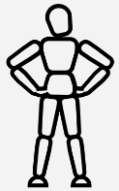
Throwing an AquaWrench into the Kernel

Senior Managing
Consultant @
IBM X-Force Red

Twitter: @FuzzySec

Who mI



Ruben Boonen (b33f)

All about hacking Windows on endpoint

What Is AquaWrench?

1

Bridging the gap between Administrator and Kernel

2

Perform data attacks on Kernel memory

3

Manipulate OS security and bootstrap unsigned Kernel code

Local attacker with Administrator privileges

Administrator ⟶ Kernel

- Not a security boundary!

Not commonly seen in real-world attacks

- e.g. White Lambert
  - 3-letter agency
  - Passive network backdoor

Setting The Scene

“Why is it uncommon?”

⟶ 64-bit Windows Vista

Driver signing requirements enforce a soft security boundary

↪ Buy a code signing certificate (privacy concerns)

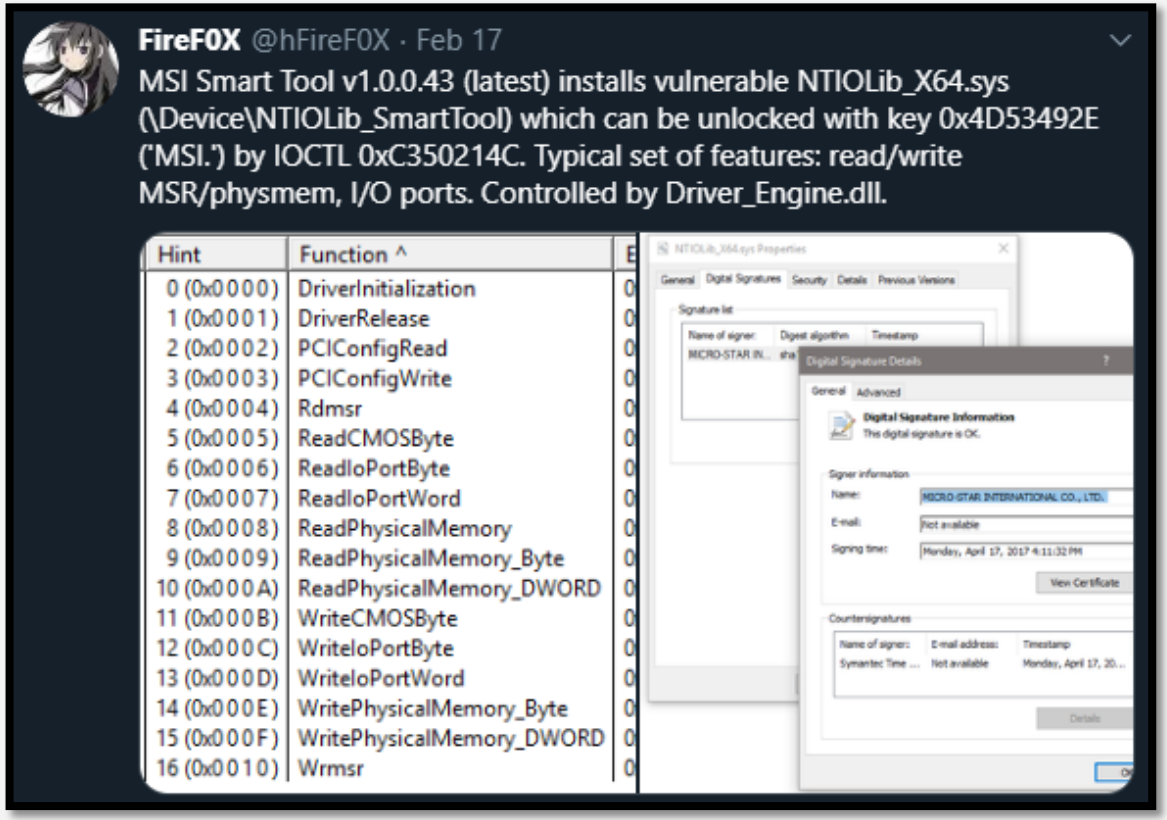
↪ Abuse a stolen or leaked certificate (can h4zz game hack?)

↪ Exploit the Kernel or a 3rd party driver vulnerability

Turla APT

↪ Uroboros Root Kit (2014) -> VBoxDrv.sys (CVE-2008-3431)

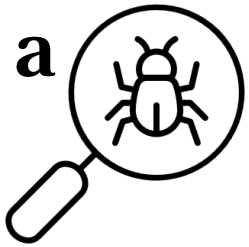
It's the wild west out there .. not kidding



Lots of vendors with terrible code ⟶ https://github.com/hfiref0x/KDU

3rd Party YOLO Driver Development

Selecting a class



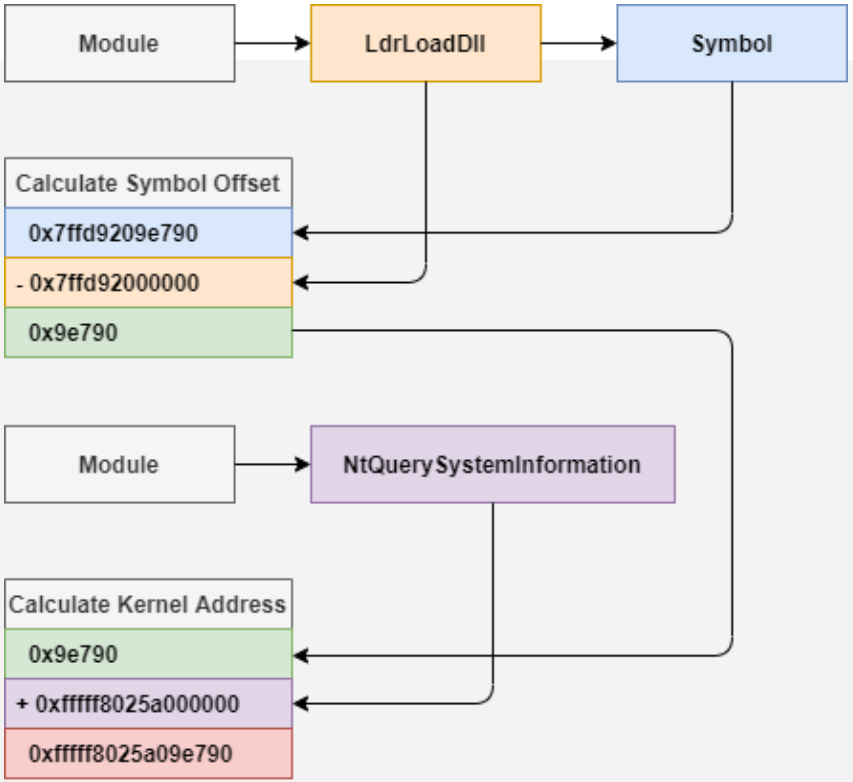
There are many signed drivers with many bug classes

↳ Some more useful than others (e.g. MSR R/W -> LSTAR 0xc0000082)

Mapping physical memory is extremely common and OP

↳ Implied arbitrary Kernel R/W

AquaWrench R/W - Part 1

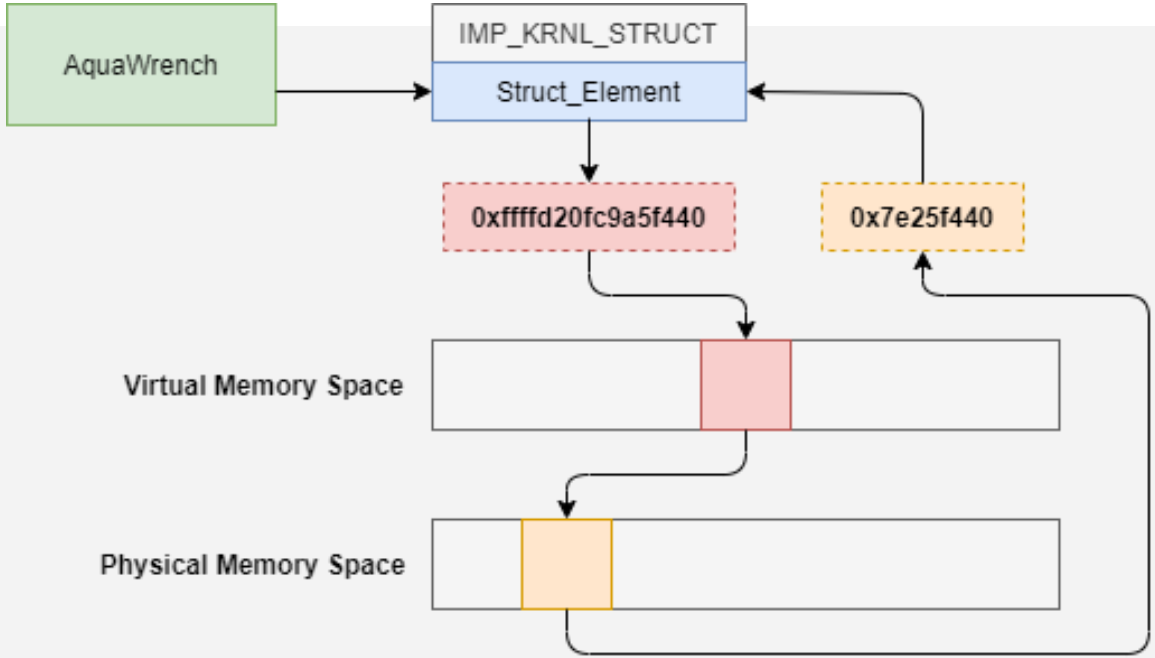


We want to modify X in ring0
but how do we find X VA?

Can KASLR save you?

- KASLR is not a security boundary for local, non-LowIL users
- “I Got 99 Problems But a Kernel Pointer Ain't One” – Alex Ionescu / Recon 2013

AquaWrench R/W - Part 2



Ok, so we can find kernel addresses for Symbols we want to modify

- What about VA -> PA?
- https://github.com/FuzzySecurity/Sharp-Suite/tree/master/VirtToPhys

Kernel R/W, what now

We have to be careful in ring0 but we own the OS

- Patch data structures
- Modify security constants
- Repurpose execution vectors or existing memory to run arbitrary code

Protected Processes

New extended model since in Windows 8.1

- EPROCESS.Protection > PS_PROTECTION
- Use code integrity to allow only trusted code to load into a process

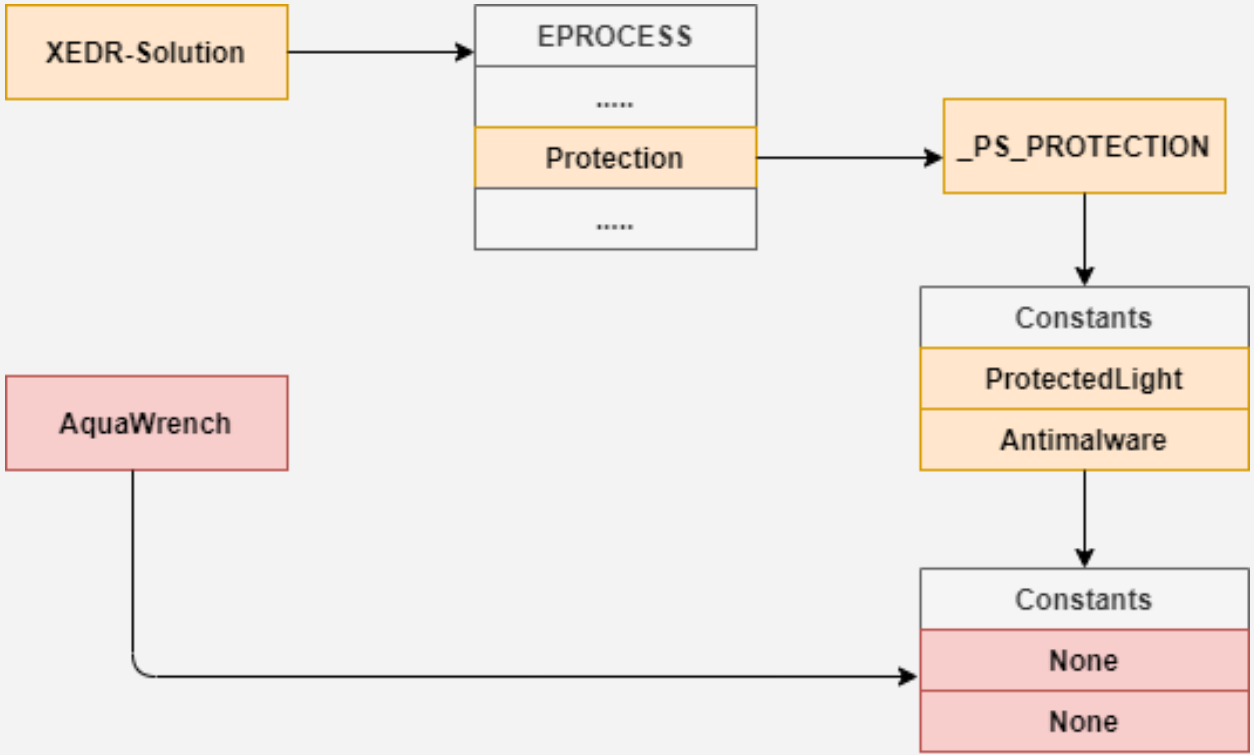
Effective, you can only open a handle with ≥ protection level

- Not a security boundary, can you guess why? 

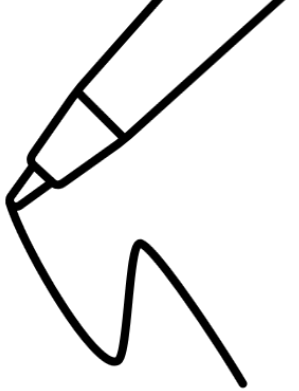
Commonly used to safeguard critical OS components and EDR solutions

Demo 1

For access; the attacker can modify his protection or change the protection of the target process



What if we need more?

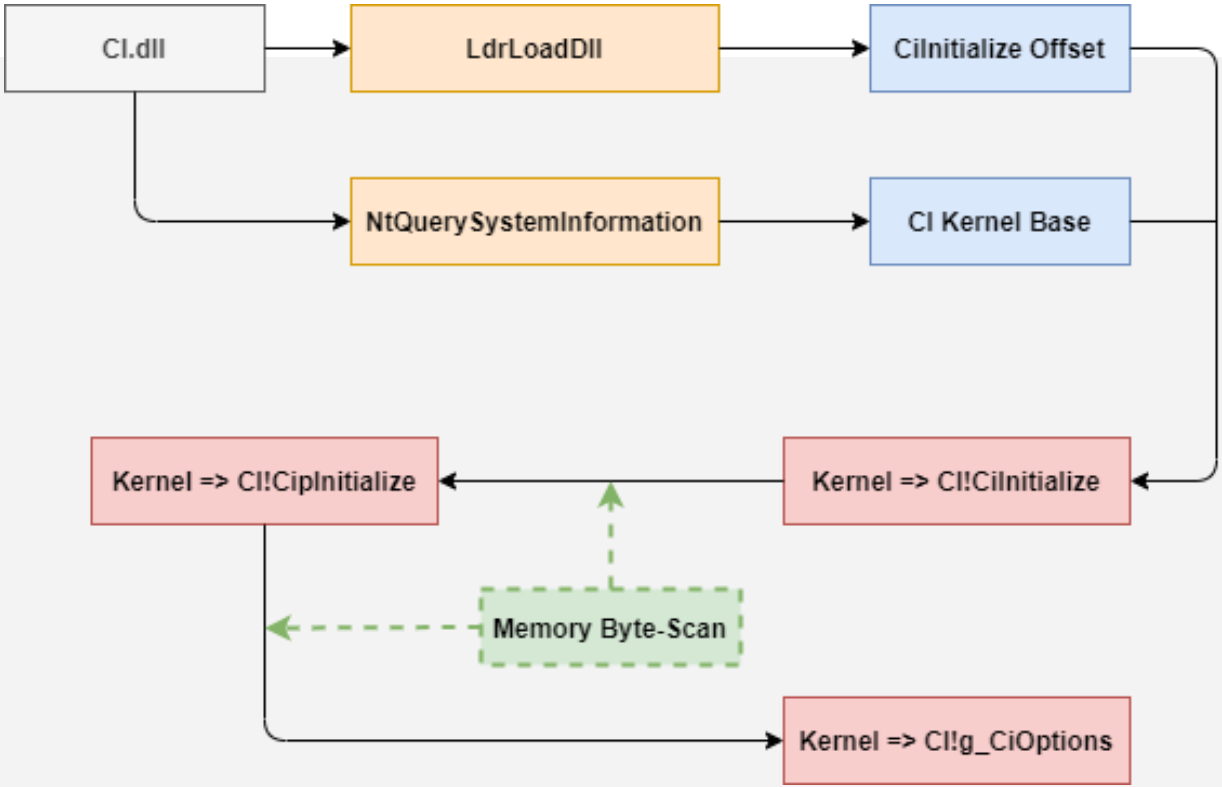


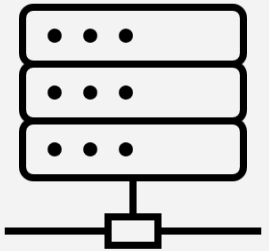
Zapping Kernel memory is fine but it would be great if we could write and load our own driver modules.

- We can toggle Driver Signature Enforcement (DSE) on/off
  - Win 7 ⟶ nt!g_CiEnable
  - Win 8+ ⟶ ci!g_CiOptions
  - Tricky to find
  - Caught while modified ⟶ PatchGuard BSOD

Demo 2

(ノ☉_☉)ノ彡┻┻



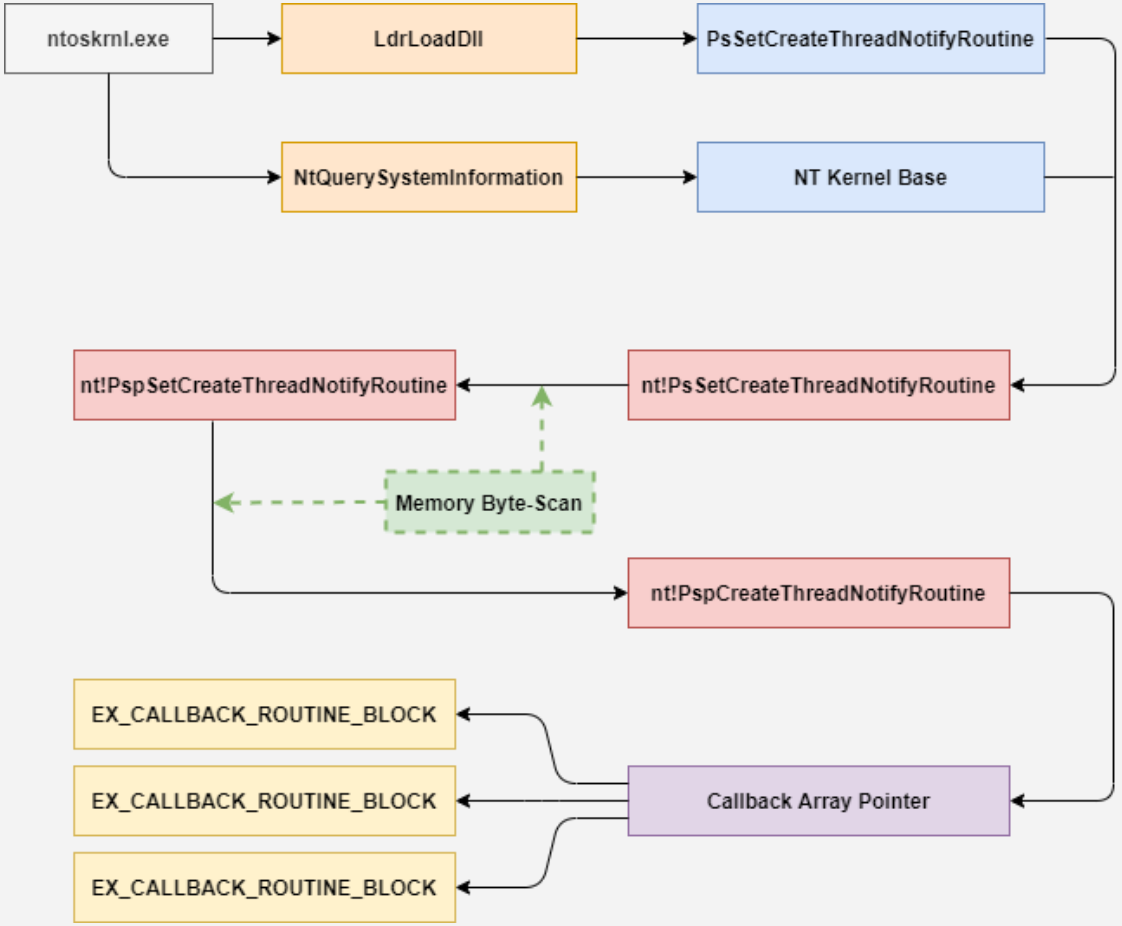


You can't catch what you don't see!

As attackers we would prefer is to generate as little signal as possible

Two simple examples:

- ⟶ Hooking NtTraceEvent, the gatekeeper for ETW (credit to @_batsec)
- ⟶ Zeroing out Kernel Notification routines (credit to @brsn76945860)



Demo 3

Invisible things alone
are the things
that shall remain

Pretty spooky right?

Current driver payloads

- Driver payload using InfinityHook to sinkhole all ETW event data

Reflective driver loading is in development

- Dynamically write shellcode to driver memory space
- Patch NtGdi* syscalls for code exec, these wrap simple function pointers (@Blomster81 & @j00ru)
- Requires special driver boilerplate code

Todo

- Integrate reflective WinDivert payload & write bindings
- Plunder capabilities from 3rd party frameworks (e.g. BlackBone)

MSFT is well aware of these techniques

↳ We are borrowing heavily from Kernel exploitation

Data Corruption Protection

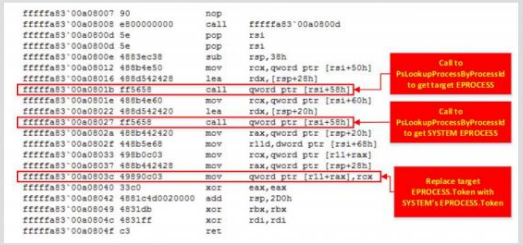
Introducing: Kernel Data Protection

Problem: Kernel exploits in Windows leverage data corruption to obtain privilege escalation

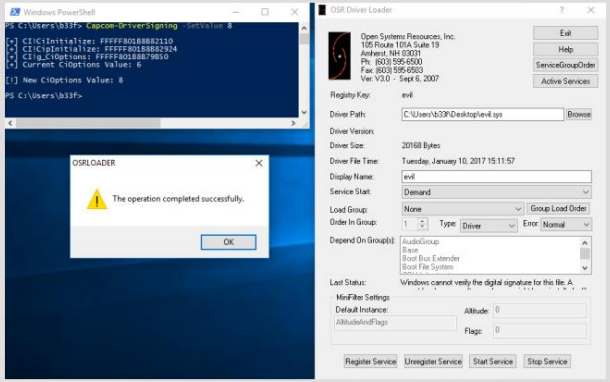
Current State: Hypervisor-based code integrity prevents dynamic code injection and enforces signing policy

Prevent code injection is not enough, kernel has many sensitive data structures

Kernel Data Protection (KDP) uses Secure Kernel to enforce immutability

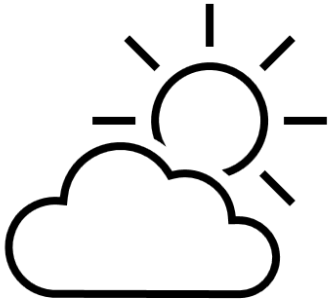


CVE-2016-7256 exploit: Open type font elevation of privilege



Corrupting Code Integrity Globals (credit: FuzzySec)

~ Dave Weston (@dwizzzleMSFT) Bluehat Shanghai 2019



Nothing New Under The Sun

Hypervisor-Protected Code Integrity (HVCI) stops many of these attacks

You really should strictly control the driver inventory on your estate

- Device Guard driver whitelisting will prevent any unauthorized driver loads
- Sysmon EventID 6 can record all driver load events

References

KDU (@hFireF0X)
https://github.com/hfiref0x/KDU

VirtToPhys (@FuzzySec)
https://github.com/FuzzySecurity/Sharp-Suite/tree/master/VirtToPhys

Spectre (@BillDemirkapi)
https://github.com/d4stiny/spectre

TelemetrySourcerer (@Jackson_T)
https://github.com/jthuraisamy/TelemetrySourcerer

Sysmon Enumeration Overview (@dotslashroot)
https://ackroute.com/post/2017/08/08/sysmon-enumeration-overview/

Universally Evading Sysmon and ETW (@_batsec_)
https://blog.dylan.codes/evading-sysmon-and-windows-event-logging/

Removing Kernel Callbacks Using Signed Drivers (@brsn76945860)
https://br-sn.github.io/Removing-Kernel-Callbacks-Using-Signed-Drivers/

Mimidrv In Depth: Exploring Mimikatz's Kernel Driver (@matterpreter)
https://posts.specterops.io/mimidrv-in-depth-4d273d19e148

I Got 99 Problem But a Kernel Pointer Ain't One (@aionescu)
https://recon.cx/2013/slides/Recon2013-Alex%20Ionescu-I%20got%2099%20problems%20but%20a%20kernel%20pointer%20ain't%20one.pdf

Exploiting a Windows 10 PagedPool off-by-one overflow (@j00ru)
https://j00ru.vexillium.org/2018/07/exploiting-a-windows-10-pagedpool-off-by-one/

A quick insight into the Driver Signature Enforcement (@j00ru)
https://j00ru.vexillium.org/2010/06/insight-into-the-driver-signature-enforcement/

Unraveling the Lamberts Toolkit
https://securelist.com/unraveling-the-lamberts-toolkit/77990/

Conclusions