

Dieser Artikel erschien ursprünglich in c't 18/2011, Seite 178

Tatort Internet

S02E04: Operation am offenen Herzen

Frank Boldewin - 03.10.2011

Es ist einer der wenigen Samstagnachmittage, an denen die Sonne scheint. Ich überlege gerade, wie viel Fleisch ich wohl für den geplanten Grill-Abend einkaufen soll, als mein Handy klingelt. Mit nervöser Stimme eröffnet mir Hans, dass er sich anscheinend 'nen Virus eingefangen habe.



[Artikelanfang](#)
[Festplattenzugriffe](#)
[Dateisystem](#)
[Boot-Tricks](#)

Seite « 1 2 3 4 »

[English version](#)

Als ich zurückfrage, wie er denn darauf komme, gerät der sonst so selbstsichere Maschinenbaustudent ins Stottern: „Äh, naja, ich hab mir doch gestern einen neuen Rechner gekauft, inklusive Microsoft Office. Allerdings nur als 30-Tage-Testversion und ich hatte kein Geld mehr für die Vollversion. Und da dachte ich ...“, „... da lad ich mir mal eben einen Crack runter und spar mir das Geld“, vervollständige ich den Satz. „Und jetzt passieren auf einmal seltsame Dinge auf deinem System, stimmt's?“

„Genau, aber woher weißt Du...?“, wundert sich Hans. Ich denk insgeheim: „Weil jeden Tag ein Dummer aufsteht“, während ich ihm ruhig erkläre, dass er keineswegs der Erste ist, der sich mit solchen Problemen an mich wendet.

Dann zieh ich ihm die Fakten Stück für Stück aus der Nase: Nach dem Start des angeblichen Cracks verschwand die ausführbare Datei wie von Zauberhand vom Desktop. Sonst passierte nichts – jedenfalls nichts Sichtbares. Doch seitdem signalisierte der Router quasi permanenten Internet-Verkehr, obwohl keine Anwendungen gestartet waren, die Traffic hätten produzieren können. Nach einem Neustart war kurz Ruhe und dann ging das Geblinke der Router-LED erneut los. Da Hans noch etwas bei mir gut hat, schwing ich mich auf mein Fahrrad und mach mich auf den Weg zu ihm.

Ein erster Blick auf das System und mein siebter Sinn sagen mir, dass ich hier an der Oberfläche nicht fündig werde, sondern deutlich tiefer bohren muss. Zu dumm, dass ich mein Memory-Dump-Analyse-System gestern im Büro gelassen habe. Also werd ich wohl den Rechner direkt analysieren müssen: lokales Kernel-Debugging – quasi eine Operation am offenen Herzen.

„No risk, no fun!“, schießt mir durch den Kopf, während ich in Gedanken die Ärmel hochkremple. Ganz real installiere ich von meinem schreibgeschützten USB-Stick

708996

International: **The H, The H Security, The H Open Source**

Copyright © 2011 Heise Zeitschriften Verlag Content Management by InterRed

Security > Hintergrund > Tatort Internet

Sklavenarbeit

Ein Interrupt-Handler stellt dazu via `ExQueueWorkItem` beziehungsweise `IoQueueWorkItem` ein sogenanntes Work-Item mit einem Zeiger auf eine Callback-Routine in die Warteschlange. Kommt das Work-Item an die Reihe, entfernt einer der System Worker Threads es aus der Warteschlange und führt dann die Callback-Routine aus, die die anstehende Arbeit für den Handler erledigt. Diese Threads laufen immer unter dem System-Prozess mit der Prozess-ID 4.



- Artikelanfang
- Festplattenzugriffe
- Dateisystem
- Boot-Tricks

Seite « 1 2 3 4 »

Genau solche Work-Items legt TDL über den Umweg sogenannter Asynchronous Process Calls (APCs) an.

English version

Mal sehen, ob ich auf Hansens PC wirklich welche finde. Dazu lasse ich mir mit `!process 0 f system` möglichst detailliert Thread- und Stack-Zustände des System-Prozesses anzeigen.

```

[Process 0 system]
PID: 252 0x1454938 SessionId: zone Cid: 0064 Feb: 00000000 ParentCid: 8000
DirBase: 0x00000000 ObjectTable: 0x1000000 HeaderCount: 243
Image System
VadPool 0x125298 Vads 4 CIdse 0 Private 3 Modified 741 Locked 8
DeviceMap 0x004419
IoMin 0
KdnapTime 0x001278
UserTime 0x00080080
KernelTime 0x0005343
QuotaPoolUsage[PagePool] 0
QuotaPoolUsage[NonPagePool] 0
Working Set Size (now min max) 178 8 145 (2368B 832 1360KB)
PeakKdnapTime 510
VirtualSize 1 Mb
VmVirtualSize 2 Mb
PageFaultCount 461
MemoryPriority 246260000
BasePriority 0
ContextSwitchCount 0

THREAD 0x1454938 Cid 0004 0064 Feb: 00000000 Win32Thread: 00000000 WAIT: (0xFreePage) KernelMode Non-Alertable
KSPID: 0000 NotificationTimer
Not implemented
DeviceMap 0x004419
 owning Process 0 Image: (Unknown)
Attached Process 0x1454938 Image: System
Exit Start TickCount 1234 Ticks: 27 (0 00 00 00 421)
Context Switch Count 974
UserTime 0x00080080
KernelTime 0x00080080
Start Address atPhaseInitialization (0x00000000)
Stack last f8ac2010 Current f8ac174c Base f8ac2000 Limit f8ac0000 Call 0
Priority 0 BasePriority 0 PriorityIncrement 0 DecrementCount 0
ChildProc RetAddr
f8ac17b4 0x000017a nt!KiSwapContext+0x26 (FPO: [Frame EEFF] [0.8.4])
f8ac17c8 0x0045970a nt!KiSwapThread+0x44 (FPO: [0.0.1])
f8ac17d8 0x00459b0d nt!PcdExitFsmDispatchContext+0x20 (FPO: [Non-FPO])
f8ac1844 0x0000466d nt!KdCaroPageThread+0x11 (FPO: [Non-FPO])
f8ac18dc 0x00004628 nt!PhaseInitialization+0x1145 (FPO: [Non-FPO])

THREAD 0x1454938 Cid 0004 0064 Feb: 00000000 Win32Thread: 00000000 WAIT: (Executive) KernelMode Non-Alertable
KSPID: 0000 NotificationTimer
Not implemented
DeviceMap 0x004419
 owning Process 0 Image: (Unknown)
Attached Process 0x1454938 Image: System
Exit Start TickCount 1234 Ticks: 26 (0 00 00 00 562)
Context Switch Count 88
UserTime 0x00080080
KernelTime 0x00080080
Start Address atExpWorkerThread (0x00000000)
Stack last f8af2010 Current f8af1840 Base f8af2000 Limit f8af0000 Call 0
Priority 12 BasePriority 12 PriorityIncrement 0 DecrementCount 0
ChildProc RetAddr
f8af1878 0x000017a nt!KiSwapContext+0x26 (FPO: [Frame EEFF] [0.8.4])
f8af1888 0x0045970a nt!KiSwapThread+0x44 (FPO: [0.0.1])
f8af189c 0x00459b0d nt!PcdExitFsmDispatchContext+0x20 (FPO: [Non-FPO])
f8af18dc 0x00004628 nt!PhaseInitialization+0x1145 (FPO: [Non-FPO])

WARNING: Phase if not in any known address. PCFollowing frame may be wrong
f8af1878 0x00003950 ExNt0x00000000
f8af18dc 0x00459b0d nt!PcdExitFsmDispatchContext+0x20 (FPO: [Non-FPO])
f8af18dc 0x00459b0d nt!PcdExitFsmDispatchContext+0x20 (FPO: [Non-FPO])

```

[\[http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=2;view=zoom;zoom=2\]](http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=2;view=zoom;zoom=2)

 [\[http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=2;view=zoom;zoom=2\]](http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=2;view=zoom;zoom=2)

708996

International: **The H, The H Security, The H Open Source**

Copyright © 2011 Heise Zeitschriften Verlag Content Management by InterRed

Security > Hintergrund > Tatort Internet

Ganz tief unten

Ich hangel mich weiter und schau mir nun mit `dt _driver_object` das Treiber-Objekt an, das laut Device-Object bei `0x82166c40` liegt. Wieder erscheint ein ungültiger Typ 0 – richtig wäre 4. Immerhin der Name ist auf den korrekten Unicode-String `"\Driver\atapi"` gesetzt und auch die Funktionen wie `DriverInit` und `DriverStartIO` zeigen auf den Original-Windows-Atapi-Treiber. Doch das „MajorFunction Array“, das die I/O-Operationen des Treibers wie `IRP_MJ_READ` und `IRP_MJ_WRITE` enthält, liegt erneut im bereits einschlägig bekannten Adressbereich `0x820dxxxx` – höchst verdächtig. Die schau ich mir doch mal genauer an.



Artikelanfang
Festplattenzugriffe
Dateisystem
Boot-Tricks

Seite « 1 2 3 4 »

English version

```
kd> dt _driver_object 0x8217af1
nt!_driver_object {
+0x000 Type          0x0
+0x002 Size          0x34
+0x004 ReferenceCount 0x0
+0x008 DeviceObject  0x82166c40 _DEVICE_OBJECT
+0x00c NextDevice    0x82144330 _DEVICE_OBJECT
+0x010 AttachedDevice 0x8217cab8 _DEVICE_OBJECT
+0x014 CurrentIrp     (null)
+0x018 Timer          (null)
+0x01c Flags          0x050
+0x020 Characteristics 0x100
+0x024 Tpb            (null)
+0x028 DeviceExtension 0x8217ebas Void
+0x02c DeviceType     2
+0x030 StackSize      1
+0x034 Queue          _unnamed
+0x038 AlignmentRequirement 1
+0x03c DeviceQueue    _KEVENT
+0x03e DeviceQueue    _KEVENT
+0x040 ActiveThreadCount 0
+0x044 SecurityDescriptor 0x8217ebas Void
+0x048 DeviceLock      _KEVENT
+0x04c SectorSize      0
+0x050 SectorsPerTrack 1
+0x054 DeviceObjectExtension 0x8217ed28 _DEVICE_OBJECT_EXTENSION
+0x058 Reserved        (null)
}
kd> dt _driver_object 0x82166c40
nt!_driver_object {
+0x000 Type          0x0
+0x002 Size          0x18
+0x004 DeviceObject  0x8201cb90 _DEVICE_OBJECT
+0x008 Flags          4
+0x00c DriverStart    0x820d3000 Void
+0x010 DriverSize     0x17400
+0x014 DriverSection  0x820d3c38 Void
+0x018 DriverExtension 0x8216ca40 _DRIVER_EXTENSION
+0x01c DriverName      UNICODE_STRING "\Driver\atapi"
+0x020 HardwareDatabase 0x00661d8 UNICODE_STRING "REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM"
+0x024 FastIoDispatch (null)
+0x028 DriverInit      0x820d7517 long atapi!CaDriverEntry+0
+0x030 DriverStartIo   0x820d97c6 void atapi!IdePortStartIo+0
+0x034 DriverUnload    0x820d2204 void atapi!IdePortUnload+0
+0x038 MajorFunction   [7] 0x820d3440 long atapi!IdePortIo+0
}
```

[<http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=3;view=zoom;zoom=4>]



[<http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=3;view=zoom;zoom=4>]

Dazu muss ich den Code gar nicht allzu genau studieren. Schon bei einer oberflächlichen Analyse wird mir klar, was hier passiert. Das mit dem Befehl `uf` erzeugte Disassembly zeigt dank Symbol-Tabelle viele bekannte Aufrufe der File System Runtime Library wie `FsRtlProcessFileLock` oder `FsRtlAllocatePool`. Darüber hinaus finden sich in den Funktionen typische Return-Codes wie `0xc0000123`

708996

International: **The H, The H Security, The H Open Source**

Copyright © 2011 Heise Zeitschriften Verlag Content Management by InterRed

Security > Hintergrund > Tatort Internet

Routinemäßig überprüfe ich noch, ob sich die Usermode-Komponente ebenfalls ins Windows-API einklinkt. Das Kommando `!chkimg` gleicht die aktuellen Bibliotheksfunktionen mit dem Microsoft Symbol Server ab und entdeckt solche Hooks dabei sehr zuverlässig. Beim Check der System-Bibliotheken `ntdll.dll` und `mswsock.dll` meldet es Fehler, die ziemlich sicher auf Hooks zurückzuführen sind. Ich gehe ihnen allerdings jetzt nicht mehr weiter nach.



[Artikelanfang](#)
[Festplattenzugriffe](#)
[Dateisystem](#)
[Boot-Tricks](#)

Seite « 1 2 3 4 »

English version

```
kd> !chkimg -d ntdll.dll
7c91deb6-7c91deba 5 bytes - ntdll!ZwProtectVirtualMemory
[ b8 89 00 00 00:e9 4f 21 50 84 ]
7c91ea32-7c91ea36 5 bytes - ntdll!NtWriteVirtualMemory (+0xb7c)
[ b8 15 01 00 00:e9 d3 15 51 84 ]
7c91eae0-7c91eaf0 5 bytes - ntdll!KiUserExceptionDispatcher (+0xba)
[ 8b 4c 24 04 8b:e9 1b 15 4f 84 ]
15 errors : ntdll.dll (7c91deb6-7c91eaf0)
kd> !chkimg -d mswsock.dll
719b405f-719b4063 5 bytes - mswsock!WSACloseSocket
[ 6a 44 68 d0 41:e9 a8 bf 0e 90 ]
719b4342-719b4346 5 bytes - mswsock!WSARecv (+0x2e3)
[ 6a 44 68 b8 44:e9 c5 bc 48 8f ]
719b5847-719b584b 5 bytes - mswsock!WSASend (+0x1505)
[ 6a 40 68 80 59:e9 c0 a7 50 8f ]
15 errors : mswsock.dll (719b405f-719b584b)
```

[<http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=4;view=zoom;zoom=9>]



[<http://www.heise.de/security/artikel/Tatort-Internet-Operation-am-offenen-Herzen-1338967.html?artikelseite=4;view=zoom;zoom=9>]

Die zusammengetragenen Fakten reichen mir, um mit den Dateinamen und markanten Windbg-Ausgaben via Google zu bestätigen, dass es sich hier tatsächlich um den neuesten Abkömmling der TDL-Rootkit-Familie handelt. Das TDL4-Rootkit ist derzeit das einzige Schadprogramm, das sich auch auf einem Windows 7 64 Bit im Kernel einnisten kann.

Boot-Spielchen

Eigentlich soll bei einem 64-Bit-Windows der Patchguard verhindern, dass Treiber in den Kernel geladen werden, die keine gültige Signatur tragen. Deshalb knipst TDL4 diese Schutzfunktion aus, indem es in einer sehr frühen Boot-Phase dem System vorgaukelt, im Systemrestore-Modus „WinPE“ zu booten. Denn der aktiviert Patchguard nicht.

708996

International: **The H, The H Security, The H Open Source**

Copyright © 2011 Heise Zeitschriften Verlag Content Management by InterRed

