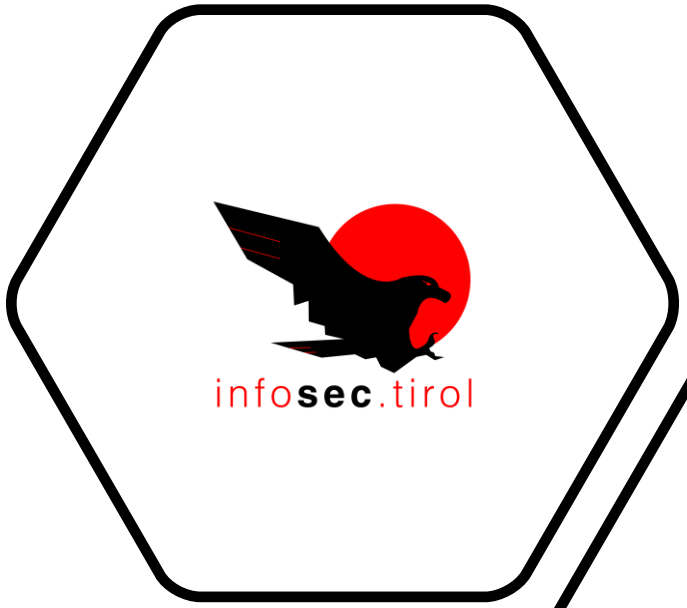


Master of Puppets

How to tamper the EDR?

Whoami

Daniel Feichter:

- Founder of **Infosec Tirol** (www.infosec.tirol)
- Twitter @VirtualAllocEx
- Martial arts fan and fully convinced EDR user

Focus on:

- Offensive security/red teaming
- Antivirus & EDR products
- IT-security research
- Windows Internals
- Defense evasion
- Windows hardening (client/server)

We take a look at

i. ATT&CK T1562.001: Impair Defenses: Disable or Modify Tools

- How to **disable main functionality of EPP/EDR's,** by **targeted, controlled, tampering** of specific **EPP/EDR components?**

  Without relying on:

  i. EDR uninstall password

  ii. Using (EDR) uninstall software

  iii. Disabling EDR by Security Center GUI

ii. Disclaimer: just my personal research/experience

iii. Applies to **multiple products**

  i. Few days ago, seen in the wild, AvosLocker Ransomware

We want to achieve

Deep dive AV/EPP/EDR products on Windows

Functional connection between **different components user- and kernel space**

a) **User space:** processes, services, registry keys

b) **Kernel space:** callback routines, EDR drivers

Controlled disabling key components, **to permanently avoid**

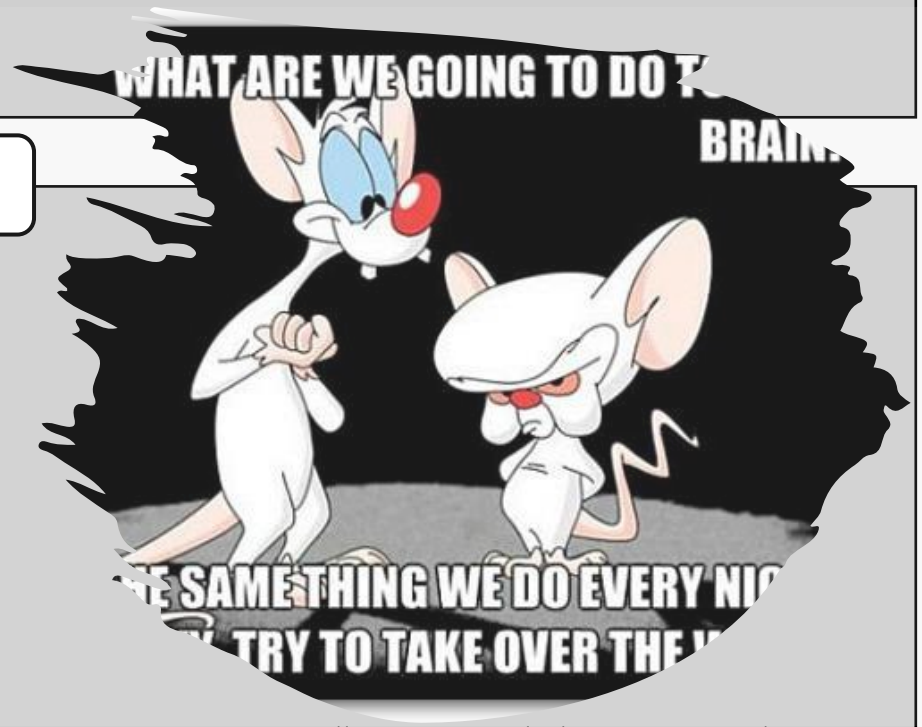
a) **Antivirus module:** dynamically and in-memory prevention

b) **EDR module:**

i. Detections and telemetry footprint

ii. Host isolation and real time response (remote shell)

iii. EDR recovery feature



Reference: https://www.pinterest.com/pin/768074911421378920/

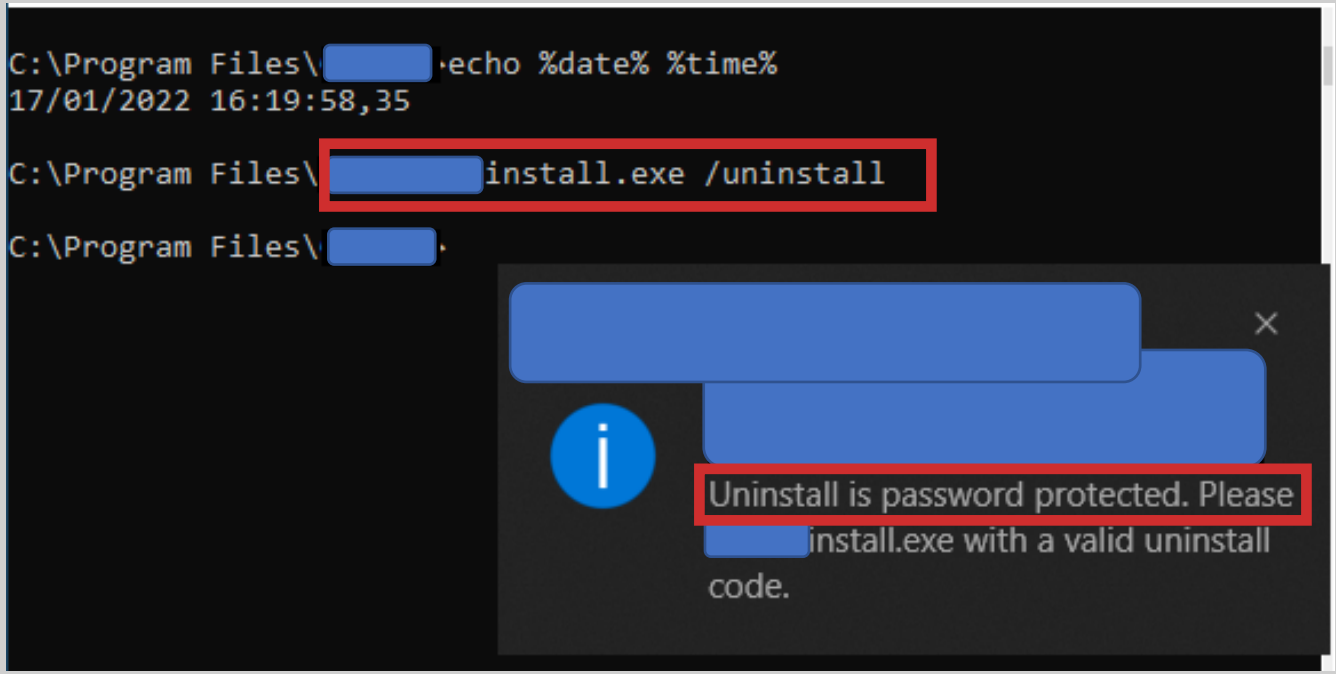
No zero days

Necessary requirements

a) **Privileged user (**high- or system integrity) or **Unprivileged User** (medium integrity)

b) Despite, most EDR annoying

Why not uninstall the EDR?



Give me a scenario

Red team engagement

a) Initial access: phishing or similar

b) Local privilege escalation: PrintNightmare CVE-2021-1675 or other misconfig

c) Compromised host: (other) **useful open user session**

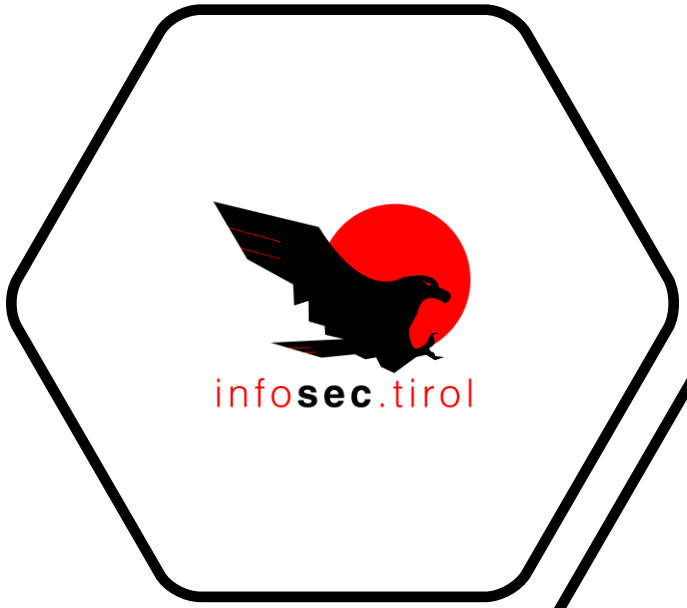
   i. Nice, but installed EDR is tough

Steal credentials or impersonate useful user

a) OS credential dumping: LSASS memory -> T1003.001

b) Access token manipulation: token impersonation/theft -> T1134.001

c) But First -> targeted, **controlled disabling EDR main functionality**



User space

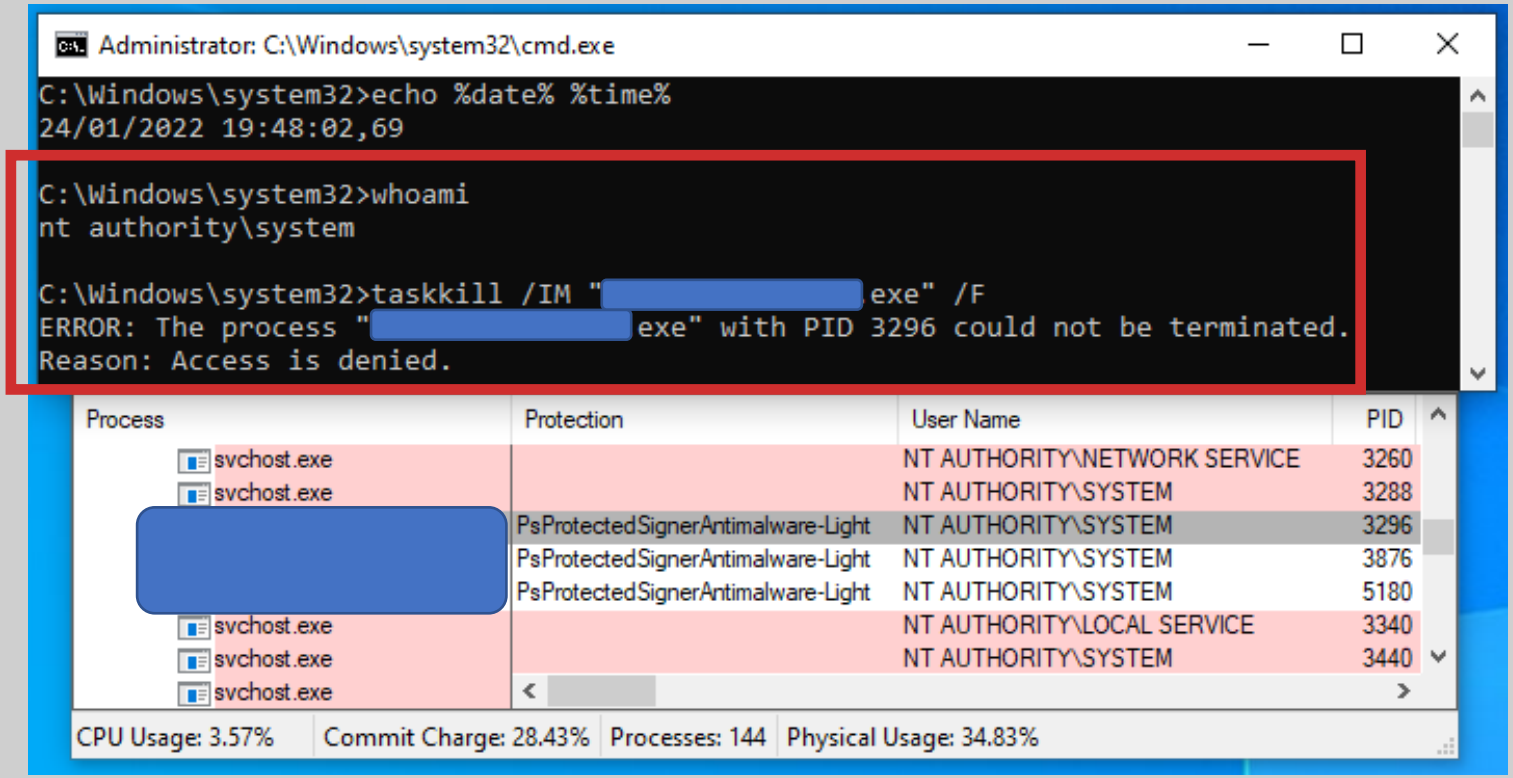
First step: EDR processes

User-space: EDR process tampering

EDR process termination

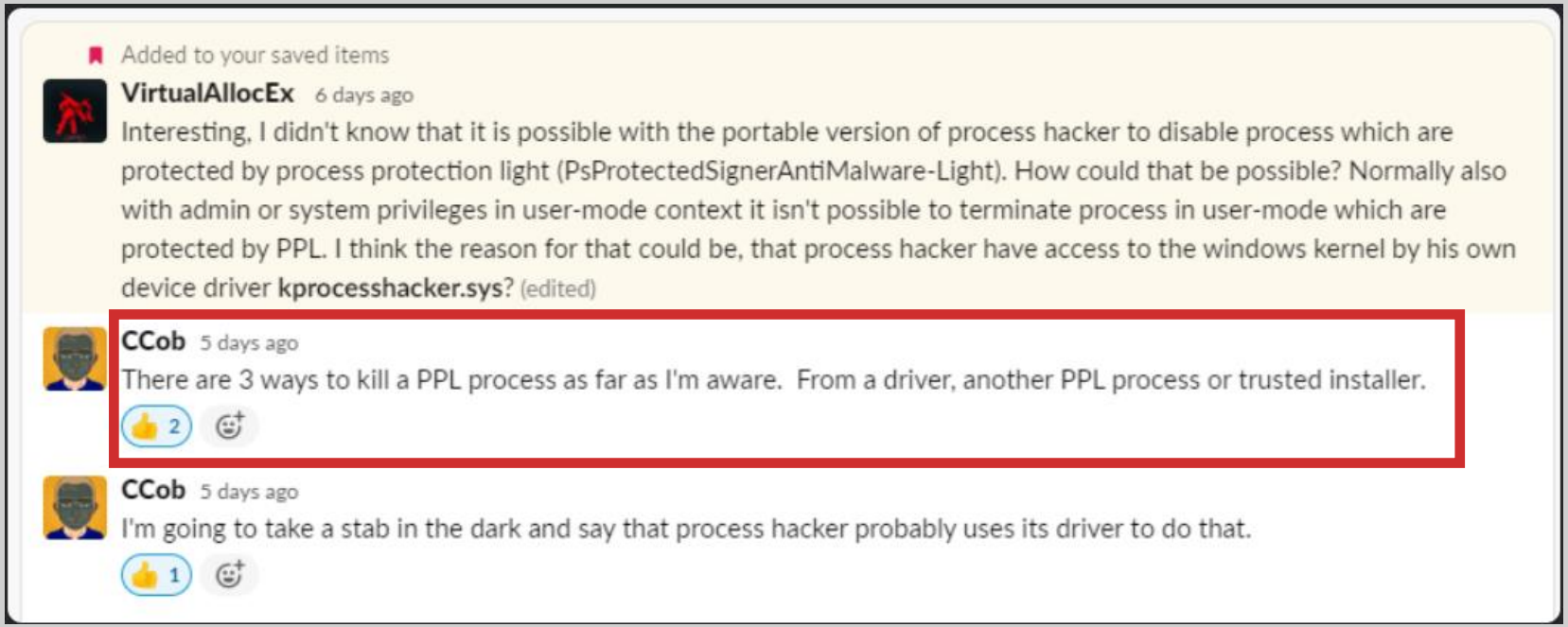
a) Try to kill EDR process in system session -> **despite system integrity** not being allowed

Normally, initialized as **P**rotected **P**rocess **L**ight (PPL)



User-space: EDR process tampering

How to terminate PPL process(es)? -> creds to @_EThicalChaos_

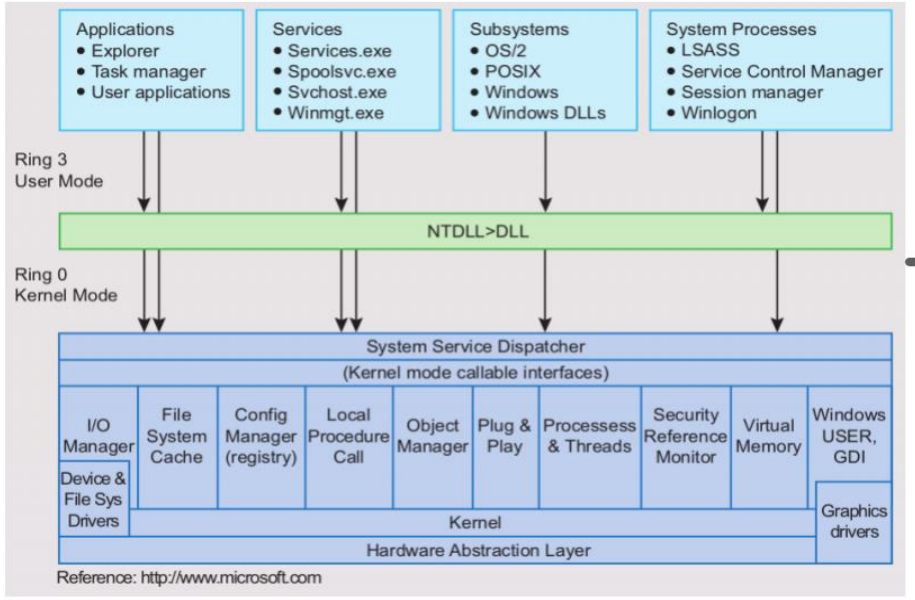


Concept of vulnerable (device) driver

a) Get **access kernel space** -> **vulnerable device driver** RTCore64 CVE 2019-16098

b) **Remove PPL flag** and terminate unprotected process or directly terminate PPL process

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
```

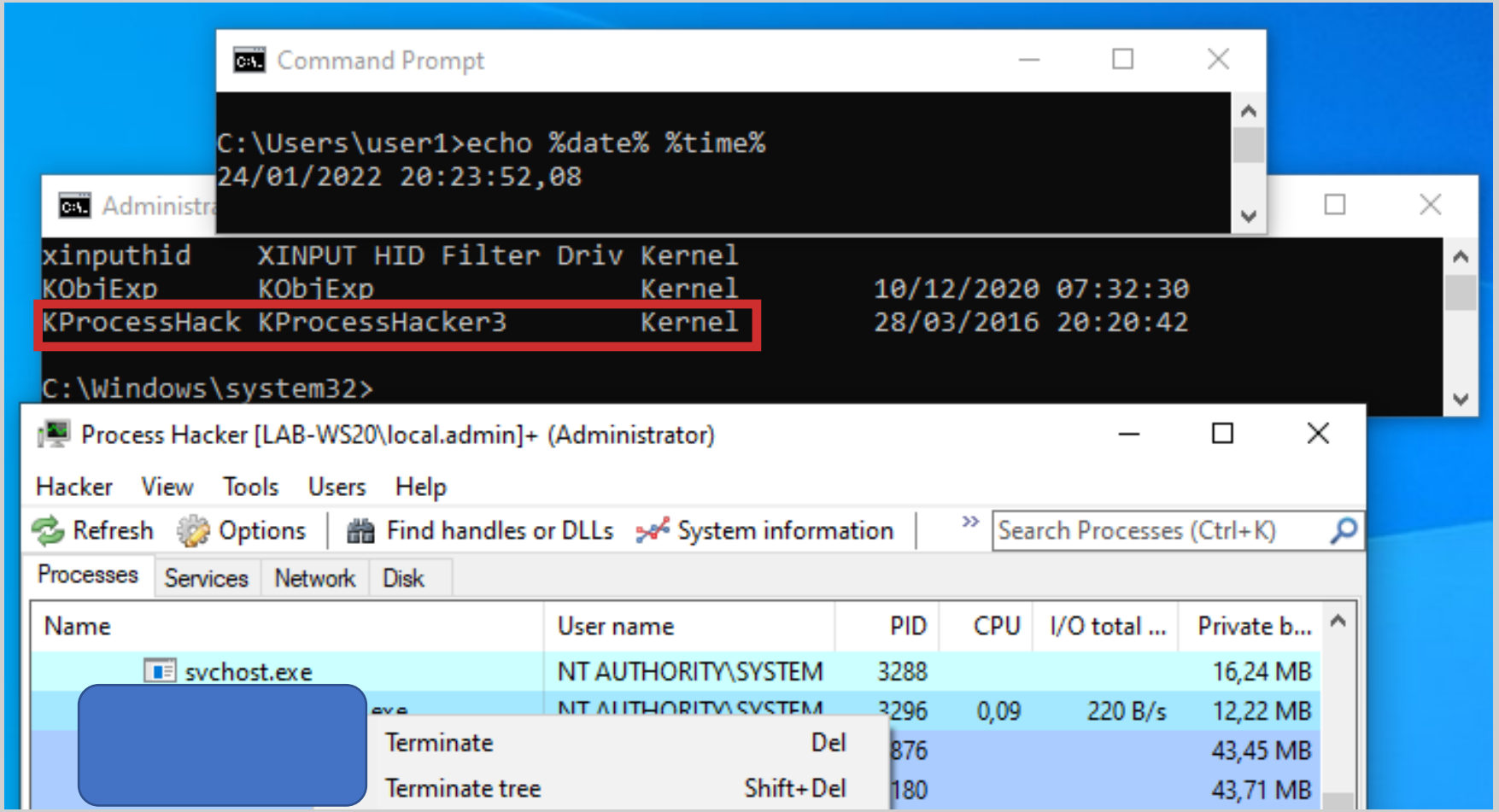
```
C:\cache>echo %date% %time%
17/01/2022 15:45:12,00

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



User-space: EDR process tampering

Conclusion EDR process tampering

a) With access to kernel space -> EDR process(es) termination possible

b) But normally:

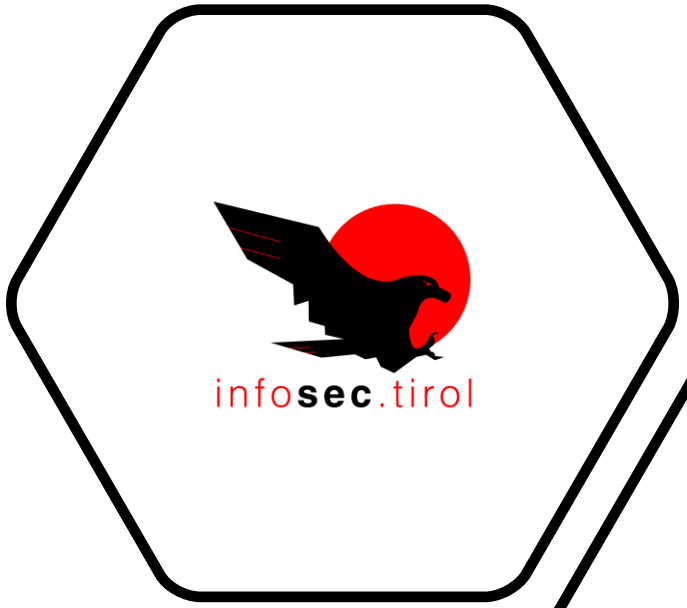
- Termination is **only temporary**
- **Watchdog function** restarts killed Process(es) -> until now, no details available

c) Restart Time depends on the respective EDR product

d) Until now, **no permanent termination** of PPL EDR Process(es) possible

e) Until now, no permanent disabling of necessary EDR components achieved

f) **We must dig deeper…**

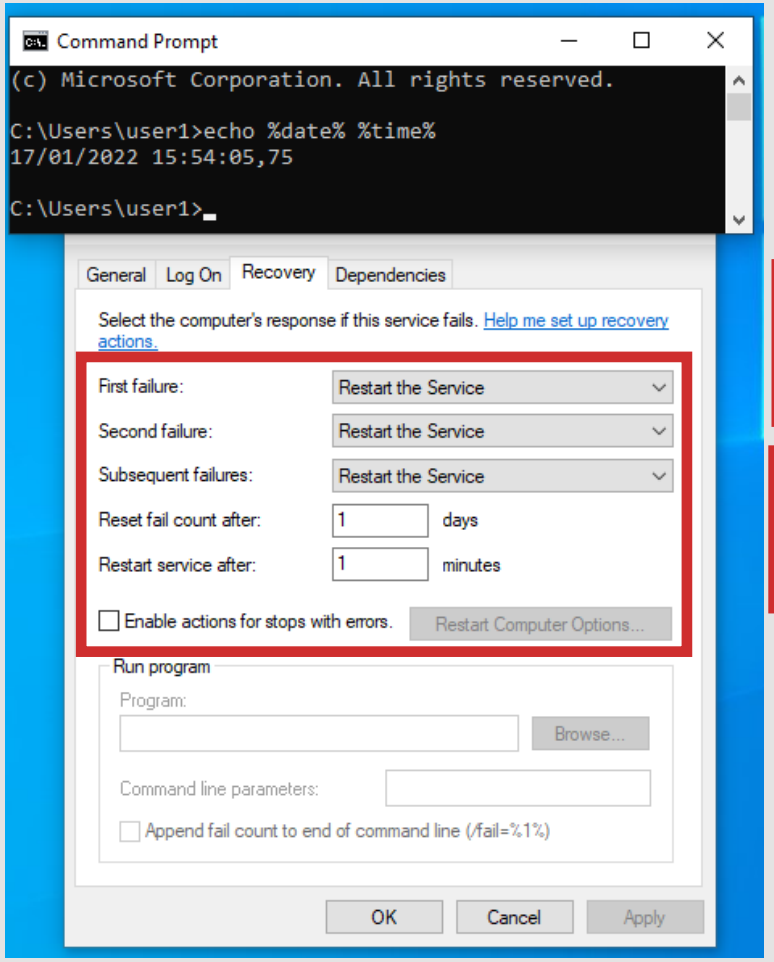


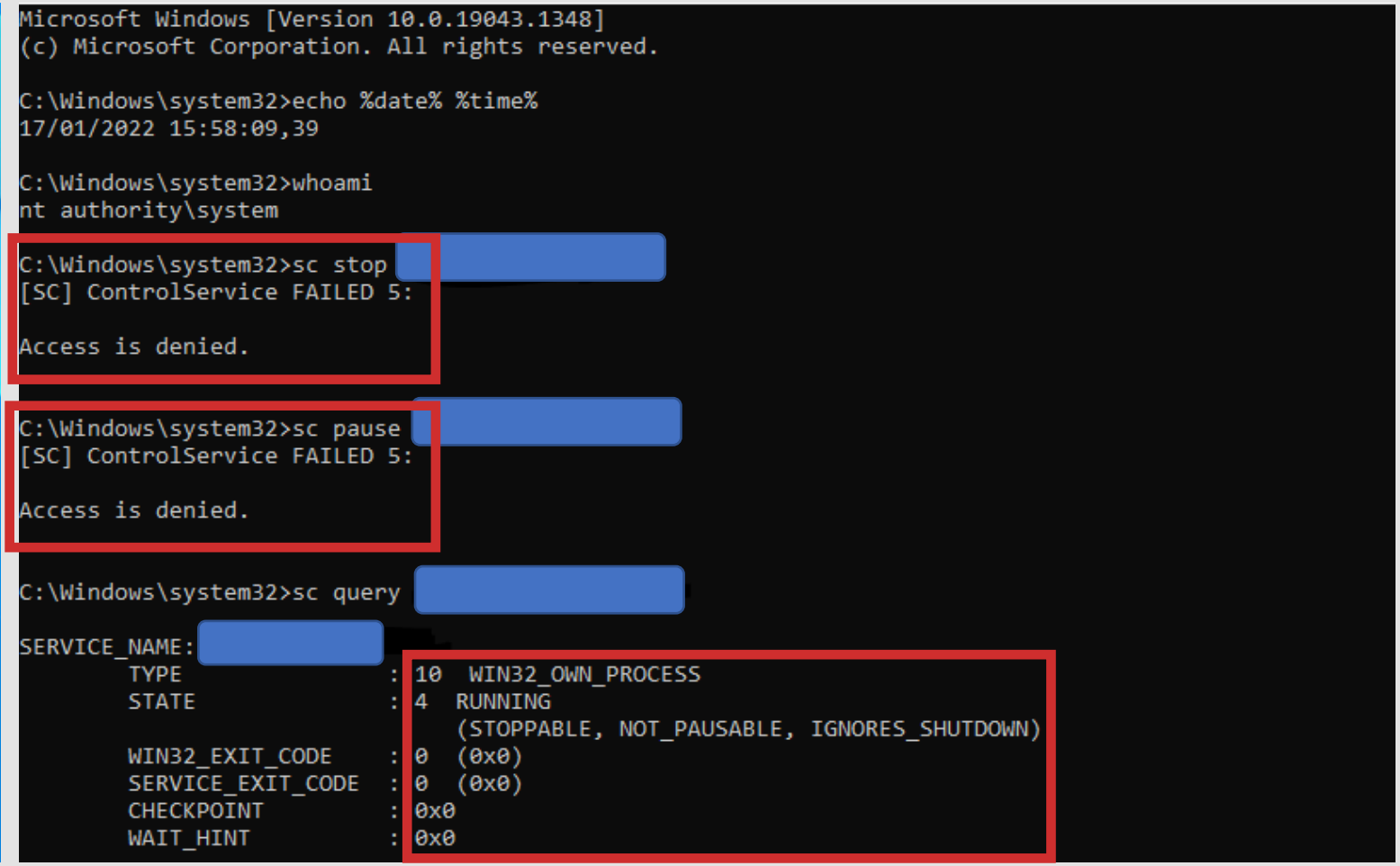
User space

Second step: EDR services

User-space: EDR service tampering

Identify connected, **protected service**





User-space: EDR service tampering

Conclusion EDR service tampering

a) Responsible watchdog for restarting the terminated PPL EDR process(es)

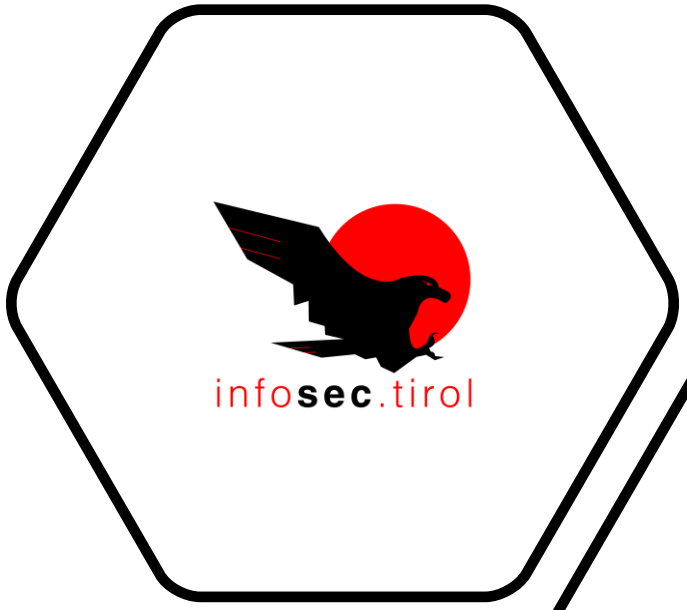
b) Initialization as protected service by ELAM driver
   i. EDR, a closer look at protected services

c) From user space -> **despite system integrity, access denied**

d) Until now, no permanent disabling of protected EDR service possible

e) Until now, **no permanent disabling** of necessary EDR components **achieved**

f) We must dig deeper…

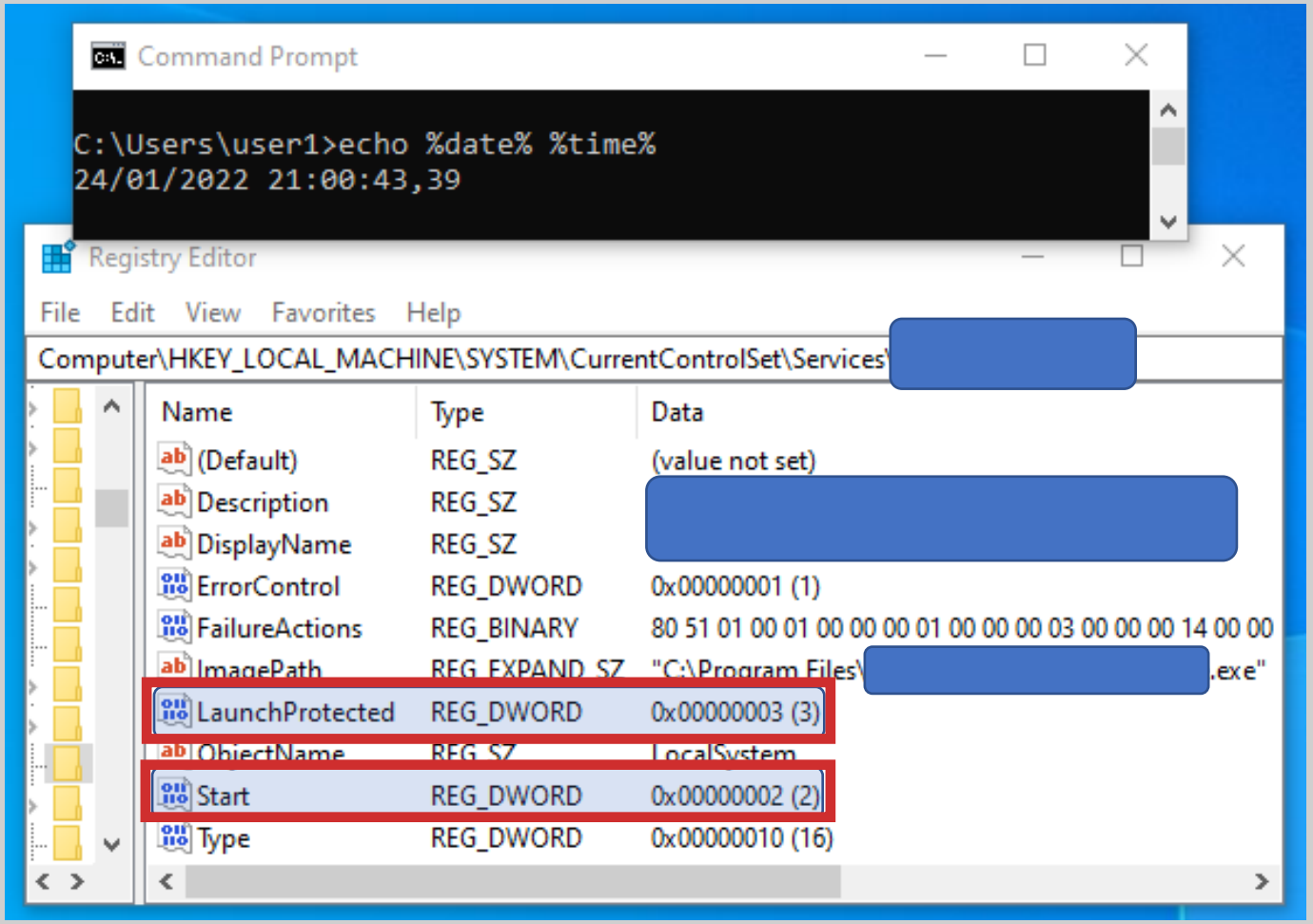


User space

Third step: EDR registry keys

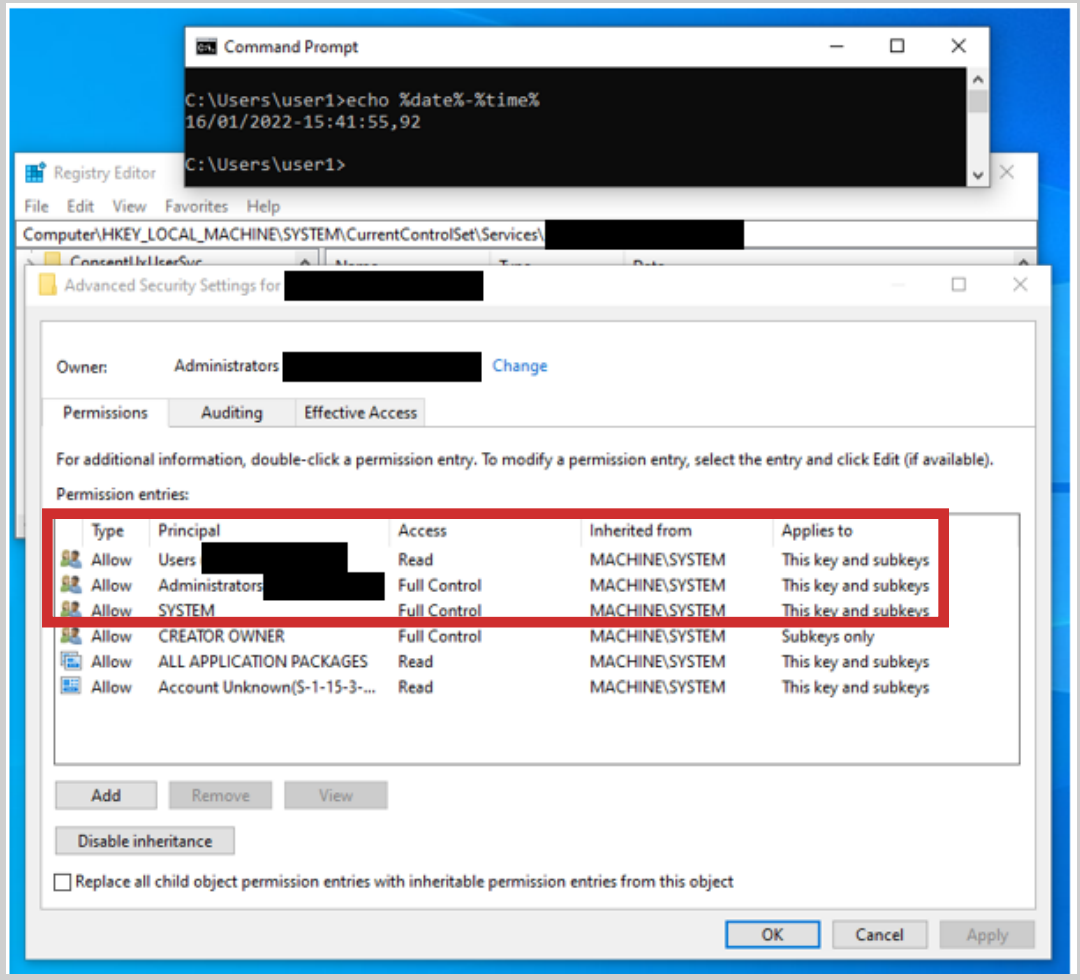
User-space: EDR registry tampering

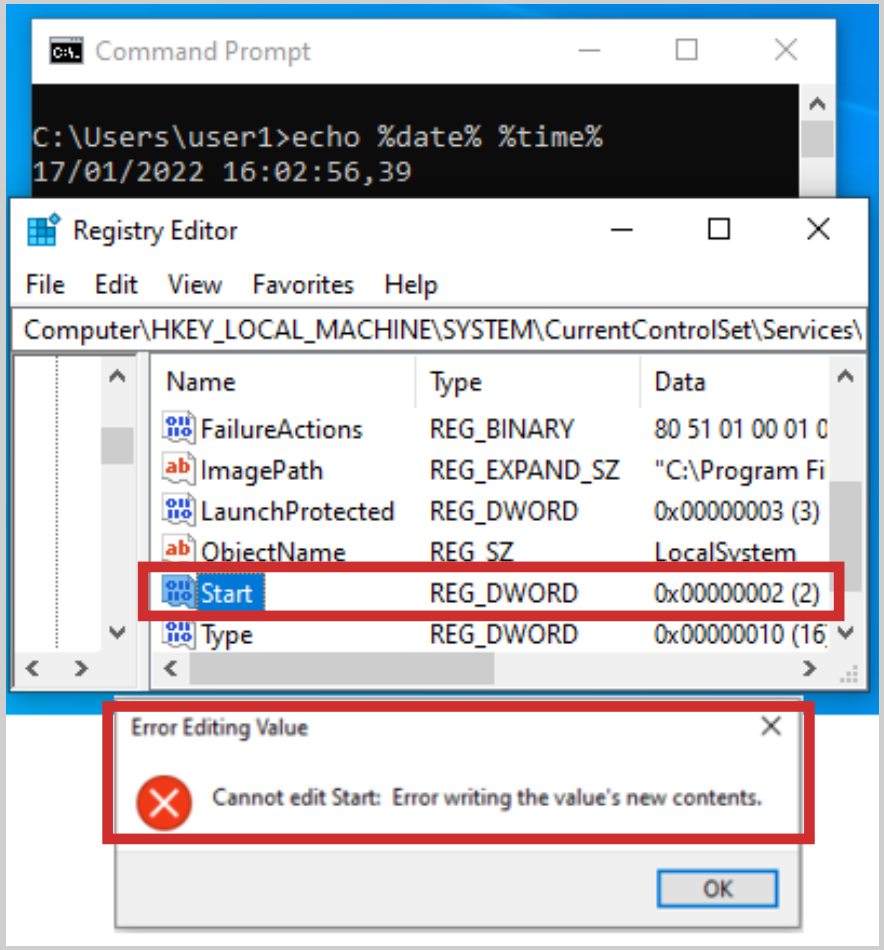
Protected service, identify **reg keys / sub keys / entries**



User-space: EDR registry tampering

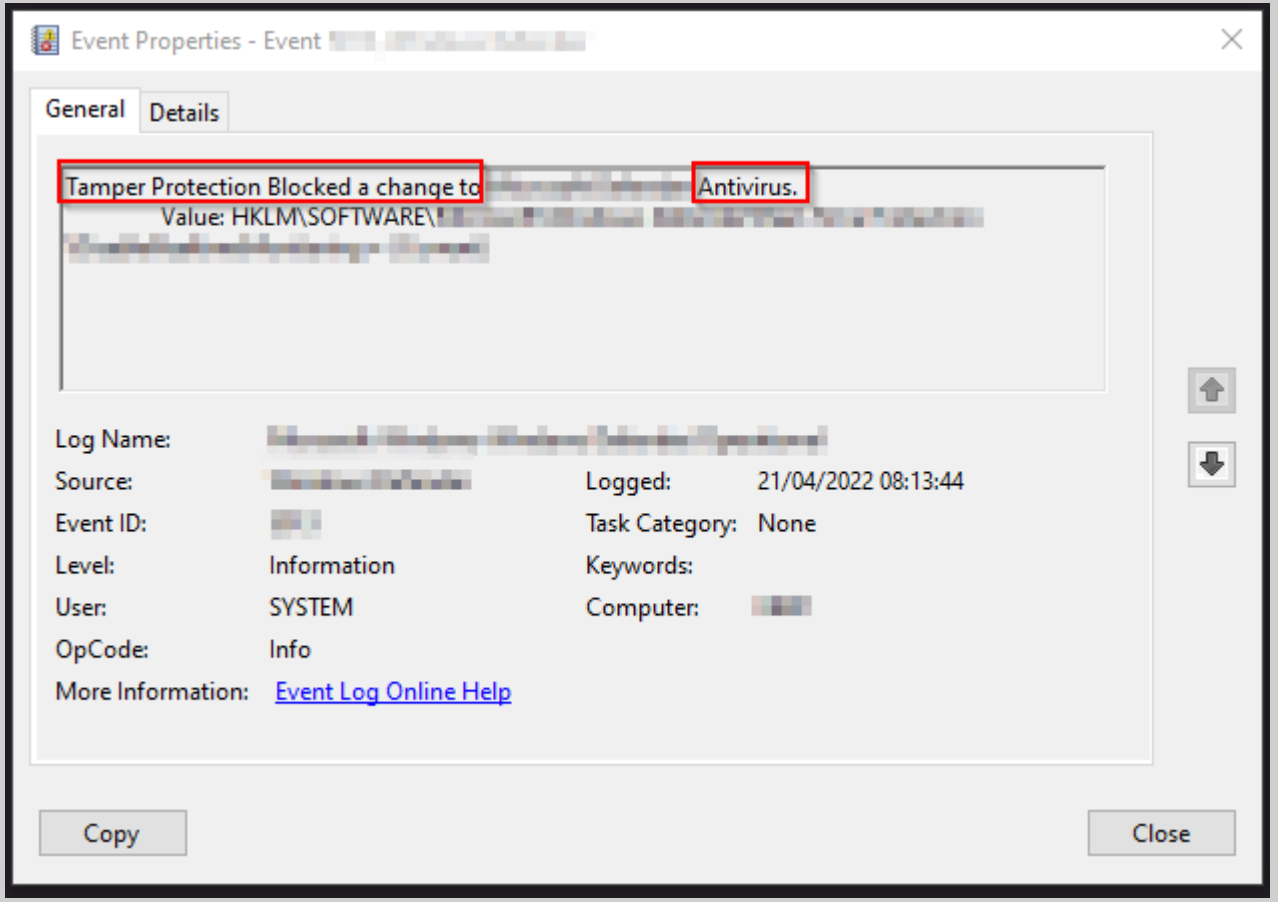
Check user permissions on necessary registry keys





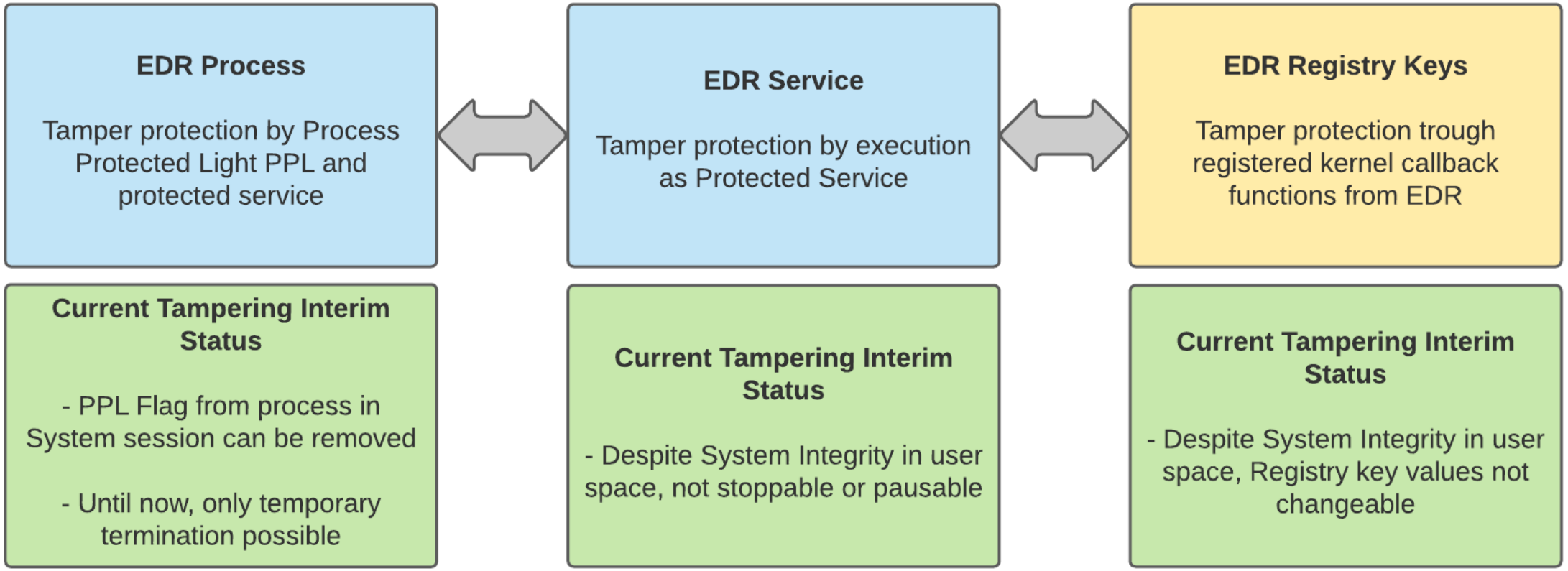
User-space: EDR registry tampering

Depending on product -> we possibly create tampering alerts



Interim status user space tampering

EDR components in Windows user space



Interim status user space tampering

Despite system integrity user space -> necessary EDR user-space components **can't be permanently disabled**:

a) EDR process -> protected by PPL

b) EDR service -> executed as protected service (ELAM Driver)

c) EDR registry keys -> protection mechanism until yet unknown

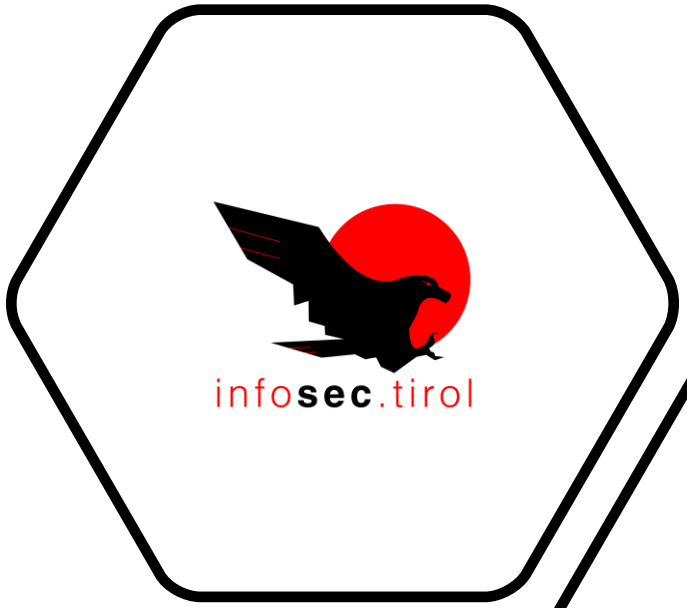
Reference: https://www.saturdaymorningsforever.com/2016/12/pinky-and-brain.html

But now we know:

a) Initialization of **user space component,** determined by **START entry** in **protected service registry key**

b) Tampering the registry key -> **maybe the key element** to permanent disable EDR user-space component

c) But something **protects EDR registry keys** against tampering



Kernel space

Fourth Step: Callback Routines

Kernel-space: EDR callback routines

What are kernel callback routines? Why kernel callbacks? Responsible tasks?

a) Since introduction of Kernel Patch Protection aka PatchGuard (Windows XP SP3 x64)

„officially“ syscall hooking no longer possible -> EDR forced to user-space API hooking

b) Despite PatchGuard, it is possible to get **telemetry/data** from **Windows kernel:**

i. **ProcessNotify** (process creation, user space DLL injection / user space API-hooking)

ii. ThreadNotify (process injection)

iii. LoadImageNotify (DLL mapping, suspicious image loading)

c) Important to **collect telemetry** on endpoint **(threat hunting)**

Kernel-space: EDR callback routines

Besides; (could) be responsible, **protecting reg 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: EDR user space service disabling

User space component disabled, impact?

a) **Impact** on EDR **user space component and functionality**,

when **ProcessNotify** callback gets **patched?**

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

Conclusion EDR user space service

Tampering the EDR ProcessNotify callback, impact?

a) EDR user space **DLL injection / API-hooking temporary disabled**

   i. **Prevention** capabilities antivirus module

   ii. **Detection** capabilities EDR module, for example **telemetry collection** process creation

b) **Registry key protection** from EDR user-space component (Protected Service)

   i. Patch specific callback routine -> **START entry** value can be changed (from 2 to 4)

   ii. Value 4 is **equal to disabled** -> **after reboot,** EDR user space component (protected service and PPL process) disabled

Conclusion EDR user space service

From red team perspective, **just** disabling user space service -> **inefficient!**

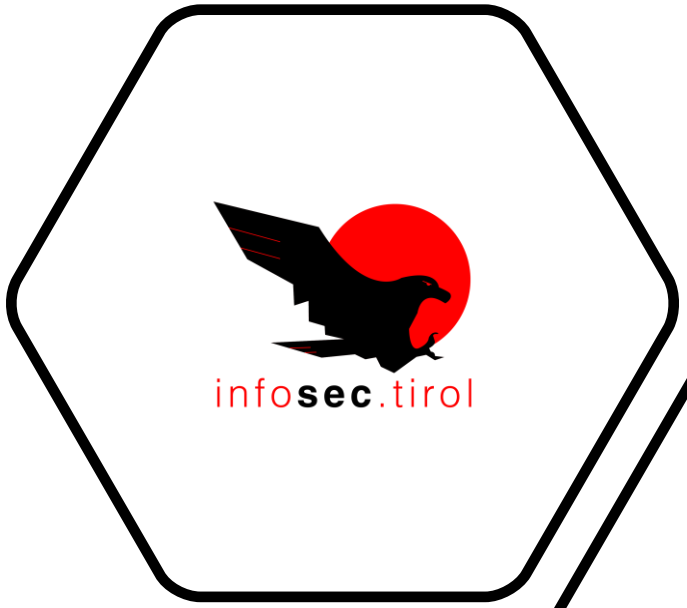
a) Despite **permanent disabled user space component**, after necessary host reboot:

i. **EDR re-registers** all previously patched callbacks

- Thereby, in case of re-registered ProcessNotify -> user space **Dll injection, API-hooking** again active
- Thereby, **prevention and detection** (telemetry collection) again active

ii. Regardless, **host isolation** and **real time response** (remote shell) still possible

iii. Regardless, EDR **recovery** still possible



Kernel space

Final Step: EDR Minifilter driver

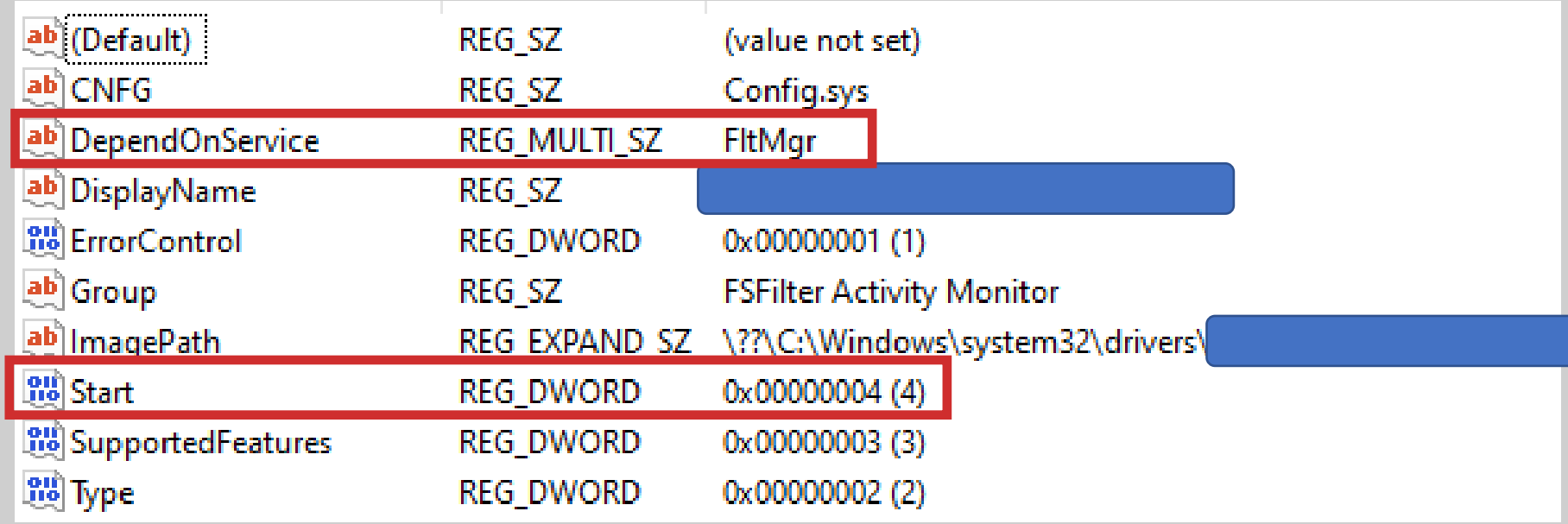
Kernel-space: EDR minifilter driver

What is a minifilter driver? For what do EDRs use it? Responsible tasks?

a) **EDR kernel component** which is:

i. Used to **register kernel callback routines** and **register Windows Security Center**

ii. Still active, even if EDR user space service is already disabled



Maybe our EDR key element?

Product-independent controlled disabling of EDR, to **permanently avoid:**

a) **User space Dll injection**

   i. AV -> Prevention (dynamically and in-memory)

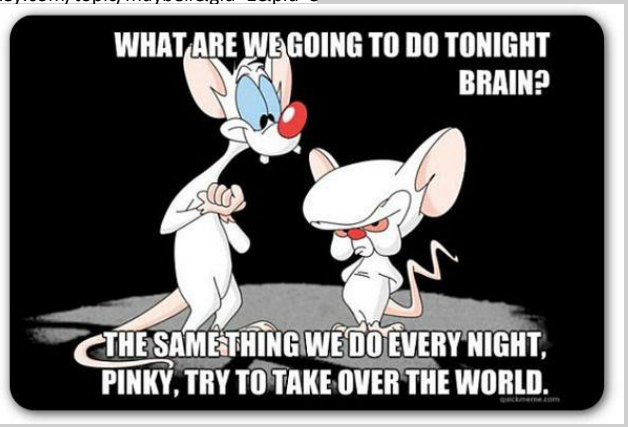
   ii. EDR -> Detection and telemetry collection

b) **Host isolation**

c) **Real time response** (remote shell)

d) **EDR recovery** of partly disabled EDR

Reference: https://www.memesmonkey.com/topic/maybe#&gid=1&pid=3



Reference: https://www.pinterest.com/pin/768074911421378920/

Second Demo: EDR minifilter driver tampering

Disable registration of EDR minifilter driver, impact?

a) **How to tamper** the EDR minifilter driver? -> remember EDR **registry keys**

b) Final round -> knockout the EDR!

https://www.deviantart.com/littlebasty98/art/Taekwondo-wallpaper-580014925

Conclusion EDR minifilter driver tampering

Disabling the EDR minifilter driver (could) have **permanent impact on:**

Permanently getting rid of:

a) Kernel callback registration, ProcessNotify, ThreadNotify, LoadImageNotify etc.

b) Thereby, **reg key protection disabled**

c) Thereby, **user space Dll injection** / API-hooking **disabled**

i. **Prevention capabilities antivirus part;** despite still-active user space component (protected service and PPL process) -> prevention no longer works efficiently, mimikatz.exe etc.

ii. **Detection capabilities** EDR part -> telemetry collection disabled

-> could be a bad day for Threat Hunter 😜

Conclusion EDR minifilter driver tampering

Disabling the EDR minifilter driver (could) have **permanent impact on:**

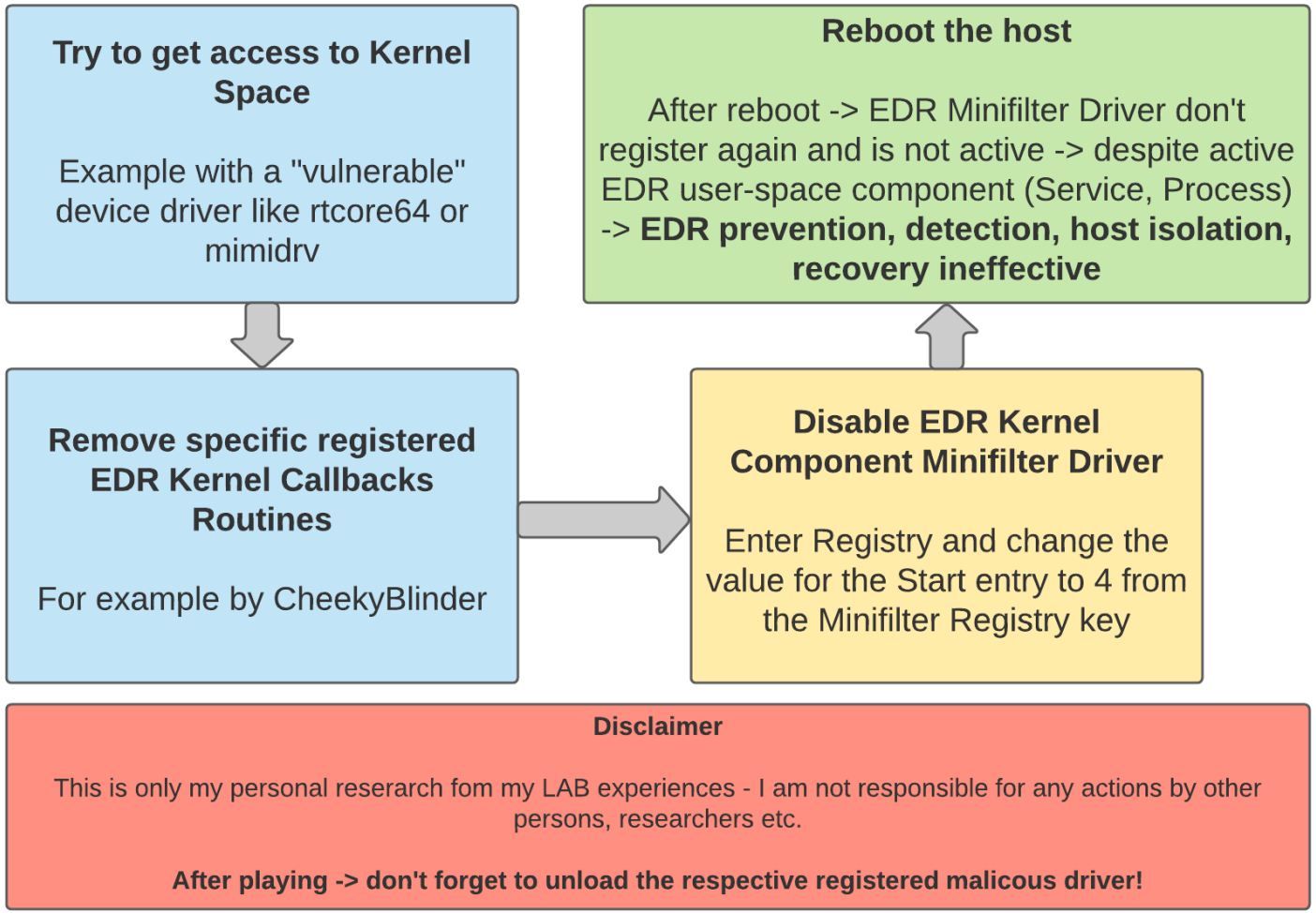
Furthermore, impact on compromised host:

b) Host isolation no longer possible

c) Real time response (remote shell) no longer possible

d) EDR recovery feature e.g., update or repair function in web console, no longer possible

Conclusion EDR minifilter driver tampering



Three Key take-aways

Despite successful tampering, **EDR do raise the bar more and more**

- EDR **tampering** is a **very isolated** process -> always think about **whole attack chain**

Remember our scenario, escalate from an **unprivileged user towards domain admin** on a host were the attacker has achieved unprivileged access and the domain admin has an open user session

Below we see possibilities for EDR products to prevent or detect key activities from the attacker, until he could reach his goal to disable the EDR by tampering specific componentes and get as quiet as possible domain admin

TA0001-Initial Access

Example the attacker try to get initial access by a phishing mail (attachment, link etc.)

TA0002-Execution

Exectuion of malware to open a command and control channel or code which is used to register the vulnerable device driver or the registration of the vulnerable driver itselt etc.

TA0004-Privilege Escalation

Depending on how the attacker tries to escalate his local privileges. Example, PrintNightmare, HiveNightmare etc.

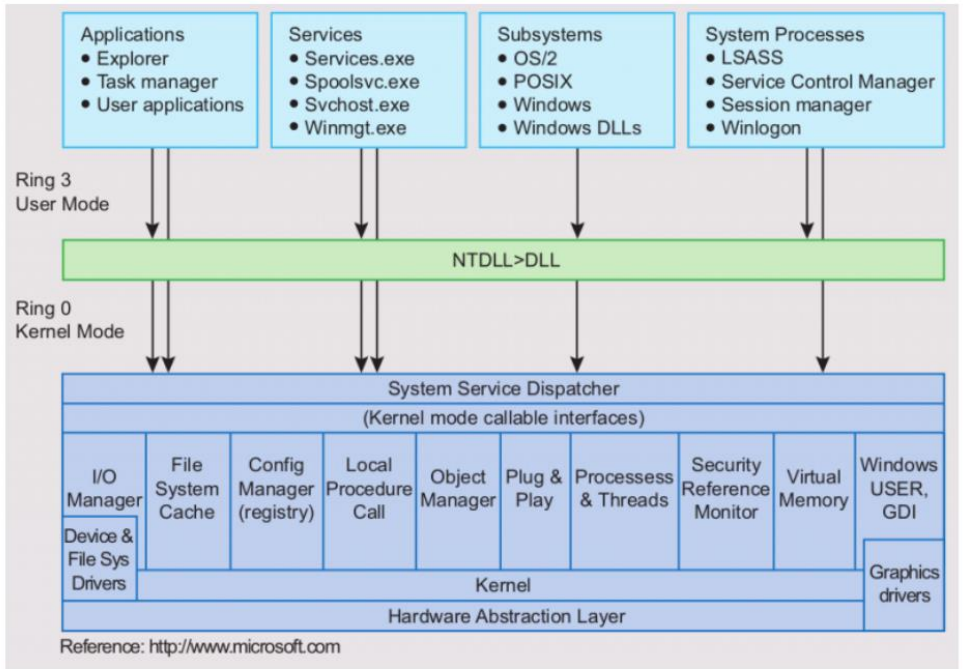
TA0005-Defense Evasion

Attempt disabling or modifying EDR (ATT&CK T1562.001). Process termination, disabling services, modifying reg keys, removing callback routines etc.

Three Key take-aways

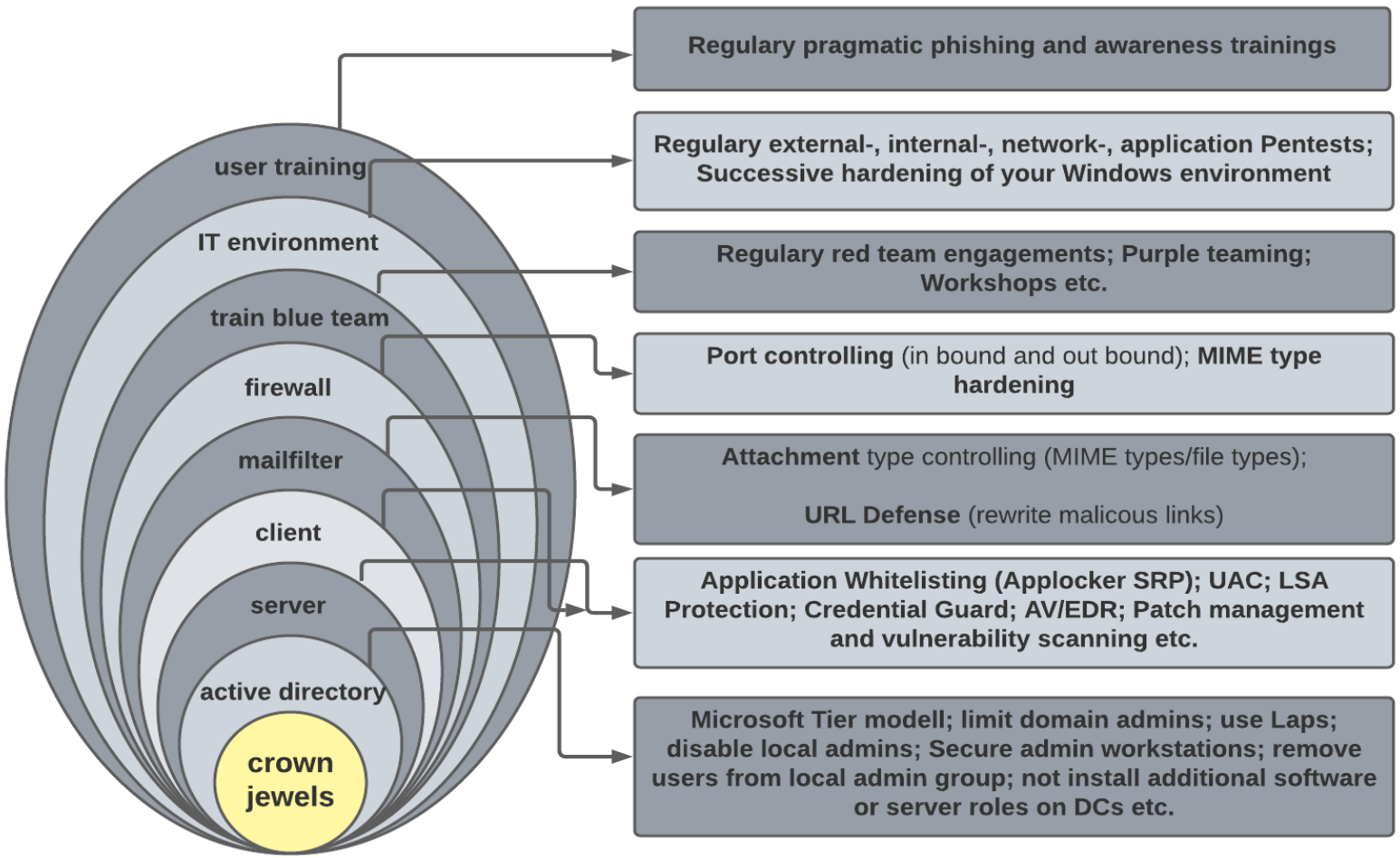
EDR tampering under Windows is **not based on vulnerabilities** in EDR products

- rather, we see that all manufacturers must adapt to the rules of the Windows OS architecture -> same (official) rules for every EDR vendor



Don't rely too much even on best (EDR) products; **Harden your Windows Environment!**

Onion layer principle



Many Thanks
BSides Munich!

Thank you for the opportunity to be a part of **BSides conference!**

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/

References

[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/.

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

[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