

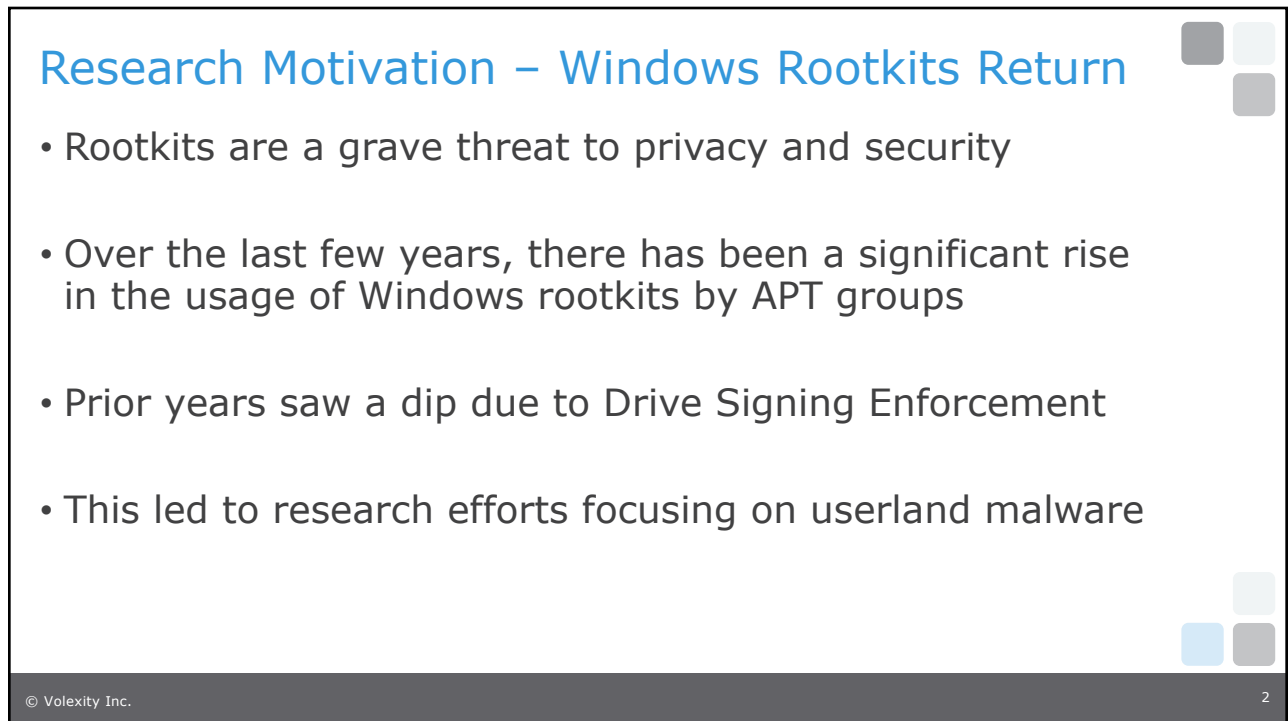
**VOLEXITY**  
CYBER SESSIONS

#DFIR in the D.M.V.

**Detecting and Triaging Modern Windows Rootkits**  
Andrew Case | *Volexity*

© Volexity Inc.

1



**Research Motivation – Windows Rootkits Return**

- Rootkits are a grave threat to privacy and security
- Over the last few years, there has been a significant rise in the usage of Windows rootkits by APT groups
- Prior years saw a dip due to Drive Signing Enforcement
- This led to research efforts focusing on userland malware

© Volexity Inc. 2

2

## Research Goals

- Develop **effective, scalable** triage techniques for detecting currently in-the-wild Windows rootkits
- Focus where malware operates - physical memory (RAM)
- Focus where historical records reside - Event Logs

© Volexity Inc.

3

3

## Why Memory Forensics?

- Across platforms, memory-only payloads are often used by malware to avoid detection and hinder analysis
- Disk and live forensics generally can find no traces of this malware
- Volatile memory is the **\*only\*** place to determine that such malware is present and to fully investigate it

*Corelump* is the main payload and resides exclusively in memory to evade detection. It contains a variety of capabilities including keylogging, capturing screenshots, exfiltrating files, running a remote shell, and running arbitrary plugins downloaded from KNOTWEED's C2 server.

Microsoft Report: [1]

© Volexity Inc.

4

4

## Why Event Logs?

- Provide timestamped, detailed records of system activity
- In many organizations, are centralized in a database that can be easily queried
- Vista+ systems have so many event log sources, that anti-forensics techniques are not generally potent

Windows Logging Cheat Sheets: [14]

5


## Loading a Driver the MS Recommended Way

- Create a service of type `SERVICE_KERNEL_DRIVER` or `SERVICE_FILE_SYSTEM_DRIVER`
- This stores the service name, driver path, etc. within the registry
- The service can then be started upon system boot or at run time by a service control program
- This creates records in the event logs, but too noisy

6

# Driver Signing Enforcement (DSE)

## Driver Signing changes in Windows 10

By  HWCert-Migrated  
Published Mar 12 2019 07:10 AM 👁 8,015 Views

**First published on MSDN on Apr 01, 2015**

NOTE: These driver signing changes correspond to the initial Windows 10 release. For driver signing changes in Windows 10, version 1607, see [this post](#).

Beginning with the release of Windows 10, **all new Windows 10 kernel mode drivers must be submitted to and digitally signed** by the [Windows Hardware Developer Center Dashboard portal](#). Windows 10 will not load new kernel mode drivers which are not signed by the portal.


Additionally, starting 90 days after the release of Windows 10, the portal will only accept driver submissions, including both kernel and user mode driver submissions, that have a valid Extended Validation ("EV") Code Signing Certificate.

We're making these changes to help make Windows more secure. **These changes limit the risk of a driver publisher's signing keys being lost or stolen** and also ensures that driver publishers are strongly authenticated.

© Volexity Inc. 7

7

# Stolen Certificates [2-4]

 COSTIN RAIU

Last night, Verisign acted promptly and revoked the second stolen certificate used to sign a version of the Stuxnet rootkit driver. As previously mentioned, this certificate [belonged to JMicron Technology Corp](#), a popular Taiwanese hardware company.

---

## Microsoft admits to signing rootkit malware in supply-chain fiasco

By **Ax Sharma** 📅 June 26, 2021 ⌚ 05:16 AM 💬 11

---

## Malware now using NVIDIA's stolen code signing certificates


By **Lawrence Abrams** 📅 March 5, 2022 ⌚ 03:45 PM 💬 4




© Volexity Inc. 8


8

## Game Changer – BYOVD [5,6]

# “Bring Your Own Vulnerable Driver” Attacks Are Breaking Windows

 **CORBIN DAVENPORT** @corbindavenport  
 JUL 22, 2022, 2:56 PM EDT | 2 MIN READ

Lazarus Adopts Bring Your Own Vulnerable Driver Attack Methodology   

 Written by Karolis Liucveikis on October 06, 2022

Lazarus Group, North Korea’s elite state-sponsored hacking group, has never been shy from adopting new techniques and tactics. In the past, the group has dabbled with ransomware blurring the lines between what was considered the realm of financially motivated hackers rather than their state-sponsored cousins. Now, according to a new report published by ESET, the group has adopted the Bring Your Own Vulnerable Driver (BYOVD) attack tactic to install Windows based rootkits.

© Volexity Inc. 9

9

## Defining `Vulnerable`

- When a driver gives kernel level access to non-intended userland processes
- This access can include read/write capabilities to physical memory and/or MSRs
- Drivers are complex and often require allowing userland to dictate parameters and operations of hardware devices —Think of your graphics card, NIC, etc.

© Volexity Inc. 10

10

## Approaches to Leveraging Vulnerable Drivers

1. Directly manipulate kernel data structures and code, such as disabling protected processes
  - Many valuable targets protected by Patch Guard
  - **Very** difficult to perform complex operations through what is essentially memory forensics
2. Disable DSE to load a second, unsigned driver (rootkit)
  - This bypasses all protection from DSE
  - Much easier to write complex code
  - The most common way, by far, seen in the wild

© Volatility Inc.

11

11

## Disabling DSE [8, 9]

- On Windows 8.1+, the *g\_CiOptions* global variable of *CI.DLL* controls DSE
- 0 = disabled, other bits undocumented and values vary across default Windows 10 versions
- Malware abuses a vulnerable driver to set *g\_CiOptions* to 0 to allow the unsigned rootkit driver to load

© Volatility Inc.

12

12

Possible Detection - Examine *g\_CiOptions* in Memory

**Thought:** Write a Volatility plugin to detect *g\_CiOptions* = 0

**Reality:** Every infected memory sample had a non-0 value

**Reason:** All rootkits tested reset *g\_CiOptions* after loading

**Verdict:** 😞


© Volexity Inc. 13

13

Patch Guard, *g\_CiOptions*, and Bypasses [12,13]

## **g\_CiOptions in a Virtualized World**

Posted on 2022-05-15 Tagged in low-level, windows, drivers



© Volexity Inc. 14

14

## Actual Detection – Event Logs [10, 11]

### Windows CodeIntegrity Operational log

Event ID	Explanation
3004	This event isn't common and may occur with or without an Application Control policy present. It typically indicates a kernel driver tried to load with an invalid signature. For example, the file may not be WHQL-signed on a system where WHQL is required.

Level	Provider	Event ID	Qualifier	Pid	User SID
Error	Microsoft-Windows-CodeIntegrity	3004	0		S-1-5-18

Key	Value
FileNameLength	73
FileNameBuffer	\Device\HarddiskVolume1\Windows\System32\drivers\MoriyaStreamWatchmen.sys
SecureRequired	0x00000001
RequestedSigningLevel	1
ProcessNameLength	6
ProcessNameBuffer	System

15

## Actual Detection – Event Logs [16, 17]

Exploit protection Security-Mitigations (Kernel Mode/User Mode) 11 Code integrity guard audit

Level	Computer Name	Provider	Event ID	Qualifier	Pid	User SID
LogAlways	EC2AMAZ-M2FC9M0	Microsoft-Windows-Security-Mitigations	11	0		S-1-5-18

ProcessPath	\Device\HarddiskVolume1\Windows\System32\svchost.exe
ProcessCommandLineLength	42
ProcessCommandLine	C:\windows\system32\svchost.exe -k netsvcs
ProcessId	3596
ProcessCreateTime	2022-08-26 17:56:40.548645
ProcessStartKey	136233888727957703
ProcessSignatureLevel	0
ProcessSectionSignatureLevel	0
ProcessProtection	0
Target ThreadId	4928
Target ThreadCreateTime	2022-08-26 17:56:40.557436
RequiredSignatureLevel	8
SignatureLevel	1
ImageNameLength	33
ImageName	Windows\Web\MoriyaServiceX64.dll

16



## Patch Guard vs Traditional Rootkit Techniques

- Besides DSE, *Kernel Patch Protection*, commonly referred to as *Patch Guard*, also hinders many previous rootkit techniques
- Patch Guard works by monitoring modifications to protected resources
- A system crash is triggered if violations found
  - If crash dumps enabled, this will 'trap' the malware inside the produced crash dump \*\*

17

```

7. [kd> !analyze -show 109]
CRITICAL_STRUCTURE_CORRUPTION (109)
This bugcheck is generated when the kernel detects that critical kernel code or
data have been corrupted. There are generally three causes for a corruption:
1) A driver has inadvertently or deliberately modified critical kernel code
or data. See http://www.microsoft.com/whdc/driver/kernel/64bitPatching.aspx
2) A developer attempted to set a normal kernel breakpoint using a kernel
debugger that was not attached when the system was booted. Normal breakpoints,
"bp", can only be set if the debugger is attached at boot. Critical floating point control register modification
breakpoints, "ba", can be set at any time. 16 : Critical floating point control register modification
17 : Local APIC modification
3) A hardware corruption occurred, e.g. failing RAM holding 18 : Kernel notification callout modification
Arguments:
Arg1: 0000000000000000. Reserved 19 : Loaded module list modification
Arg2: 0000000000000000. Reserved 1a : Type 3 process list corruption
Arg3: 0000000000000000. Failure type dependent information 1b : Type 4 process list corruption
Arg4: 0000000000000000. Type of corrupted region, can be
0 : A generic data region 1c : Driver object corruption
1 : Modification of a function or .pdata 1d : Executive callback object modification
2 : A processor IDT 1e : Modification of module padding
3 : A processor GDT 1f : Modification of a protected process
4 : Type 1 process list corruption 20 : A generic data region
5 : Type 2 process list corruption 21 : A page hash mismatch
6 : Debug routine modification 22 : A session page hash mismatch
7 : Critical MSR modification 23 : Load config directory modification
8 : Object type 24 : Inverted function table modification
9 : A processor IVT 25 : Session configuration modification
a : Modification of a system service function 26 : An extended processor control register
b : A generic session data region 27 : Type 1 pool corruption
c : Modification of a session function or .pdata 28 : Type 2 pool corruption
d : Modification of an import table 29 : Type 3 pool corruption
e : Modification of a session import table 2a : Type 4 pool corruption
f : Ps Win32 callout modification 2b : Modification of a function or .pdata
10 : Debug switch routine modification 2c : Image integrity corruption
11 : IRP allocator modification 2d : Processor misconfiguration
12 : Driver call dispatcher modification 2e : Type 5 process list corruption
13 : IRP completion dispatcher modification 2f : Process shadow corruption
14 : IRP deallocator modification 30 : Retpoline code page corruption
15 : A processor control register 101 : General pool corruption
102 : Modification of win32k.sys
    
```

18

## Drivers in Memory - Modules

```
$ python3 vol.py -f data.lime -r pretty windows.modules
Volatility 3 Framework 2.4.0
```

Offset	Base	Size	Name	Path
*   0x89017b24e610	0xf8014b600000	0x1046000	ntoskrnl.exe	\SystemRoot\system32\ntoskrnl.exe
*   0x89017b24ea20	0xf8014a920000	0x6000	hal.dll	\SystemRoot\system32\hal.dll
*   0x89017b24ebb0	0xf8014a930000	0xb000	kdcom.dll	\SystemRoot\system32\kd.dll
*   0x89017b24ed60	0xf8014a8f0000	0x28000	mcupdate.dll	\SystemRoot\system32\mcupdate_AuthenticAMD.dll
*   0x89017b260050	0xf8014a970000	0x6b000	CLFS.SYS	\SystemRoot\System32\drivers\CLFS.SYS
*   0x89017b260200	0xf8014a940000	0x27000	tm.sys	\SystemRoot\System32\drivers\tm.sys
*   0x89017b2603b0	0xf8014a9e0000	0x1a000	PSHED.dll	\SystemRoot\system32\PSHED.dll
*   0x89017b260570	0xf8014aa00000	0xb000	BOOTVID.dll	\SystemRoot\system32\BOOTVID.dll
*   0x89017b260720	0xf8014ab30000	0x6f000	FLTMGR.SYS	\SystemRoot\System32\drivers\FLTMGR.SYS
*   0x89017b2608e0	0xf8014abd0000	0x61000	msrpc.sys	\SystemRoot\System32\drivers\msrpc.sys
*   0x89017b260aa0	0xf8014aba0000	0x29000	ksecdd.sys	\SystemRoot\System32\drivers\ksecdd.sys
*   0x89017b25f980	0xf8014aa10000	0x114000	clipsps.sys	\SystemRoot\System32\drivers\clipsps.sys

[cut 200+ rows]

19

## Drivers in Memory - DRIVER\_OBJECT

```
$ python3 vol.py -f data.lime -r pretty windows.driverscan
Volatility 3 Framework 2.4.0
```

Offset	Start	Size	Service Key	Driver Name	Name
*   0x89017b24d4e0	0xf8014b600000	0x0	\Driver\PnpManager	PnpManager	\Driver\PnpManager
*   0x89017b2a44f0	0xf80150800000	0xb000	msisadrv	msisadrv	\Driver\msisadrv
*   0x89017b2a4970	0xf801507e0000	0x14000	pcw	pcw	\Driver\pcw
*   0x89017b2a6d80	0xf8014b600000	0x0	\Driver\WMIxWDM	WMIxWDM	\Driver\WMIxWDM
*   0x89017b2e04f0	0xf801503b0000	0xd1000	Wdf01000	Wdf01000	\Driver\Wdf01000
*   0x89017b2e0970	0xf80151050000	0x40000	Wof	Wof	\FileSystem\Wof
*   0x89017b2e24f0	0xf80150650000	0xcc000	ACPI	ACPI	\Driver\ACPI
*   0x89017b2fdd80	0xf8014b600000	0x0	\Driver\DeviceApi	DeviceApi	\Driver\DeviceApi
*   0x89017b30ad80	0xf8014b600000	0x0	\Driver\ACPI_HAL	ACPI_HAL	\Driver\ACPI_HAL

[cut 150+ rows]

20

## Rootkit Technique – Module and Driver Tampering

- Rootkits deploy several techniques to hide from analysis, such as:

1. Unlinking its module from the module list \*
2. Changing the driver’s metadata
  - Base address
  - Size
  - Function pointers for operations (start, unload, etc.)
3. Changing the module’s metadata

Detection: Disconnect between a module and its driver

21

## DriverModule vs DirtyMoe [15]



```
$ python3 vol.py -f dirtymoe.lime windows.driverscan
Volatility 3 Framework 2.4.0
Offset      Start Size Service Key  Driver Name  Name
0x13fa8bda0 0x0    0x0    dump_EDCB5A3C dump_EDCB5A3C \Driver\dump_EDCB5A3C

$ python3 vol.py -f dirtymoe.lime windows.drivermodule
Volatility 3 Framework 2.4.0
Offset      Known Driver Name  Service Key  Alternative Name
0x6070ca0  True  RAW              \FileSystem\RAW
0x13fa8bda0 False dump_EDCB5A3C    dump_EDCB5A3C  \Driver\dump_EDCB5A3C
```

22

## I/O Request Packets and Handlers

- I/O Request Packets (IRPs) are used to send requests to a kernel driver
- Examples:
  - Asking the NTFS driver for access to a file
  - Asking the network driver to send data
  - Asking a rootkit driver to hide the process with PID 42

23

## IRP Entries

IRP\_MJ\_CLEANUP

IRP\_MJ\_POWER

IRP\_MJ\_CLOSE

IRP\_MJ\_QUERY\_INFORMATION

IRP\_MJ\_CREATE

IRP\_MJ\_READ

IRP\_MJ\_DEVICE\_CONTROL

IRP\_MJ\_SET\_INFORMATION

IRP\_MJ\_FILE\_SYSTEM\_CONTROL

IRP\_MJ\_SHUTDOWN

IRP\_MJ\_FLUSH\_BUFFERS

IRP\_MJ\_SYSTEM\_CONTROL

IRP\_MJ\_INTERNAL\_DEVICE\_CONTROL

IRP\_MJ\_WRITE

IRP\_MJ\_PNP

24

```

$ python3 vol.py -f data.lime windows.driverirp
Ntfs | IRP_MJ_CREATE | 0xf80dfe5141f0 | ntfs
Ntfs | IRP_MJ_CREATE_NAMED_PIPE | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_CLOSE | 0xf80dfe514c90 | ntfs
Ntfs | IRP_MJ_READ | 0xf80dfe40c430 | ntfs
Ntfs | IRP_MJ_WRITE | 0xf80dfe406e80 | ntfs
Ntfs | IRP_MJ_QUERY_INFORMATION | 0xf80dfe512c40 | ntfs
Ntfs | IRP_MJ_SET_INFORMATION | 0xf80dfe4e0d70 | ntfs
Ntfs | IRP_MJ_QUERY_EA | 0xf80dfe512c40 | ntfs
Ntfs | IRP_MJ_SET_EA | 0xf80dfe512c40 | ntfs
Ntfs | IRP_MJ_FLUSH_BUFFERS | 0xf80dfe523b80 | ntfs
Ntfs | IRP_MJ_QUERY_VOLUME_INFORMATION | 0xf80dfe520f70 | ntfs
Ntfs | IRP_MJ_SET_VOLUME_INFORMATION | 0xf80dfe520f70 | ntfs
Ntfs | IRP_MJ_DIRECTORY_CONTROL | 0xf80dfe4fd9e0 | ntfs
Ntfs | IRP_MJ_FILE_SYSTEM_CONTROL | 0xf80dfe4f8b70 | ntfs
Ntfs | IRP_MJ_DEVICE_CONTROL | 0xf80dfe4c98f0 | ntfs
Ntfs | IRP_MJ_INTERNAL_DEVICE_CONTROL | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_SHUTDOWN | 0xf80dfe617c60 | ntfs
Ntfs | IRP_MJ_LOCK_CONTROL | 0xf80dfe4487e0 | ntfs
Ntfs | IRP_MJ_CLEANUP | 0xf80dfe515550 | ntfs
Ntfs | IRP_MJ_CREATE_MAILSLLOT | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_QUERY_SECURITY | 0xf80dfe520f70 | ntfs
Ntfs | IRP_MJ_SET_SECURITY | 0xf80dfe520f70 | ntfs
Ntfs | IRP_MJ_POWER | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_SYSTEM_CONTROL | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_DEVICE_CHANGE | 0xf8010812fd70 | ntoskrnl
Ntfs | IRP_MJ_QUERY_QUOTA | 0xf80dfe512c40 | ntfs
Ntfs | IRP_MJ_SET_QUOTA | 0xf80dfe512c40 | ntfs
Ntfs | IRP_MJ_PNP | 0xf80dfe568300 | ntfs
    
```

25

## Rootkit Technique – Hooked IRPs

- Rootkits will hook IRP entries for drivers of interest to control operations
- This gives very low-level control over system operations
- Patch Guard (supposedly) protects IRP entries for critical drivers

26

## DriverIRP vs Ghost Emperor [16]

In addition, we noticed some similarities between the features of Demodex and those of the Derusbi rootkit, which [was publicly described](#) in the past and also attributed to a Chinese-speaking actor. The purpose of both is to hide malicious artefacts, where notably both have an almost identical flow for hiding TCP connections by hooking the nsiproxy.sys IOCTL dispatcher. The

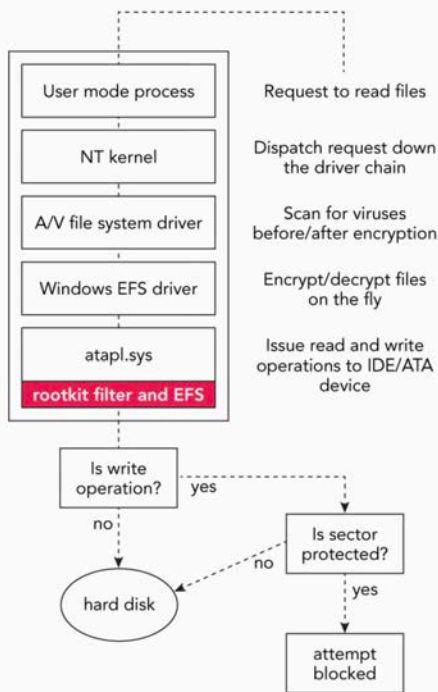
nsiproxy	IRP_MJ_DIRECTORY_CONTROL	0xf8010b0c24b0	nsiproxy
nsiproxy	IRP_MJ_FILE_SYSTEM_CONTROL	0xf8010b0c24b0	nsiproxy
nsiproxy	IRP_MJ_DEVICE_CONTROL	0xe38a8ed14168	-
nsiproxy	IRP_MJ_INTERNAL_DEVICE_CONTROL	0xf8010b0c24b0	nsiproxy

dump_audio_codec0	IRP_MJ_CREATE	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_CREATE_NAMED_PIPE	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_CLOSE	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_READ	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_WRITE	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_QUERY_INFORMATION	0xe38a8ed18278	-
dump_audio_codec0	IRP_MJ_SET_INFORMATION	0xe38a8ed18278	-

27

## Device Trees

- Windows has a layered architecture for device access so that multiple drivers can handle one request
- This helps support firewalls, AV, EDRs, etc.



28

## Device Trees in Memory

```

* 0xbf83316bb8c0    DEV    volmgr  HarddiskVolume3 N/A    FILE_DEVICE_DISK
** 0xbf83316bb8c0  ATT    volmgr  -                \Driver\fvevol  FILE_DEVICE_DISK
*** 0xbf83316bb8c0 ATT    volmgr  -                \Driver\iorate  FILE_DEVICE_DISK
**** 0xbf83316bb8c0 ATT    volmgr  -                \Driver\volume  FILE_DEVICE_DISK
***** 0xbf83316bb8c0 ATT    volmgr  -                \Driver\volsnap FILE_DEVICE_DISK

0xbf83317d3a10  DRV    NTFS    N/A    N/A    N/A
* 0xbf83317d3a10  DEV    NTFS    -      N/A    FILE_DEVICE_DISK_FILE_SYSTEM
** 0xbf83317d3a10  ATT    NTFS    -      \FileSystem\FltMgr FILE_DEVICE_DISK_FILE_SYSTEM

0xbf83318b1860  DRV    Tcpip   N/A    N/A    N/A
* 0xbf83318b1860  DEV    Tcpip   eQoS   N/A    FILE_DEVICE_NETWORK
    
```

29

## DeviceTree vs Rootkits [16, 17]

The rootkit attaches to nsiproxy.sys's device stack and intercepts IOCTLs of type IOCTL\_NSI\_GETALLPARAM (0x12000B) that are sent to it. This IOCTL is used to retrieve information about the active network connections on the system. When it is intercepted, the driver replaces the IoCompletion routine with a function that filters the results to hide its own network connections.

```

0xbf83312462e0  DRV    nsiproxy N/A    N/A    N/A
* 0xbf83312462e0  DEV    nsiproxy Nsi   N/A    FILE_DEVICE_NETWORK
** 0xbf83312462e0  ATT    nsiproxy -      \FileSystem\crtsys FILE_DEVICE_NETWORK
    
```

- 0x220304:** This IOCTL is used to register a file system filter driver's notification routine by using the IoRegisterFSRegistrationChange API. The notification routine invoked upon registration of a new file system verifies if it is an NTFS-based one and if so, creates a device object for the rootkit which is attached to the subject file system's device stack. Additionally, both the file

```

0xe38a8fa48dc0  DRV    Ntfs    N/A    N/A    N/A
* 0xe38a8fa48dc0  DEV    Ntfs    -      N/A    FILE_DEVICE_DISK_FILE_SYSTEM
** 0xe38a8fa48dc0  ATT    Ntfs    -      \FileSystem\FltMgr FILE_DEVICE_DISK_FILE_SYSTEM
*** 0xe38a8fa48dc0  ATT    Ntfs    -      \Driver\dump_audio_codec0 FILE_DEVICE_DISK_FILE_SYSTEM
    
```

30

## Rootkit Technique – GetCellRoutine Hijacking

- Since callbacks are checked by EDRs and memory forensics, rootkits want to be a bit more stealthy
- To observe and modify registry operations, GetCellRoutine hijacking can be performed instead of CmRegisterCallback

31

## Enumerating Hives

**\$ python3 vol.py -f crtsys\_apt.lime -r pretty hivelist**

Volatility 3 Framework 2.4.0

Offset | File Full Path

```
* | 0x9a869403d000 | \REGISTRY\MACHINE\SYSTEM
* | 0x9a8694077000 | \REGISTRY\MACHINE\HARDWARE
* | 0x9a86959e1000 | \Device\HarddiskVolume1\EFI\Microsoft\Boot\BCD
* | 0x9a86947fa000 | \SystemRoot\System32\Config\SOFTWARE
* | 0x9a8694ad0000 | \SystemRoot\System32\Config\DEFAULT
* | 0x9a8699513000 | \SystemRoot\System32\Config\SECURITY
* | 0x9a8699582000 | \SystemRoot\System32\Config\SAM
* | 0x9a8699643000 | \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
* | 0x9a86997d9000 | \SystemRoot\System32\Config\BBI
* | 0x9a86997f8000 | \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
* | 0x9a869a282000 | \??\C:\Windows\AppCompat\Programs\Amcache.hve
* | 0x9a8699bd3000 | \??\C:\Users\Administrator\ntuser.dat
* | 0x9a869b29b000 | \SystemRoot\System32\config\DRIVERS
```

```
'_HHIVE' : [ 0x600, {
'Signature' : [ 0x0, ['unsigned long']],
'GetCellRoutine' : [ 0x8, ['pointer64', ['void']],
...

```

32



## Volatility vs FireChili

- Volatility's new getcellroutine plugin reports any cached hive whose GetCellRoutine handler is not the kernel

```
$ python3 vol.py -f crtsys_apt.lime -r pretty getcellroutine
Volatility 3 Framework 2.4.0
| Hive | Module | Handler
* | \REGISTRY\MACHINE\SYSTEM | crtsys | 0xf800f85d4cf0
```

© Volatility Inc. 33

33

## Questions? Comments?

### Contact

[andrew@dfir.org](mailto:andrew@dfir.org)

### Social Media

@volexity, @volatility, @lsucyber, @attrc

© Volatility Inc. 34

34

## References

- [1] <https://www.microsoft.com/security/blog/2022/07/27/untangling-knotweed-european-private-sector-offensive-actor-using-0-day-exploits/>
- [2] <https://securelist.com/stuxnet-signed-certificates-frequently-asked-questions/29725/>
- [3] <https://www.bleepingcomputer.com/news/security/microsoft-admits-to-signing-rootkit-malware-in-supply-chain-fiasco/>
- [4] <https://www.bleepingcomputer.com/news/security/malware-now-using-nvidias-stolen-code-signing-certificates/>
- [5] <https://www.howtogeek.com/820374/bring-your-own-vulnerable-driver-attacks-are-breaking-windows/>
- [6] <https://www.pcrisk.com/internet-threat-news/25005-lazarus-adopts-bring-your-own-vulnerable-driver-attack-methodology>
- [7] <https://www.rapid7.com/blog/post/2021/12/13/driver-based-attacks-past-and-present/>
- [8] <https://y1k1nqfr.github.io/loading-windows-unsigned-driver/>
- [9] <https://blog.nviso.eu/2022/01/10/kernel-karnage-part-8-getting-around-dse/>
- [10] <https://securelist.com/operation-tunnelsnake-and-moriya-rootkit/101831/>
- [11] <https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/event-id-explanations>
- [12] [https://www.trustedsec.com/blog/g\\_cioptions-in-a-virtualized-world](https://www.trustedsec.com/blog/g_cioptions-in-a-virtualized-world)
- [13] <https://www.fortinet.com/blog/threat-research/driver-signature-enforcement-tampering>
- [14] <https://www.malwarearchaeology.com/cheat-sheets>
- [15] [https://www.splunk.com/en\\_us/blog/security/these-are-the-drivers-you-are-looking-for-detect-and-prevent-malicious-drivers.html](https://www.splunk.com/en_us/blog/security/these-are-the-drivers-you-are-looking-for-detect-and-prevent-malicious-drivers.html)
- [16] <https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/event-id-explanations>
- [17] <https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/overview-attack-surface-reduction?view=0365-worldwide>