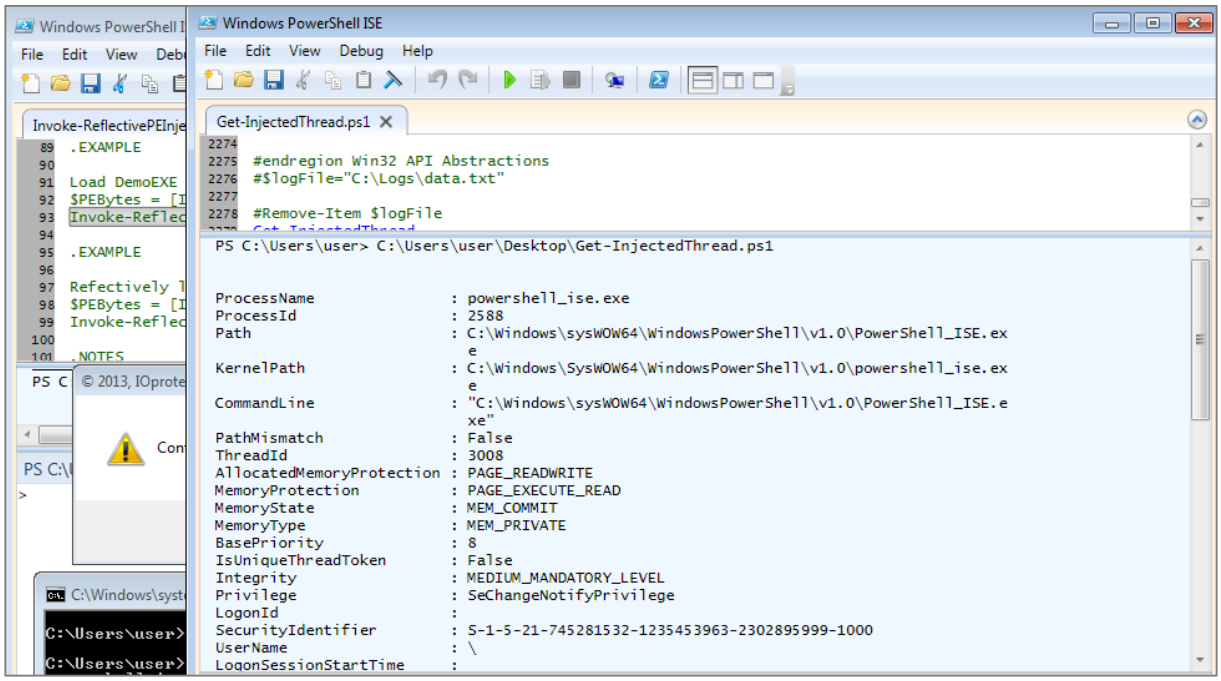

Abbildung 5: Das Skript Get-InjectedThread.ps1 identifiziert die Code-Injection in einem der beiden PowerShell_ise.exe

Auch hier zeigt das Skript den richtigen Prozess an und detektiert somit diesen Angriff.

2.3 PowerShell Empire

In diesem Fallbeispiel wird PowerShell Empire[3] genutzt. Als erstes wird ein Agent auf dem Zielsystem ausgeführt, welcher eine Remote Verbindung zum Angreifer-System (Linux) aufbaut. Danach injiziert der Angreifer Code in den lsass.exe-Prozess und erhält dadurch einen zweiten Agenten.

[3] https://www.powershellempire.com/

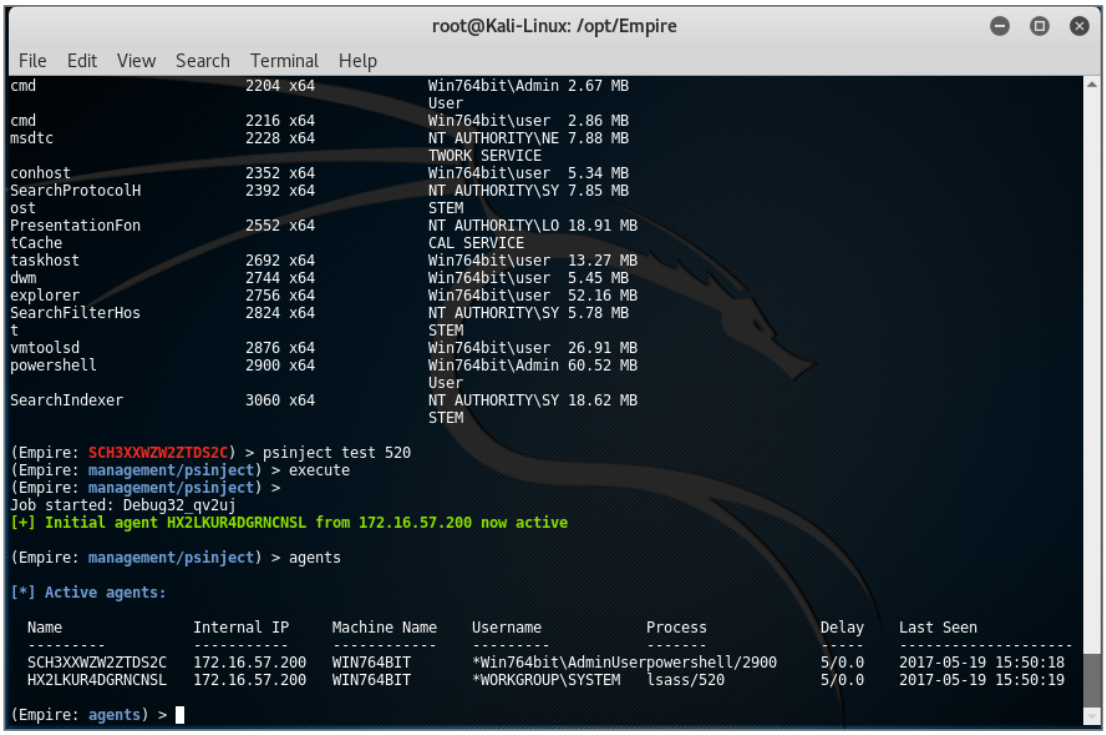

Abbildung 6: Angreifer verfügt über zwei Agenten, wobei der zweite via lsass.exe ausgeführt wurde

Lässt man Get-InjectedThread.ps1 auf dem Zielsystem ausführen, zeigt sich folgendes Bild:

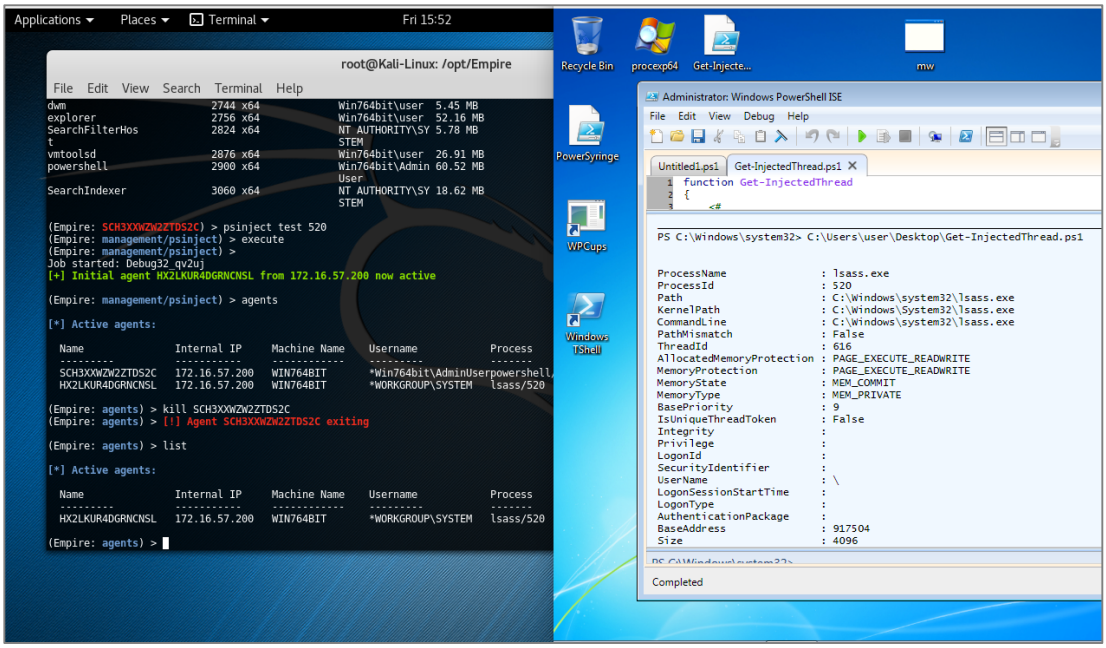

Abbildung 7: Der lsass.exe Prozess wird richtig identifiziert

Auch hier wird der entsprechende Prozess detektiert.

2.4 Metasploit Meterpreter

Meterpreter ist Teil des Metasploit Frameworks und ein sehr beliebtes Modul, um ein infiziertes Client-System via System unter Kontrolle des Angreifers fernzusteuern. Meterpreter ist ein Remote Administrations Tool und dient auch als Post Exploitation Framework. So lässt sich beispielsweise auch Mimikatz nachladen und via Meterpreter ausführen. Im Bild unten ist eine Meterpreter-Verbindung via Port 80 zu sehen. Der Angreifer kann damit Befehle mit den Rechten des eingeloggten Benutzers ausführen:

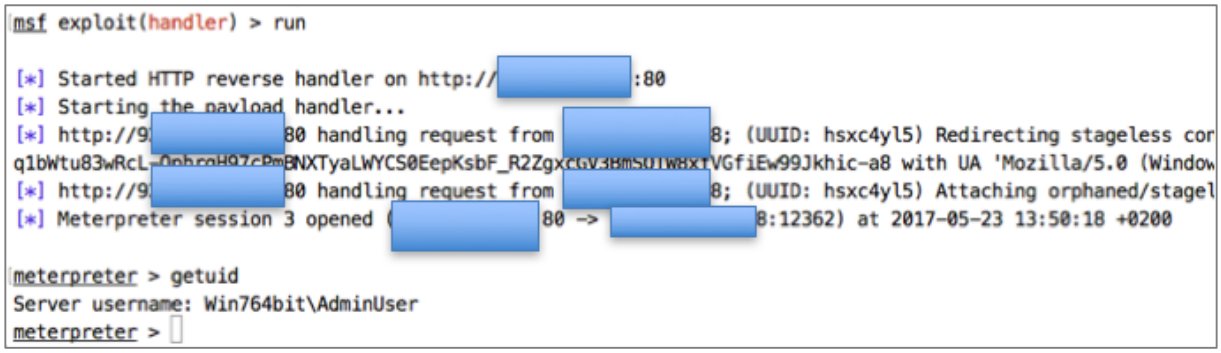

Abbildung 8: Angreifer erhält eine Remote Administrations Verbindung zu einem Windows-System via Meterpreter

In diesem Beispiel werden alle Prozesse auf dem Windows Client angezeigt sowie deren User:

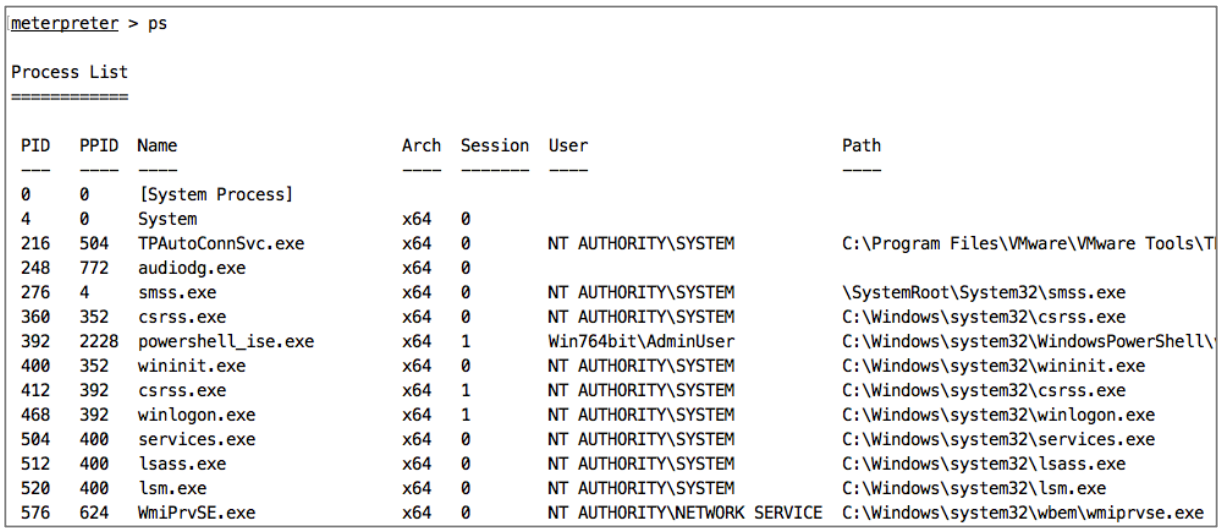

Abbildung 9: Anzeigen aller laufenden Prozesse

Damit bestimmte Post Exploitation Module wie Mimikatz etc. ausgeführt werden können, muss der Angreifer zuerst in einen anderen Prozess migrieren, welcher SYSTEM Privilegien aufweist. Dazu benötigt er Administratorenrechte. Im Bild unten wurde in den Prozess 360 (csrss.exe) migriert und anschliessend Get-InjectedThread.ps1 mit Administratorenrechten ausgeführt.

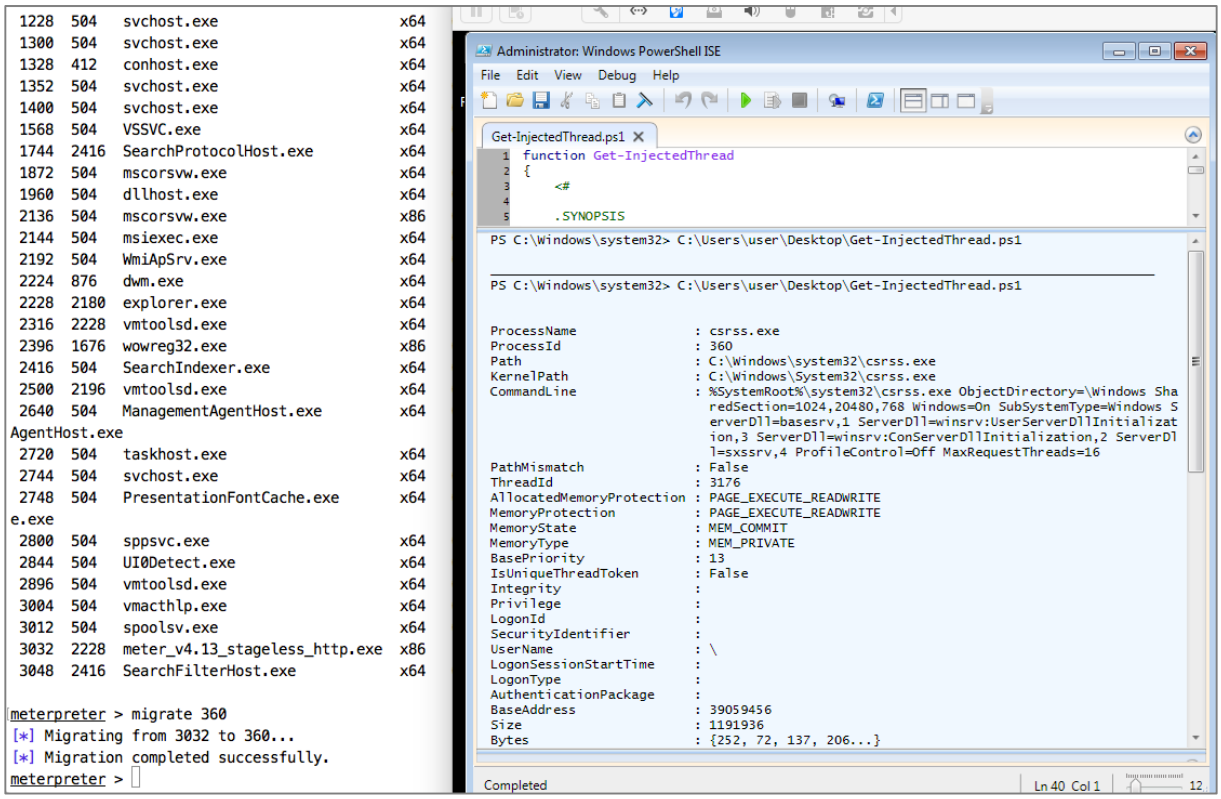

Abbildung 10: Migration in einen Prozess via Meterpreter; detektiert mit Get-InjectedThread.ps1

Der betroffene Prozess wird ebenfalls erkannt.

2.5 Ransomware mit Process Hollowing

Beim Process Hollowing startet die Malware einen legitimen Prozess, jedoch erst einmal im Suspended Mode. Beispiele für genutzte Prozesse sind explorer.exe oder lsass.exe. Bevor jedoch der erste Thread dieses Programms ausgeführt wird, ersetzt die Malware den ursprünglichen Code des legitimen Prozesses durch den Code der Malware selbst. Die Informationen im Process Environment Block (PEB) und den Dateiangaben zeigen jedoch auf die legitime Datei auf der Disk. Die groben Schritte des Angriffs sind wie folgt:

1. Start einer neuen Instanz von, z.B. explorer.exe, mit dem ersten Thread im Suspended Mode.
2. Die Datei Malware.exe öffnen und deren Inhalt in einen Buffer kopieren.
3. Speicherbereich des legitimen Prozesses freigeben. Die DLLs, die Heaps bleiben alloziert, File-Handles etc. offen.
4. Neues Speichersegment in explorer.exe allozieren. Dieser Bereich muss RWX Berechtigungen haben.
5. Kopieren des PE-Headers von malware.exe in den neu allozierten Speicherbereich in explorer.exe.
6. Kopieren jeder PE-Section des malware.exe-Prozesses in den explorer.exe-Prozess.
7. Startadresse des ersten Threads von explorer.exe an die AddressOfEntryPoint-Adresse des maliziösen Prozesses (malware.exe) setzen.
8. Thread aus dem Suspended Mode holen (Resume Thread).

Im Fallbeispiel wird folgende Datei verwendet:
eb855e1f9ca899d48215975c86b65943c37d02c87c70515f31d75b589fee641d

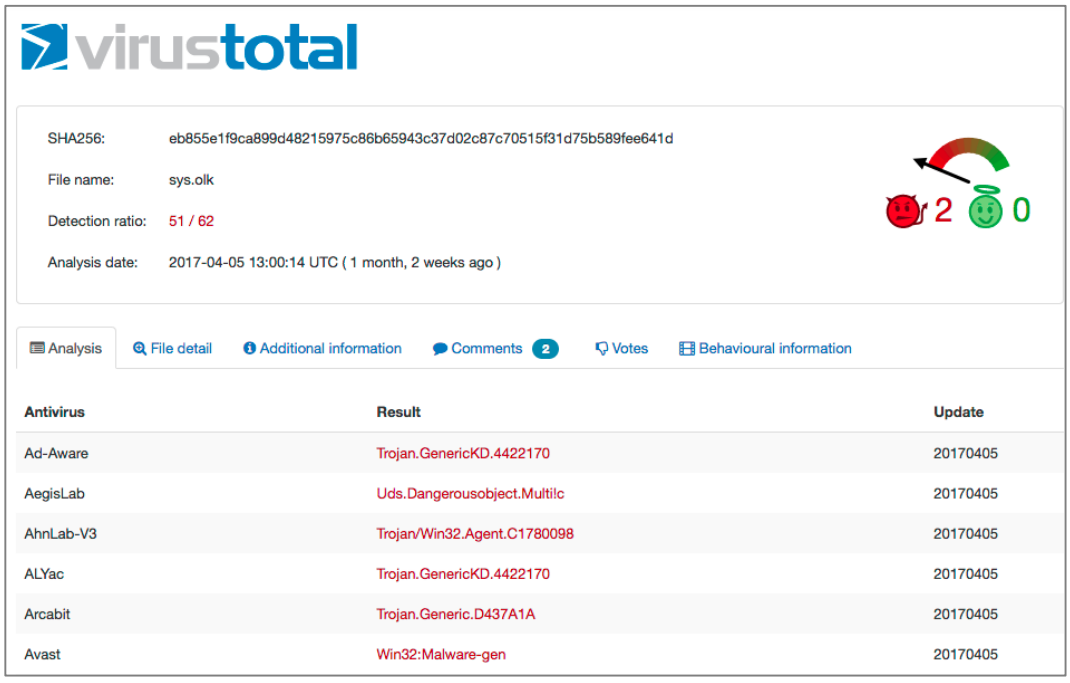

Abbildung 11: Ausgabe von VirusTotal zum Malware-Sample

Lässt man diese Datei auf einem nicht gepatchten Windows 7 System laufen, erscheint nach ein paar Sekunden ein explorer-exe-Prozess, welcher unter NT AUTHORITY\SYSTEM läuft. Diese Malware enthält damit auch einen Privilege Escalation Exploit, der im November 2016 gepatcht wurde.

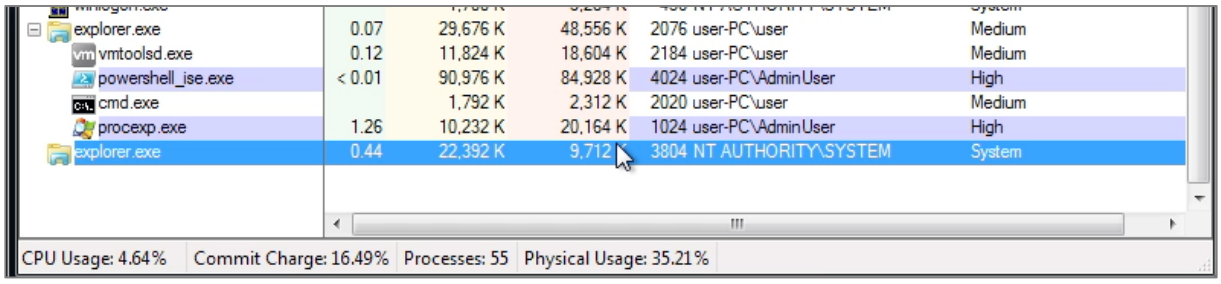

Abbildung 12: Die Schadsoftware führt via Privilege Escalation Exploit eine neue Instanz von explorer.exe als SYSTEM aus

Lässt man wiederum Get-InjectedThread.ps1 mit erhöhten Rechten ausführen, wird der betroffene Prozess detektiert:

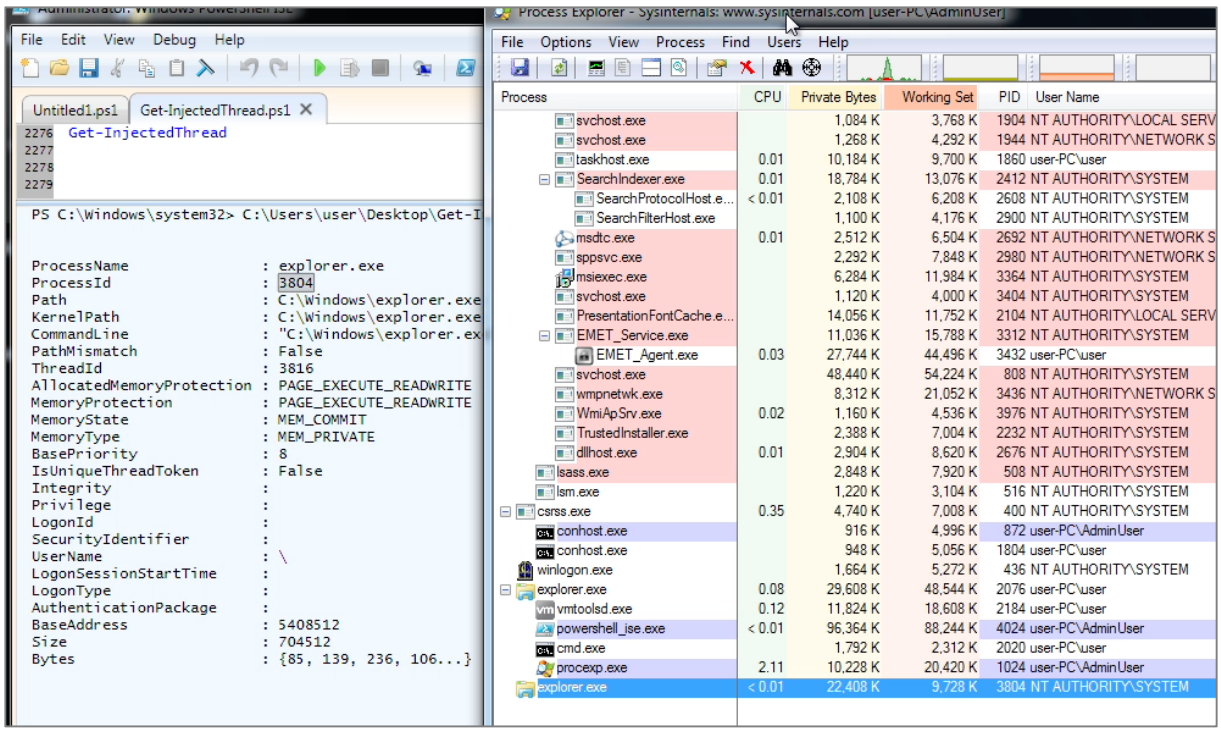

Abbildung 13: Der betroffene Prozess wird durch den Monitoring Use Case detektiert

Interessante Beobachtung: Ein paar Monate später wurde dieselbe Malware wieder ausgeführt und anschliessend via Process Explorer und Virustotal-Anbindung untersucht. Jedoch wird mit diesem Vorgehen der maliziöse Prozess nicht erkannt.

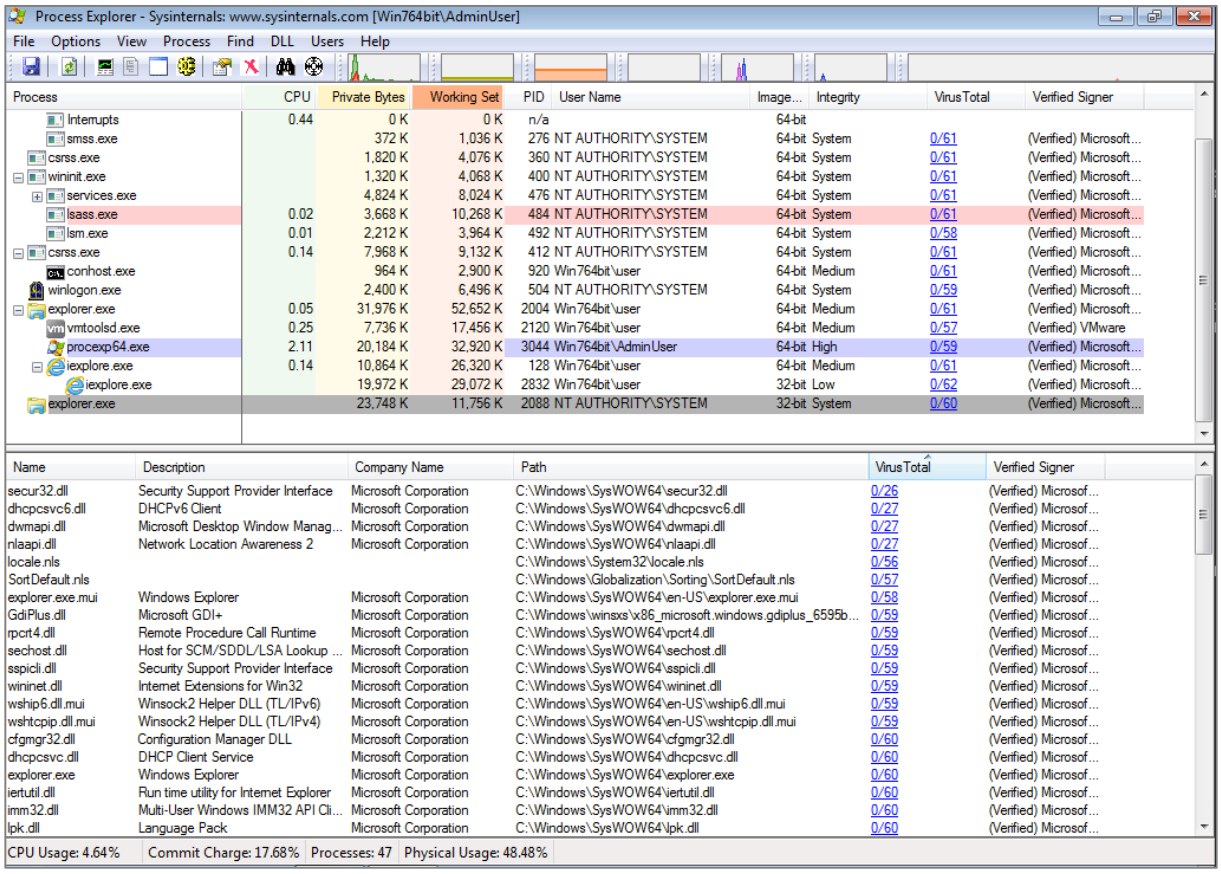

Abbildung 14: Virustotal erkennt den bösartigen Code nicht

3 Bemerkungen

Eine mögliche Einbettung dieses Monitoring Use Cases in das interne Log-Monitoring Framework kann z.B. mittels eines Scheduled Tasks und der Überwachung des entsprechenden Logfile Directories erfolgen. Zu klären ist, ob installierte Third-Party Applikationen (z.B. AV) ebenfalls diese Technik nutzen und entsprechend in eine Whiteliste aufzunehmen sind.