PENTEST HACKFEST 2022

RED OPS
INFORMATION SECURITY

How to tamper an EDR?

Master of Puppets

Join us in Arlington, VA or Live Online for **FREE**

SUMMIT: Nov 14–15 | TRAINING: Nov 16–21

SANS

Whoami

Daniel Feichter from Austria / Tyrol / Innsbruck

- Originally Industrial Engineer
  - 12 years experience in electronics and IT
- 4 years in infosec industry
- Founder RedOps GmbH (formerly Infosec Tirol)

Focus on offensive security:

- APT-test development and APT-simulation
- Endpoint security product testing
- Penetration testing
- Red teaming
- Endpoint security research, mostly antivirus & EDR

Disclaimer

- Only personal research / experience
- No claims to completeness
- EDR functionality on Windows (no zero days!)
  - Key activities require a privileged user
- Refer to EDRs with antivirus module -> EPP/EDR
- Applies to multiple products on Windows
- Vendor neutrality

We take a closer look at

- **ATT&CK T1562.001 ->** Impair Defenses: Disable or Modify Tools

- **Disable main functionalities from EDR, without relying on:**
  - EDR uninstall password / token
  - Using any uninstall software
  - Uninstalling EDR in general
  - Using Windows Security Center

- Similar seen in the wild, by **AvosLocker Ransomware**

We want to achieve

- **First Step**
  - Closer look EDR Windows user space and kernel space components
  - Functionality and relationship between them

- **Second step -> tamper EDR and** permanently get rid of:

Antivirus capabilities

Prevention based on user space API-hooking and callback telemetry collection

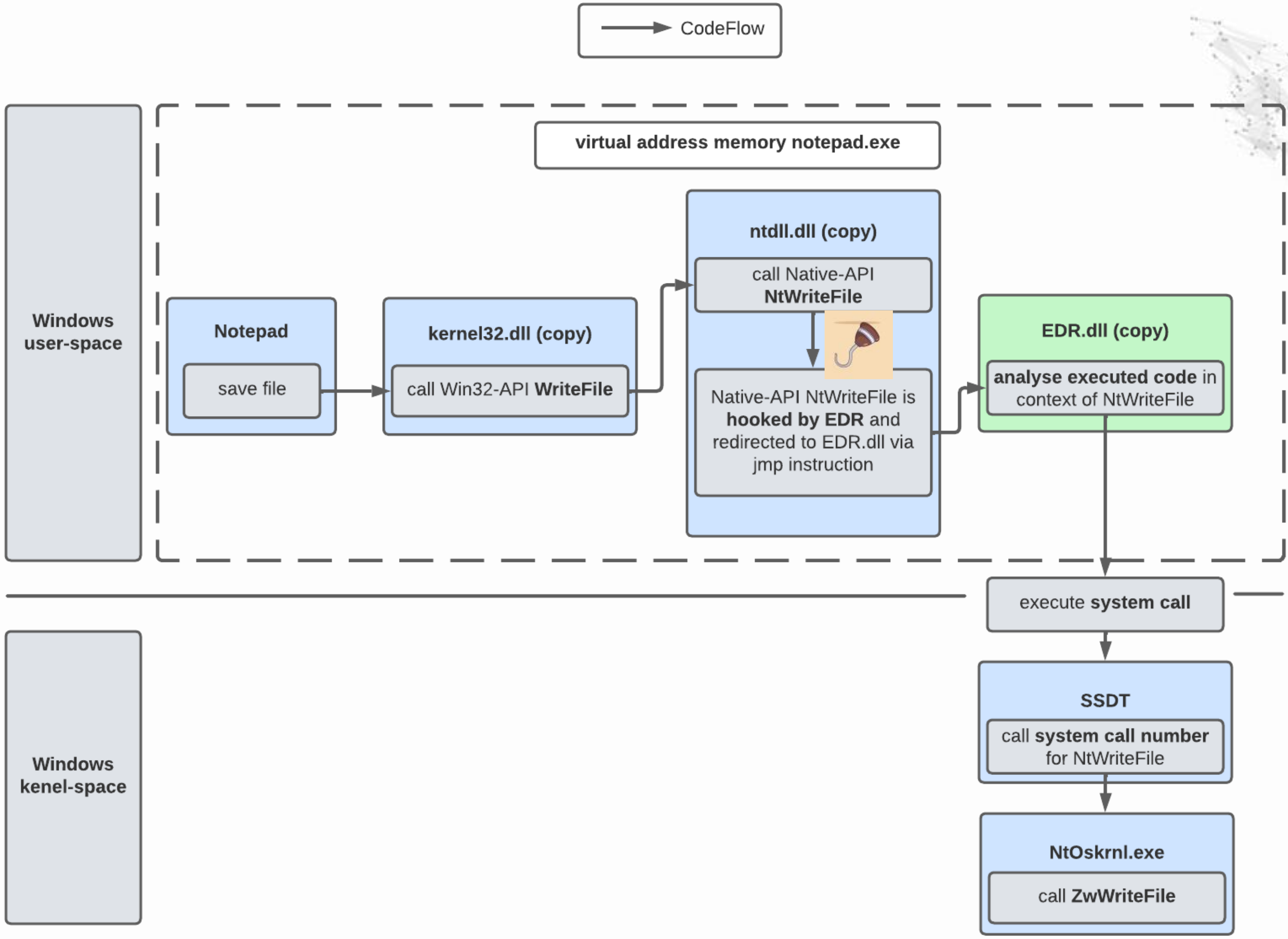
EDR capabilities

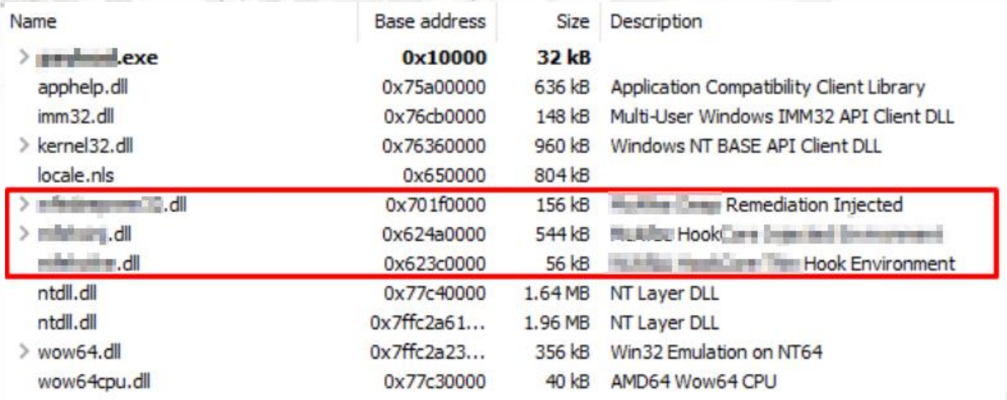
Detections based on user space API-hooking and callback telemetry collection

EDR web console capabilities

Host isolation; Real time response shell; sensor recovery

API-Hooking?


CodeFlow
Windows user-space
virtual address memory notepad.exe
Notepad
save file
kernel32.dll (copy)
call Win32-API WriteFile
ntdll.dll (copy)
call Native-API NtWriteFile
Native-API NtWriteFile is hooked by EDR and redirected to EDR.dll via jmp instruction
EDR.dll (copy)
analyse executed code in context of NtWriteFile
execute system call
Windows kenel-space
SSDT
call system call number for NtWriteFile
NtOskrnl.exe
call ZwWriteFile



Name	Base address	Size	Description
[redacted].exe	**0x10000**	**32 kB**	
apphelp.dll	0x75a00000	636 kB	Application Compatibility Client Library
imm32.dll	0x76cb0000	148 kB	Multi-User Windows IMM32 API Client DLL
kernel32.dll	0x76360000	960 kB	Windows NT BASE API Client DLL
locale.nls	0x650000	804 kB	
[redacted].dll	0x701f0000	156 kB	[redacted] Remediation Injected
[redacted].dll	0x624a0000	544 kB	[redacted] Hook[redacted]
[redacted].dll	0x623c0000	56 kB	[redacted] Hook Environment
ntdll.dll	0x77c40000	1.64 MB	NT Layer DLL
ntdll.dll	0x7ffc2a61...	1.96 MB	NT Layer DLL
wow64.dll	0x7ffc2a23...	356 kB	Win32 Emulation on NT64
wow64cpu.dll	0x77c30000	40 kB	AMD64 Wow64 CPU

Give me a scenario

- **Red team engagement**
  - Initial access: phishing or similar
  - Achieved privileged user rights: exploit or misconfiguration
  - Explore process structure -> additional useful user session open

T1003.001

OS credential dumping: LSASS memory

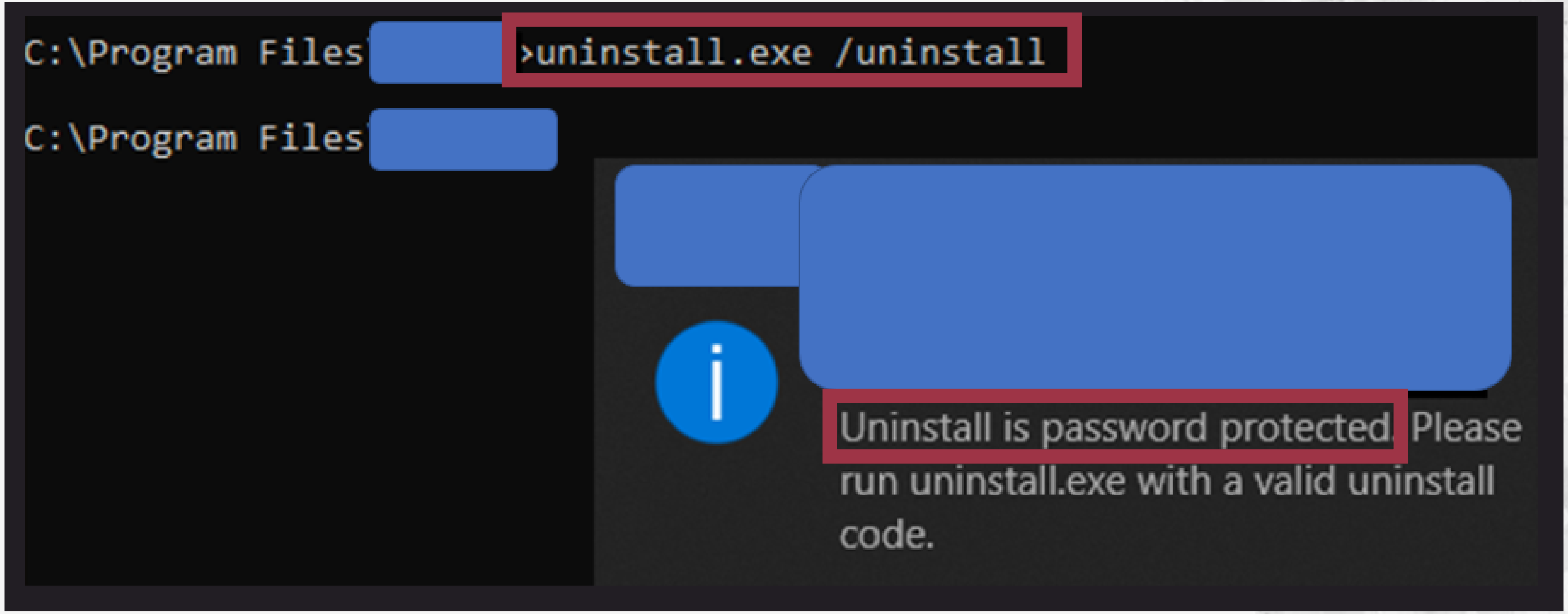
T1134.001

Access token manipulation: token impersonation/theft

- **But, installed EDR is tough!** -> Beginning of EDR tampering journey

Come on, I am already admin

- Despite privileged user, most EDRs still annoying
- Uninstallation is password protected
  - Won’t rely on uninstall password or token!



```
C:\Program Files[redacted]>uninstall.exe /uninstall

C:\Program Files[redacted]
```

[redacted]

[redacted]

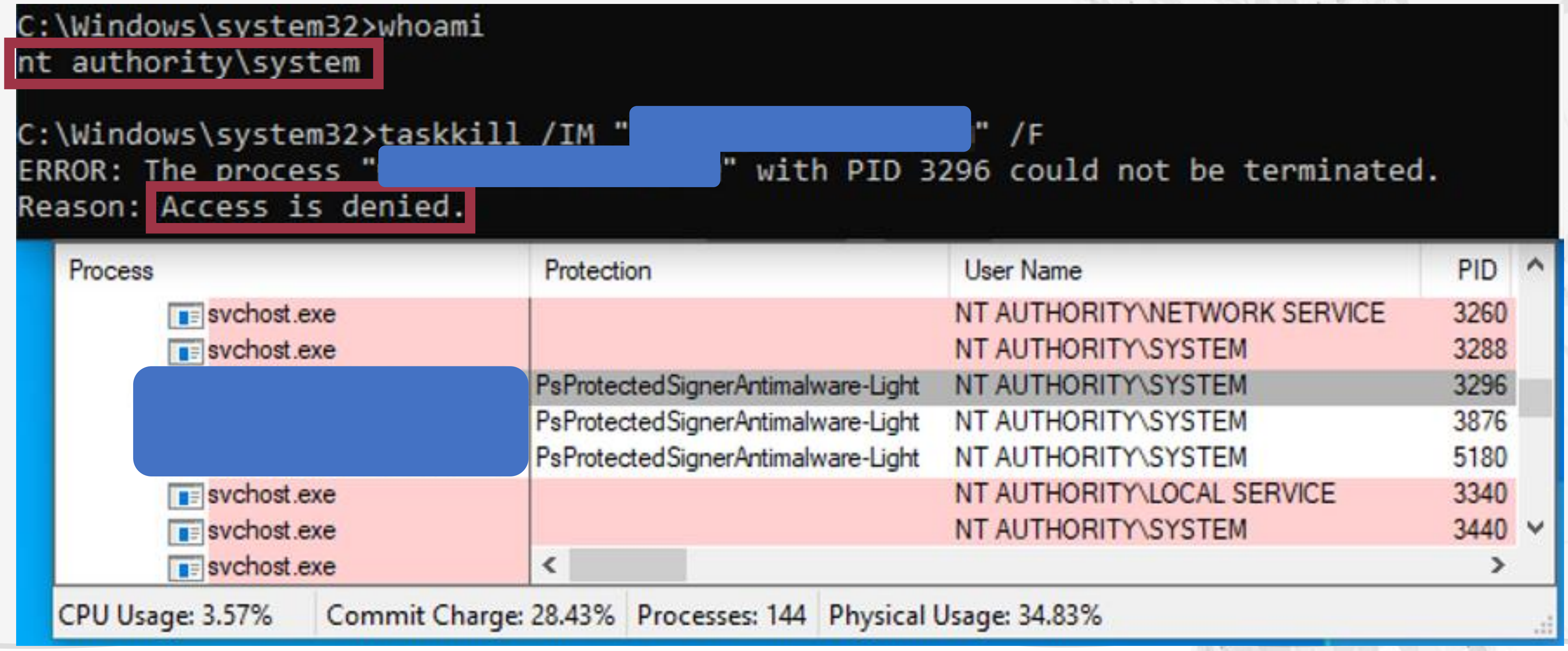
Uninstall is password protected. Please run uninstall.exe with a valid uninstall code.

User space

First step: EDR processes

User-space: EDR processes

- Normally, initialized as Protected Process Light (PPL)
- **Despite system integrity**, process termination not allowed



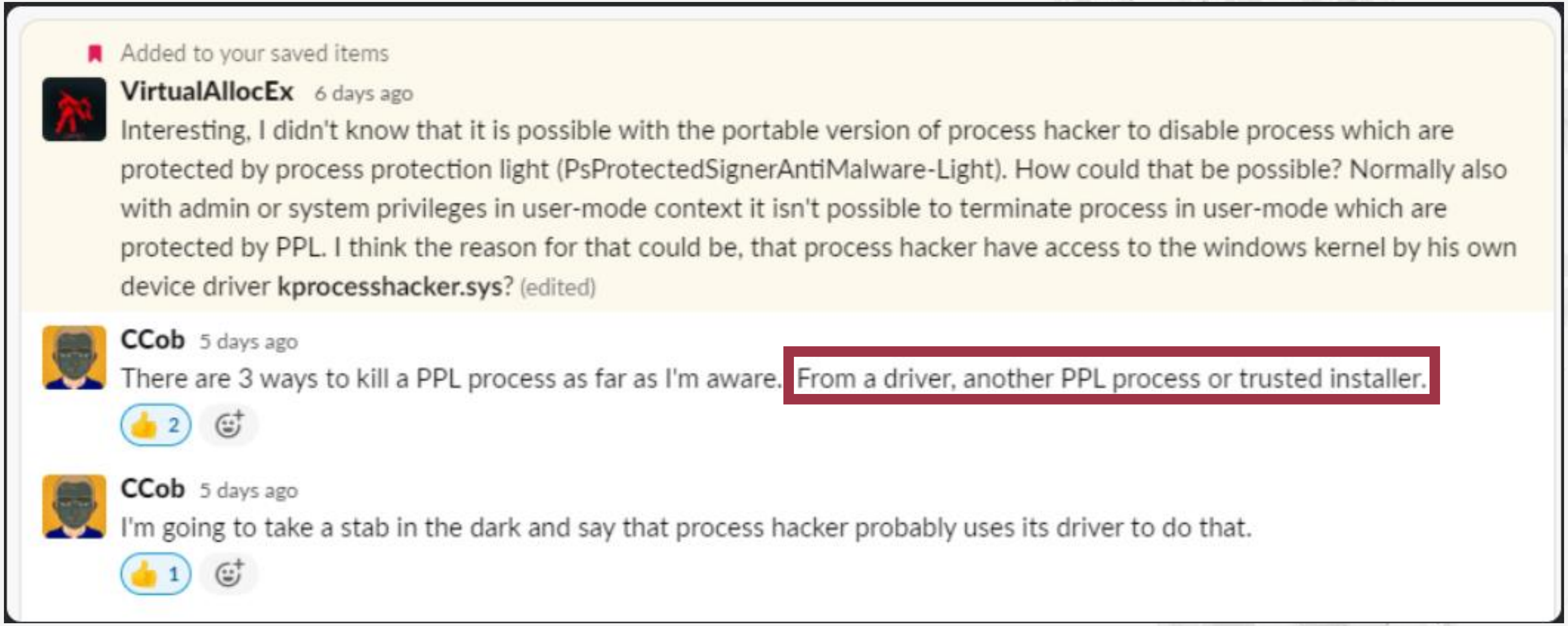
```
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>taskkill /IM "[redacted]" /F
ERROR: The process "[redacted]" with PID 3296 could not be terminated.
Reason: Access is denied.
```

Process	Protection	User Name	PID
svchost.exe		NT AUTHORITY\NETWORK SERVICE	3260
svchost.exe		NT AUTHORITY\SYSTEM	3288
[redacted]	PsProtectedSignerAntimalware-Light	NT AUTHORITY\SYSTEM	3296
[redacted]	PsProtectedSignerAntimalware-Light	NT AUTHORITY\SYSTEM	3876
[redacted]	PsProtectedSignerAntimalware-Light	NT AUTHORITY\SYSTEM	5180
svchost.exe		NT AUTHORITY\LOCAL SERVICE	3340
svchost.exe		NT AUTHORITY\SYSTEM	3440
svchost.exe			

CPU Usage: 3.57% Commit Charge: 28.43% Processes: 144 Physical Usage: 34.83%

EDR processes: disable PPL

- Signed vulnerable (device) driver -> **RTCore64 CVE 2019-16098**
- Creds to **@EthicalChaos**



Added to your saved items

VirtualAllocEx 6 days ago

Interesting, I didn't know that it is possible with the portable version of process hacker to disable process which are protected by process protection light (PsProtectedSignerAntiMalware-Light). How could that be possible? Normally also with admin or system privileges in user-mode context it isn't possible to terminate process in user-mode which are protected by PPL. I think the reason for that could be, that process hacker have access to the windows kernel by his own device driver **kprocesshacker.sys**? (edited)

CCob 5 days ago

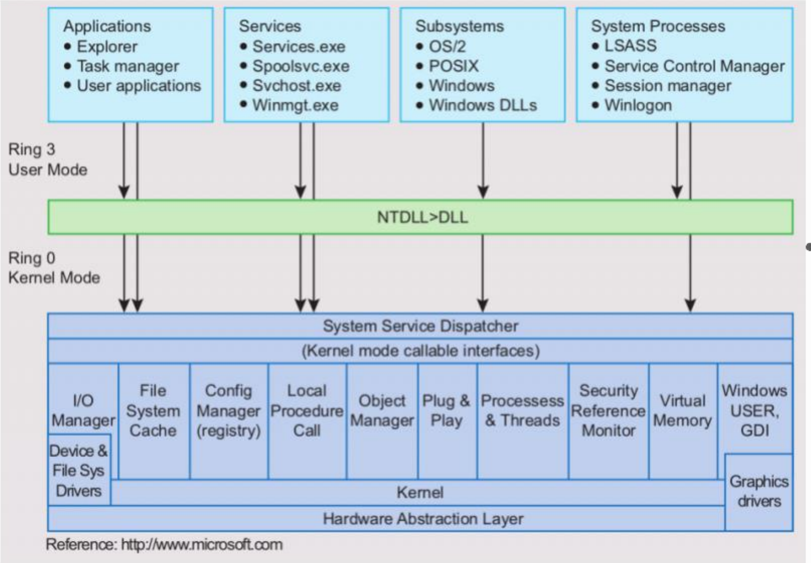
There are 3 ways to kill a PPL process as far as I'm aware. From a driver, another PPL process or trusted installer.

CCob 5 days ago

I'm going to take a stab in the dark and say that process hacker probably uses its driver to do that.

EDR processes: disable PPL

Windows
user space



https://www.aldeid.com/wiki/File:User-kernel-space.png

https://dragon-ball-super.fandom.com/

Unprivileged user
(Medium Integrity)

Local Privilege Escalation
For example PrintNightmare

https://www.deviantart.com/

Privileged user
(High or System Integrity)

Enter kernel-space
Loading vulnerable Driver mimidrv or rtcore64

Privileged user
in kernel space
(Ring 0)

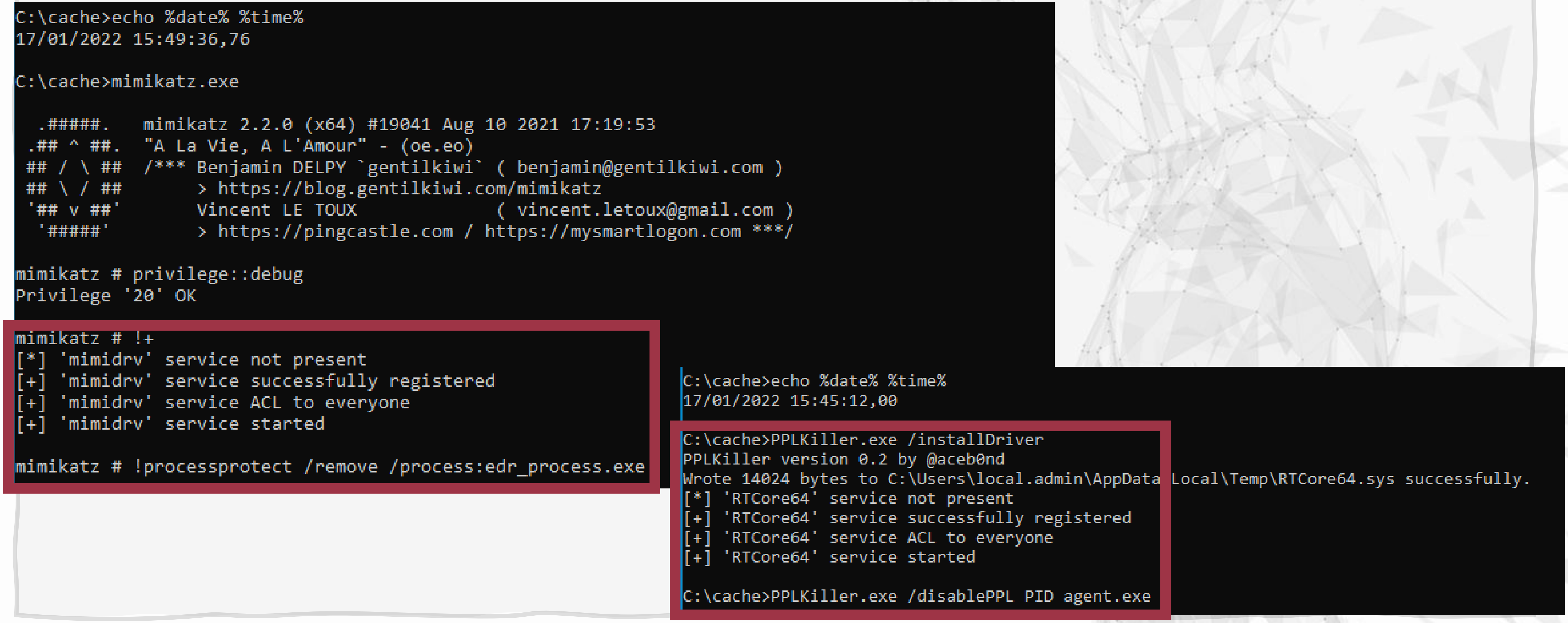
https://www.gamestop.at/

Loaded vulnerable Device Driver
acts as an bridge to access kernel space, also as an unprivileged user from user space (medium integrity Null DACL)

Windows
kernel space

User-space: EDR process tampering

- Tool Time -> **PPL Killer** -> driver rtcore64.sys or **Mimikatz** -> mimidrv.sys

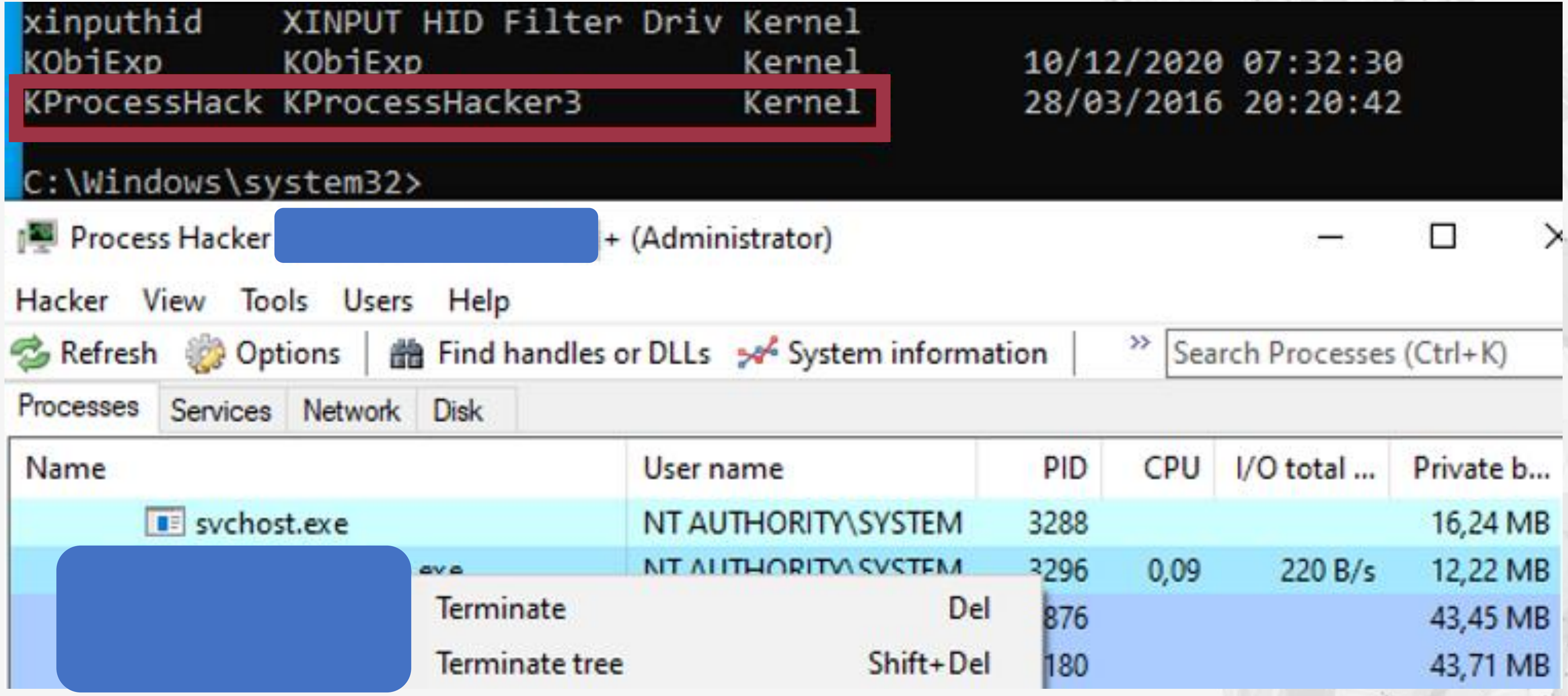


```
C:\cache>echo %date% %time%
17/01/2022 15:49:36,76

C:\cache>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processprotect /remove /process:edr_process.exe
```

```
C:\cache>echo %date% %time%
17/01/2022 15:45:12,00

C:\cache>PPLKiller.exe /installDriver
PPLKiller version 0.2 by @aceb0nd
Wrote 14024 bytes to C:\Users\local.admin\AppData Local\Temp\RTCore64.sys successfully.
[*] 'RTCore64' service not present
[+] 'RTCore64' service successfully registered
[+] 'RTCore64' service ACL to everyone
[+] 'RTCore64' service started

C:\cache>PPLKiller.exe /disablePPL PID agent.exe
```

User-space: EDR process tampering

- Tool Time -> execute **Process Hacker** as privileged user



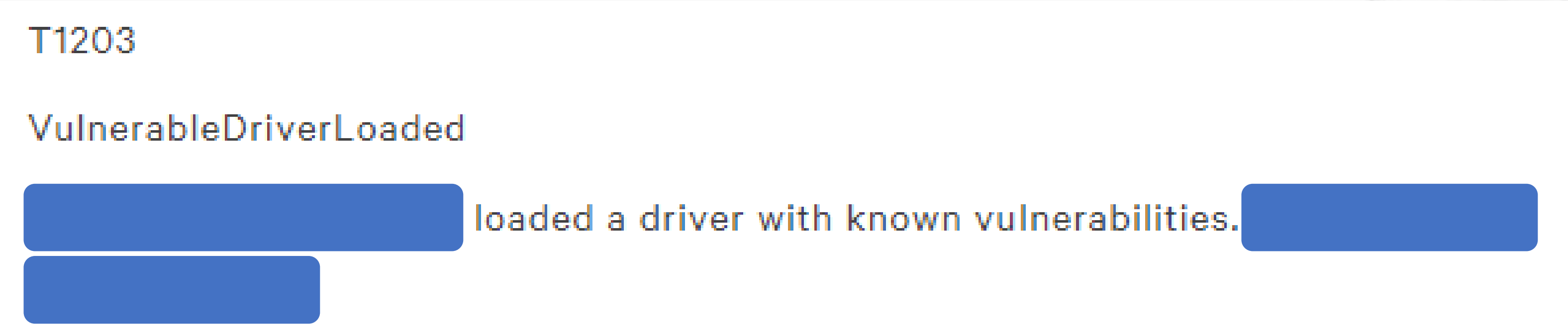
```
xinputhid    XINPUT HID Filter Driv Kernel
KObjExp      KObjExp                Kernel              10/12/2020 07:32:30
KProcessHack KProcessHacker3        Kernel              28/03/2016 20:20:42

C:\Windows\system32>
```

Process Hacker [redacted] + (Administrator)

Name	User name	PID	CPU	I/O total ...	Private b...
svchost.exe	NT AUTHORITY\SYSTEM	3288			16,24 MB
[redacted] .exe	NT AUTHORITY\SYSTEM	3296	0,09	220 B/s	12,22 MB
		876			43,45 MB
		180			43,71 MB

User-space: EDR process tampering

- EDR vendors are aware -> improving their products
  - Started to **blacklist** and **block known drivers with vulnerabilities**
  - Depending on product, bypassing is necessary

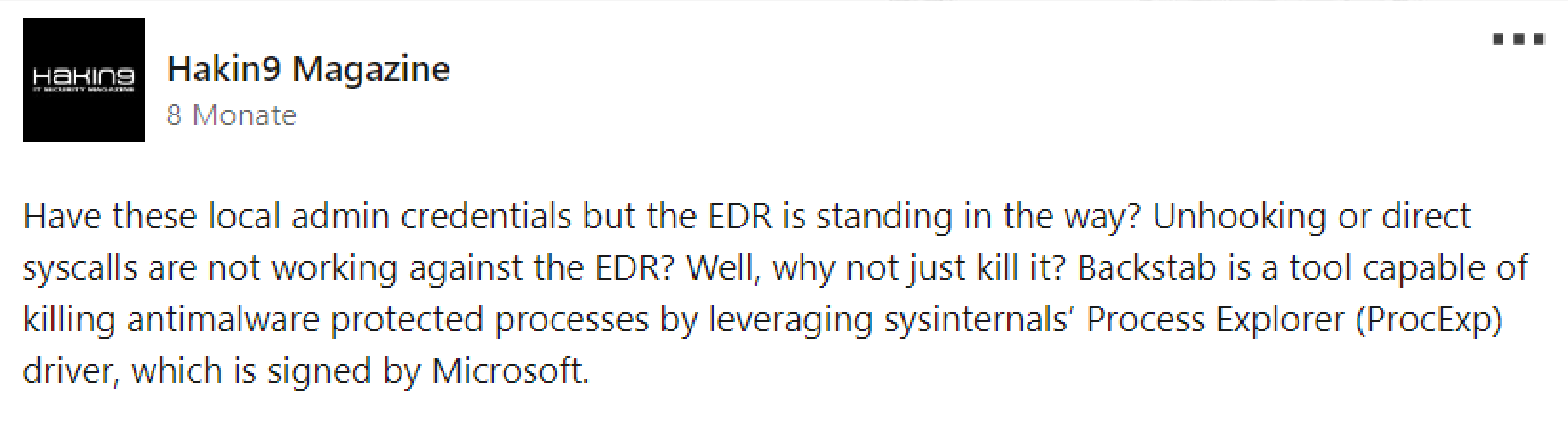


T1203

VulnerableDriverLoaded

[redacted] loaded a driver with known vulnerabilities. [redacted]

User-space: EDR process tampering



Hakin9 Magazine

8 Monate

Have these local admin credentials but the EDR is standing in the way? Unhooking or direct syscalls are not working against the EDR? Well, why not just kill it? Backstab is a tool capable of killing antimalware protected processes by leveraging sysinternals' Process Explorer (ProcExp) driver, which is signed by Microsoft.

Reference: https://www.linkedin.com/feed/update/urn:li:activity:6902622063433986048/

Process termination

Only temporary, gets restarted again and again

Process terminated

Even between gap, process was terminated and gets restarted, EDR works fine

EDR Killed?

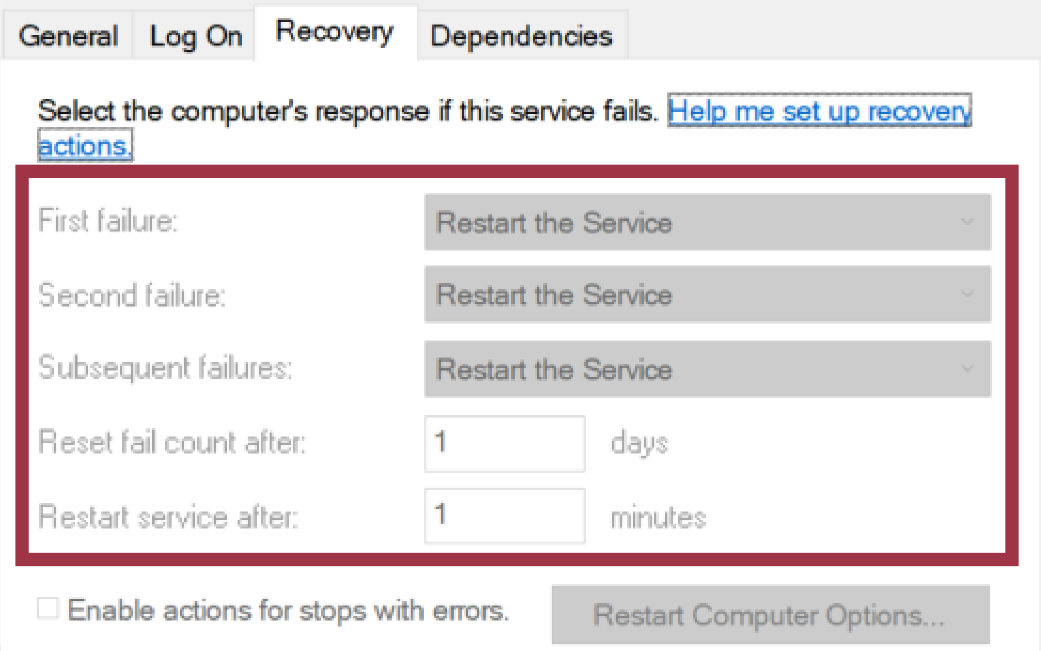
Much to less to get temporary or permanently rid of an EDR!

User space

Second step: EDR services

User-space: EDR services

- Identify EDR service, connected to EDR PPL process
- EDR user space service + EDR user space process = **EDR user space component**
- **Responsible for restarting** terminated PPL EDR process(es)



Select the computer's response if this service fails. Help me set up recovery actions.

First failure: Restart the Service

Second failure: Restart the Service

Subsequent failures: Restart the Service

Reset fail count after: 1 days

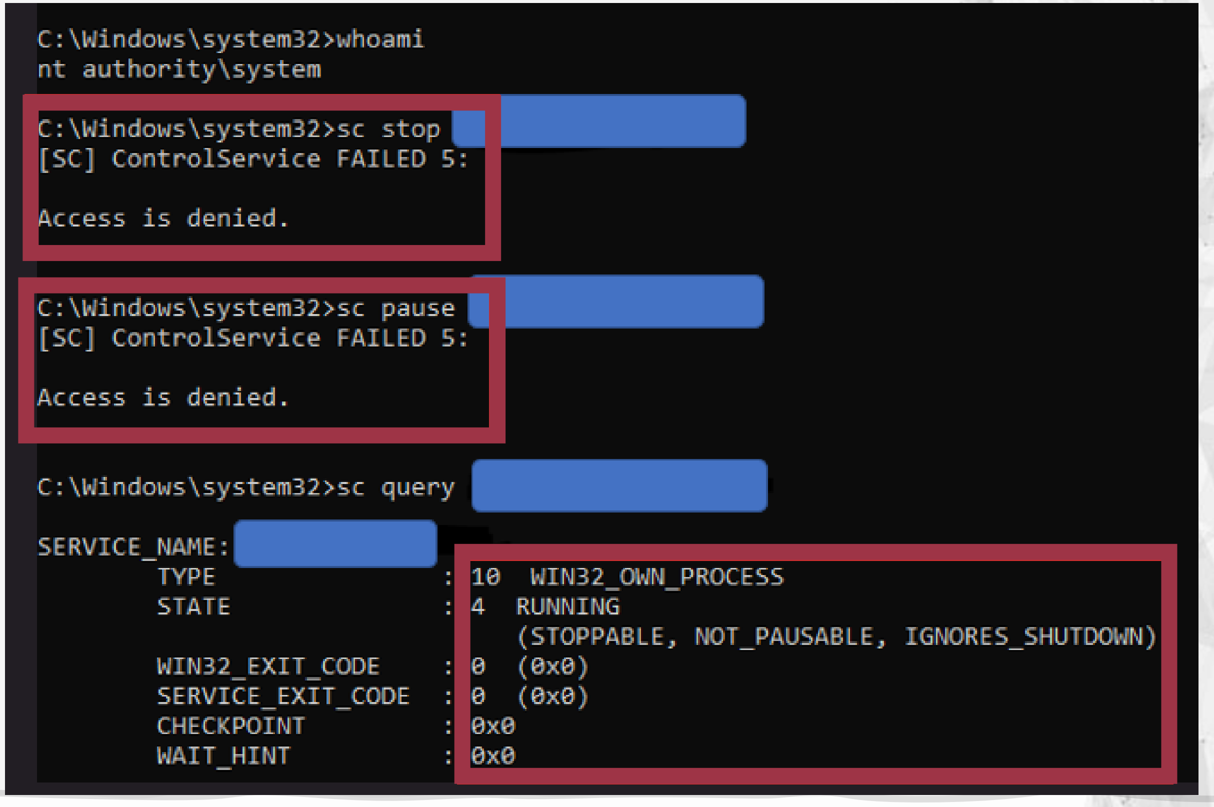
Restart service after: 1 minutes

□ Enable actions for stops with errors.

User-space: EDR services

- Initialization as protected service by **ELAM driver**

- **Despite system integrity**, not possible (also not temporary) to pause, stop, disable etc.



```
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>sc stop [redacted]
[SC] ControlService FAILED 5:

Access is denied.

C:\Windows\system32>sc pause [redacted]
[SC] ControlService FAILED 5:

Access is denied.

C:\Windows\system32>sc query [redacted]

SERVICE_NAME: [redacted]
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 4  RUNNING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

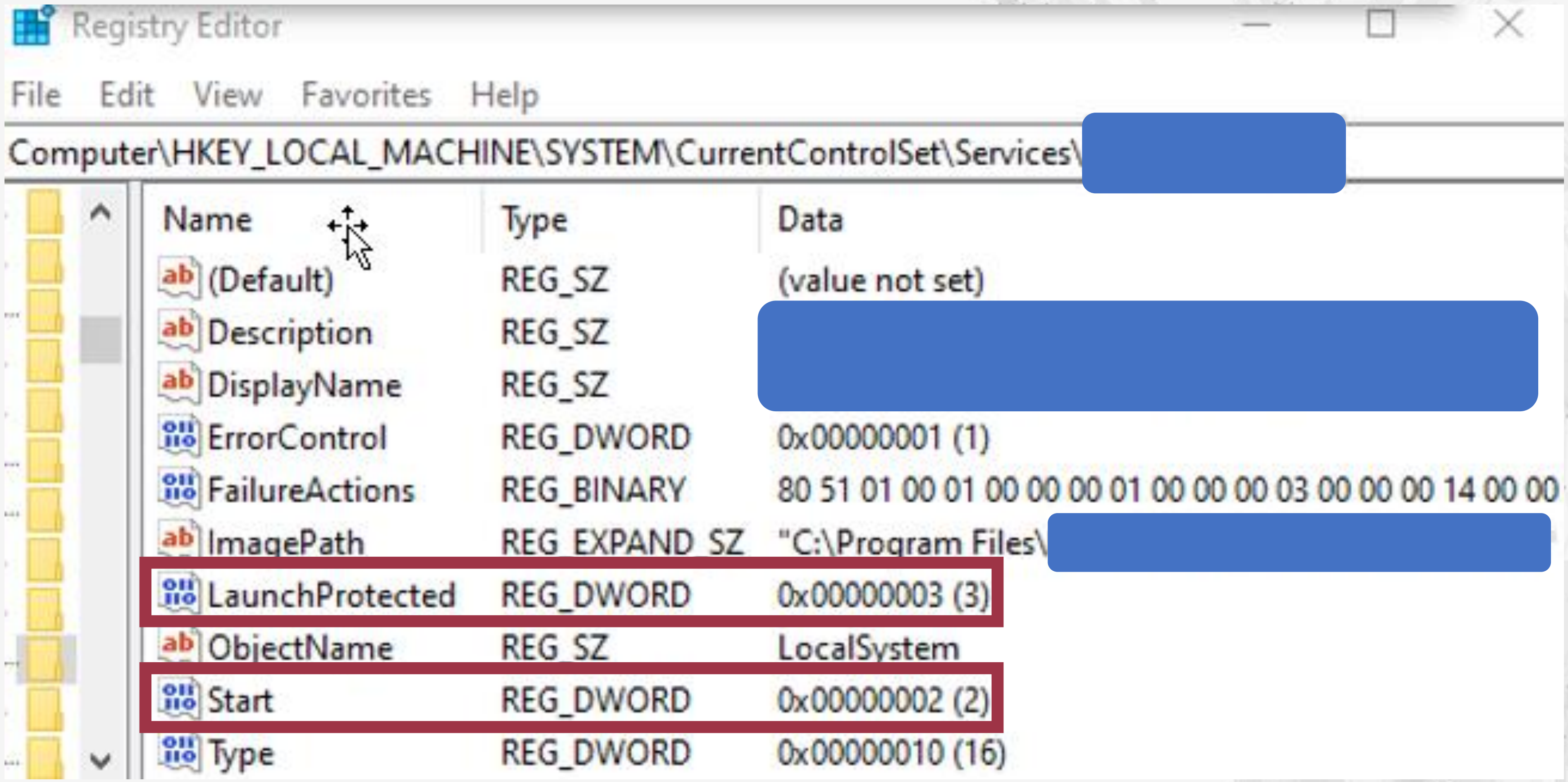
SANS OFFENSIVE OPERATIONS

User space

Third step: EDR registry keys

User-space: EDR registry keys

- Identify **reg keys / sub keys / entries** from EDR user space component (service)

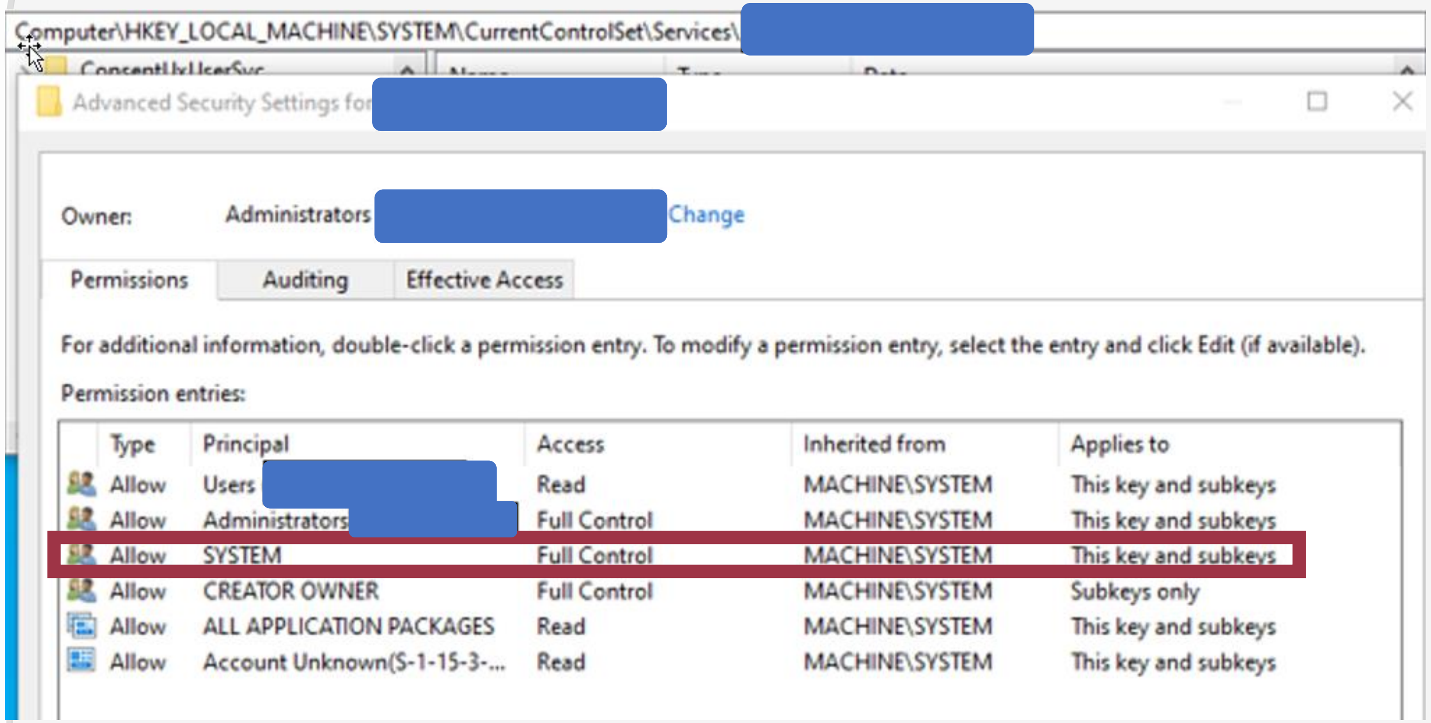


Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\[redacted]

Name	Type	Data
(Default)	REG_SZ	(value not set)
Description	REG_SZ	[redacted]
DisplayName	REG_SZ	[redacted]
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	80 51 01 00 01 00 00 00 01 00 00 00 03 00 00 00 14 00 00
ImagePath	REG_EXPAND_SZ	"C:\Program Files\[redacted]
LaunchProtected	REG_DWORD	0x00000003 (3)
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16)

User-space: EDR registry tampering

- **Start entry ->** value 2 = **autoload** and value 4 = **disabled**
- Try to tamper start entry -> tamper protection -> despite system integrity not possible



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ [redacted]

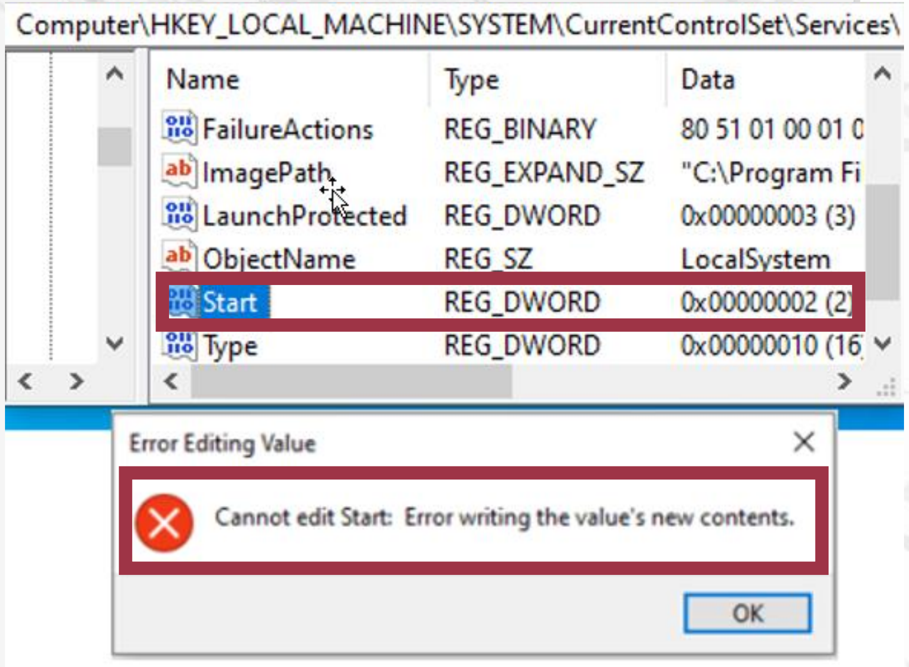
Advanced Security Settings for [redacted]

Owner: Administrators [redacted] Change

For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	Users [redacted]	Read	MACHINE\SYSTEM	This key and subkeys
Allow	Administrators [redacted]	Full Control	MACHINE\SYSTEM	This key and subkeys
Allow	SYSTEM	Full Control	MACHINE\SYSTEM	This key and subkeys
Allow	CREATOR OWNER	Full Control	MACHINE\SYSTEM	Subkeys only
Allow	ALL APPLICATION PACKAGES	Read	MACHINE\SYSTEM	This key and subkeys
Allow	Account Unknown(S-1-15-3-...	Read	MACHINE\SYSTEM	This key and subkeys



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\

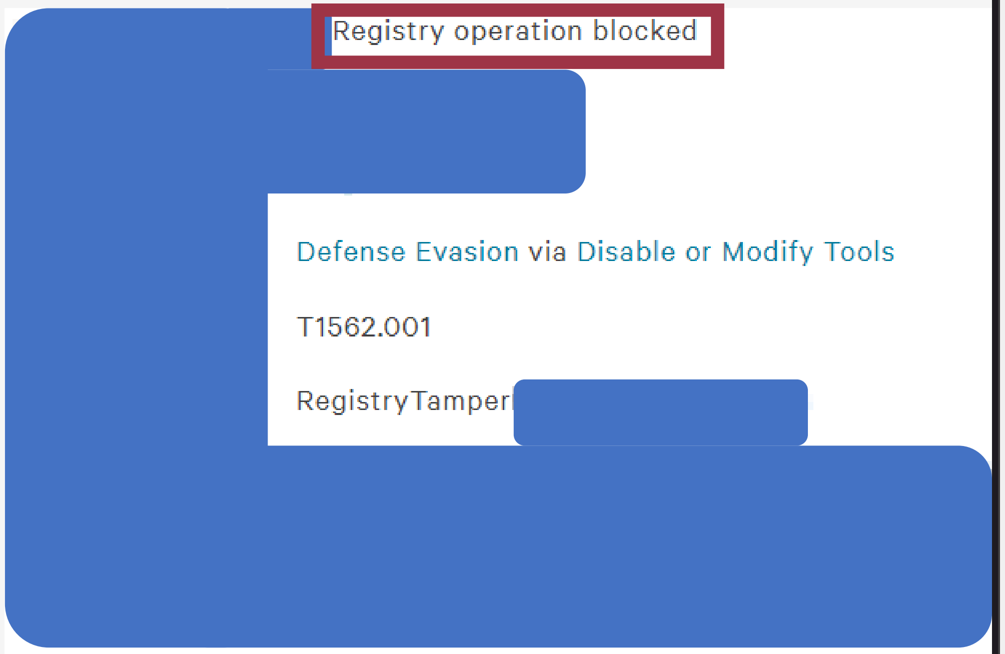
Name	Type	Data
FailureActions	REG_BINARY	80 51 01 00 01 0
ImagePath	REG_EXPAND_SZ	"C:\Program Fi
LaunchProtected	REG_DWORD	0x00000003 (3)
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16

Error Editing Value

Cannot edit Start: Error writing the value's new contents.

User-space: EDR registry tampering

- Depending on product -> we (possibly) create **tamper protection alerts**



Registry operation blocked

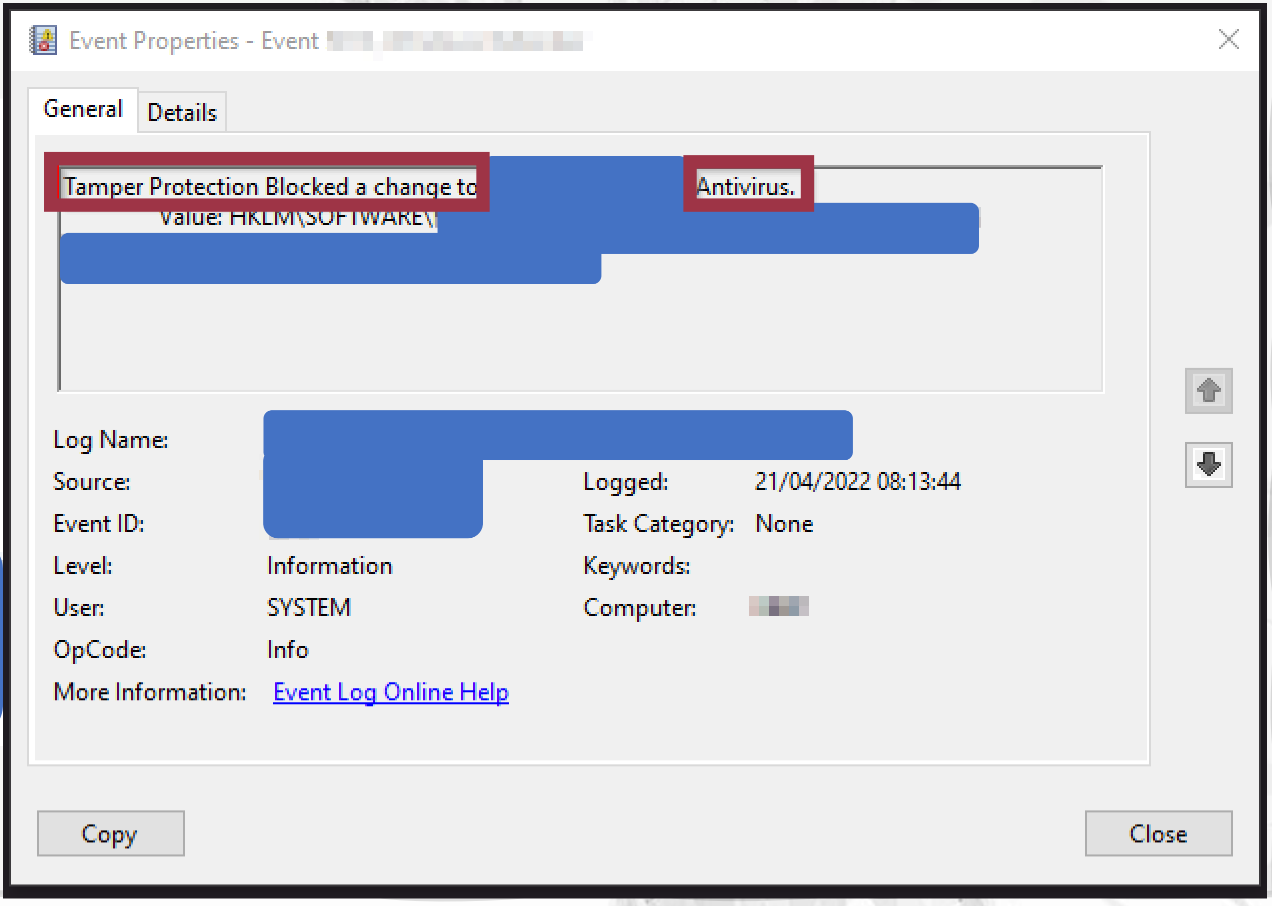
[redacted]

Defense Evasion via Disable or Modify Tools

T1562.001

RegistryTamper[redacted]

[redacted]



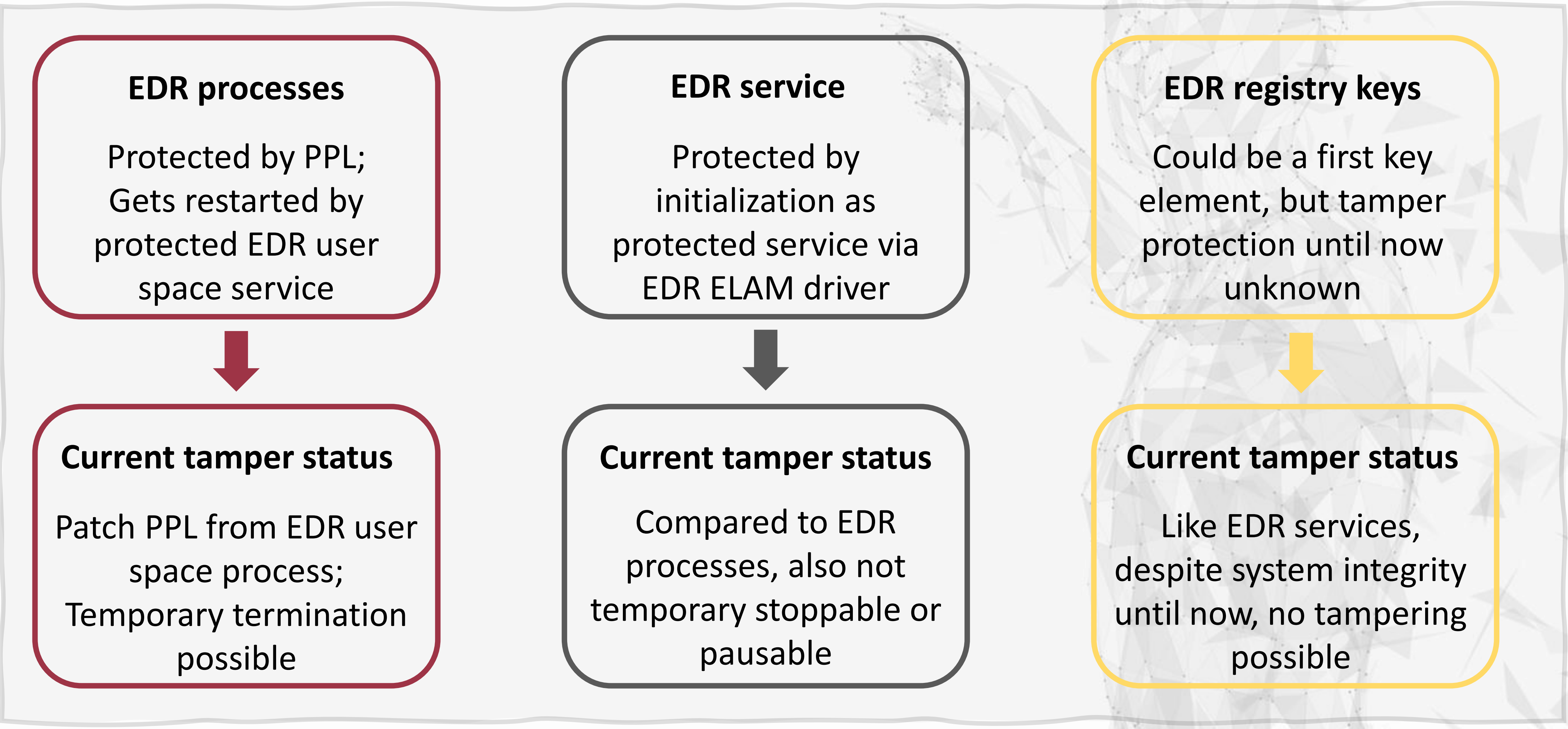
Event Properties - Event [redacted]

Tamper Protection Blocked a change to [redacted] Antivirus.
Value: HKLM\SOFTWARE\[redacted]

Log Name: [redacted]
Source: [redacted]
Event ID: [redacted]
Level: Information
User: SYSTEM
OpCode: Info
More Information: Event Log Online Help

Logged: 21/04/2022 08:13:44
Task Category: None
Keywords:
Computer: [redacted]

Interim status: EDR user space tampering



EDR processes

Protected by PPL; Gets restarted by protected EDR user space service

Current tamper status

Patch PPL from EDR user space process; Temporary termination possible

EDR service

Protected by initialization as protected service via EDR ELAM driver

Current tamper status

Compared to EDR processes, also not temporary stoppable or pausable

EDR registry keys

Could be a first key element, but tamper protection until now unknown

Current tamper status

Like EDR services, despite system integrity until now, no tampering possible

SANS
OFFENSIVE
OPERATIONS

Kernel space

Fourth step: Callback routines

Kernel-space: EDR callback routines

- **Kernel Patch Protection aka PatchGuard**
  - (Officially) hooks in kernel space not longer allowed
  - Forced to user space -> user space API hooking
  - Despite Patchguard, different kernel callbacks can be registered:

ProcessNotifyRoutine

User space DLL-injection / user space API-hooking

ThreadNotifyRoutine

Process injections

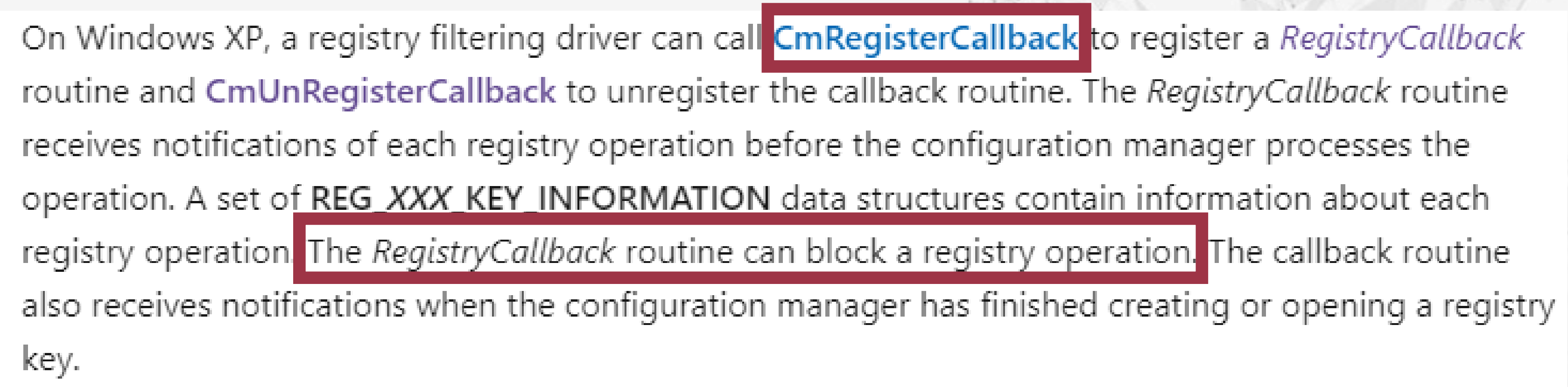
LoadImageNotify Routine

DLL mapping, suspicious image loading

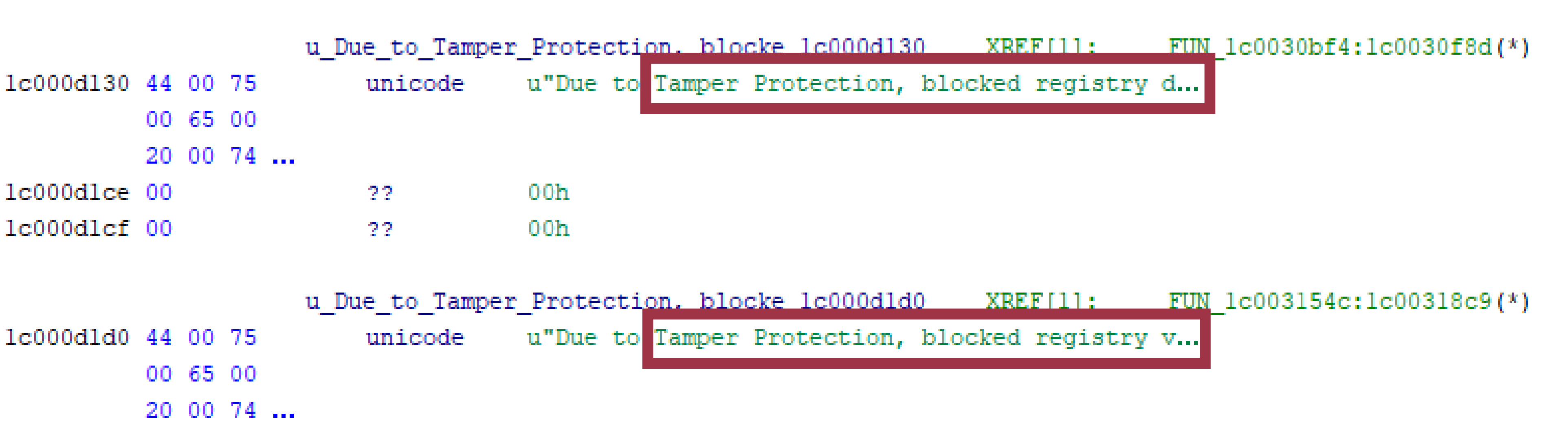
EDR sensor -> telemetry collection in general (processes, threads, images etc.)

Kernel-space: EDR callback routines

- Besides, used by EDRs to **protect their own registry keys** against tampering!



On Windows XP, a registry filtering driver can call CmRegisterCallback to register a *RegistryCallback* routine and CmUnRegisterCallback to unregister the callback routine. The *RegistryCallback* routine receives notifications of each registry operation before the configuration manager processes the operation. A set of REG_*XXX*_KEY_INFORMATION data structures contain information about each registry operation. The *RegistryCallback* routine can block a registry operation. The callback routine also receives notifications when the configuration manager has finished creating or opening a registry key.



```
                         u_Due_to_Tamper_Protection,_blocke_1c000d130     XREF[1]:     FUN_1c0030bf4:1c0030f8d(*)
1c000d130 44 00 75          unicode     u"Due to Tamper Protection, blocked registry d…
          00 65 00
          20 00 74 …
1c000d1ce 00                ??          00h
1c000d1cf 00                ??          00h

                         u_Due_to_Tamper_Protection,_blocke_1c000d1d0     XREF[1]:     FUN_1c003154c:1c00318c9(*)
1c000d1d0 44 00 75          unicode     u"Due to Tamper Protection, blocked registry v…
          00 65 00
          20 00 74 …
```

First demo: disable EDR user space compon.

- **Using gained knowledge to:**
  - Only **disable permanently** the **EDR user space component** and what’s the impact on:

Antivirus capabilities

Prevention based on user space API-hooking and callback telemetry collection

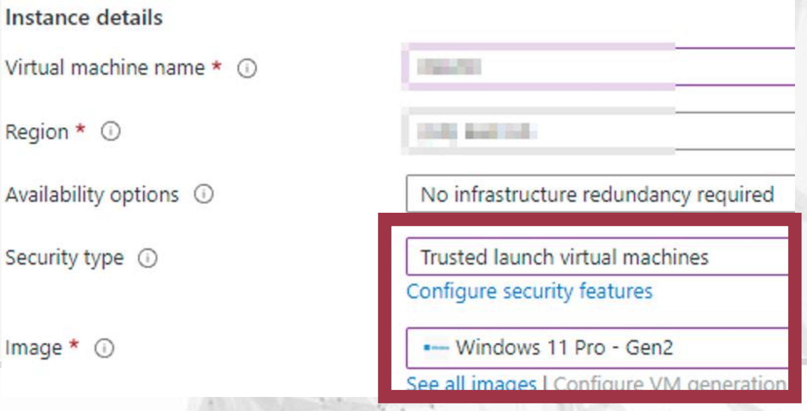
EDR capabilities

Detections based on user space API-hooking and callback telemetry collection

EDR web console capabilities

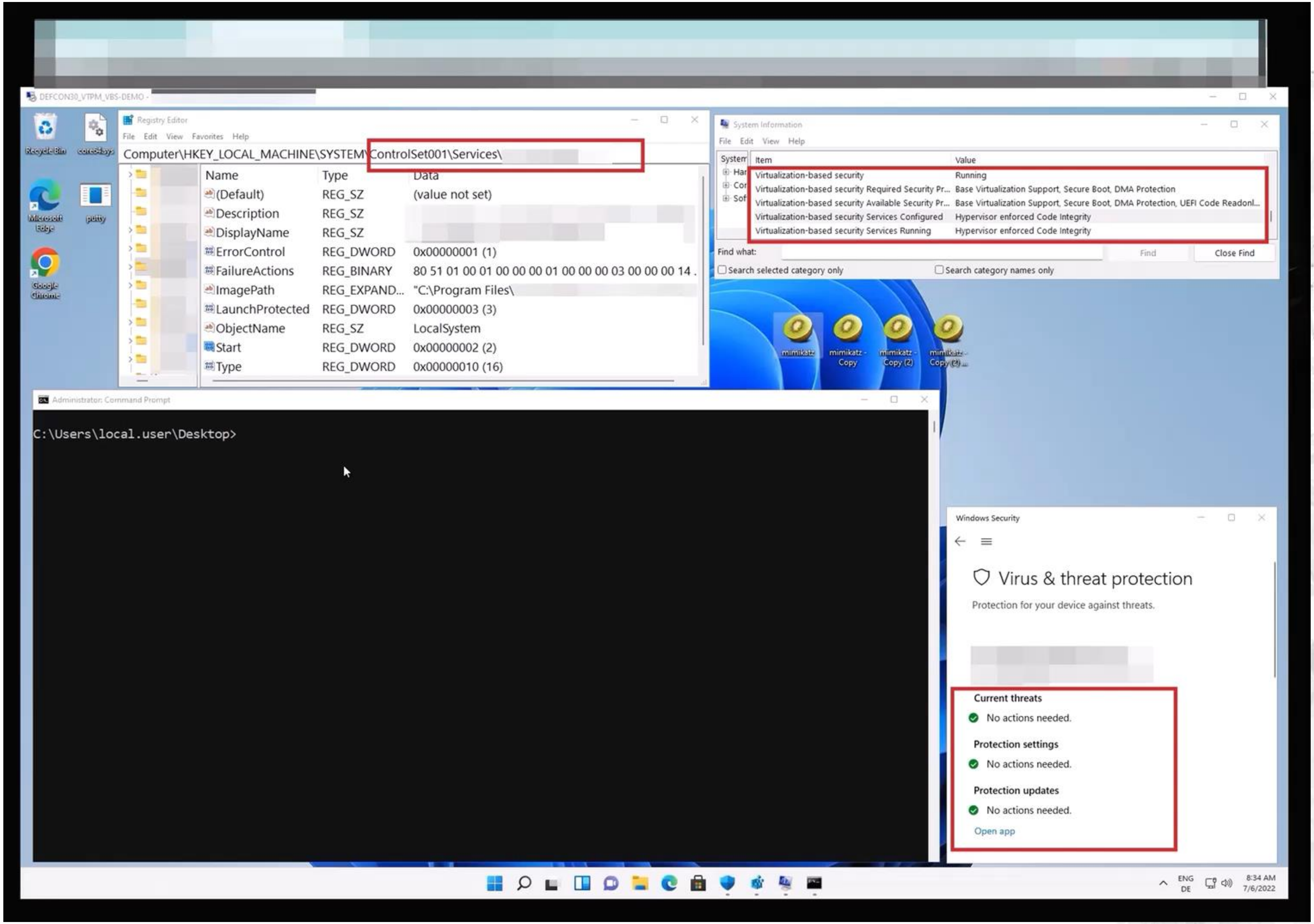
Host isolation; Real time response shell; sensor recovery

- **All creds** for the POC CheekyBlinder to @brsn76945860
- Have a look at his amazing blog https://br-sn.github.io/



Instance details

Virtual machine name * ⓘ	[redacted]
Region * ⓘ	[redacted]
Availability options ⓘ	No infrastructure redundancy required
Security type ⓘ	Trusted launch virtual machines Configure security features
Image * ⓘ	Windows 11 Pro - Gen2 See all images | Configure VM generation



Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\ [redacted]

Name	Type	Data
(Default)	REG_SZ	(value not set)
Description	REG_SZ	[redacted]
DisplayName	REG_SZ	[redacted]
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	80 51 01 00 01 00 00 00 01 00 00 00 03 00 00 00 14 .
ImagePath	REG_EXPAND...	"C:\Program Files\ [redacted]
LaunchProtected	REG_DWORD	0x00000003 (3)
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16)

Item	Value
Virtualization-based security	Running
Virtualization-based security Required Security Pr...	Base Virtualization Support, Secure Boot, DMA Protection
Virtualization-based security Available Security Pr...	Base Virtualization Support, Secure Boot, DMA Protection, UEFI Code Readonl...
Virtualization-based security Services Configured	Hypervisor enforced Code Integrity
Virtualization-based security Services Running	Hypervisor enforced Code Integrity

Find what: Find Close Find

□ Search selected category only □ Search category names only

```
C:\Users\local.user\Desktop>
```

Virus & threat protection

Protection for your device against threats.

[redacted]

Current threats

✓ No actions needed.

Protection settings

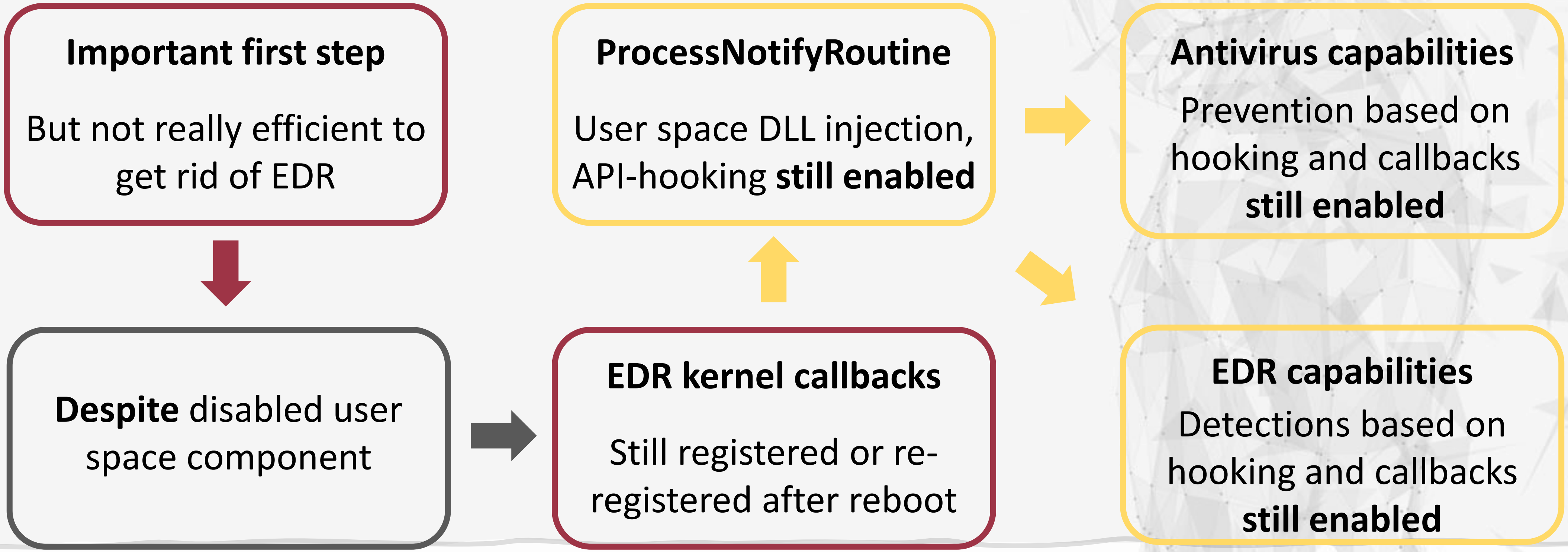
✓ No actions needed.

Protection updates

✓ No actions needed.

Conclusion: first demo

- **If write access kernel space:**
  - Patch EDR callbacks -> registry key tamper protection disabled -> set Start entry value 4 -> disable permanently EDR user space component:



Important first step

But not really efficient to get rid of EDR

Despite disabled user space component

EDR kernel callbacks

Still registered or re-registered after reboot

ProcessNotifyRoutine

User space DLL injection, API-hooking **still enabled**

Antivirus capabilities

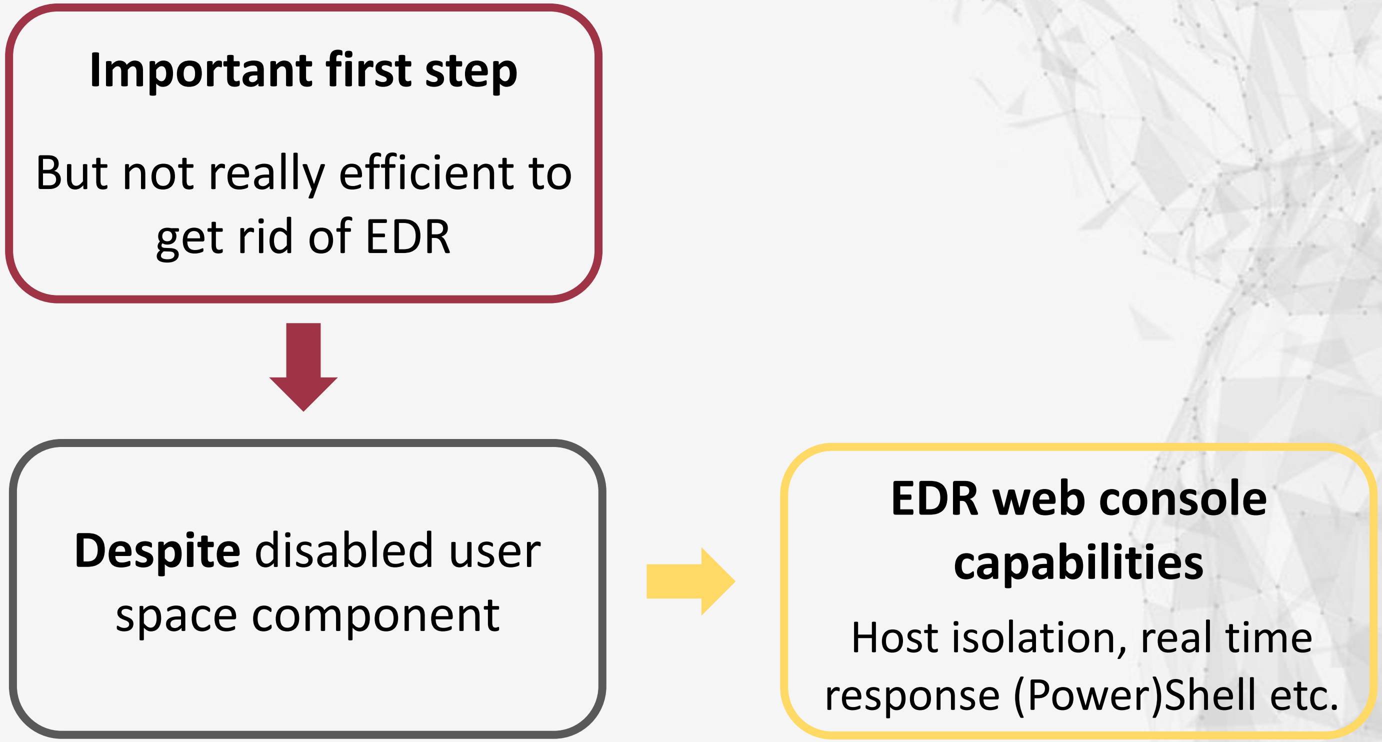
Prevention based on hooking and callbacks **still enabled**

EDR capabilities

Detections based on hooking and callbacks **still enabled**

Conclusion: first demo

- **EDR (web console) capabilities still enabled**



Important first step

But not really efficient to get rid of EDR

Despite disabled user space component

EDR web console capabilities

Host isolation, real time response (Power)Shell etc.

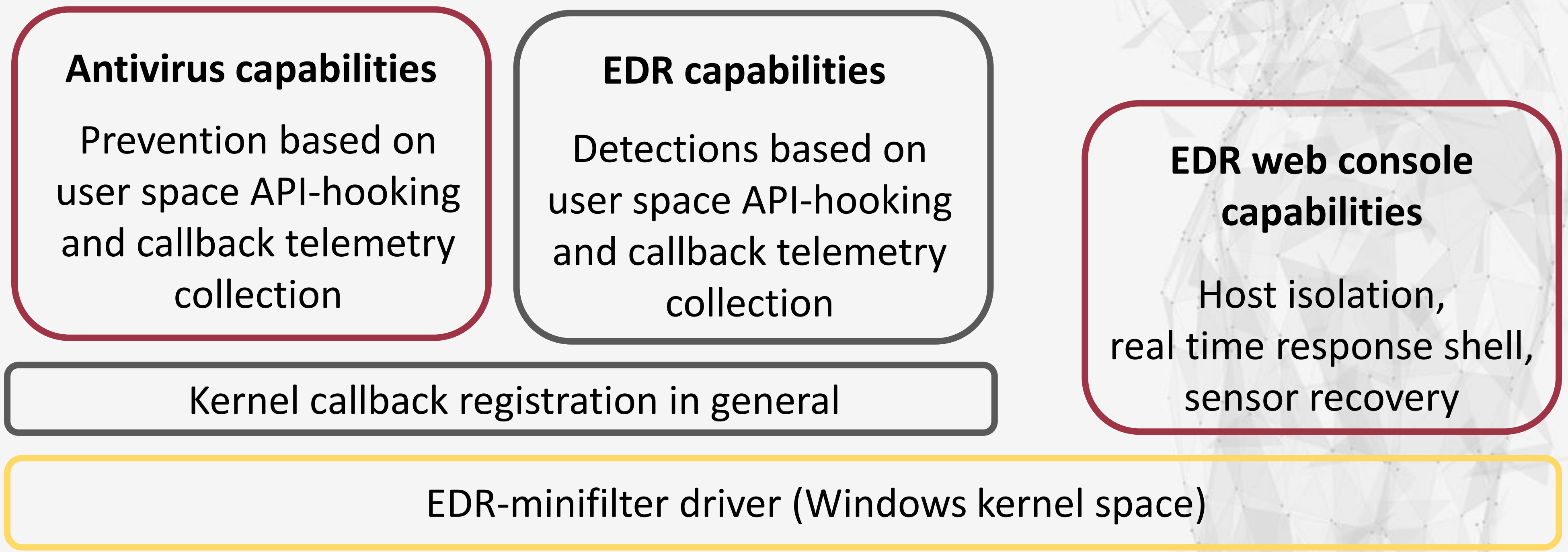
SANS
OFFENSIVE
OPERATIONS

Kernel space

Final step: Minifilter driver

Kernel-space: EDR minifilter driver

- Completely independent component -> our **key element**
  - Despite disabled user-space component enabled
  - Depending on product, responsible for:



Antivirus capabilities

Prevention based on user space API-hooking and callback telemetry collection

EDR capabilities

Detections based on user space API-hooking and callback telemetry collection

EDR web console capabilities

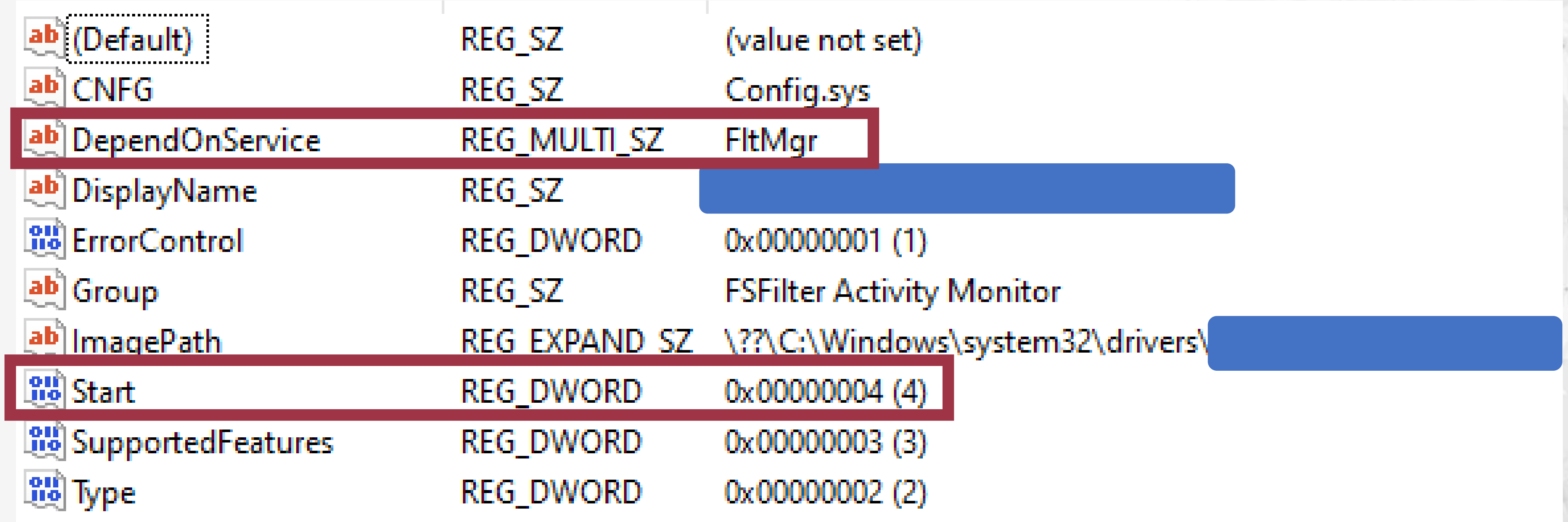
Host isolation, real time response shell, sensor recovery

Kernel callback registration in general

EDR-minifilter driver (Windows kernel space)

Kernel-space: EDR minifilter driver

- **How to get rid of?**
  - Minifilter has a **separate registry key**
  - Similar entries as EDR user space component reg key -> remember, **Start entry value 4**



(Default)	REG_SZ	(value not set)
CNFG	REG_SZ	Config.sys
DependOnService	REG_MULTI_SZ	FltMgr
DisplayName	REG_SZ	[redacted]
ErrorControl	REG_DWORD	0x00000001 (1)
Group	REG_SZ	FSFilter Activity Monitor
ImagePath	REG_EXPAND_SZ	\??\C:\Windows\system32\drivers\[redacted]
Start	REG_DWORD	0x00000004 (4)
SupportedFeatures	REG_DWORD	0x00000003 (3)
Type	REG_DWORD	0x00000002 (2)

Second demo: disable EDR minifilter driver

- **Using gained knowledge to:**
    - Only **permanently disable** initialization of **EDR minifilter driver** (kernel component)
    - EDR User space component stays enabled
    - What's the impact on:

Antivirus capabilities

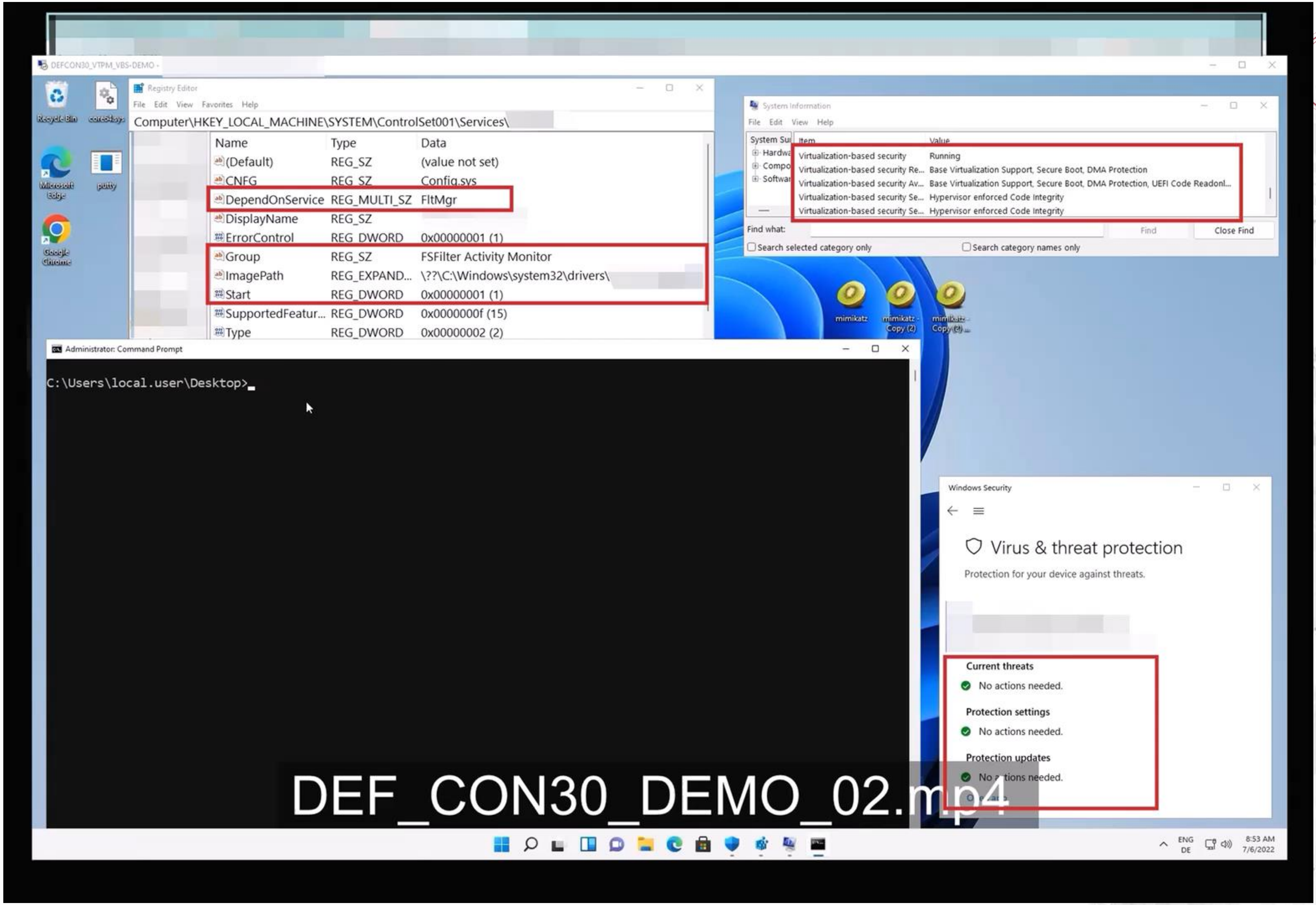
Prevention based on user space API-hooking and callback telemetry collection

EDR capabilities

Detections based on user space API-hooking and callback telemetry collection

EDR web console capabilities

Host isolation; Real time response shell; sensor recovery



Registry Editor

Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\[redacted]

Name	Type	Data
(Default)	REG_SZ	(value not set)
CNFG	REG_SZ	Config.sys
DependOnService	REG_MULTI_SZ	FltMgr
DisplayName	REG_SZ	
ErrorControl	REG_DWORD	0x00000001 (1)
Group	REG_SZ	FSFilter Activity Monitor
ImagePath	REG_EXPAND...	\??\C:\Windows\system32\drivers\[redacted]
Start	REG_DWORD	0x00000001 (1)
SupportedFeatur...	REG_DWORD	0x0000000f (15)
Type	REG_DWORD	0x00000002 (2)

System Information

Item	Value
Virtualization-based security	Running
Virtualization-based security Re...	Base Virtualization Support, Secure Boot, DMA Protection
Virtualization-based security Av...	Base Virtualization Support, Secure Boot, DMA Protection, UEFI Code Readonl...
Virtualization-based security Se...	Hypervisor enforced Code Integrity
Virtualization-based security Se...	Hypervisor enforced Code Integrity

Find what:

□ Search selected category only □ Search category names only

Administrator: Command Prompt

```
C:\Users\local.user\Desktop>_
```

Windows Security

Virus & threat protection

Protection for your device against threats.

[redacted]

Current threats

No actions needed.

Protection settings

No actions needed.

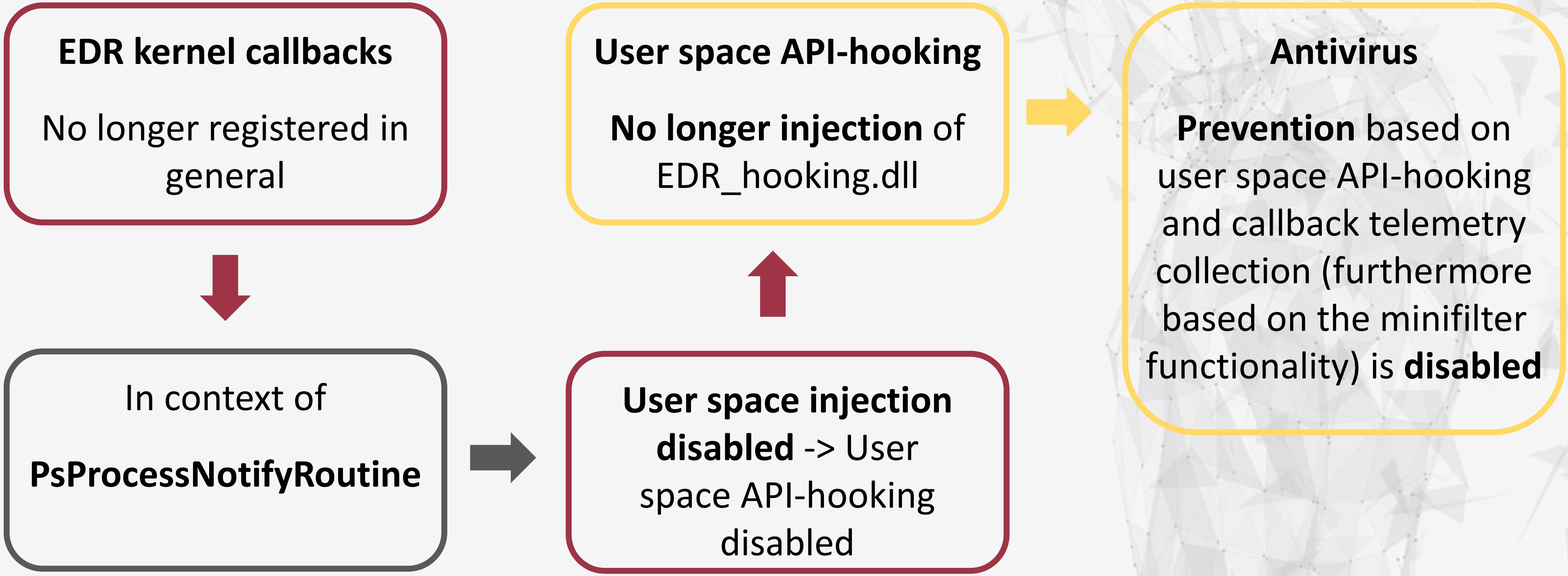
Protection updates

No actions needed.

DEF_CON30_DEMO_02.mp4

Conclusion: second demo

- Permanently **disabling EDR minifilter, much stronger impact!**
- Permanently impact on **antivirus capabilities:**



EDR kernel callbacks

No longer registered in general

In context of

PsProcessNotifyRoutine

User space injection disabled -> User space API-hooking disabled

User space API-hooking

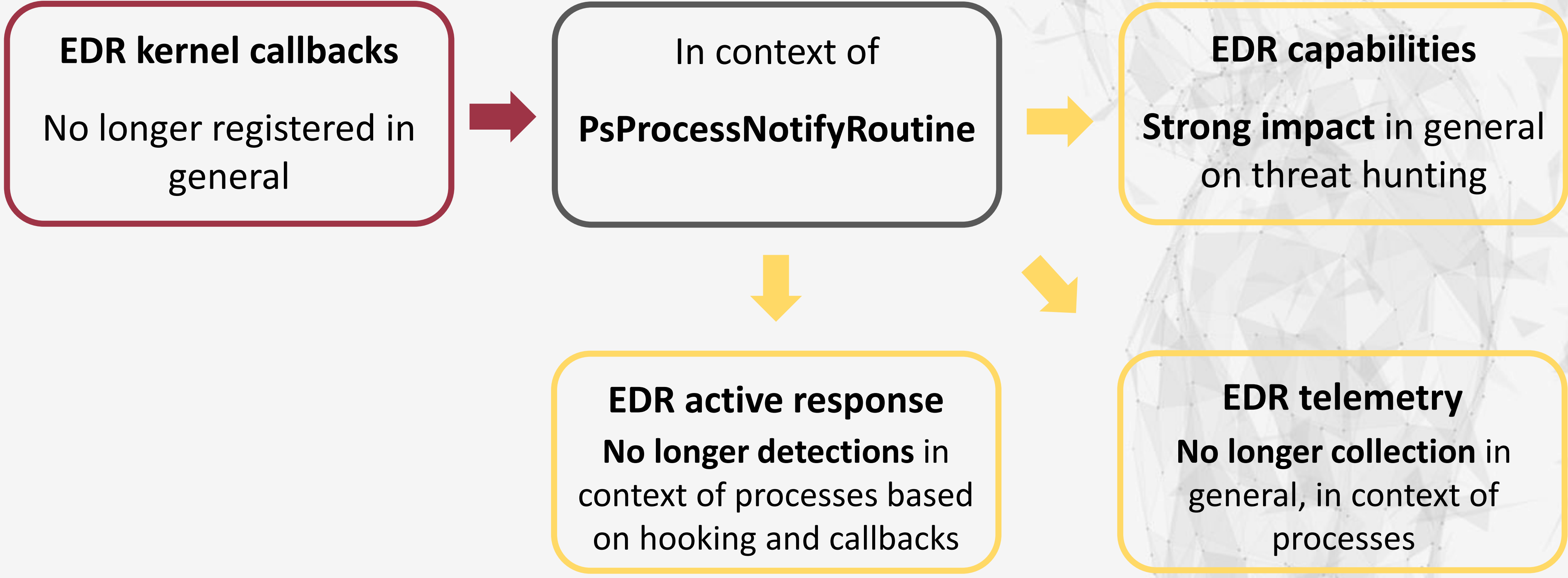
No longer injection of EDR_hooking.dll

Antivirus

Prevention based on user space API-hooking and callback telemetry collection (furthermore based on the minifilter functionality) is **disabled**

Conclusion: second demo

- Permanently **disabling EDR minifilter, much stronger impact!**
- Permanently impact on **EDR capabilities:**



EDR kernel callbacks

No longer registered in general

In context of

PsProcessNotifyRoutine

EDR capabilities

Strong impact in general on threat hunting

EDR active response

No longer detections in context of processes based on hooking and callbacks

EDR telemetry

No longer collection in general, in context of processes

Conclusion: second demo

- Permanently disabling **EDR minifilter driver**, **much stronger impact!**
  - **Disabling the EDR minifilter driver itself**
    - Permanently impact on Blue team EDR **web console features:**

Host isolation

Based on EDR sensor,
host isolation
no longer possible

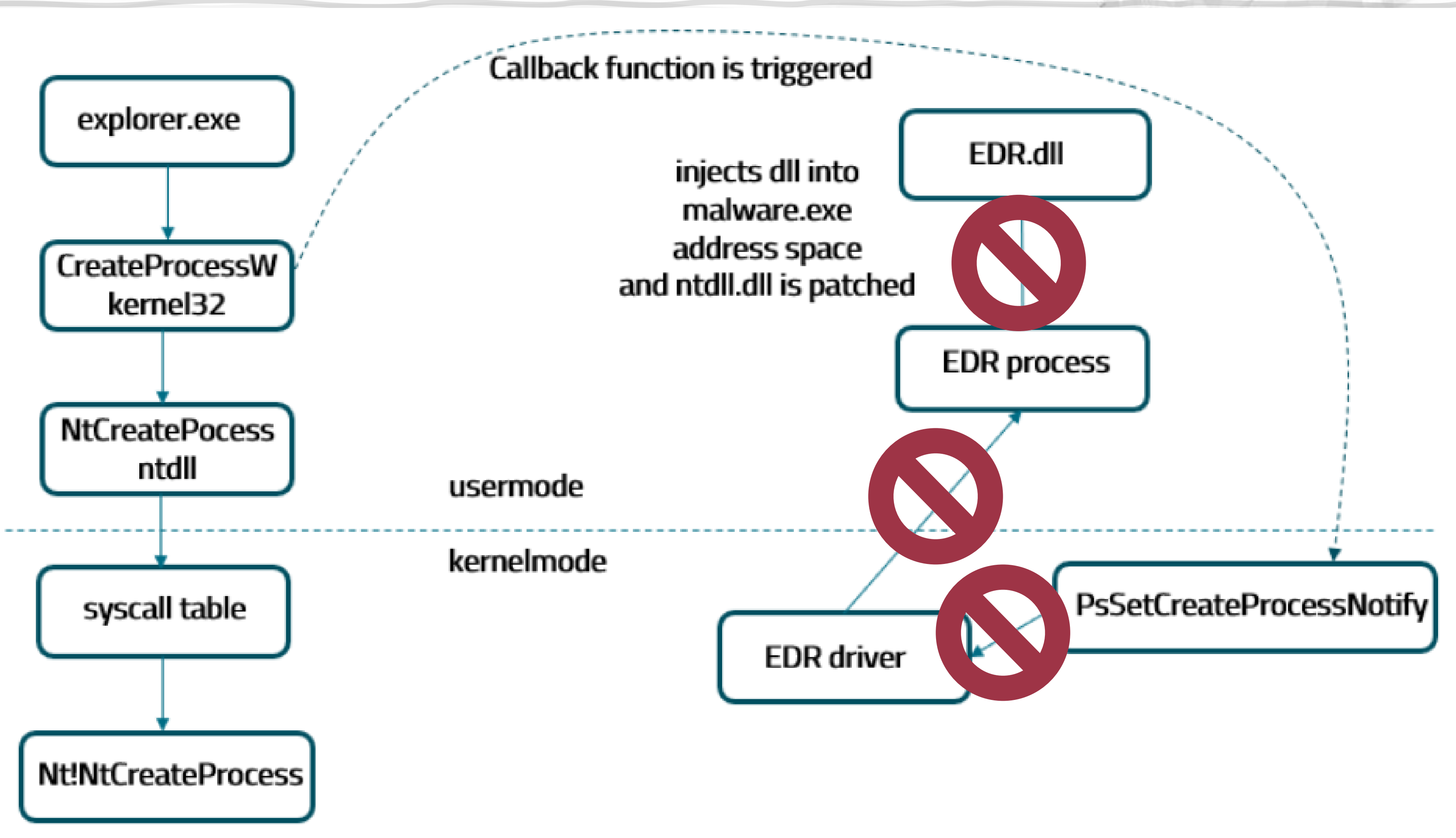
Real time response

Based on EDR sensor,
EDR remote (Power)Shell
no longer possible

EDR sensor recovery

Based on EDR sensor,
no longer possible

Why is the impact so strong?

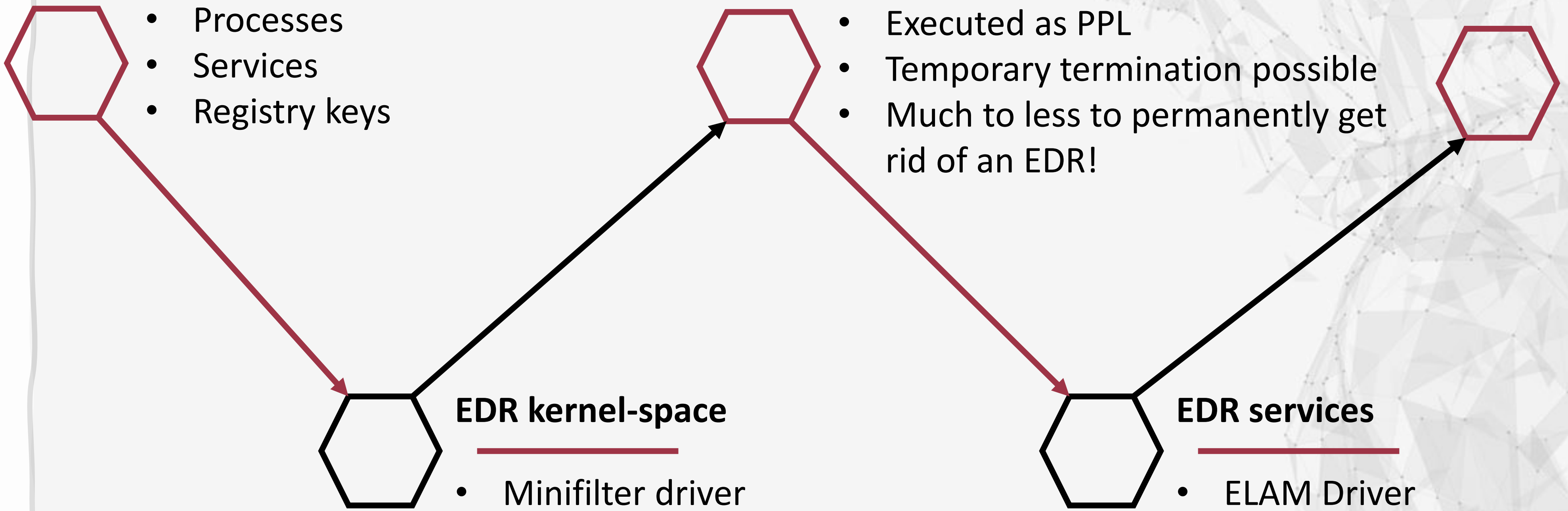


Reference: https://www.naksyn.com/edr%20evasion/2022/09/01/operating-into-EDRs-blindspot.html

SANS
OFFENSIVE
OPERATIONS

Summary

Summary



EDR user-space

- Processes
- Services
- Registry keys

EDR kernel-space

- Minifilter driver
- ELAM driver

EDR processes

- Executed as PPL
- Temporary termination possible
- Much to less to permanently get rid of an EDR!

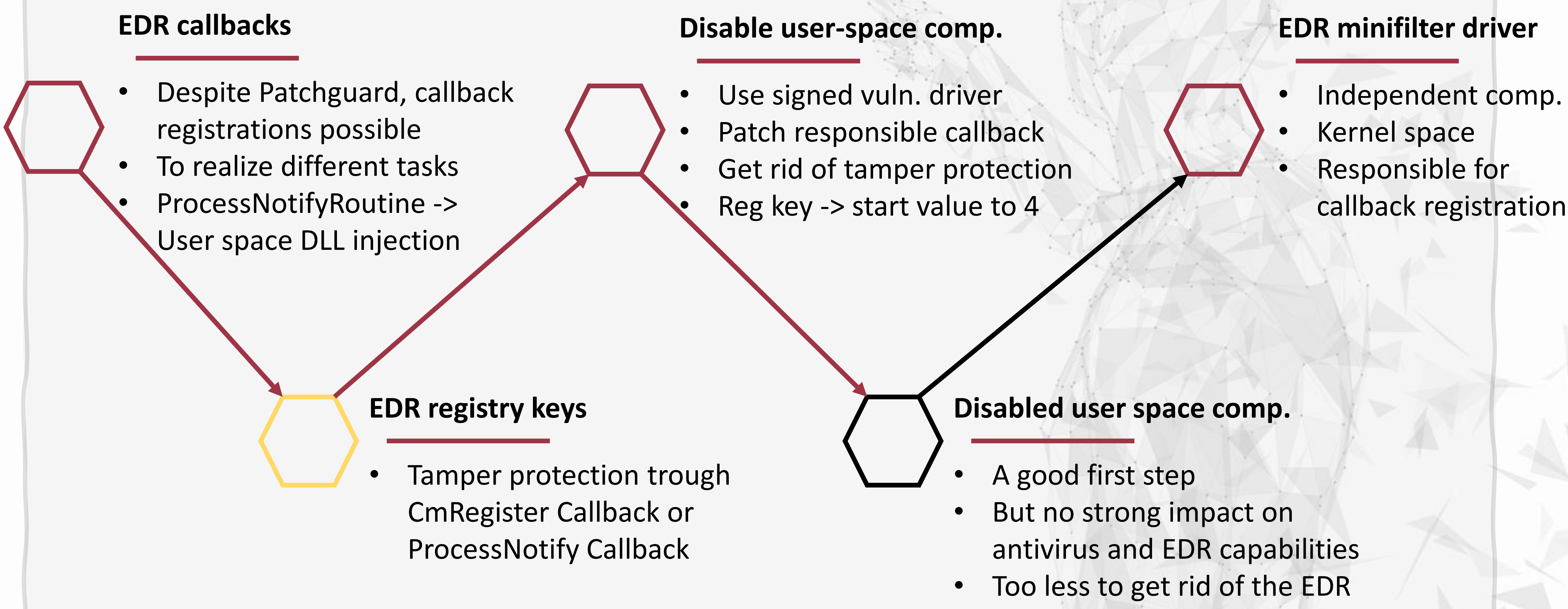
EDR services

- ELAM Driver
- Executed as protected service
- Also, not temporary pausable stoppable etc.

User-space comp.

- PPL process
- +
- Protected service

Summary



EDR callbacks

- Despite Patchguard, callback registrations possible
- To realize different tasks
- ProcessNotifyRoutine -> User space DLL injection

EDR registry keys

- Tamper protection trough CmRegister Callback or ProcessNotify Callback

Disable user-space comp.

- Use signed vuln. driver
- Patch responsible callback
- Get rid of tamper protection
- Reg key -> start value to 4

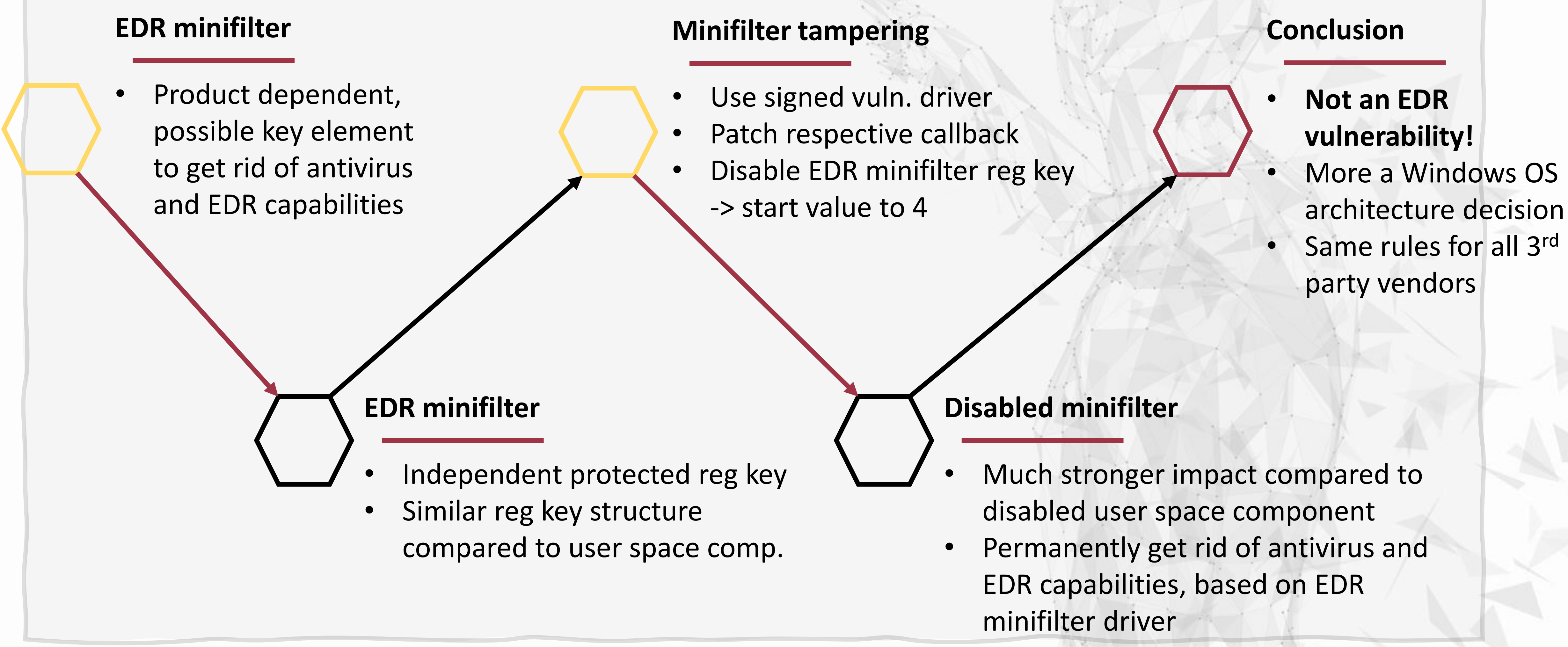
Disabled user space comp.

- A good first step
- But no strong impact on antivirus and EDR capabilities
- Too less to get rid of the EDR

EDR minifilter driver

- Independent comp.
- Kernel space
- Responsible for callback registration

Summary



EDR minifilter

- Product dependent, possible key element to get rid of antivirus and EDR capabilities

EDR minifilter

- Independent protected reg key
- Similar reg key structure compared to user space comp.

Minifilter tampering

- Use signed vuln. driver
- Patch respective callback
- Disable EDR minifilter reg key -> start value to 4

Disabled minifilter

- Much stronger impact compared to disabled user space component
- Permanently get rid of antivirus and EDR capabilities, based on EDR minifilter driver

Conclusion

- **Not an EDR vulnerability!**
- More a Windows OS architecture decision
- Same rules for all 3rd party vendors

Replicate Research-LAB

- VM with (latest) Windows 10 Pro or 11 Pro
  - VBS enabled or disabled -> Try both
- Business EDR or free/trial Antivirus
- Master of Puppets Blog Post
- Process Hacker
- CheekyBlinder
- TelemetrySourcerer
- EDRSandblast
- Backstab
- PPLKiller
- Mimikatz
- Rastamouse – Driver Development Course

© Daniel Feichter – RedOps GmbH (2022)

Thank you, Arlington!

- **Thanks** for the amazing opportunity to be a part of **SANS Hack Fest 2022** and thanks to the **greatest community!**

- Thanks to my girlfriend **Brigitte** for the **unique, amazing support over the last 10 years!**

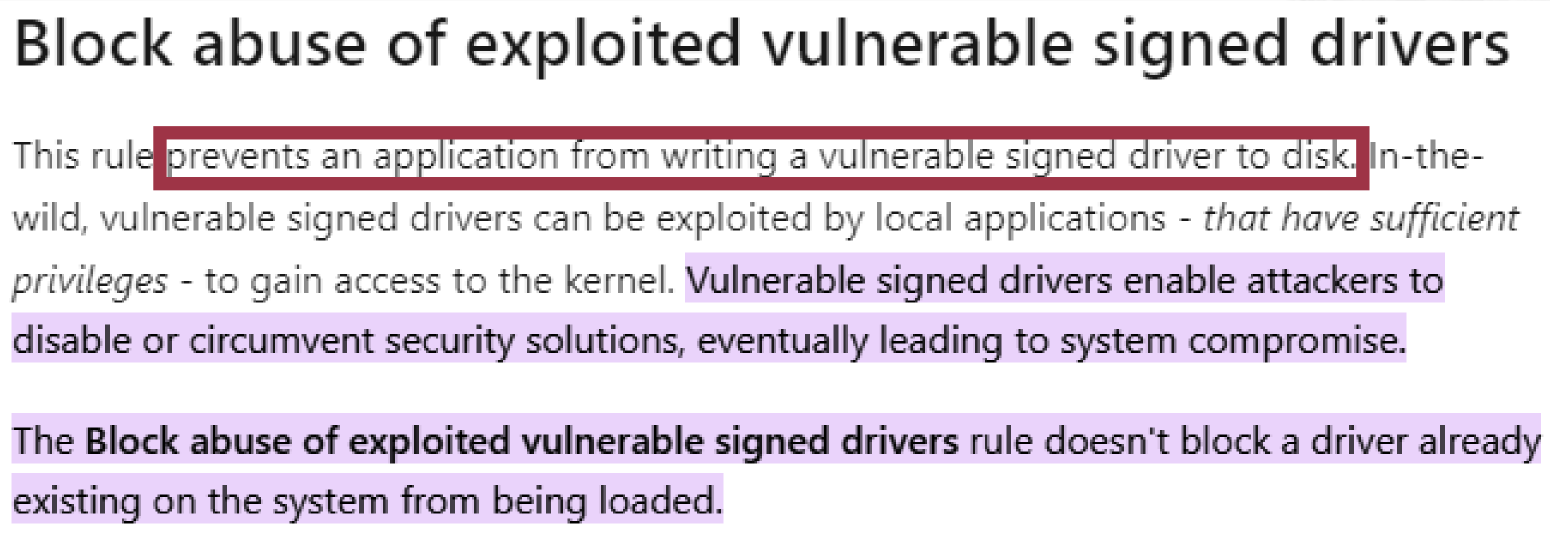
- Thanks to my colleagues **Andreas Clementi** and **Robert Rostek** for supporting me, since my first day in infosec!

Join us in Arlington, VA or Live Online for **FREE**

SUMMIT: Nov 14–15 | TRAINING: Nov 16–21

Blue Team: Mitigation

- Key element is that the attacker escalate privileges and get access to kernel space, in case of vulnerable drivers we should try to mitigate this:
- **In case of Windows Defender:**
  - ASR Rule: Block abuse of exploited vulnerable signed drivers



Block abuse of exploited vulnerable signed drivers

This rule prevents an application from writing a vulnerable signed driver to disk. In-the-wild, vulnerable signed drivers can be exploited by local applications - *that have sufficient privileges* - to gain access to the kernel. Vulnerable signed drivers enable attackers to disable or circumvent security solutions, eventually leading to system compromise.

The **Block abuse of exploited vulnerable signed drivers** rule doesn't block a driver already existing on the system from being loaded.

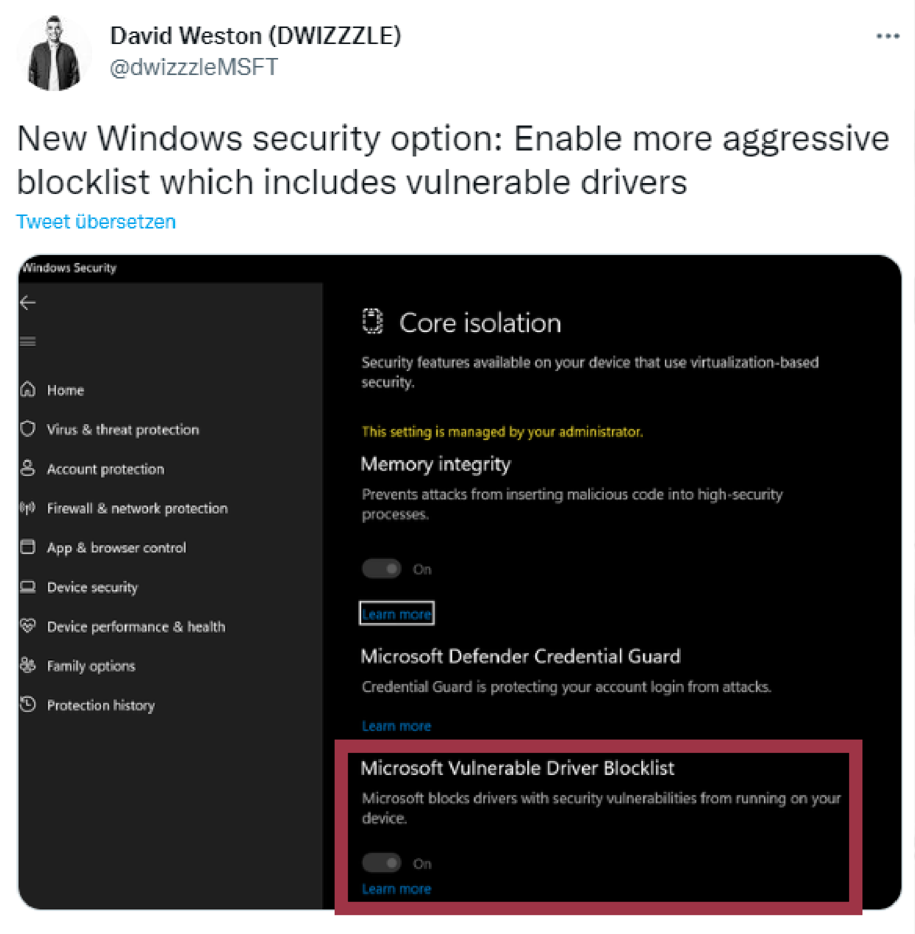
Quelle: https://docs.microsoft.com/en-us/microsoft-365/security/defender-endpoint/attack-surface-reduction-rules-reference?view=o365-worldwide

Blue Team: Mitigation

- **Windows Device Guard VBS/HVCI:**
  - Microsoft Vulnerable Driver Blocklist
  - More aggressive additional hardening with WDAC

Organizations that want a more aggressive block list than Microsoft's measured approach can add their own drivers to the list using the WDAC Policy Wizard.

Resource: https://www.techrepublic.com/article/how-microsoft-blocks-vulnerable-malicious-drivers-defender-third-party-security-tools-windows-11/



David Weston (DWIZZZLE)
@dwizzzleMSFT

New Windows security option: Enable more aggressive blocklist which includes vulnerable drivers

Tweet übersetzen

Windows Security

Core isolation

Security features available on your device that use virtualization-based security.

This setting is managed by your administrator.

Memory integrity

Prevents attacks from inserting malicious code into high-security processes.

On

Learn more

Microsoft Defender Credential Guard

Credential Guard is protecting your account login from attacks.

Learn more

Microsoft Vulnerable Driver Blocklist

Microsoft blocks drivers with security vulnerabilities from running on your device.

On

Learn more

Resource: https://twitter.com/dwizzzleMSFT/status/150821736[illegible]1142

References

[1] Yosifovich, Pavel; Ionescu, Alex; Solomon, David A.; Russinovich, Mark E. (2017): Windows internals. Part 1: System architecture, processes, threads, memory management, and more. Seventh edition. Redmond, Washington: Microsoft Press. http://proquest.tech.safaribooksonline.de/9780133986471.

[2] Pavel Yosifovich (2019): Windows 10 System Programming, Part 1: CreateSpace Independent Publishing Platform.

[3] Microsoft (2017): Filtering Registry Calls. https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/filtering-registry-calls.

[4] Microsoft (2018): CmRegisterCallbackEx function (wdm.h). https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nc-wdm-ex_callback_function

[5] Microsoft (2018): CmUnRegisterCallback function (wdm.h). https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-cmunregistercallback.

[6] @Truneski (2020): Windows Kernel Programming Book Review. https://truneski.github.io/blog/2020/04/03/windows-kernel-programming-book-review/

[7] Matteo Malvica (2020): Silencing the EDR. How to disable process, threads and image-loading detection callbacks https://www.matteomalvica.com/blog/2020/07/15/silencing-the-edr/.

[8] Matteo Malvica (2020): Kernel exploitation: weaponizing CVE-2020-17382 MSI Ambient Link driver https://www.matteomalvica.com/blog/2020/09/24/weaponizing-cve-2020-17382/

[9] Christopher Vella (2020): EDR Observations. https://christopher-vella.com/2020/08/21/EDR-Observations.html.

[10] BR-SN (2020): Removing Kernel Callbacks Using Signed Drivers. https://br-sn.github.io/Removing-Kernel-Callbacks-Using-Signed-Drivers/

References

[11] https://www.cyberwarfare.live/blog/how-edr-hooks-API-calls-part1

[12] https://synzack.github.io/Blinding-EDR-On-Windows/

[13] https://github.com/SadProcessor/SomeStuff/blob/master/Invoke-EDRCheck.ps1

[14] https://docs.microsoft.com/en-us/windows/win32/api/winsvc/ns-winsvc-service_launch_protected_info

[15] https://sourcedaddy.com/windows-7/values-for-the-start-registry-entry.html

[16] https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/types-of-windows-drivers

[17] https://courses.zeropointsecurity.co.uk/courses/offensive-driver-development

[18] https://www.ghacks.net/2022/03/28/windows-defender-vulnerable-driver-blocklist-protects-against-malicious-or-exploitable-drivers/

[19] https://www.techrepublic.com/article/how-microsoft-blocks-vulnerable-malicious-drivers-defender-third-party-security-tools-windows-11/

[20] https://github.com/eclypsium/Screwed-Drivers/blob/master/presentation-Get-off-the-kernel-if-you-cant-drive-DEFCON27.pdf

[21] https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Studien/SiSyPHuS_Win10/AP7/SiSyPHuS_AP7_node.html

[22] https://www.naksyn.com/edr%20evasion/2022/09/01/operating-into-EDRs-blindspot.html

[23] https://fourcore.io/blogs/edr-detections-bypasses-and-other-shenanigans