Windows built-in Sandbox Disables Microsoft Defender and other EDR/AVs : Attack Detection and Prevention via MemoryRanger

Denis Pogonin

Igor Korkin

2022

WHO WE ARE

Denis Pogonin

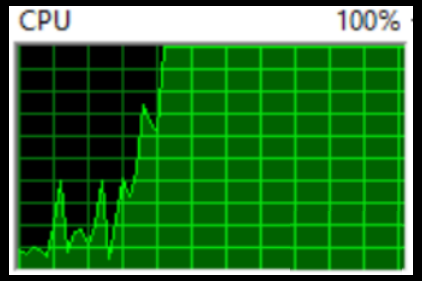
- Bachelor of Information Security
- National Research Nuclear University MEPhI
- Cryptology and Cybersecurity Department

Igor Korkin, PhD

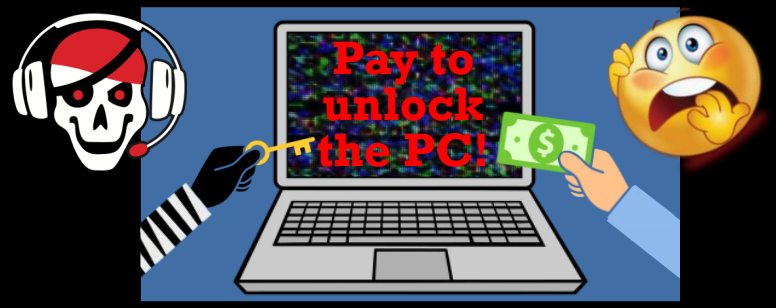
- Independent Security Researcher
- Speaker at CDFSL, BlackHat, HITB, SADFE, ROOTCON, Texas Cyber Summit
- sites.google.com/site/igorkorkin

Different Malware have Different Goals

Gain full remote access



Perform mining



Encrypt\wipe files

Packing, encryption and obfuscation

Malware is not protected

Malware is protected

Disable or evade Microsoft Defender

Microsoft Defender is running on over 500 000 000 PCs

> Windows Defender is protecting more than 50% of the Windows ecosystem, so we're a big target, and everyone wants to evade us to get the maximum number of victims

Tanmay Ganacharya

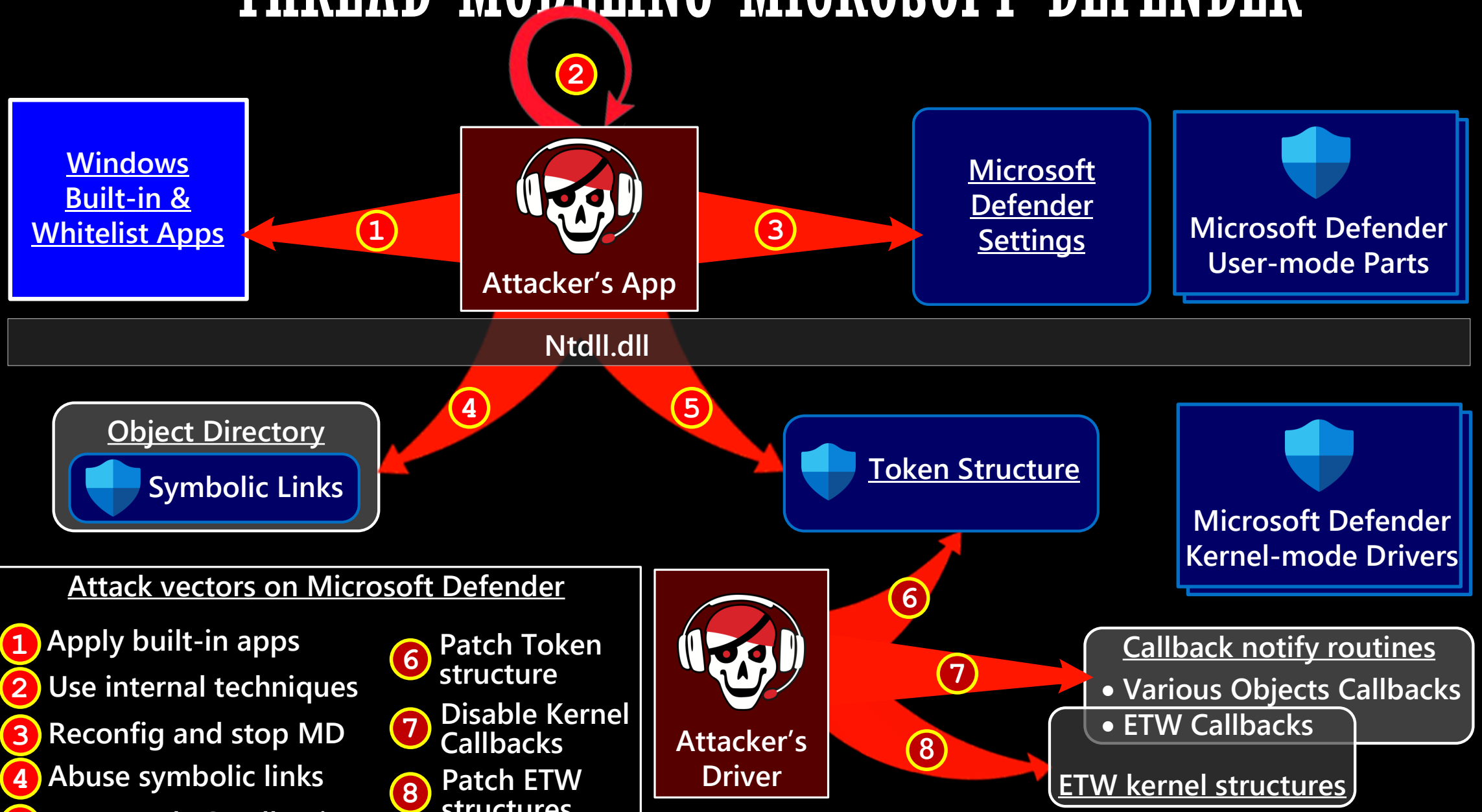
Partner Director for Security Research

@ Microsoft Defender for Endpoint

* Top Microsoft Defender expert: These are the threats security hasn't yet solved, ZDNet, 2019
https://www.zdnet.com/article/top-windows-defender-expert-these-are-the-threats-security-hasnt-yet-solved

Attack Vectors on Microsoft Defender

THREAD MODELING MICROSOFT DEFENDER



USER-MODE ATTACKS: 1, 2

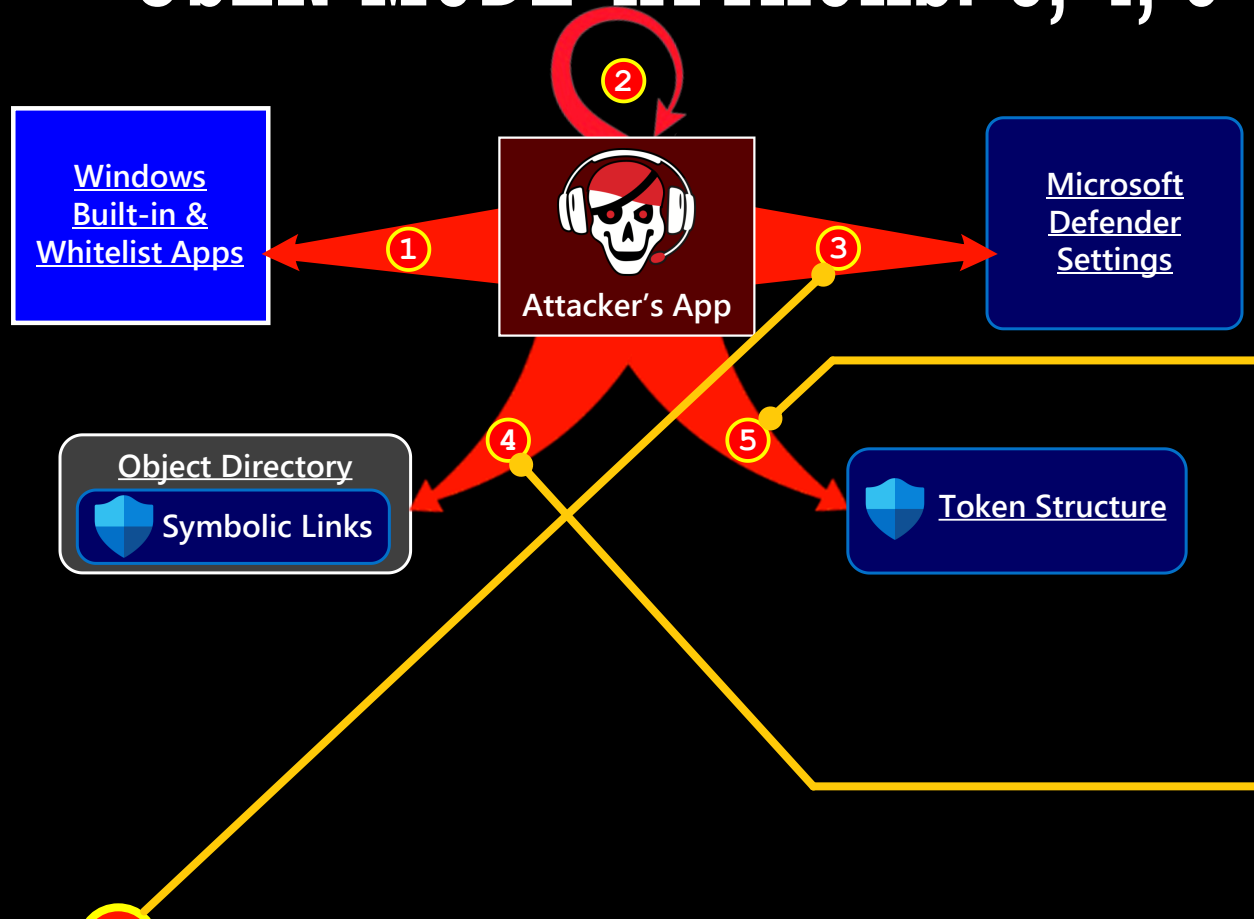
2 Attacks via internals techniques

• Packing, Obfuscation, Encryption,	FuckThatPacker
• Token Impersonation: Abusing TrustedInstaller to stop MD	DefenderSwitch, T1, DefenderStop
• Custom loader bypass kernel callbacks	DarkLoadLibrary

1 Attacks via Windows Built-in and Whitelist Apps

• PowerShell Reflection	PowerShellRunner, P1, P2
• Herpaderping	H1
• Code Injection into the whitelisted apps	I1, UltimateWDACBypassList
• LockBit Ransomware abuse Windows Defender command line to load payload, L1 • Microsoft published a list of built-in apps that can be abused to bypass Windows Defender, M1	

USER-MODE ATTACKS: 3, 4, 5



5 User-mode Sandboxing

Use syscalls to modify Token structure: • Token Integrity Level • Token Privileges	SandboxDefender, PPLGuard ('trust label')
→Fixed at June, 2022	KillDefender_BOF

4 Abuse Symbolic links

Unload WdFilter via abusing NT symbolic links redirection	unDefender

3 Modify Microsoft Defender settings

PowerShell cmdlets	• Stop scanning and other security features • Add exclusion path • Block attaching to newly created volumes	TrickBot Trojan, Zloader Trojan, MosaicLoader, Kraken Botnet, P1, P2
WMI		DeroHE ransomware (n.b. wmic.exe is deprecated now)
Group Policy		Egregor ransomware, Defender Control
Registry values		R1

KERNEL-MODE ATTACKS

Token Structure

Callback notify routines
- Various Objects Callbacks
- ETW Callbacks

ETW kernel structures

6 Attacks on Token structure

Clear PPL to terminate WD apps	P1, P2, P3
Kernel-mode Sandboxing via patching token structure: • Token Integrity Level • Token Privileges	A newly proposed idea

7 Disable Various Kernel Callbacks

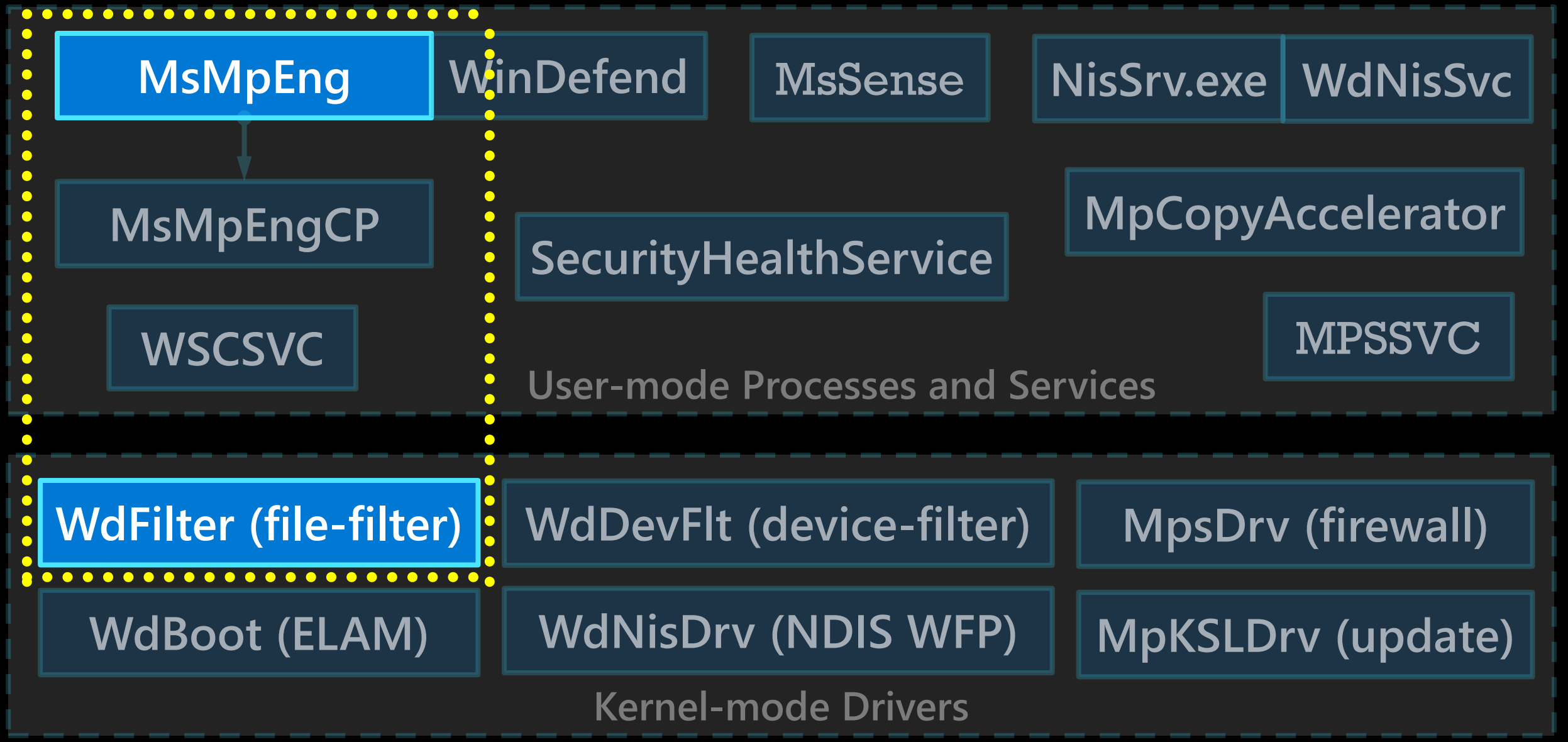
• Process, Thread, Image Callbacks • Registry Callbacks • Objects Callbacks • Minifilter Callbacks • Windows Filtering Platform Callouts • ETW Callbacks	CheekyBlinder, C1, C2, C3, C4, Lazarus Group's Rootkit Attack

8 Attacks on ETW structures

• Patching various ETW structures	ETW1

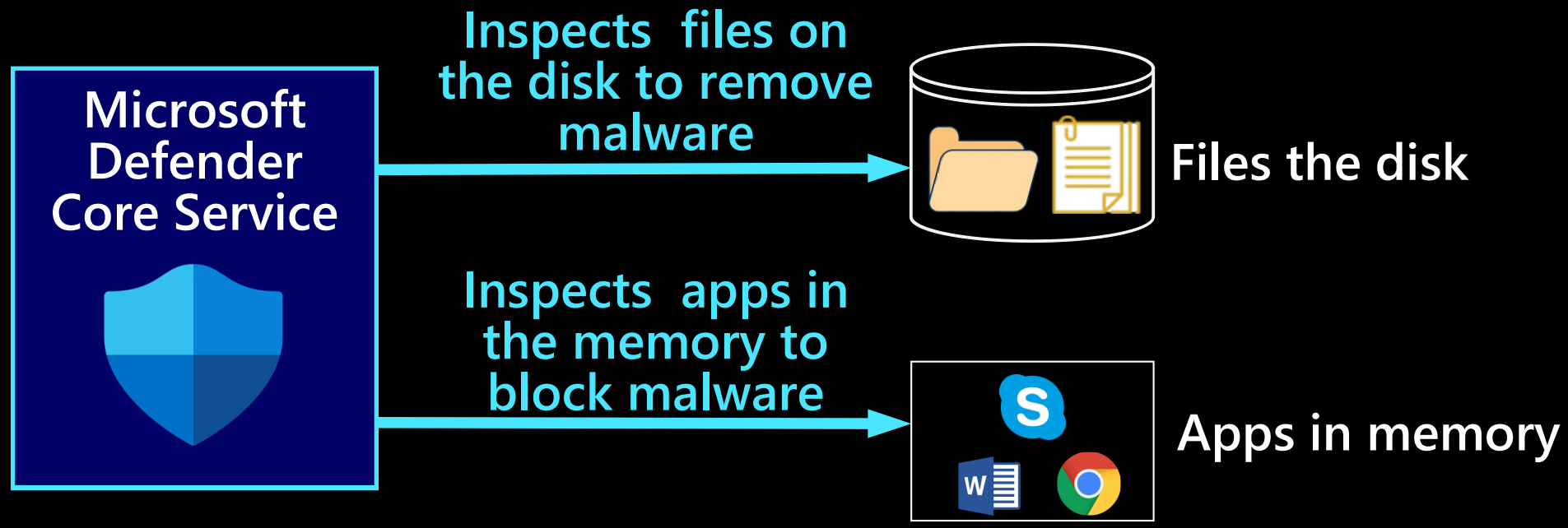
Microsoft Defender: components

MICROSOFT DEFENDER: ABOUT 10 APPS + 6 DRIVERS



- %ProgramData%\Microsoft\Microsoft Defender\Platform
- %Windir%\System32\drivers\wd

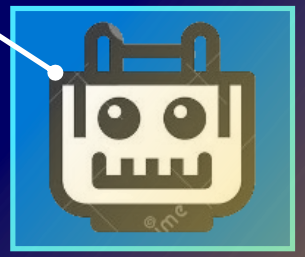
Microsoft Defender is inspecting disk and memory content



MpEngine.dll: Microsoft Malware Protection Engine

MpEngine.dll is the core of AV detection with 45,000+ functions

Microsoft Defender Core Service



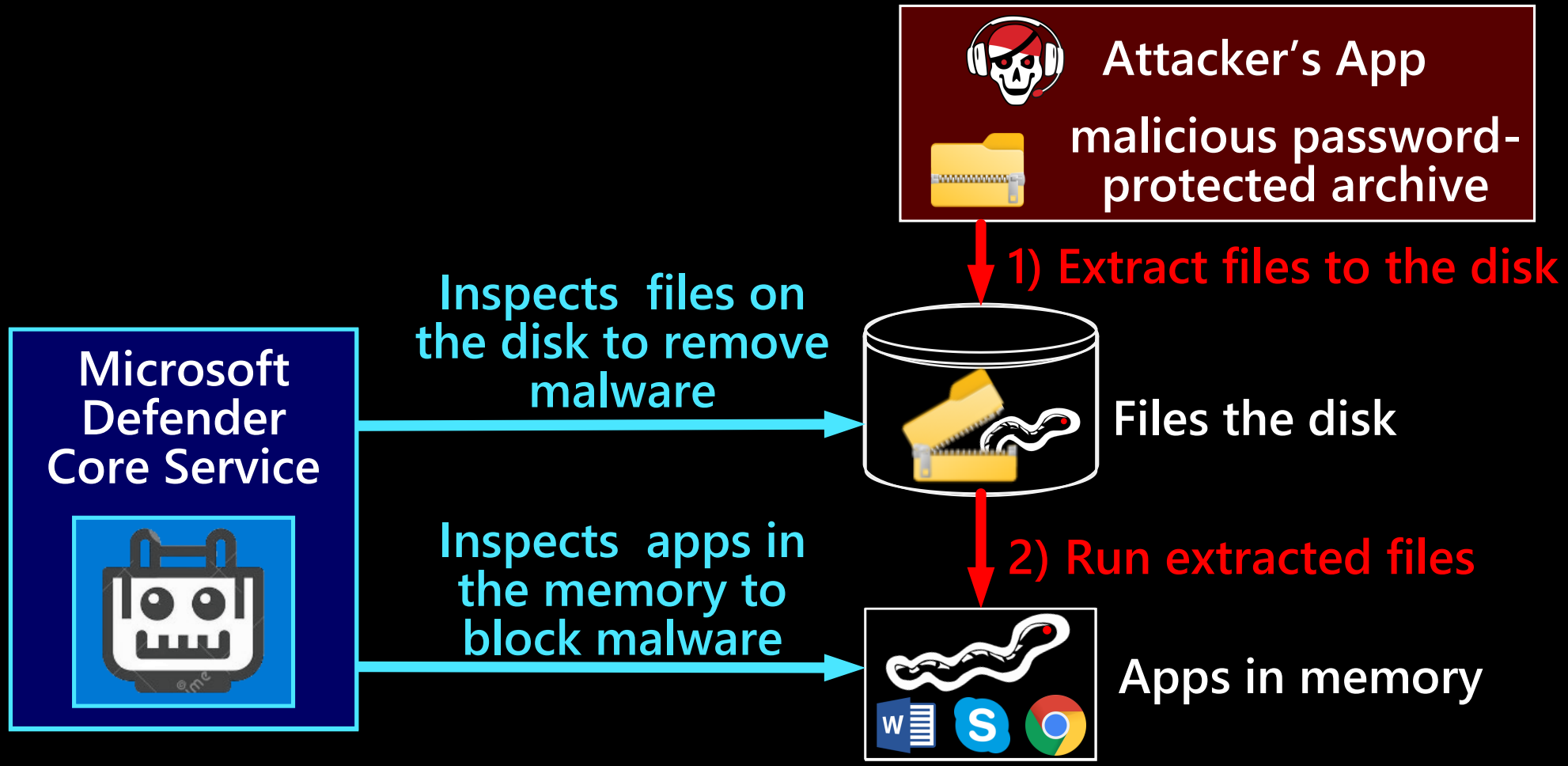
Inspects files on the disk to remove malware

Files the disk

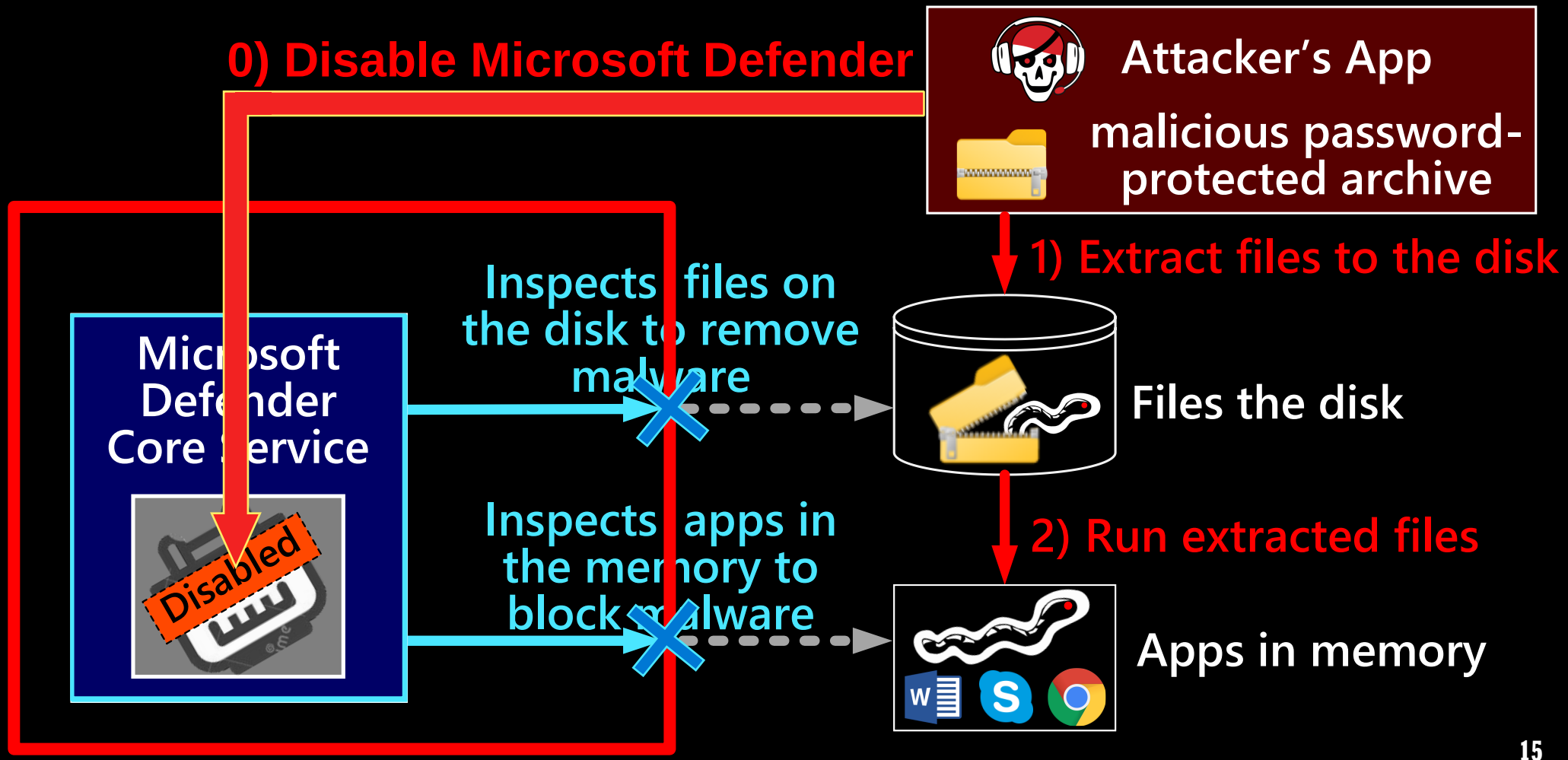
Inspects apps in the memory to block malware

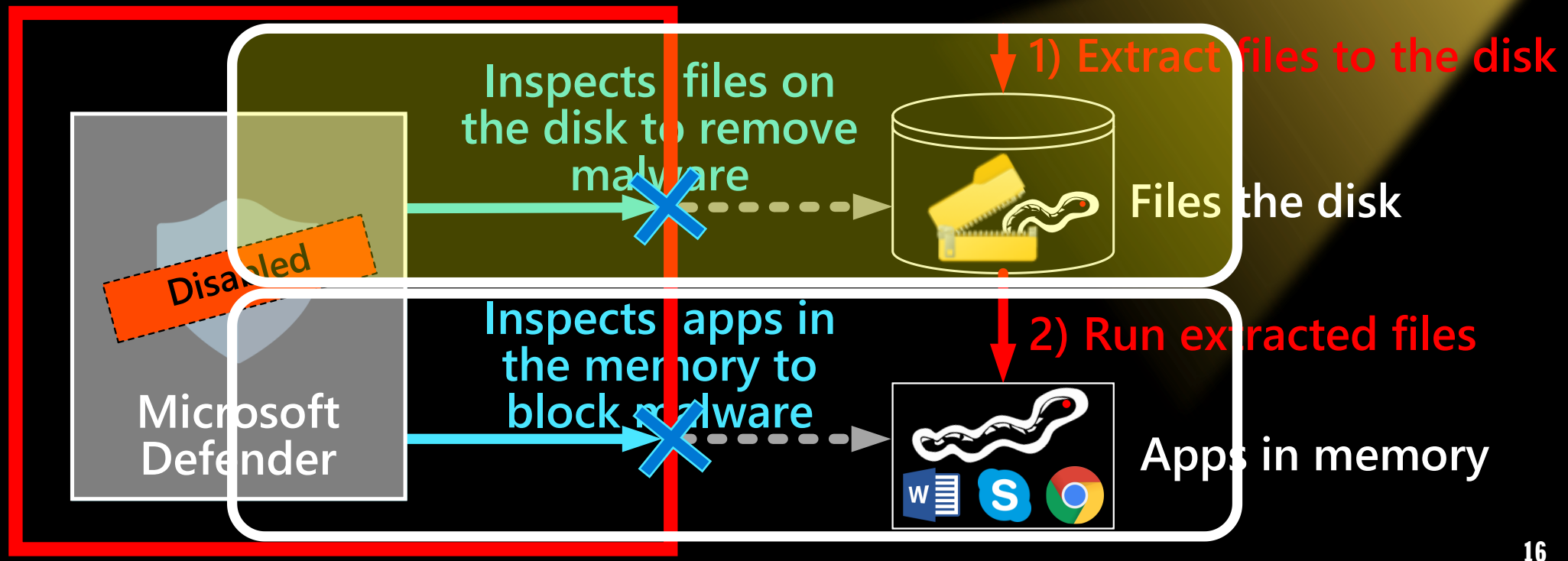
Apps in memory

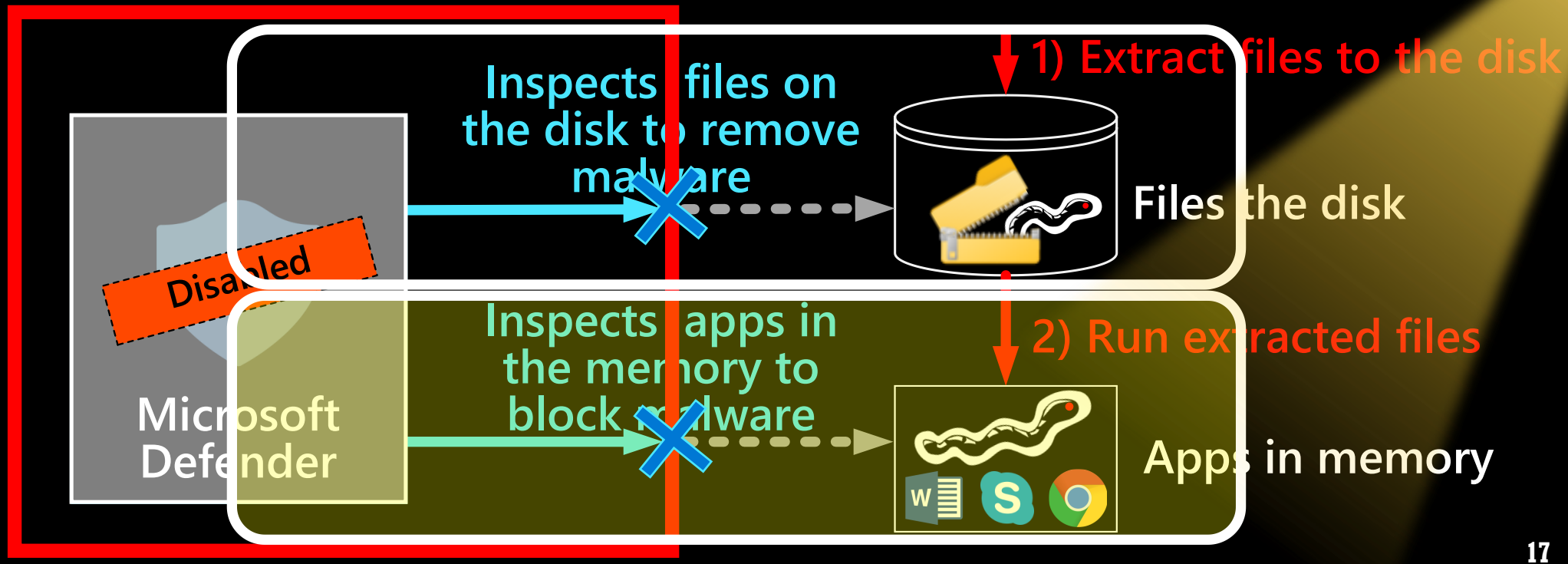
Microsoft Defender detects malware in files and memory



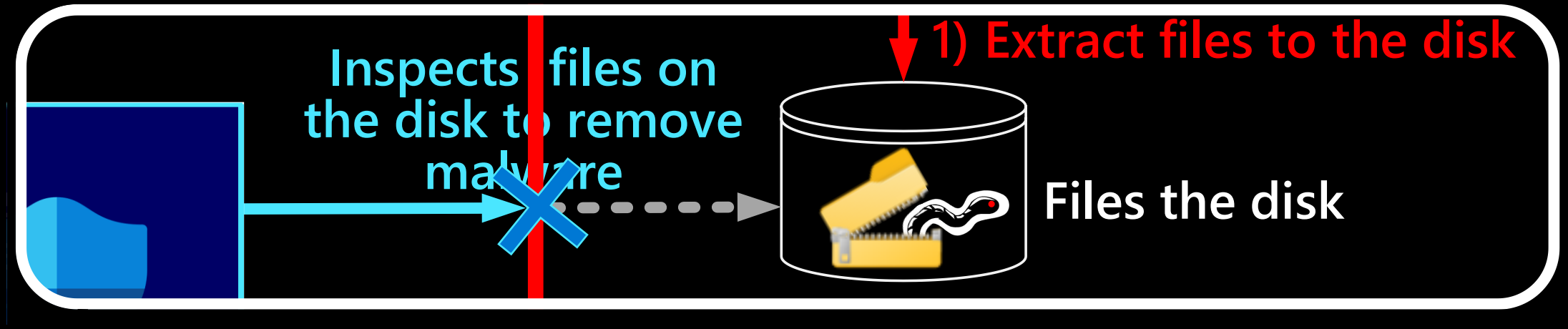
Attackers want to block inspections of both files and apps




Inspects files on the disk to remove malware
1) Extract files to the disk
Files the disk
Disabled
Microsoft Defender
Inspects apps in the memory to block malware
2) Run extracted files
Apps in memory

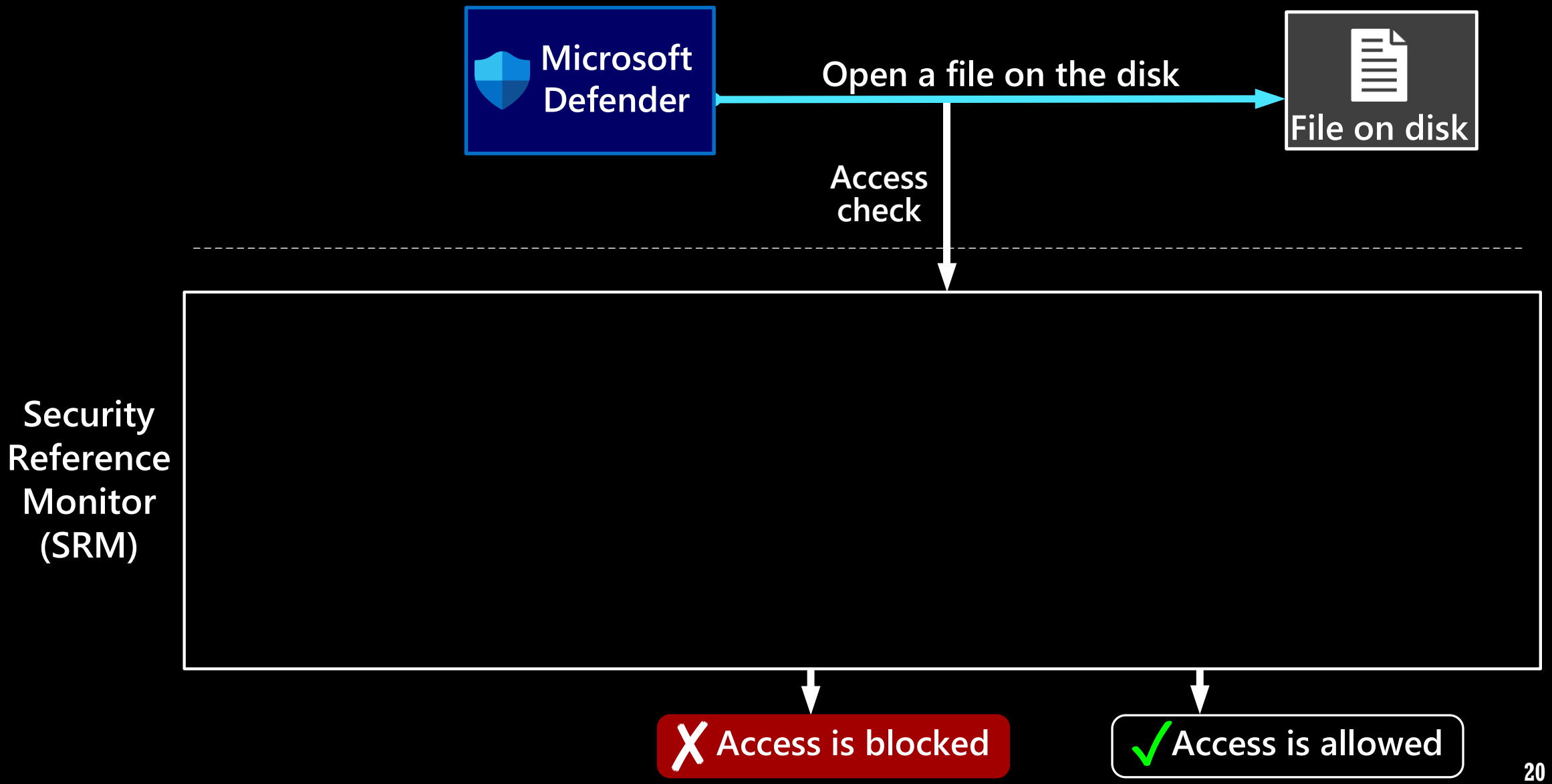

Inspects files on the disk to remove malware
1) Extract files to the disk
Files the disk
Disabled
Microsoft Defender
Inspects apps in the memory to block malware
2) Run extracted files
Apps in memory

Step1: Attackers want to block files inspection

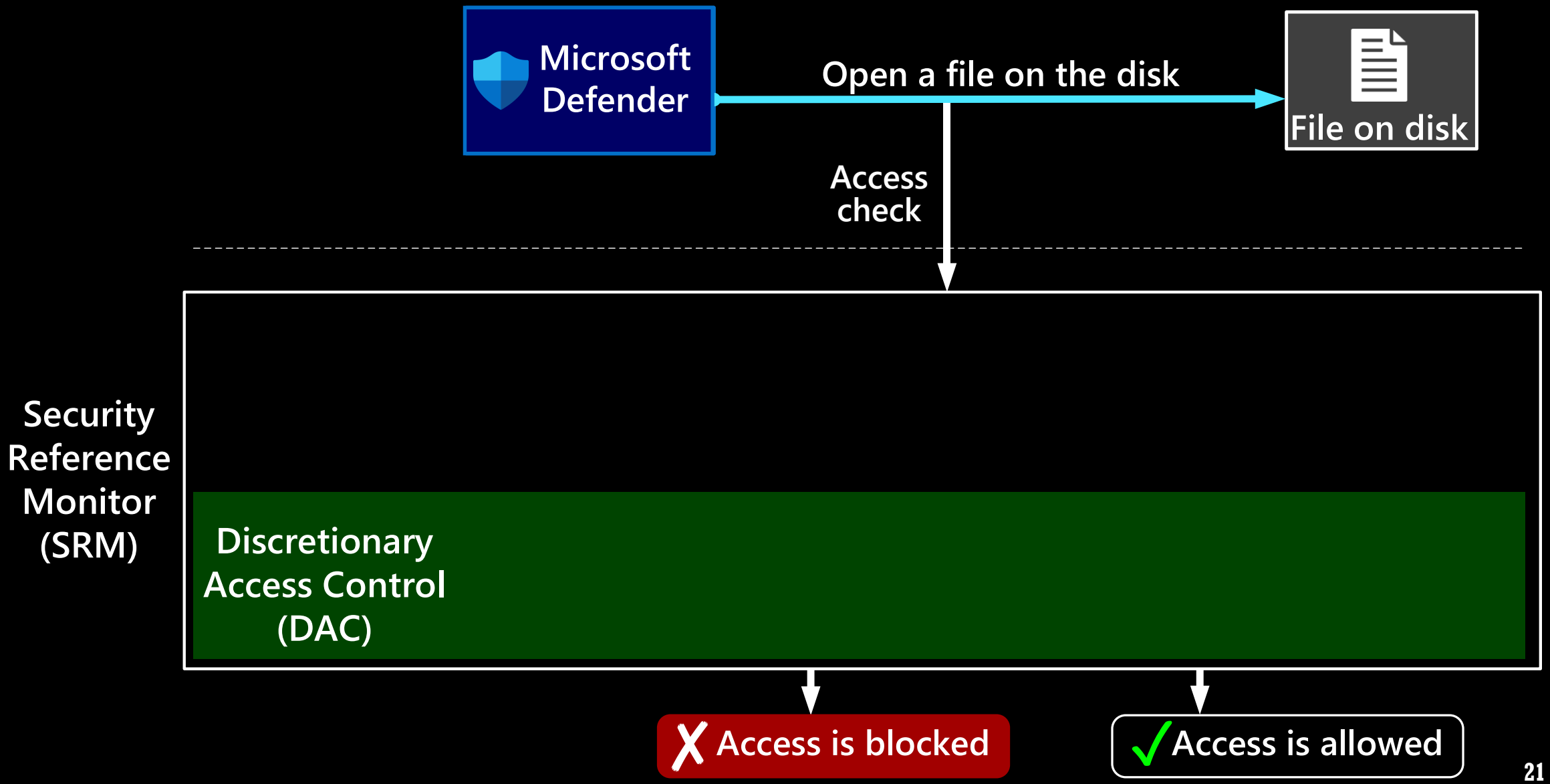


MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT

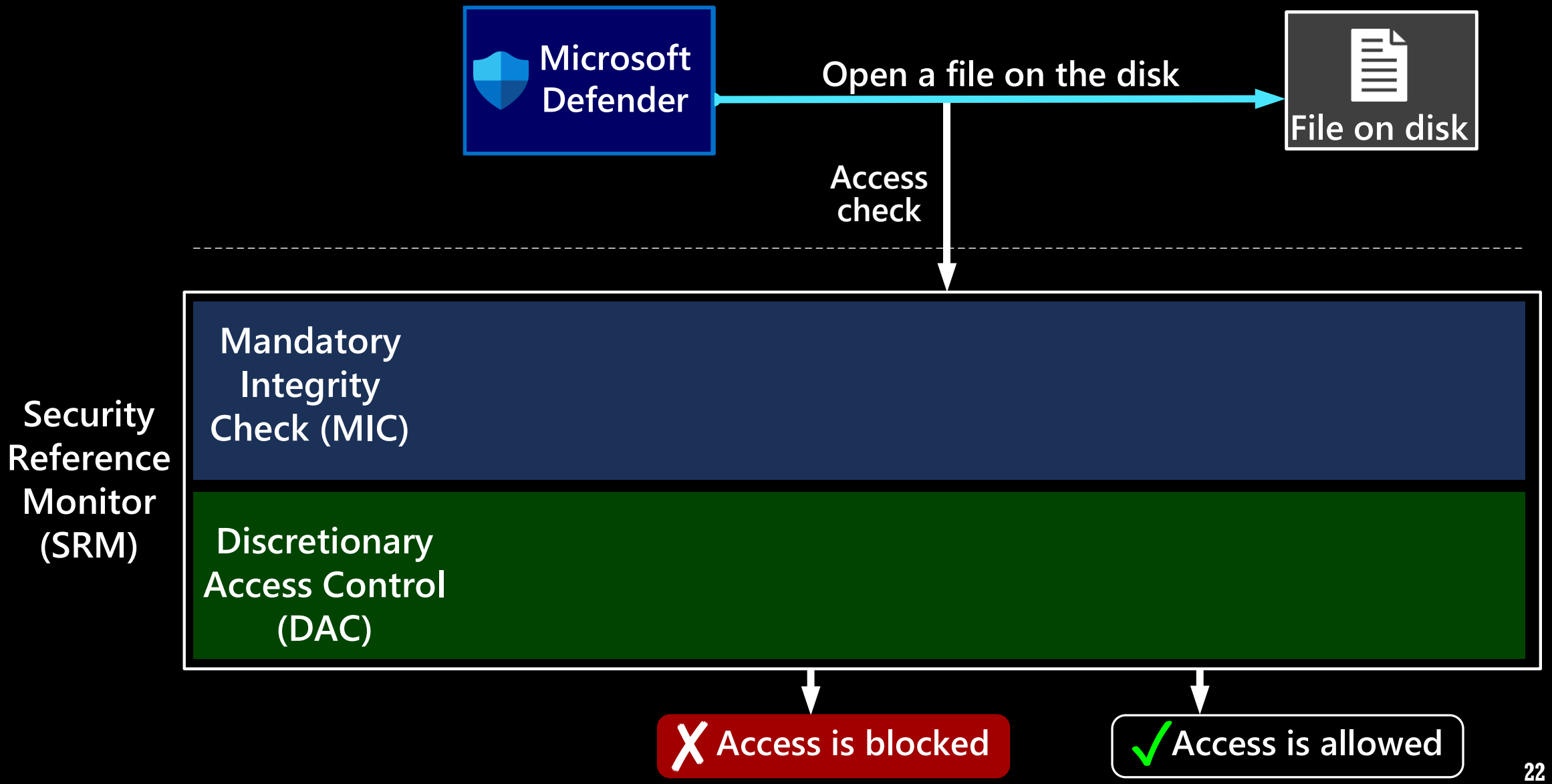
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



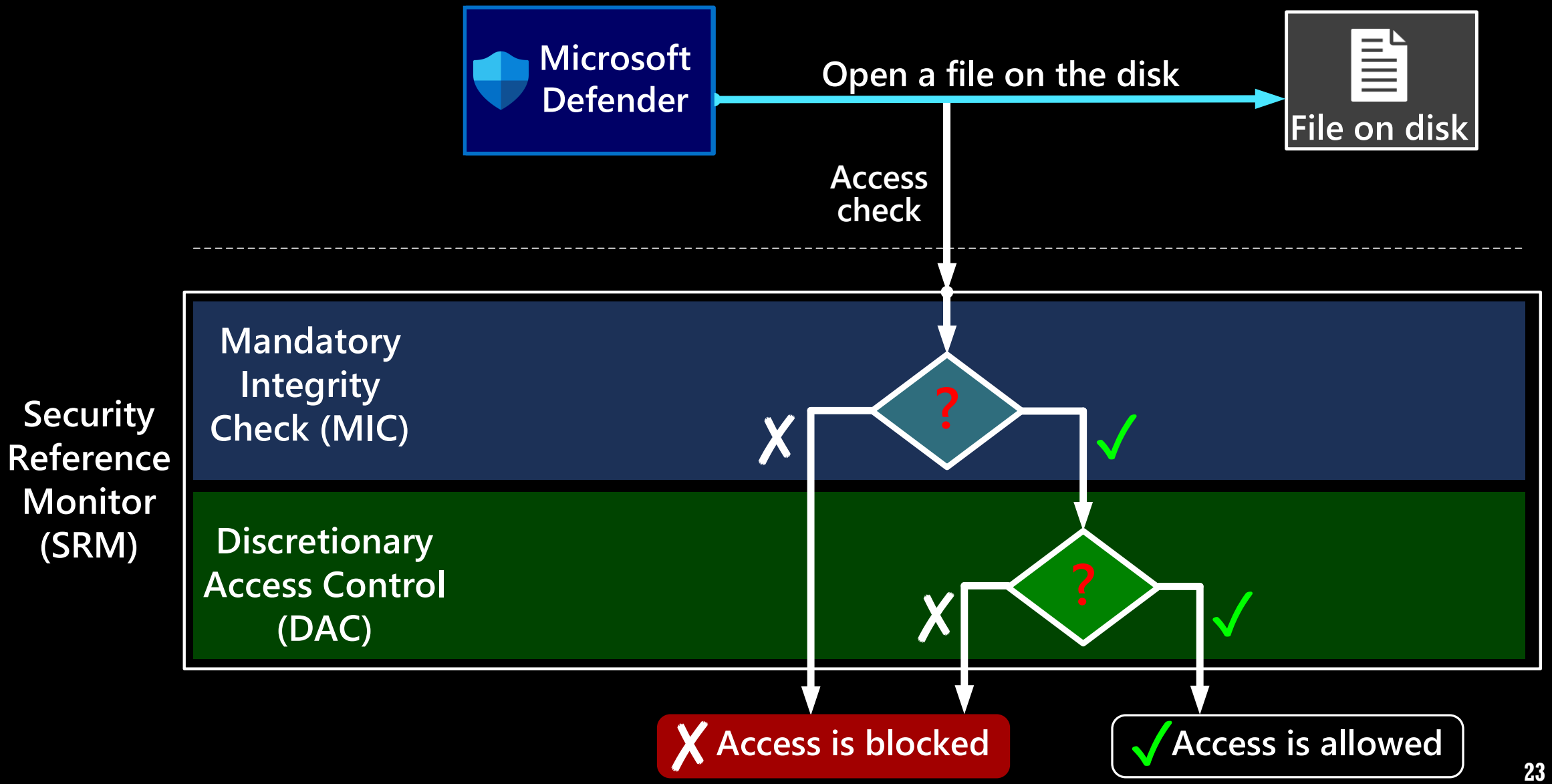
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



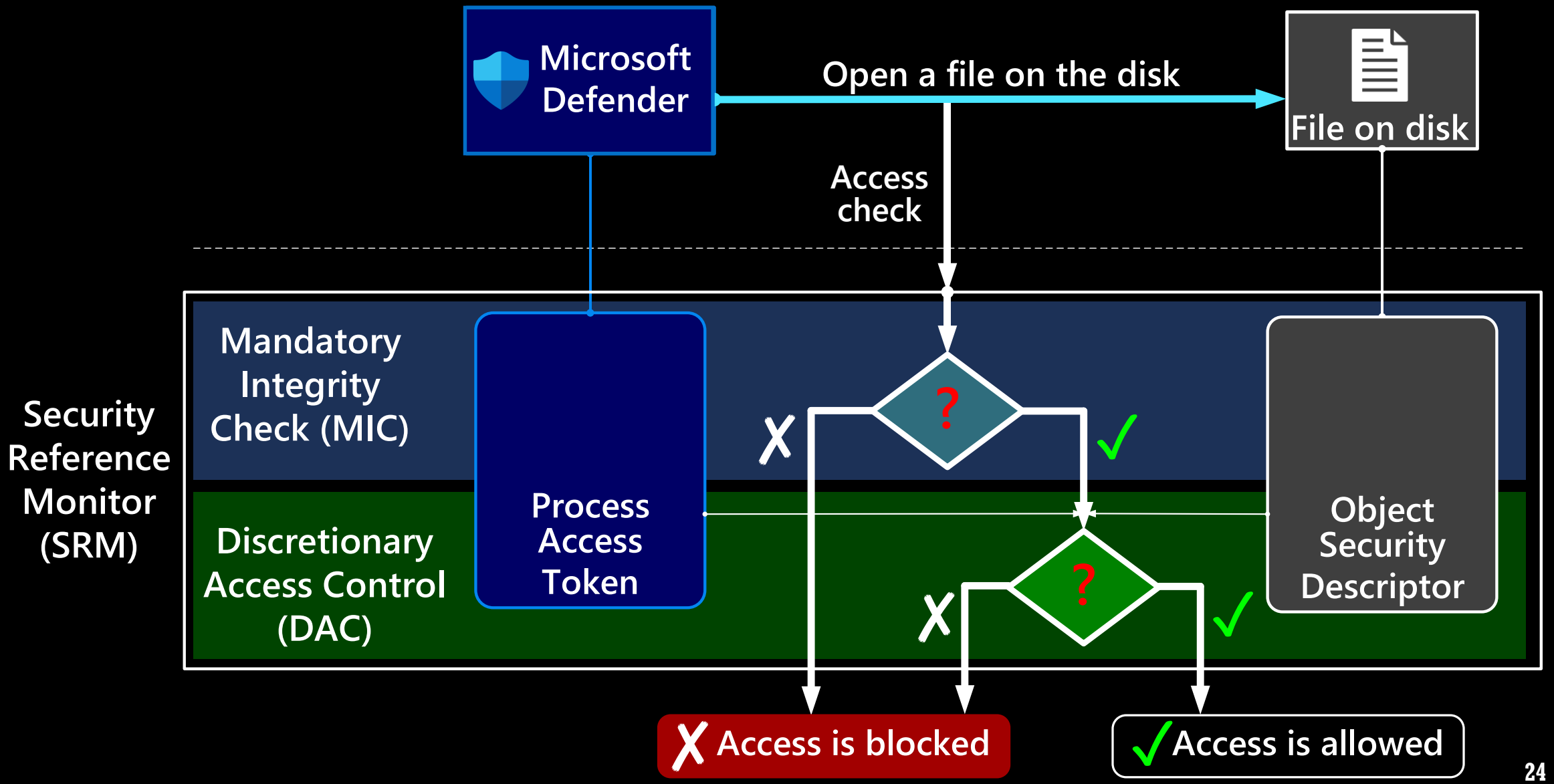
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



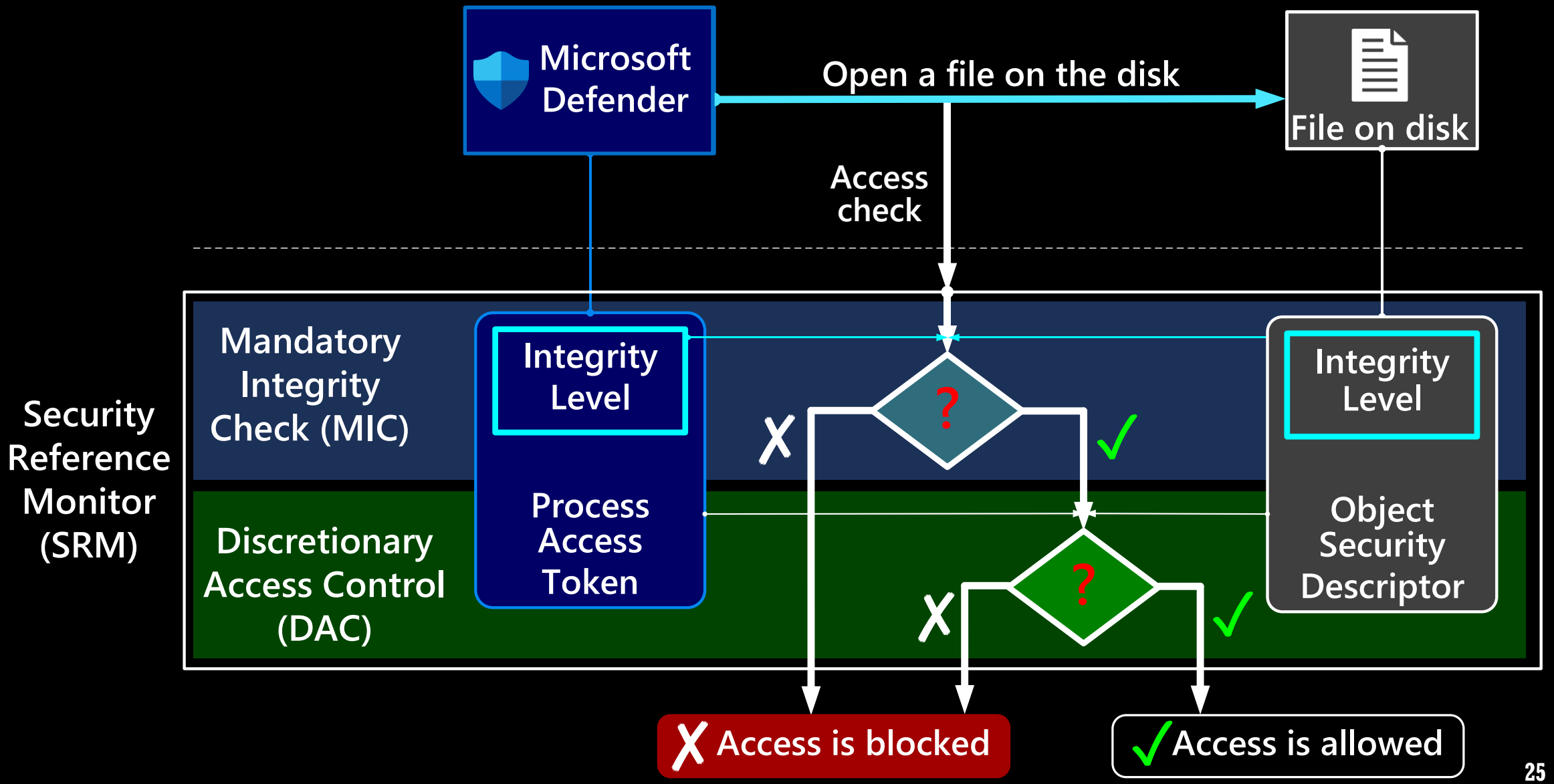
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



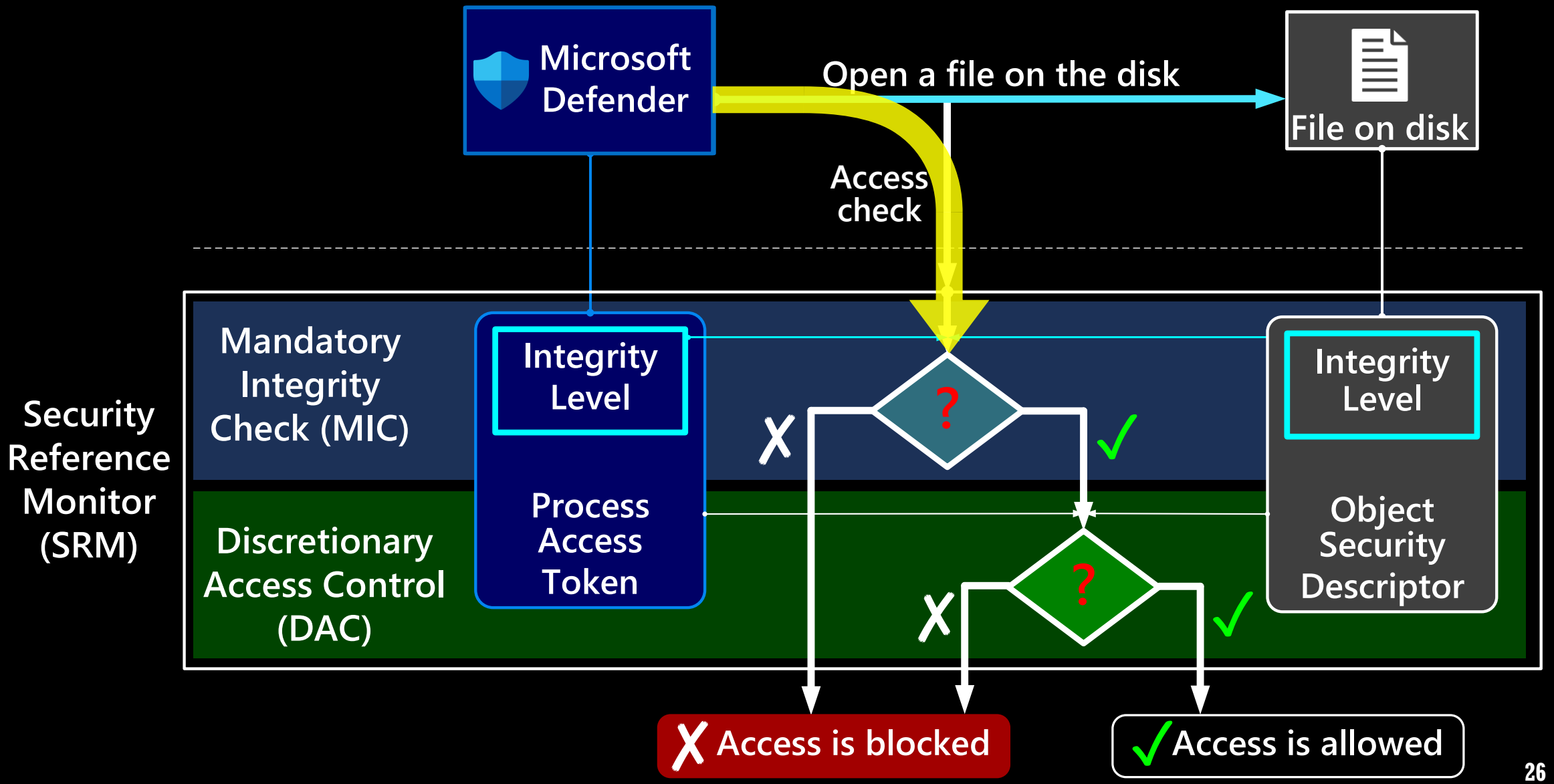
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



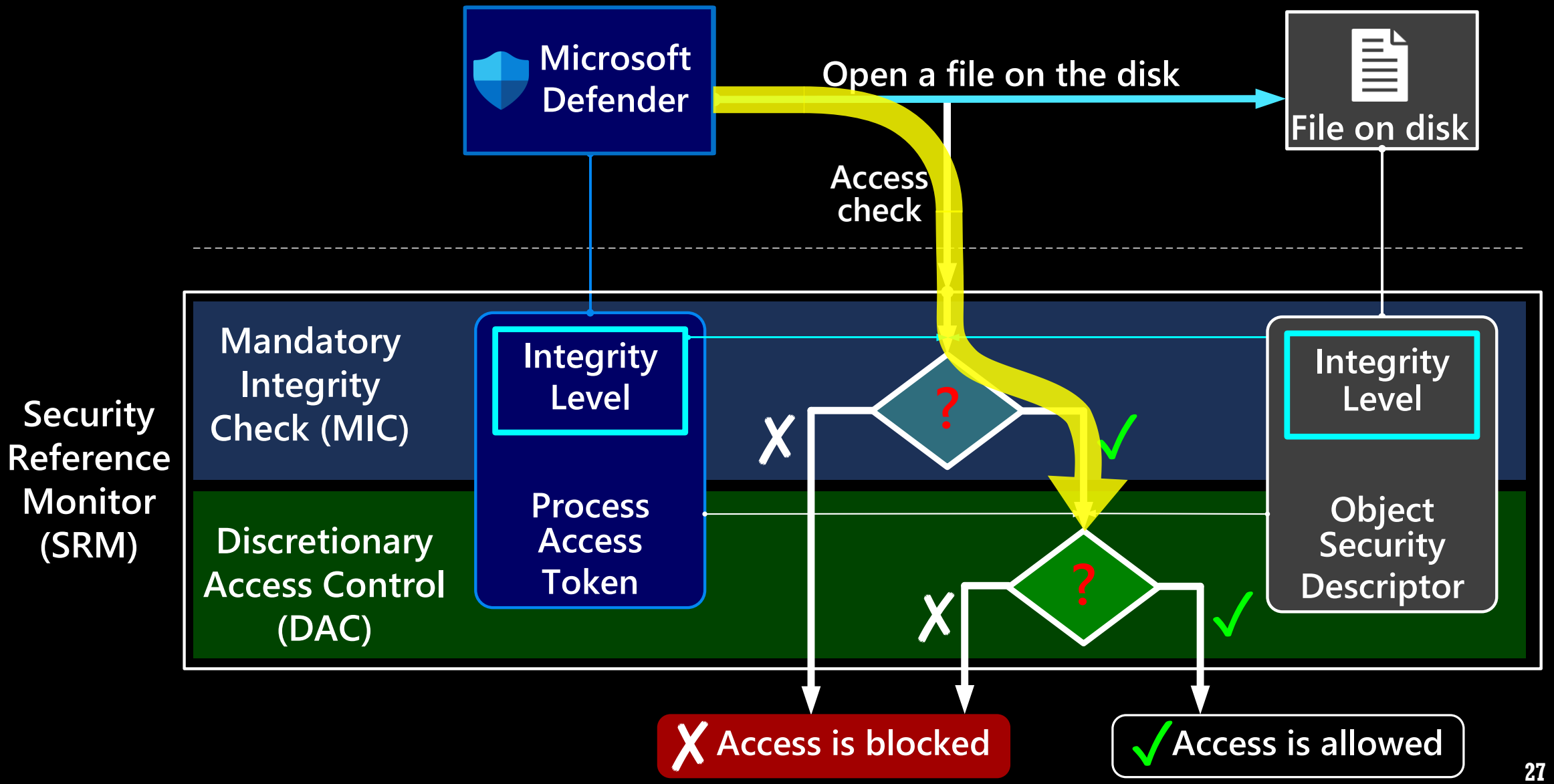
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



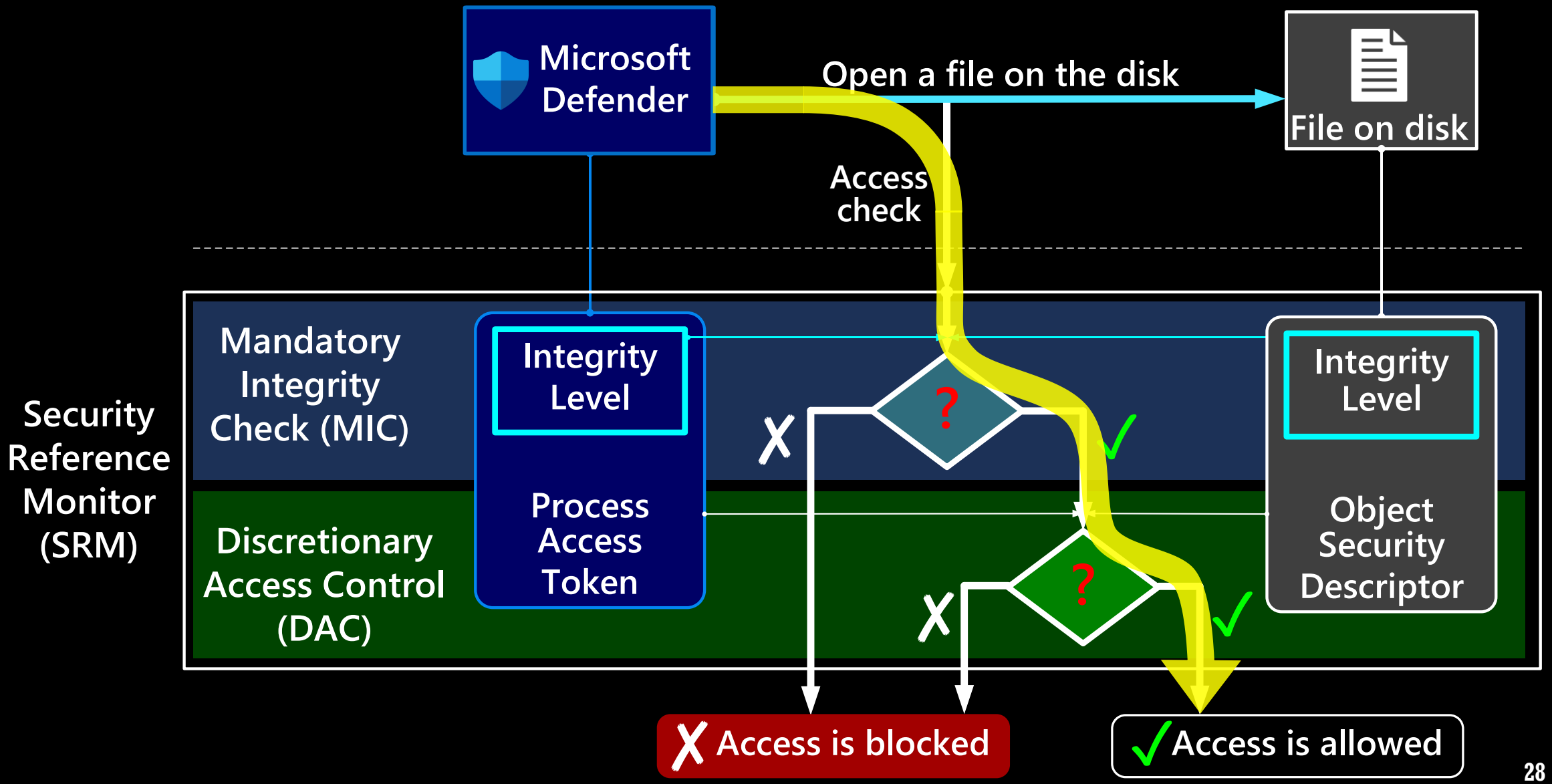
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



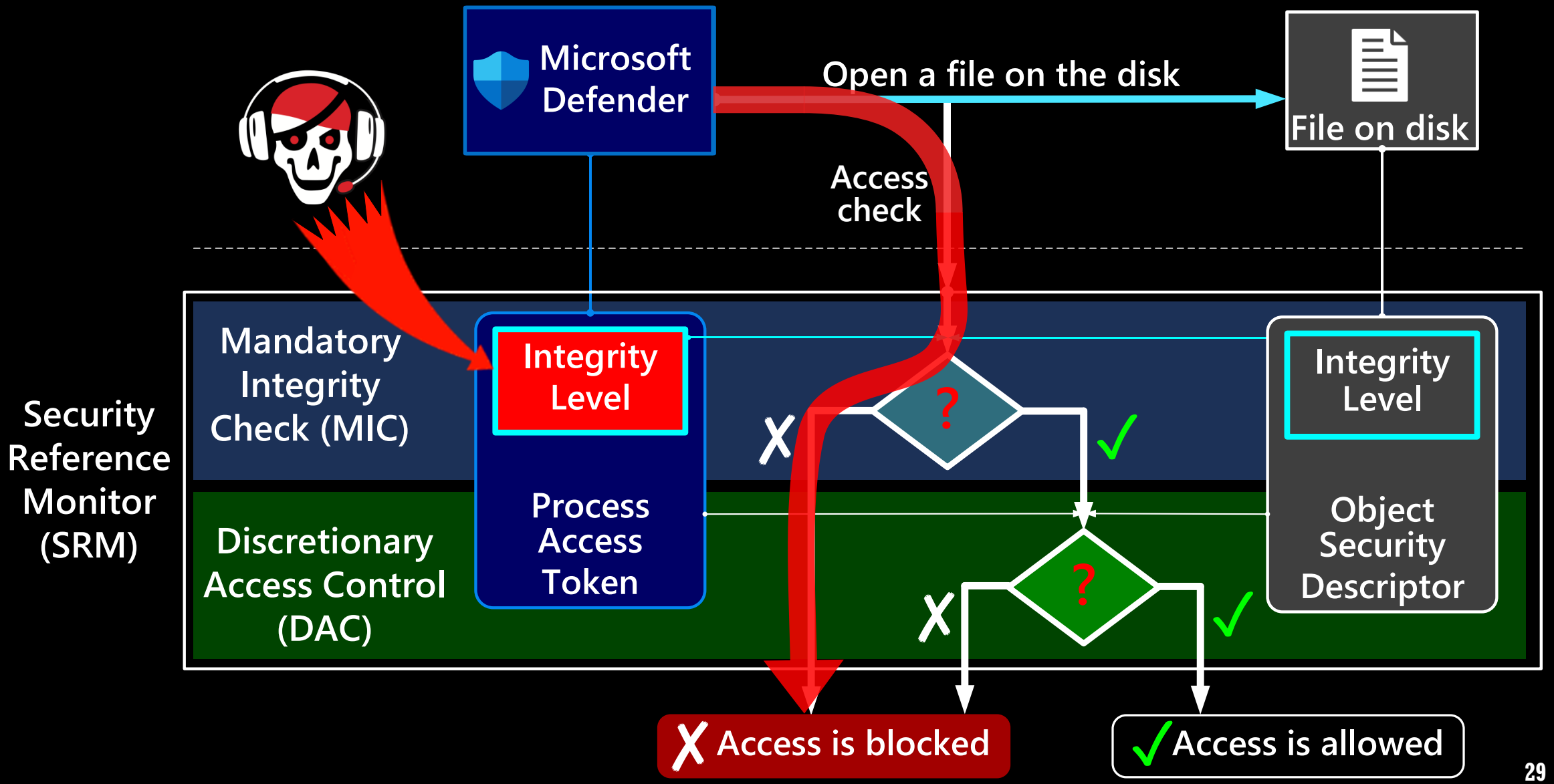
MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT

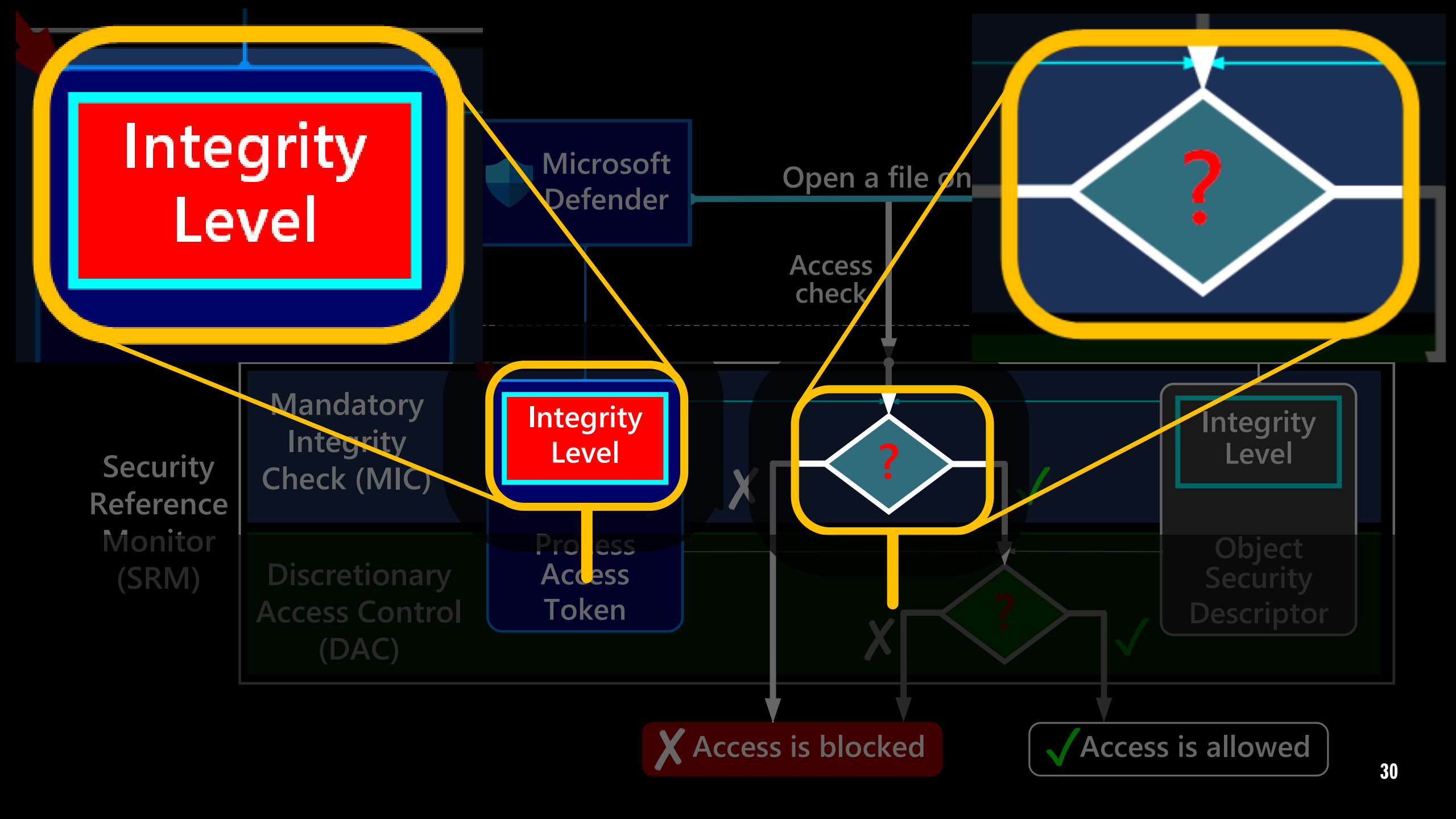


MICROSOFT DEFENDER OPENS A FILE TO INSPECT IT



PATCHING THE INTEGRITY LEVEL CAN BLOCK OPENING A FILE




Integrity Level
Microsoft Defender
Open a file on
Access check
?
Security Reference Monitor (SRM)
Mandatory Integrity Check (MIC)
Integrity Level
Process Access Token
Integrity Level
Discretionary Access Control (DAC)
Object Security Descriptor
✗ Access is blocked
✓ Access is allowed

STEP 1.1 INTEGRITY LEVELS: INTRO

MIC: Integrity Levels

Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

MIC: Integrity Levels

Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

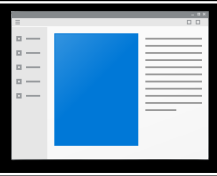
MIC: Integrity Levels

Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

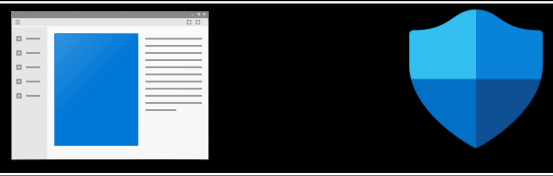
MIC: Integrity Levels

Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

MIC: Integrity Levels

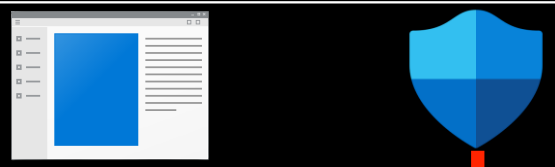
Integrity Levels	Examples
System	 
High	
Medium	 
Untrusted	 
Low	

MIC: Integrity Levels

Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

$$if\ (Level_{APP} \geq Level_{FILE}) \rightarrow access\ allowed$$
$$else \rightarrow access\ denied$$

MIC: Integrity Levels

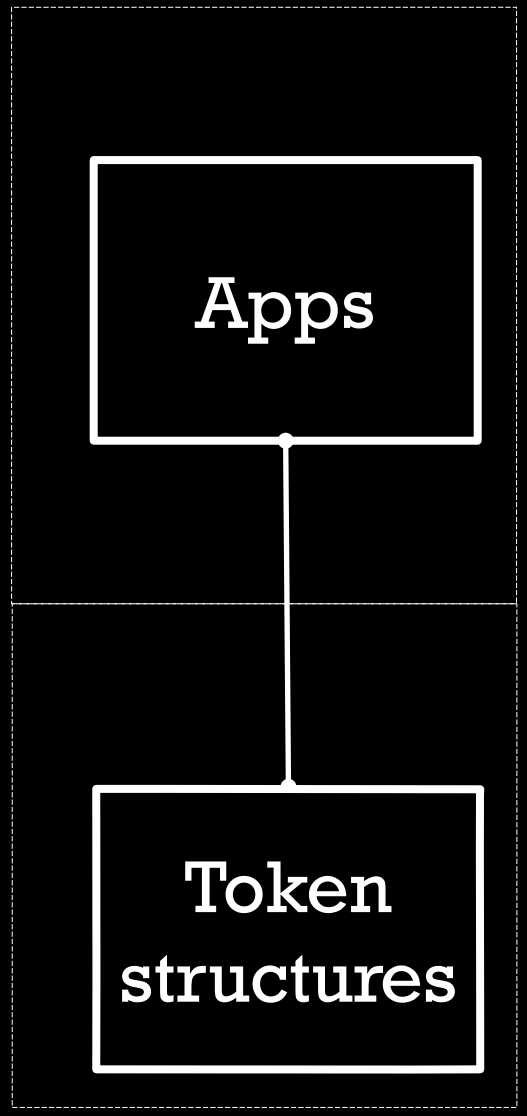
Integrity Levels	Examples
System	
High	
Medium	
Untrusted	
Low	

$$if\ (Level_{APP} \geq Level_{FILE}) \rightarrow access\ allowed$$
$$else \rightarrow access\ denied$$

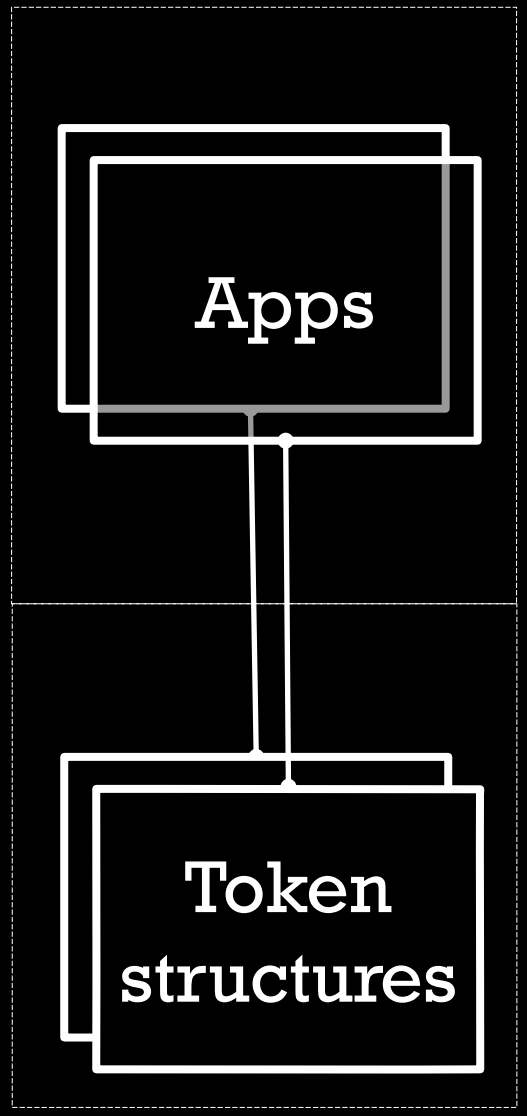
How Integrity levels are stored in memory?

STEP 1.2 INTEGRITY LEVELS: DATA

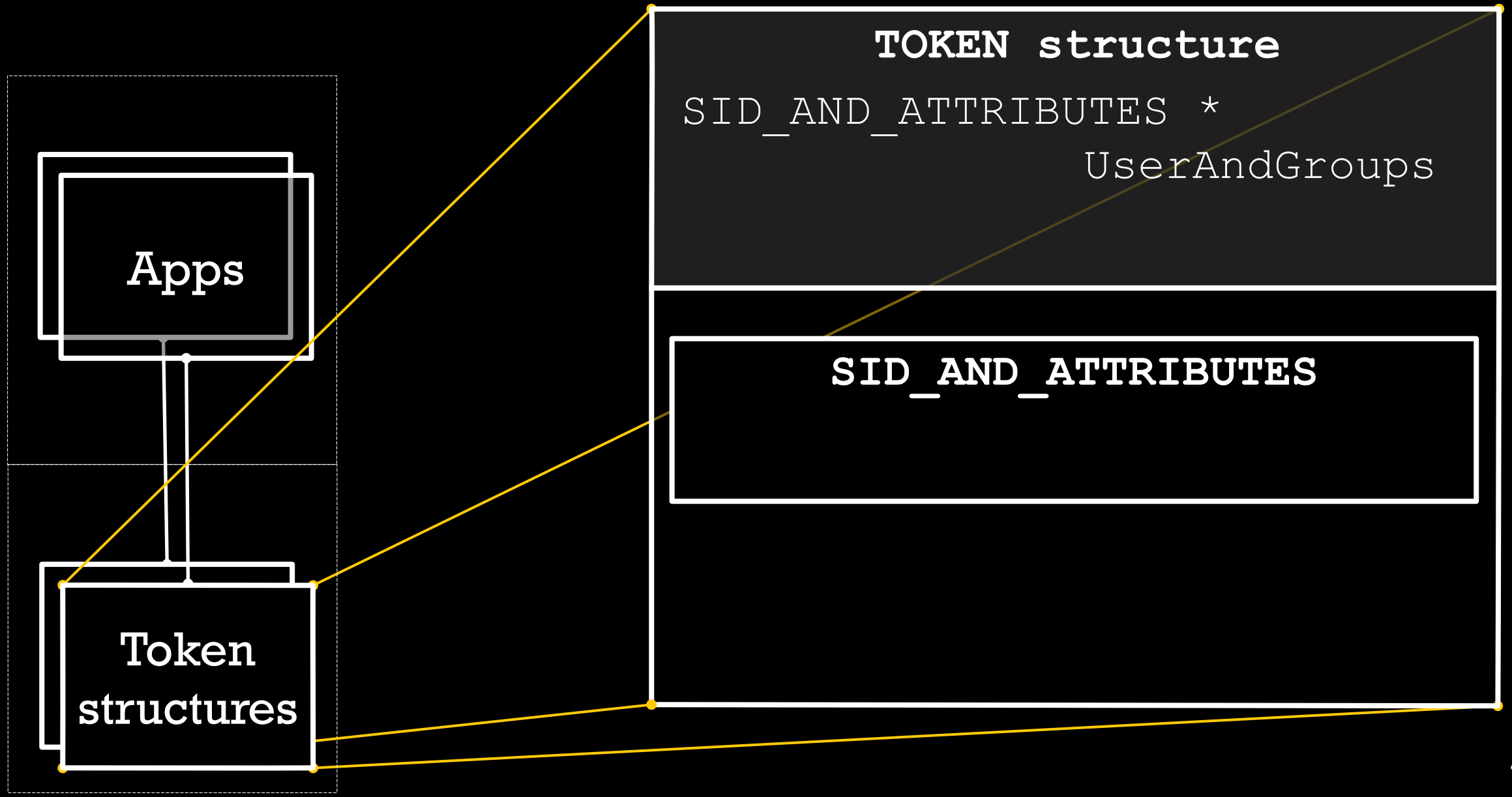
INTEGRITY LEVEL: TOKEN AND INDEX



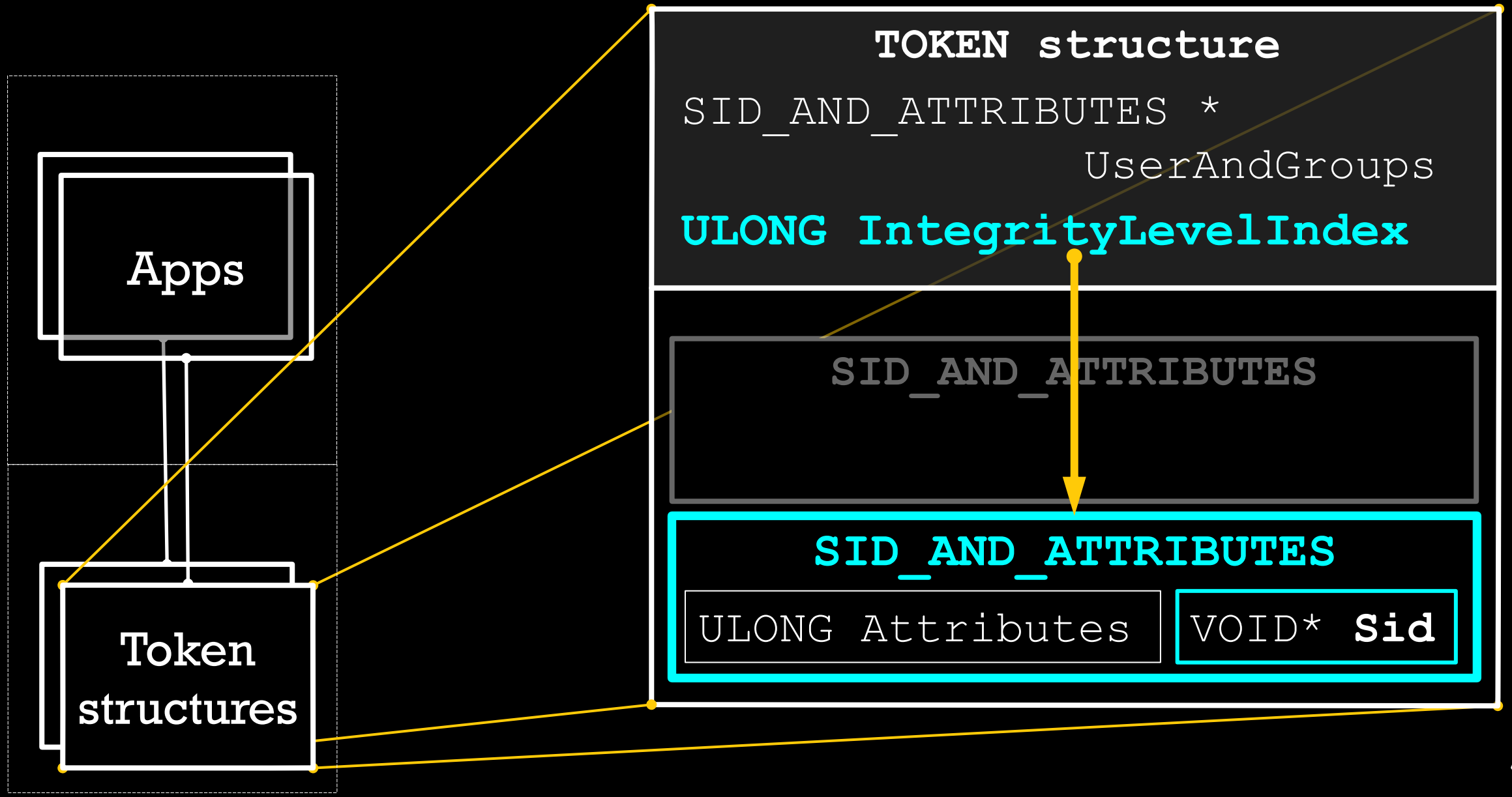
INTEGRITY LEVEL: TOKEN AND INDEX



INTEGRITY LEVEL: TOKEN AND INDEX



INTEGRITY LEVEL: TOKEN AND INDEX

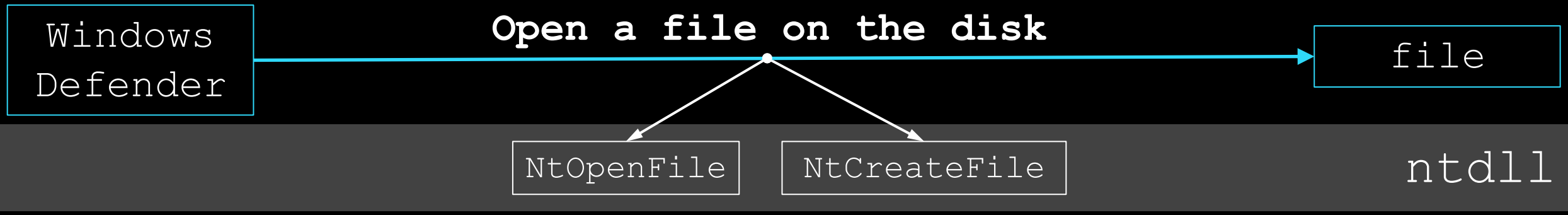


How IntegrityLevelIndex is used during Access Check?

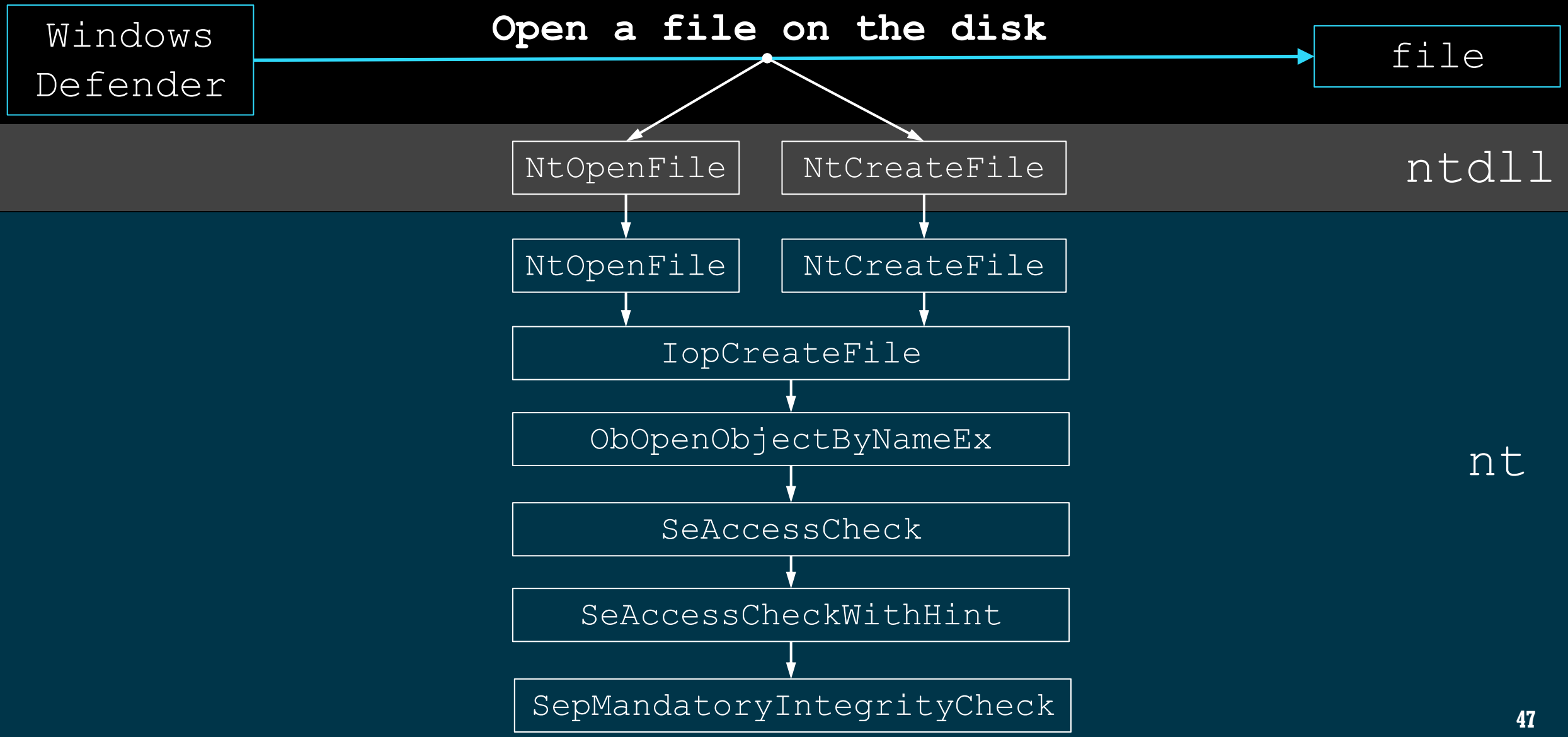
STEP 1.3 INTEGRITY LEVELS: CODE

MIC: Check Integrity Level

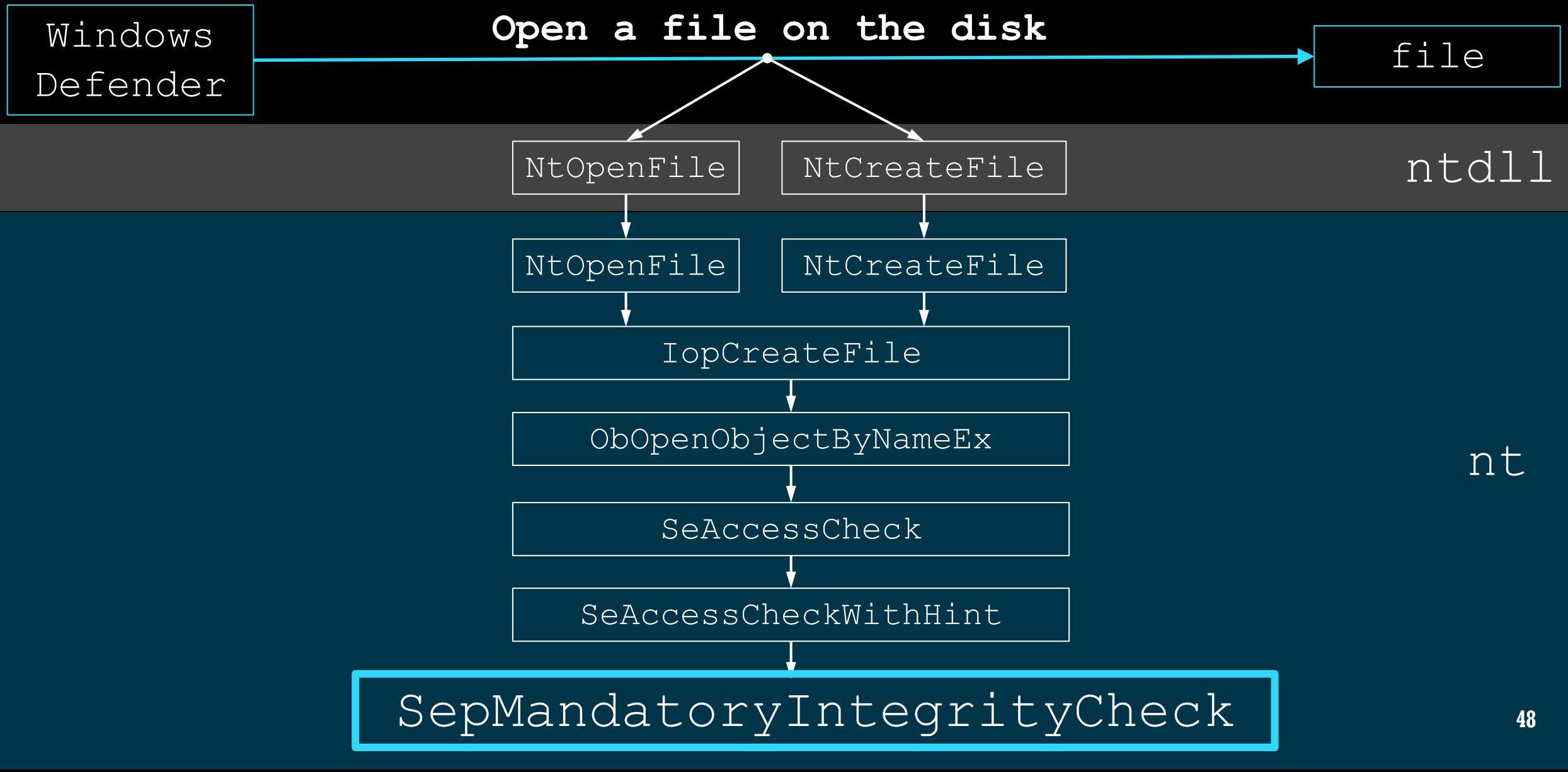
MIC: Check Integrity Level



MIC: Check Integrity Level



MIC: Check Integrity Level



```
NTSTATUS SepMandatoryIntegrityCheck(IN PTOKEN Token)
{
        NTSTATUS status;
        PSID ProcSid;
        ULONG64 index = Token->IntegrityLevelIndex;
        if (index == -1)
        {
                ProcSid = SeUntrustedMandatorySid;
        }
        else
        {
                ProcSid = Token->UserAndGroups[index]->Sid;
        }
        ...
        return status;
}
```

```
NTSTATUS SepMandatoryIntegrityCheck(IN PTOKEN Token)
{
        NTSTATUS status;
        PSID ProcSid;
        ULONG64 index = Token->IntegrityLevelIndex;
        if (index == -1)
        {
                ProcSid = SeUntrustedMandatorySid;
        }
        else
        {
                ProcSid = Token->UserAndGroups[index]->Sid;
        }
        ...
        return status;
}
```

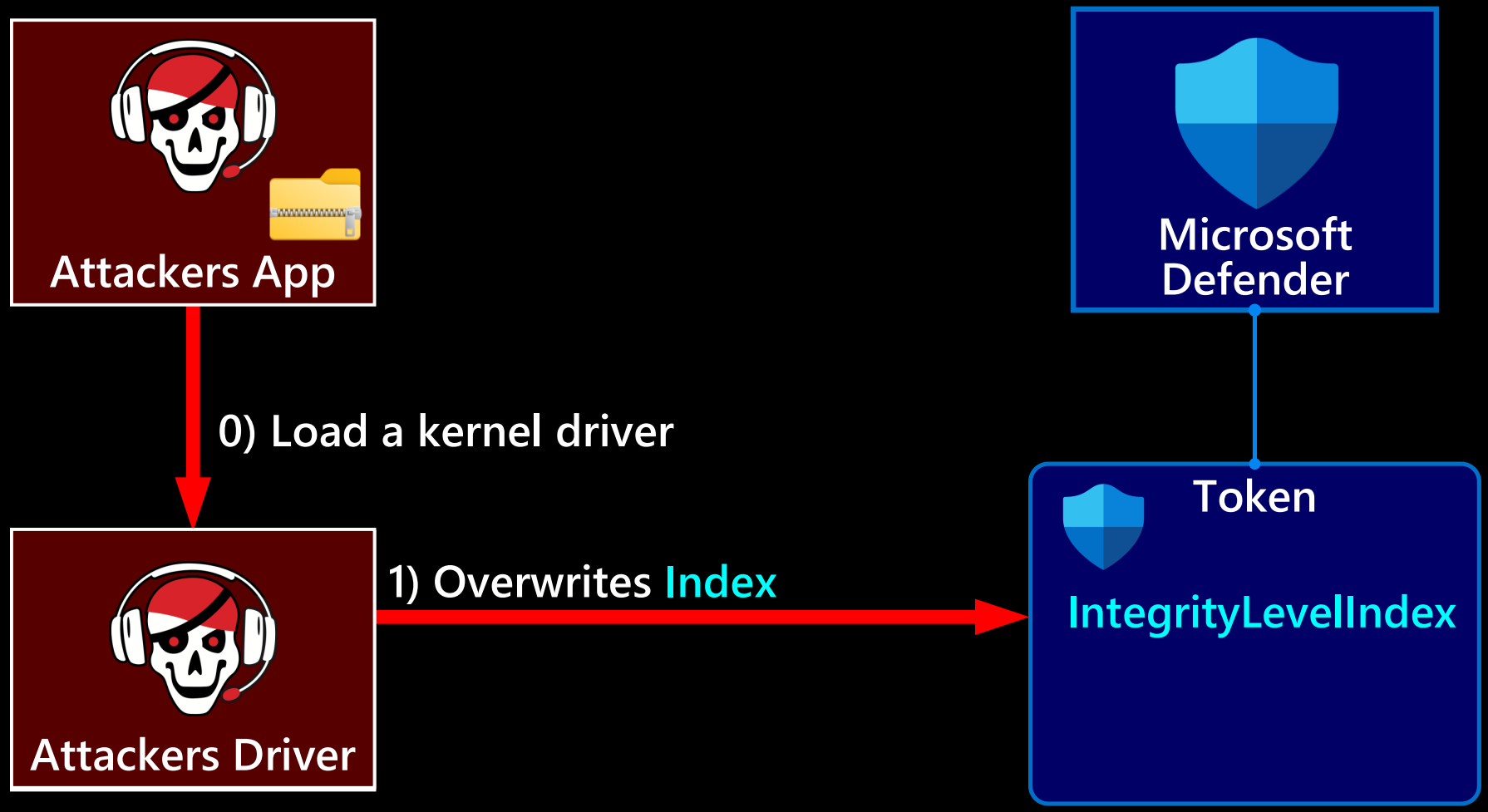
```
NTSTATUS SepMandatoryIntegrityCheck(IN PTOKEN Token)
{
        NTSTATUS status;
        PSID ProcSid;
        ULONG64 index = Token->IntegrityLevelIndex;
        if (index == -1)
        {
                ProcSid = SeUntrustedMandatorySid;
        }
        else
        {
                ProcSid = Token->UserAndGroups[index]->Sid;
        }
        ...
        return status;
}
```

```
NTSTATUS SepMandatoryIntegrityCheck(IN PTOKEN Token)
{
        NTSTATUS status;
        PSID ProcSid;
        ULONG64 index = Token->IntegrityLevelIndex;
        if (index == -1)
        {
                ProcSid = SeUntrustedMandatorySid;
        }
        else
        {
                ProcSid = Token->UserAndGroups[index]->Sid;
        }
        ...
        return status;
}
```

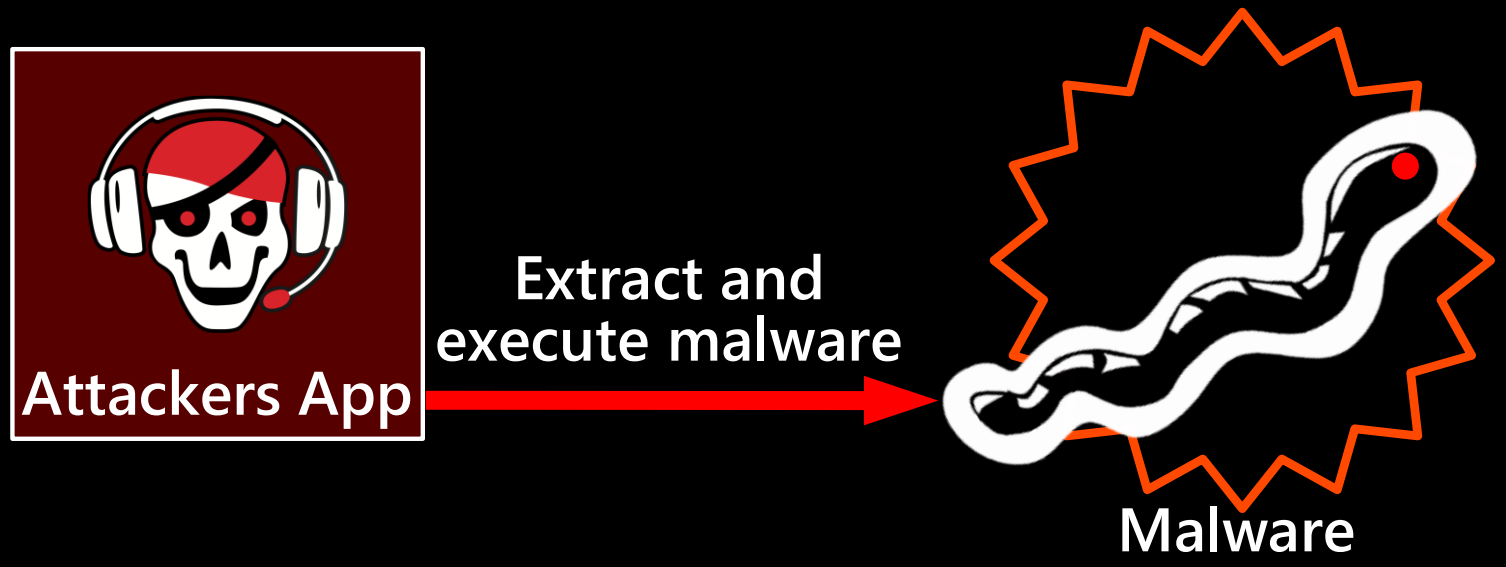
Patching IntegrityLevelIndex blocks files inspection

STEP 1.4 INTEGRITY LEVELS: ATTACK

Attackers lower Integrity Level for Microsoft Defender

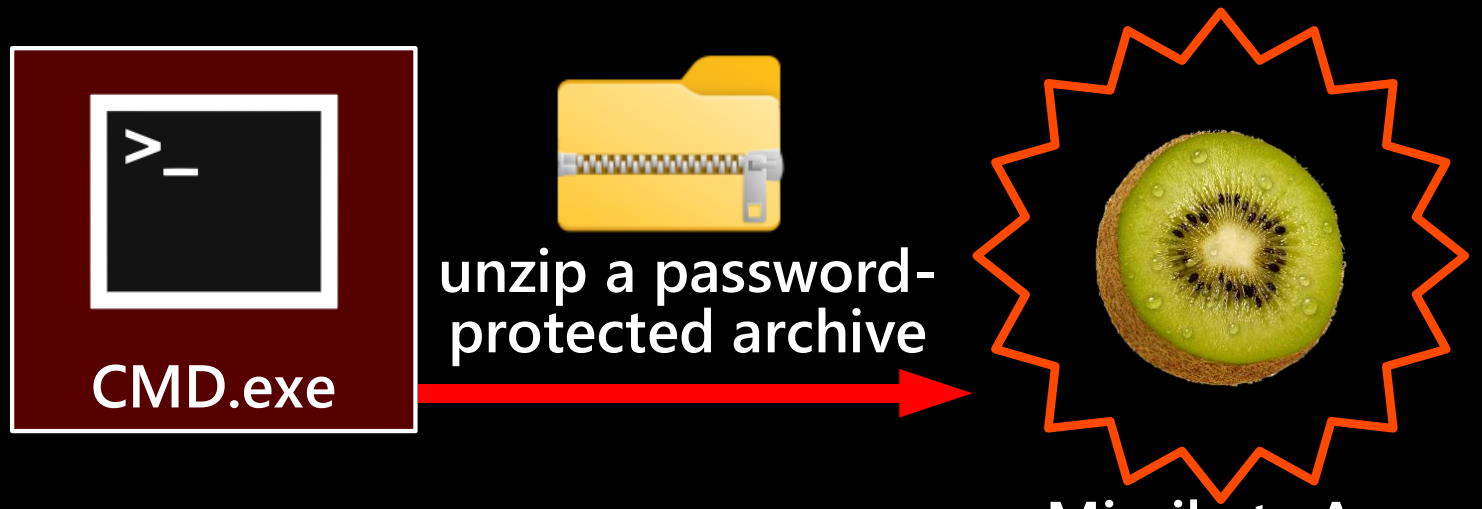


Attackers check that Microsoft Defender is ON via executing a malware from a password-protected archive



Attackers check that Microsoft Defender is ON via executing a malware from a password-protected archive

Attackers check that Microsoft Defender is ON via executing a malware from a password-protected archive

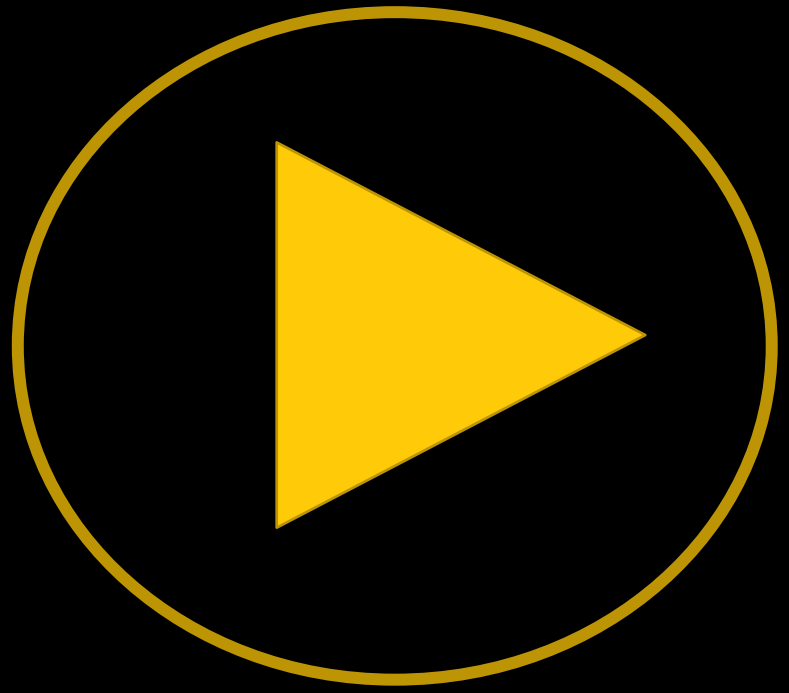


clear_extract_and_check.bat:

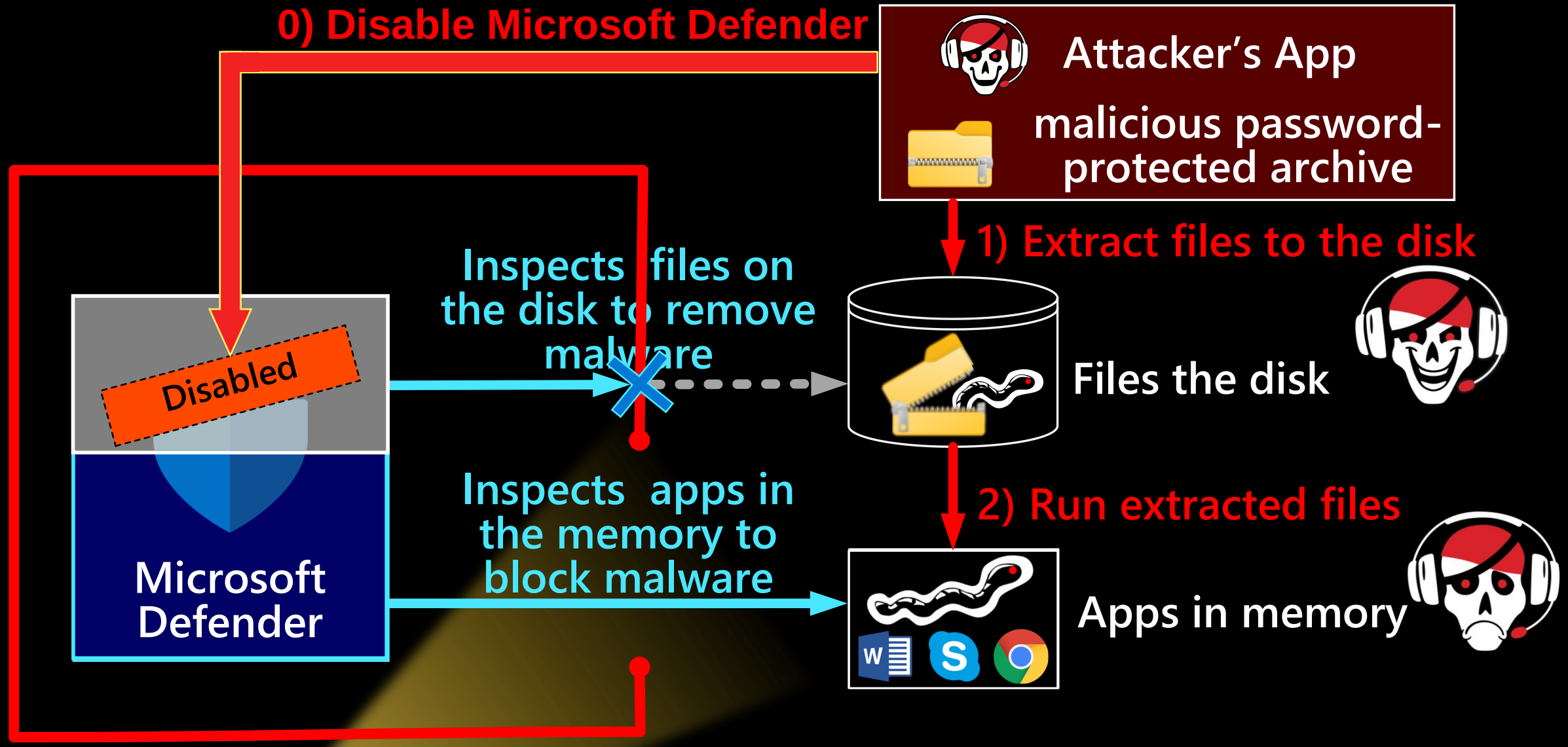
```
rmdir mimikatz /S/Q

7z.exe x "mimikatz.zip" -aos -o"mimikatz" -pinfected
dir    "mimikatz\mimikatz_trunk\x64"
start "mimikatz\mimikatz_trunk\x64\mimikatz.exe"
```

ATTACK ON MIC: DEMO



The online version is here –
https://www.youtube.com/embed/AJV4UVaw8kg?vq=hd1440


0) Disable Microsoft Defender
Attacker's App
malicious password-protected archive
1) Extract files to the disk
Inspects files on the disk to remove malware
Files the disk
Disabled
Microsoft Defender
Inspects apps in the memory to block malware
2) Run extracted files
Apps in memory

SUMMARY

- Microsoft Defender app removes malware files via call:

```
FILE_DISPOSITION_INFORMATION file_info;
file_info.DeleteFile = TRUE;
NtSetInformationFile(mlwr_handle, &file_info);
```

- CMD fails to launch mimikatz with

  STATUS_VIRUS_INFECTED (0xC0000906)

  that is returned to block launching a malware app

SUMMARY

- Microsoft Defender app removes malware files via call:

```
FILE_DISPOSITION_INFORMATION file_info;
file_info.DeleteFile = TRUE;
NtSetInformationFile(mlwr_handle, &file_info);
```

- CMD fails to launch mimikatz with

  STATUS_VIRUS_INFECTED (0xC0000906)

  that is returned to block launching a malware app

Which driver returns this status?

Microsoft Defender Antivirus Mini-Filter Driver (WdFilter)

- It register a mini-filter via FltRegisterFilter()
- It prevents launching a malware app via post-create callback

Microsoft Defender Antivirus Mini-Filter Driver (WdFilter)

- It register a mini-filter via FltRegisterFilter()
- It prevents launching a malware app via post-create callback

```
FLT_POSTOP_CALLBACK_STATUS WdFilterPostCreate(...)
{
    if (infected) {
        FltCancelFileOpen(Instance, FileObject);
        IoStatus.Status = STATUS_VIRUS_INFECTED;
    }
}
```

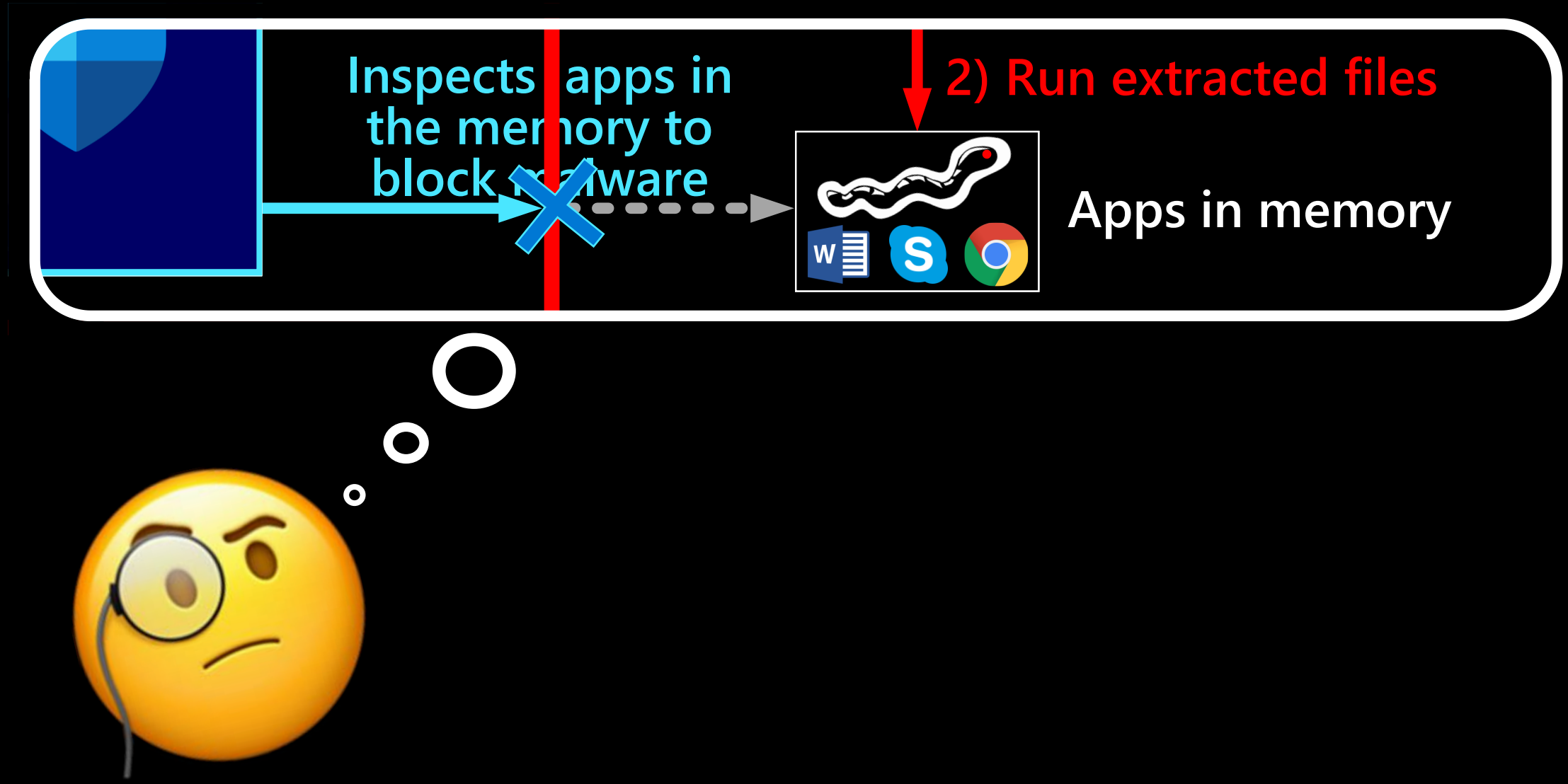
Microsoft Defender Antivirus Mini-Filter Driver (WdFilter)

- It register a mini-filter via FltRegisterFilter()
- It prevents launching a malware app via post-create callback

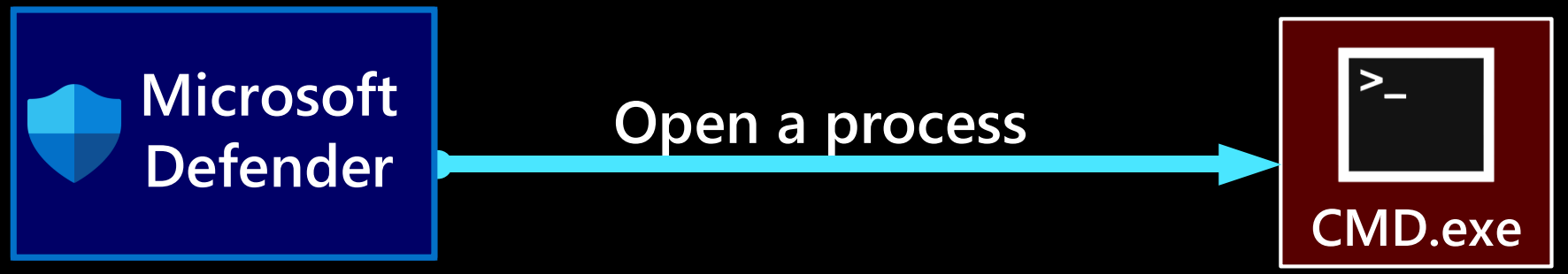
```
FLT_POSTOP_CALLBACK_STATUS WdFilterPostCreate(...)
{
    if (infected) {
        FltCancelFileOpen(Instance, FileObject);
        IoStatus.Status = STATUS_VIRUS_INFECTED;
    }
}
```

Defender is still able to access apps memory. But how?

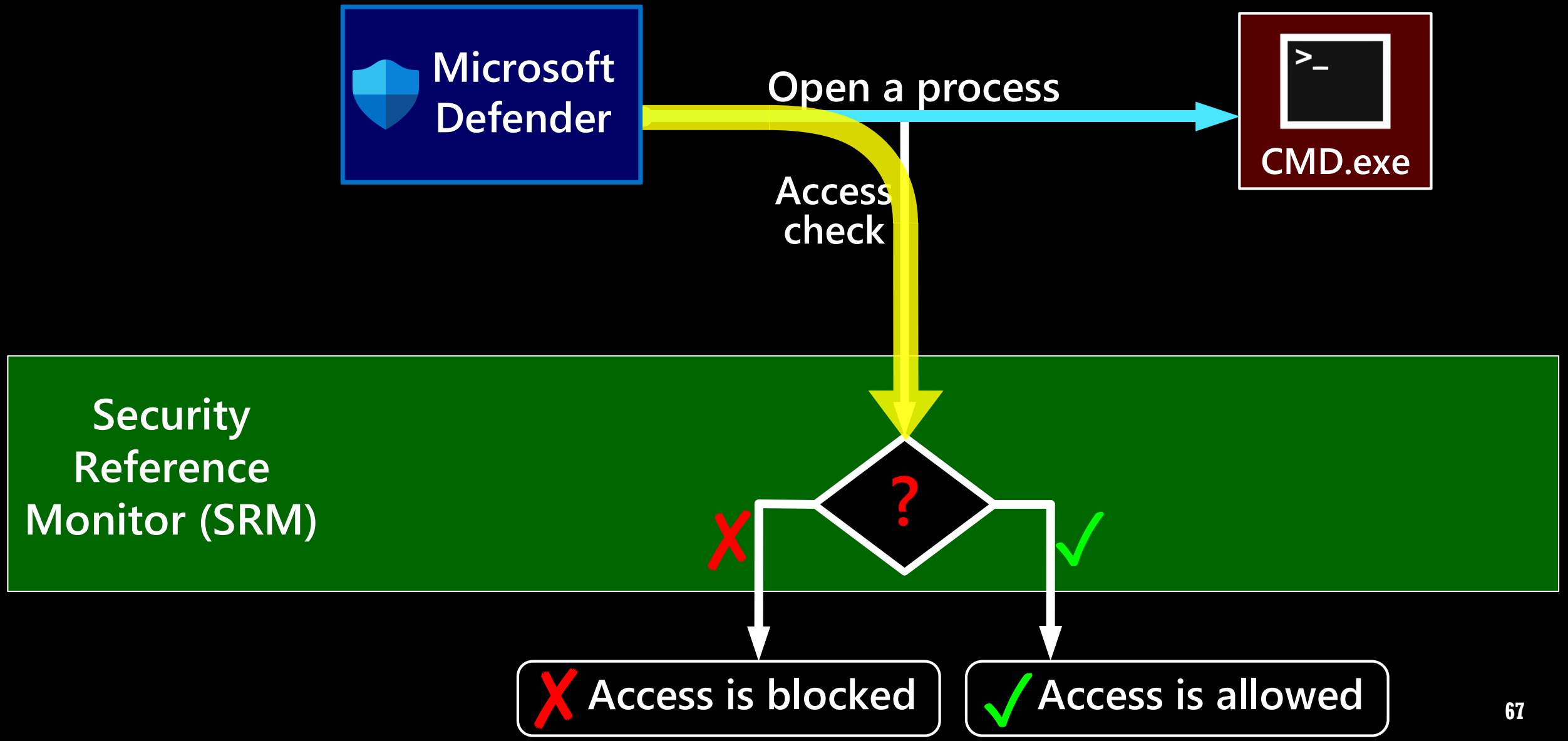
Step2: Attackers want to block apps memory inspection



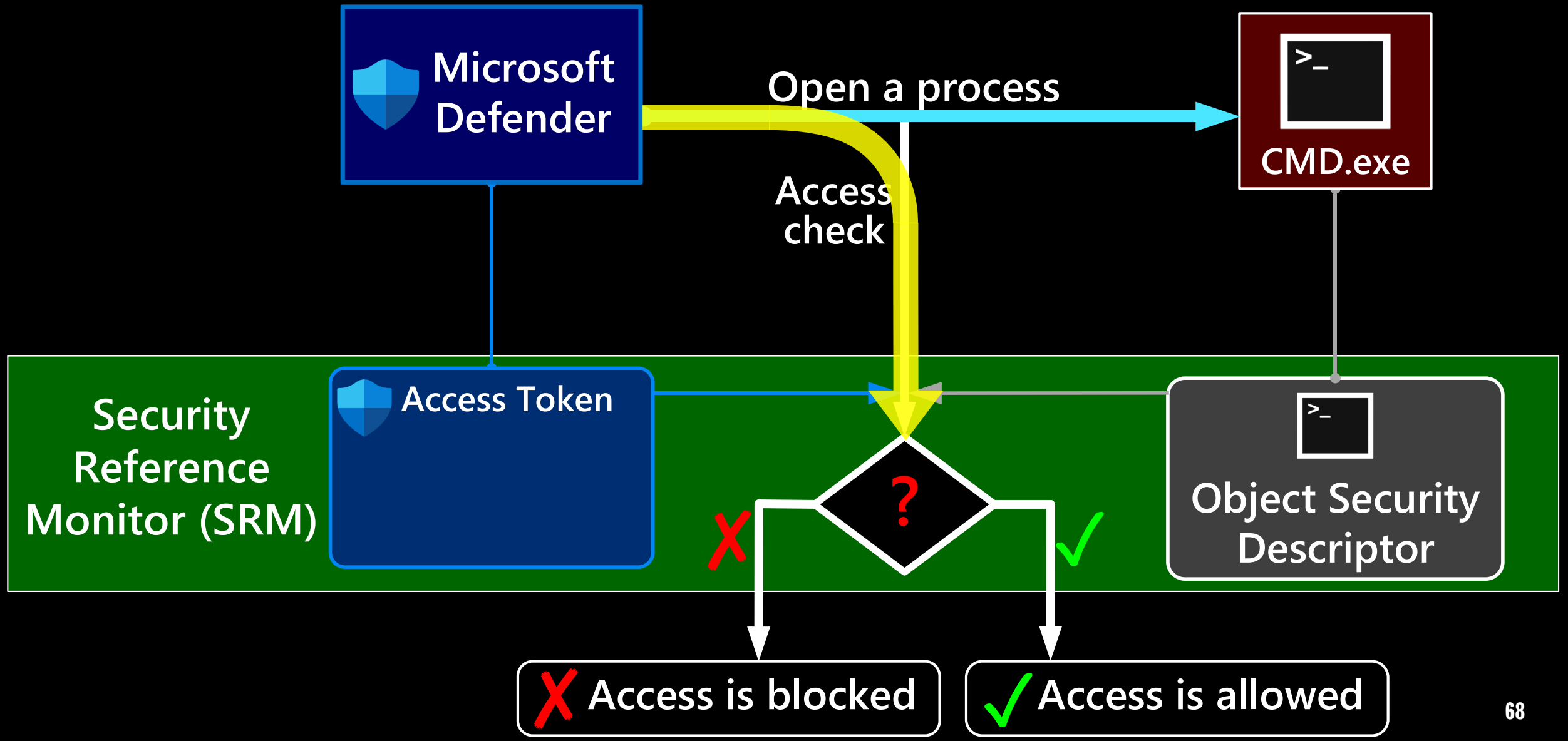
DEFENDER OPENS A PROCESS



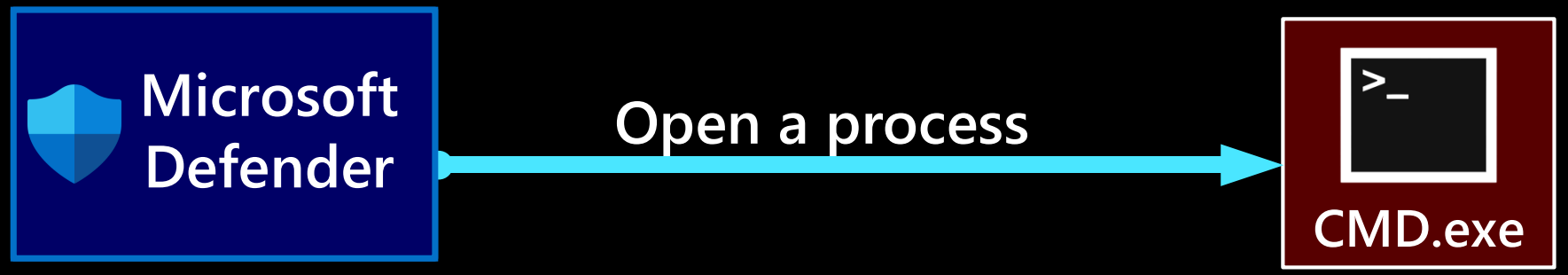
DEFENDER OPENS A PROCESS



DEFENDER OPENS A PROCESS



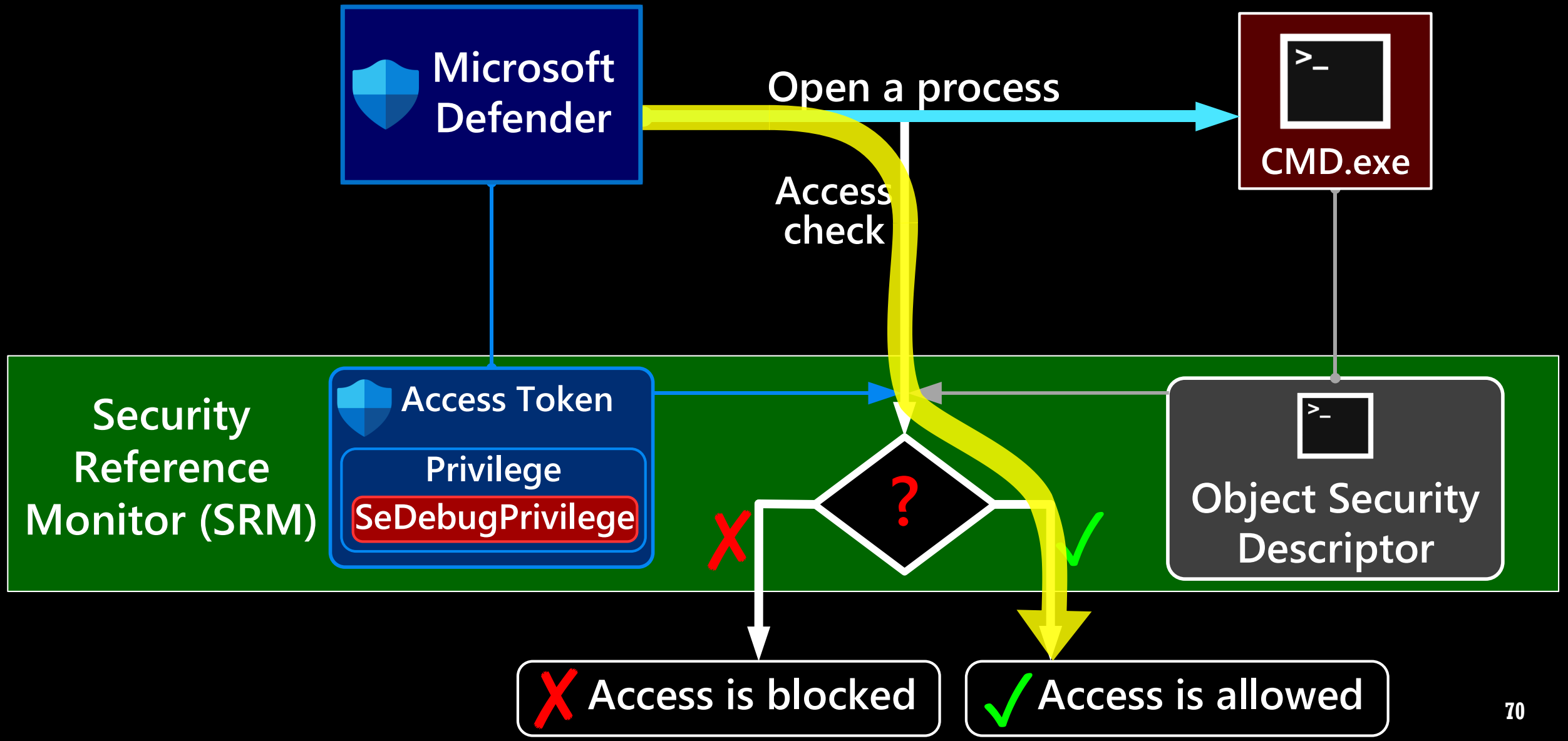
DEBUG PRIVILEGE ALLOWS TO GET FULL ACCESS TO ALL APPS



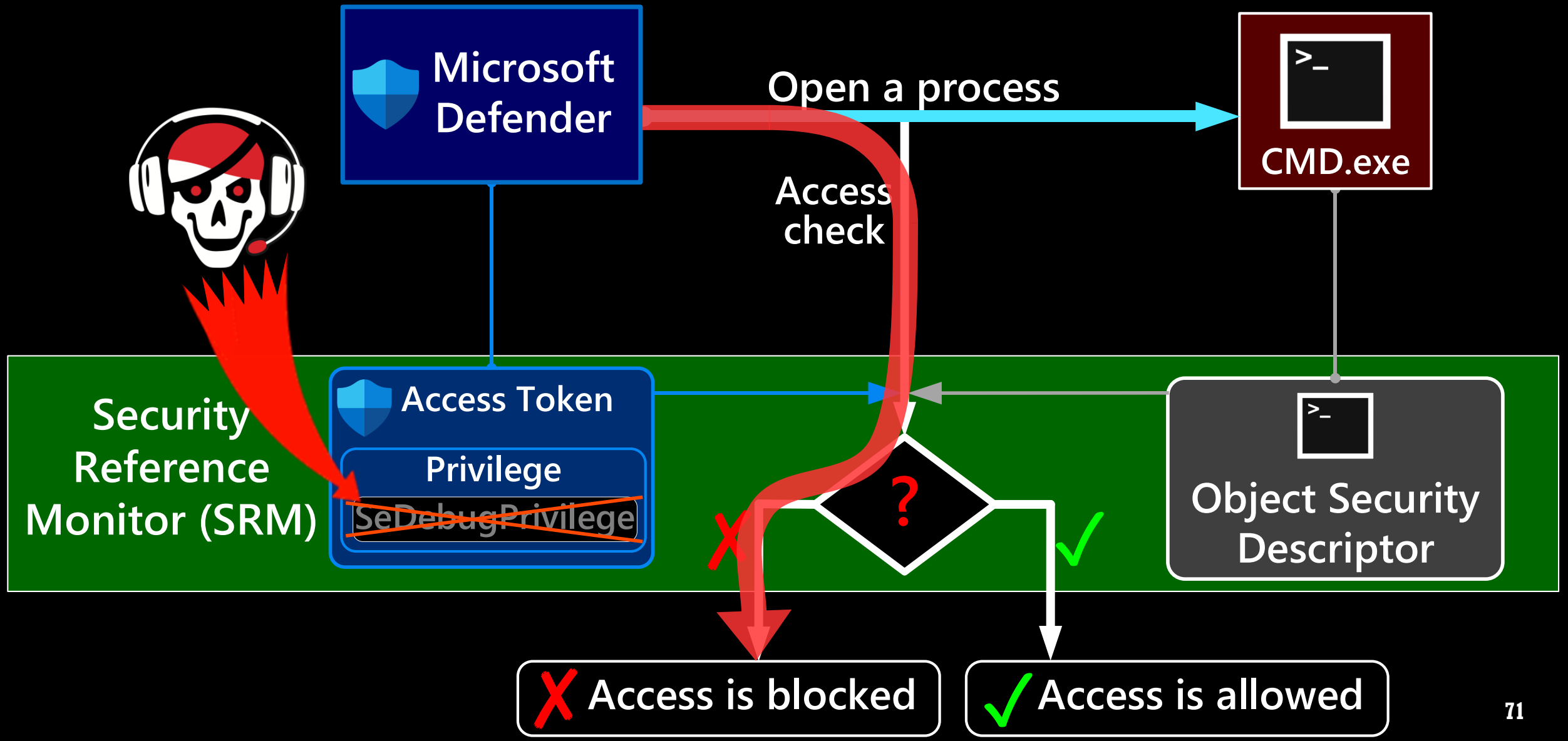
Microsoft Defender has a *SeDebugPrivilege* privilege to open apps owned by another account

Name	User name
MsMpEng.exe	NT AUTHORITY\SYSTEM
cmd.exe	DESKTOP-2FNCGCH\igork

DEFENDER OPENS A PROCESS



PATCHING PRIVILEGES CAN BLOCK OPENING AN APP



TOKEN PRIVILEGES

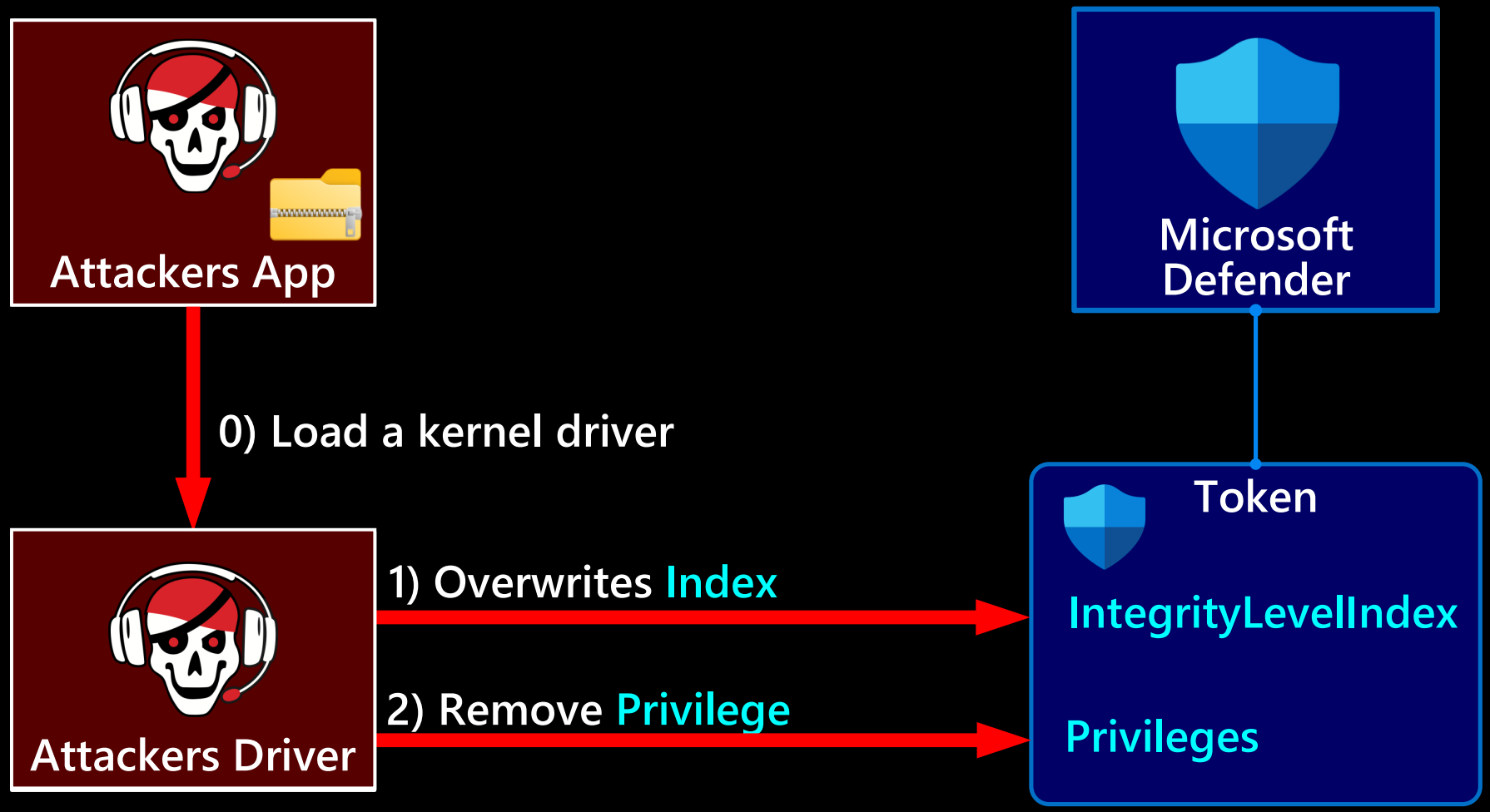
```
typedef struct _SEP_TOKEN_PRIVILEGES
{
    UINT64 Present;
    UINT64 Enabled;
    UINT64 EnabledByDefault;
} SEP_TOKEN_PRIVILEGES, *PSEP_TOKEN_PRIVILEGES;
```

Experiments:

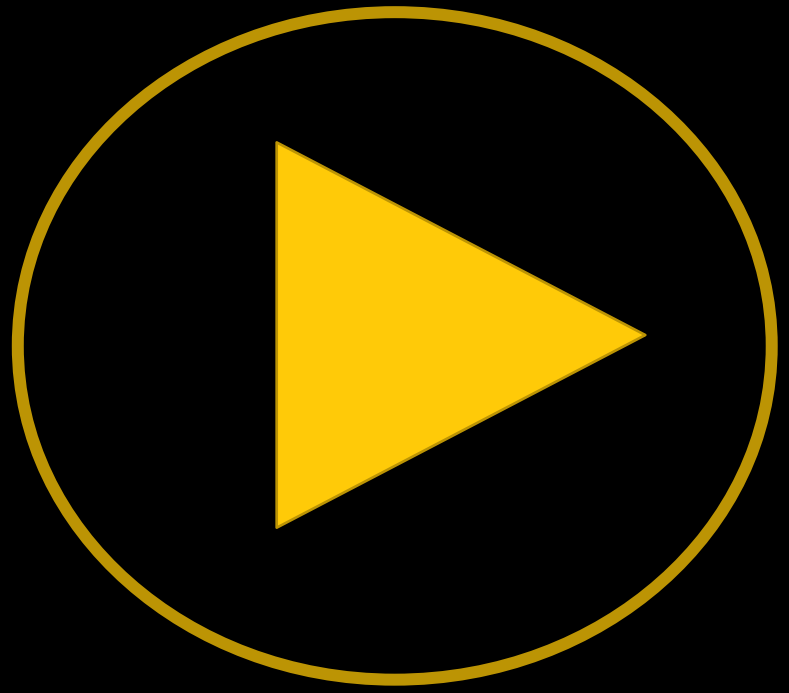
revoking *SeDebugPrivilege*-bit from the field named *Enabled* prevents Microsoft Defender from inspecting apps memory.

Attackers can disable Microsoft Defender completely!!

Attackers patch two values: Integrity Level Index and Privilege

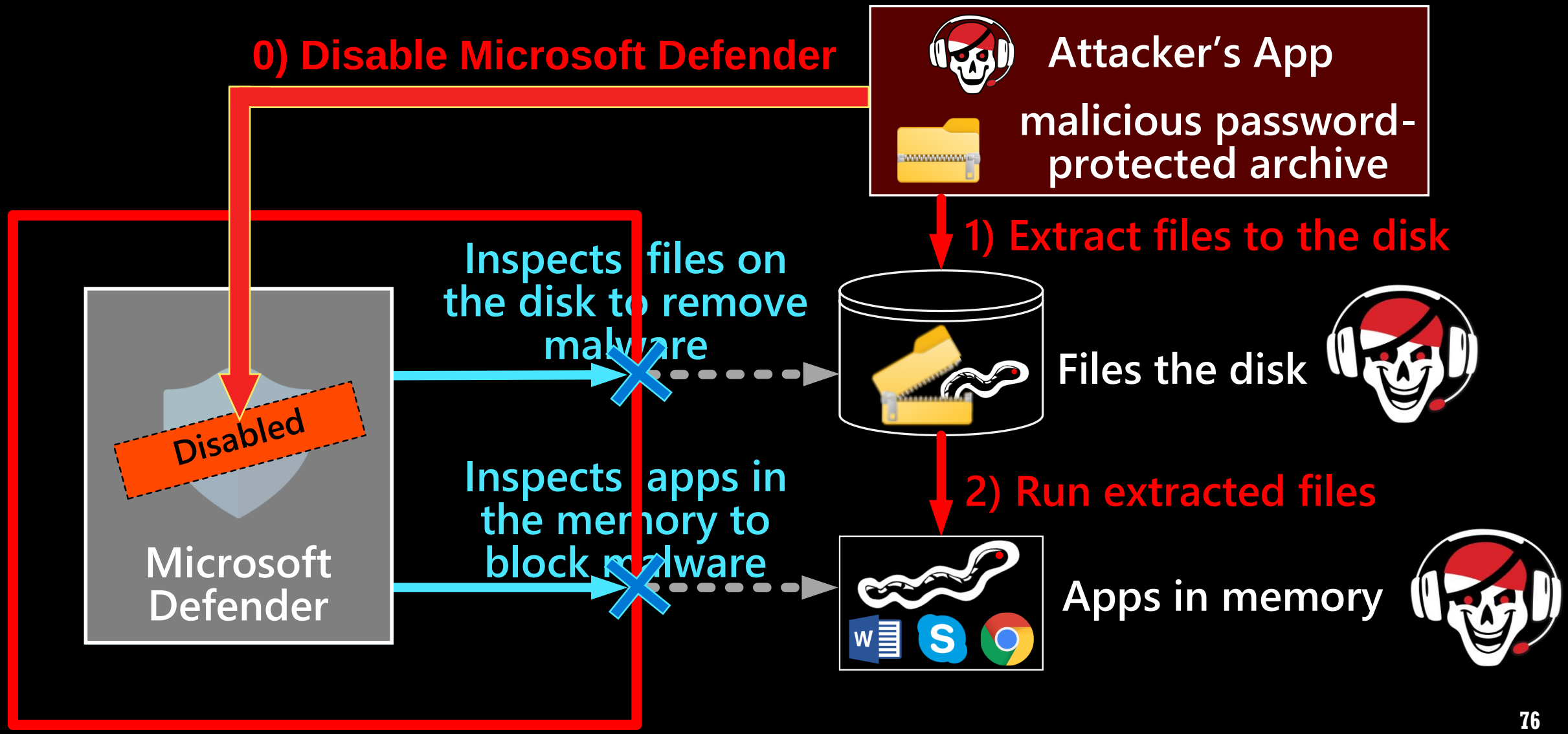


DEMO: ATTACK ON MIC + TOKEN PRIVILEGE

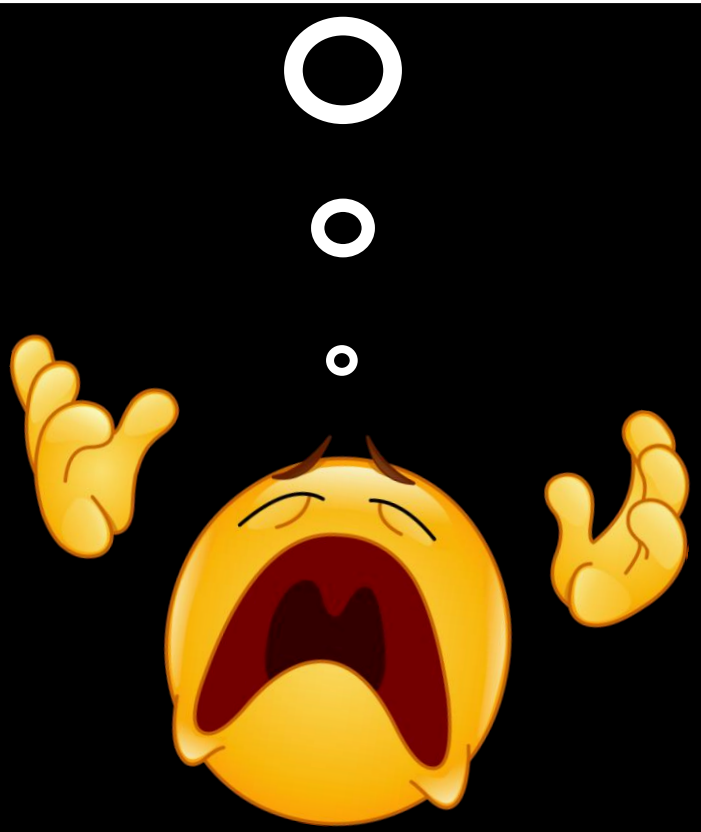


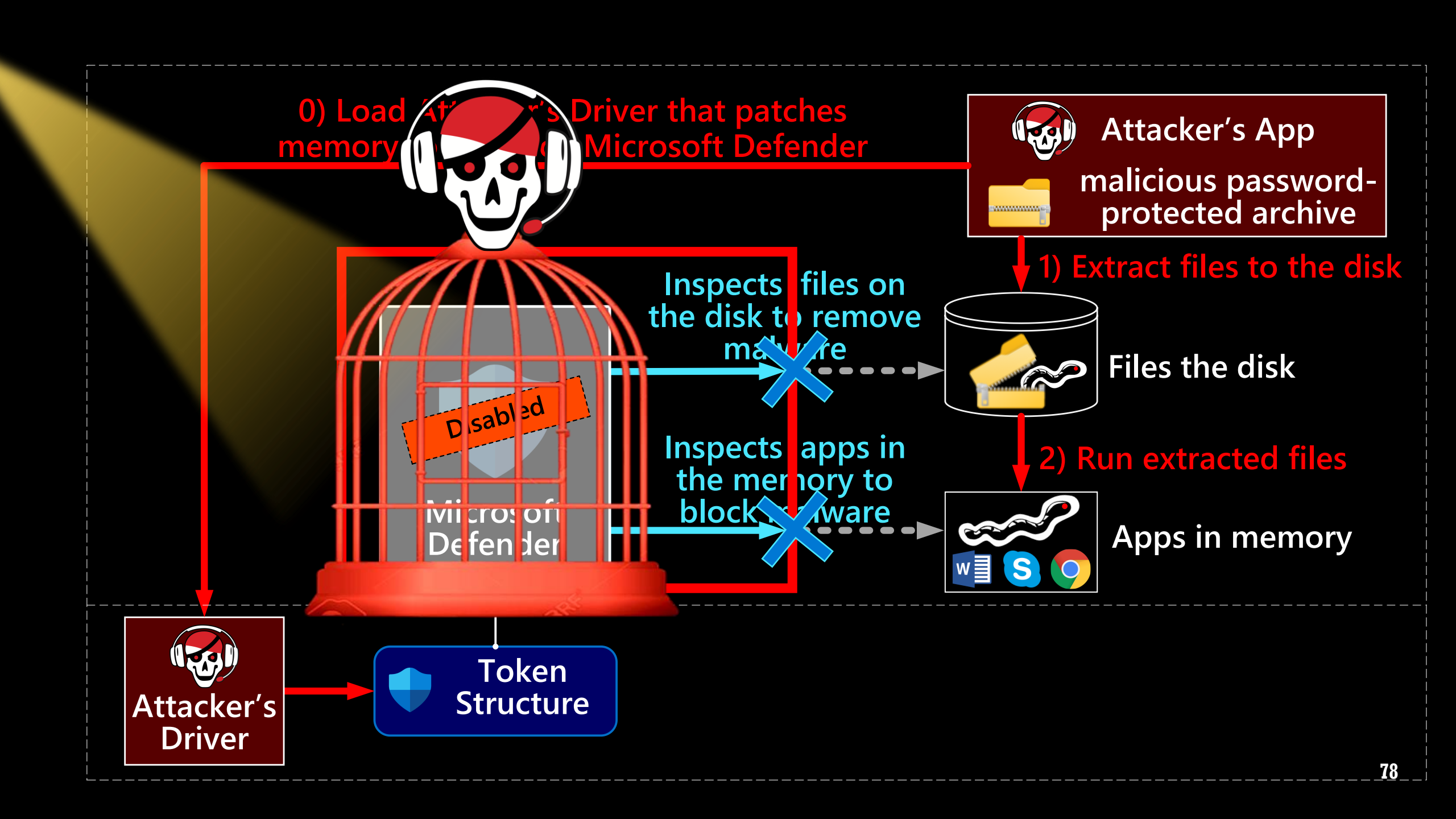
The online version is here –
https://www.youtube.com/embed/ihhUUd9qJTY?vq=hd1440

MICROSOFT DEFENDER IS COMPLETELY DISABLED

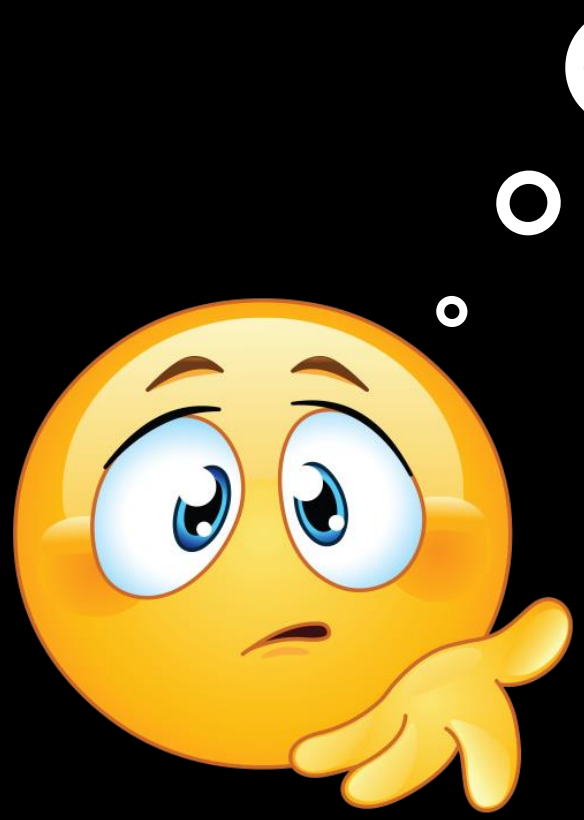


Kernel-mode attack
has disabled Microsoft Defender
without triggering any security reaction




0) Load Attacker's Driver that patches memory of Microsoft Defender
Attacker's App
malicious password-protected archive
1) Extract files to the disk
Inspects files on the disk to remove malware
Files the disk
Disabled
Microsoft Defender
Inspects apps in the memory to block malware
2) Run extracted files
Apps in memory
Attacker's Driver
Token Structure

The attack has blinded Microsoft Defender.
What about other AVs?

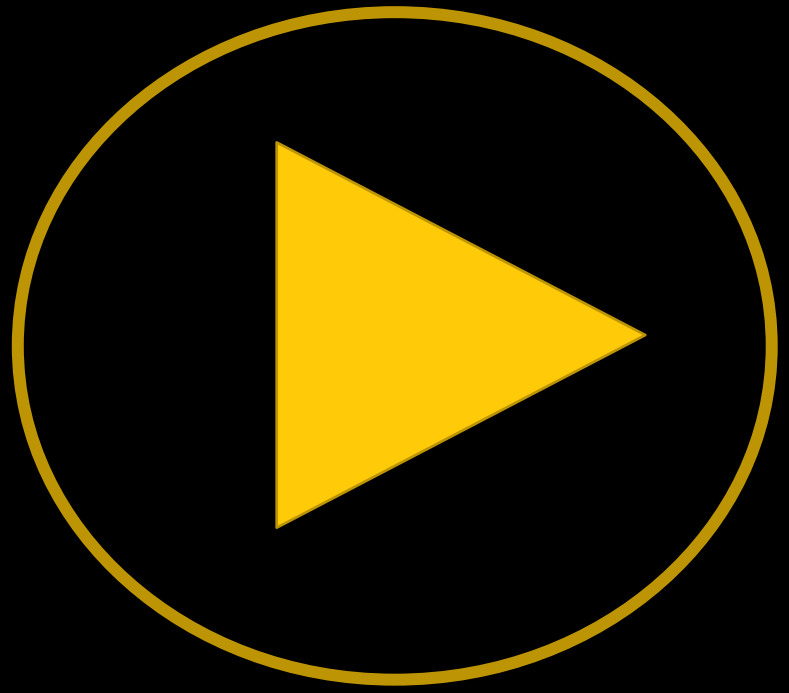


KERNEL-MODE SANDBOXING ATTACK BLINDS TOP AVs

AV Name	AV ability to detect malicious files	AV ability to detect malicious processes
Microsoft Defender 360 TOTAL SECURITY McAfee AVG avast WEBROOT Malwarebytes	Disabled	
kaspersky	Disabled	Enabled
TREND MICRO	Enabled, but AV cannot remove malware files	Disabled

Disclaimer: The purpose is to provide technical review only. This analysis is not designed to promote any solutions. We do respect all antiviruses and endpoint security solutions.

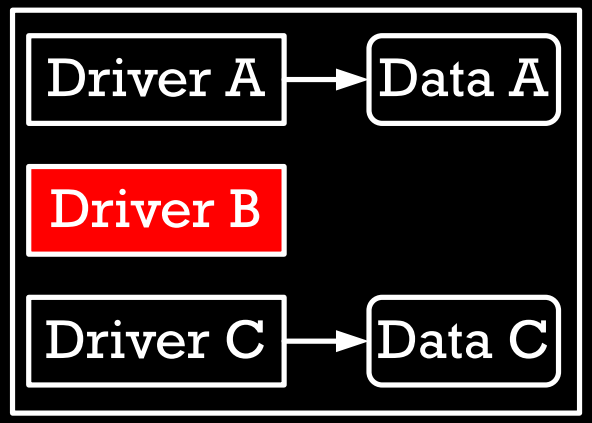
Disabling AVG AntiVirus from Denis



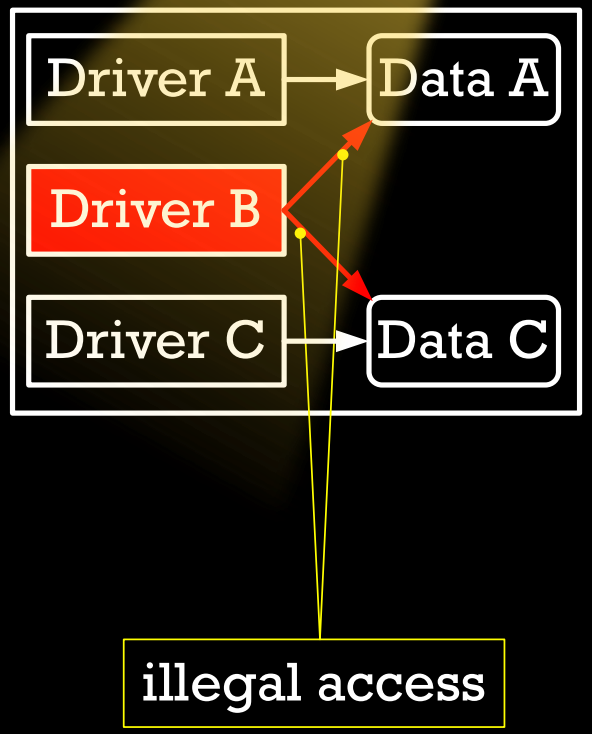
Disclaimer: The purpose is to provide technical review only. This analysis is not designed to promote any solutions.
We do respect all antiviruses and endpoint security solutions.

MemoryRanger
Defends
Microsoft Defender

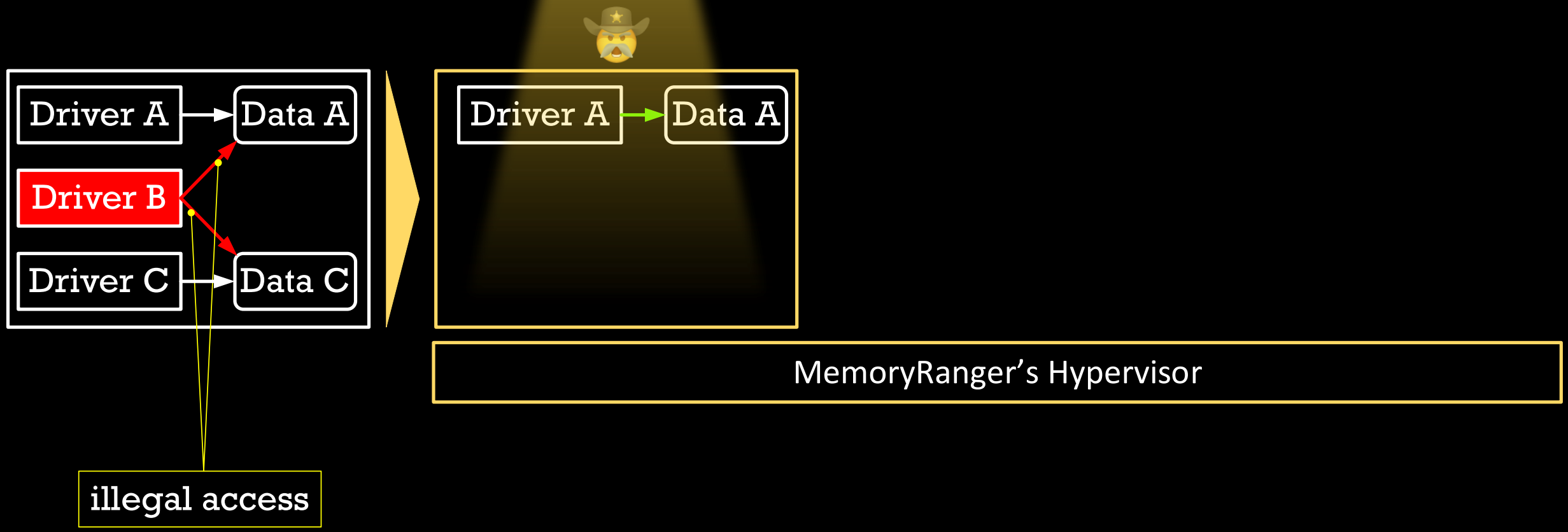
MemoryRanger: Intro



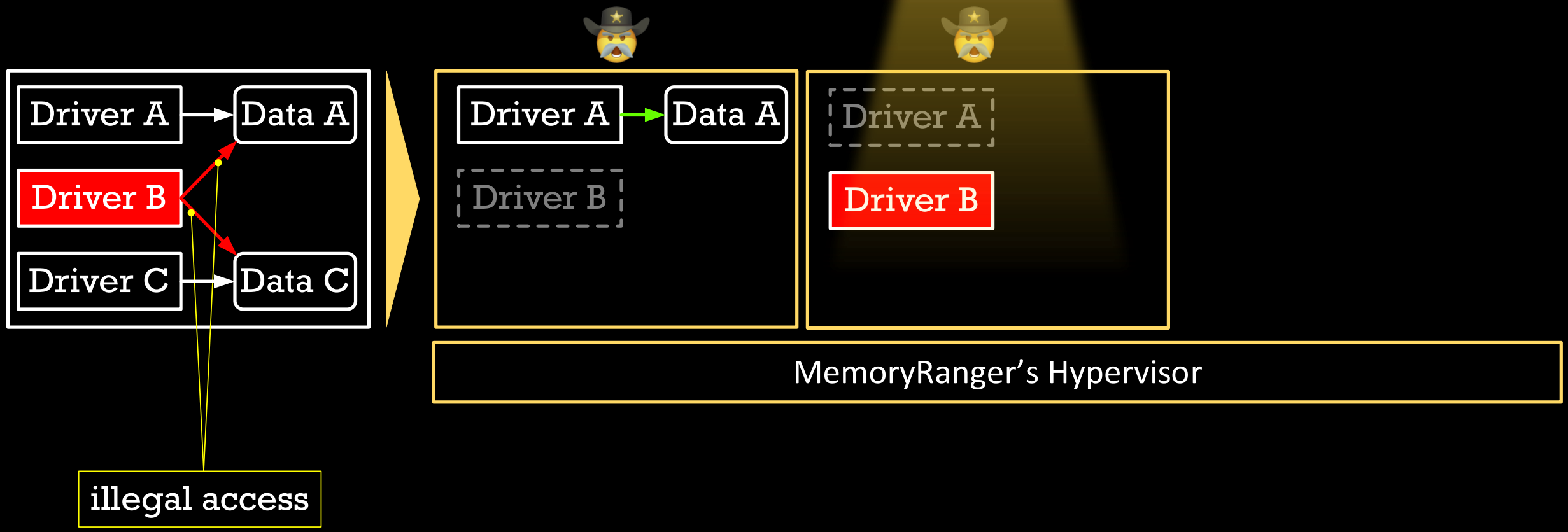
MemoryRanger: Intro



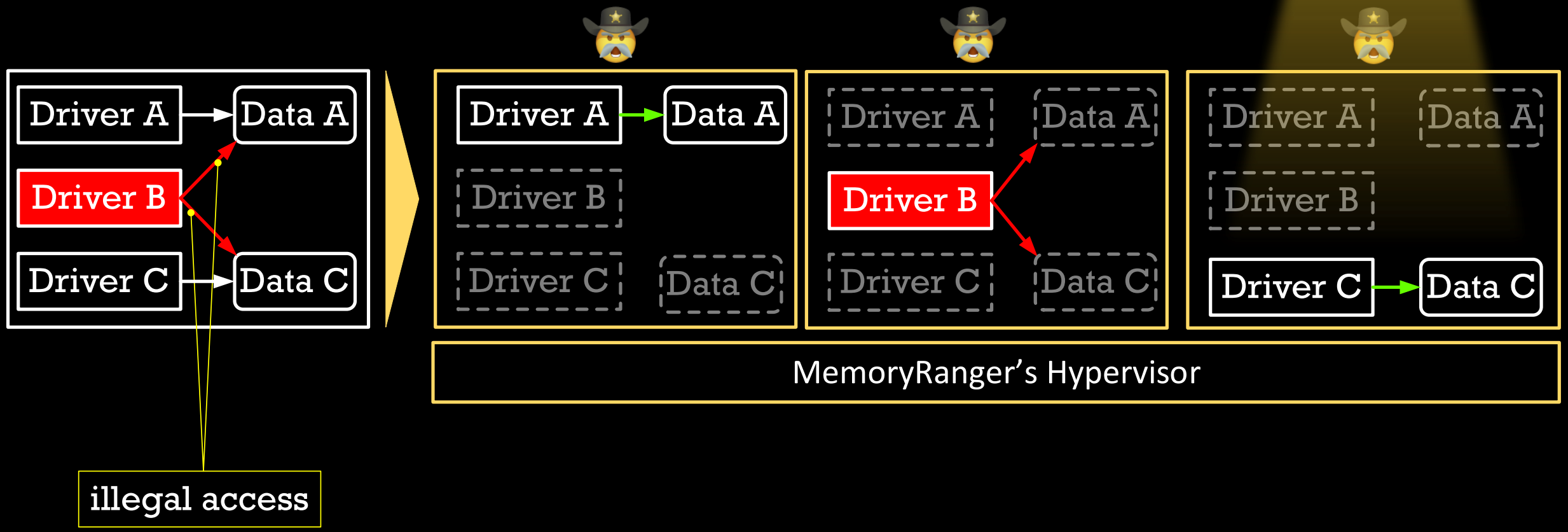
MemoryRanger: Intro



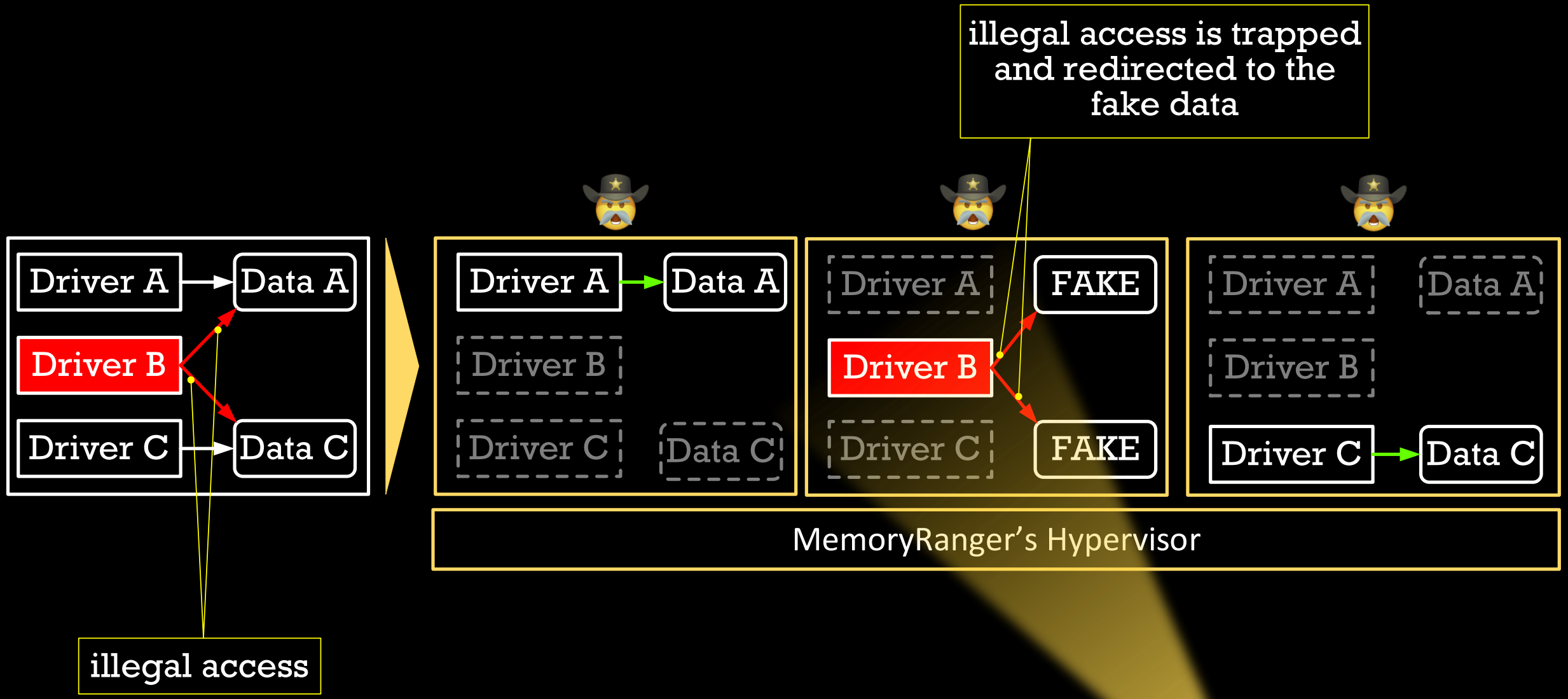
MemoryRanger: Intro



MemoryRanger: Intro



MemoryRanger: Intro



MemoryRanger: Features

- Components:
  - user-mode control app
  - kernel-mode driver to register OS callbacks
  - hypervisor based dispatcher based on Intel VT-x and EPT technologies

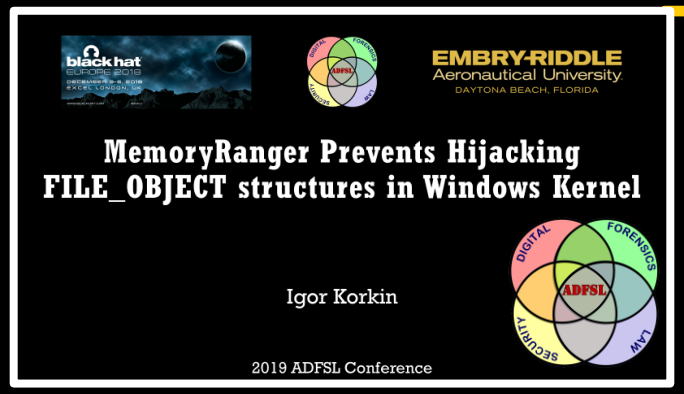
- The key features:
  - Runs kernel-mode drivers into isolated memory enclaves
  - Allows different memory access configuration for each memory enclave
  - Number of enclaves can be increased in runtime (while VBS has fixed 2 enclaves only)

- Technical features:
  - Hooks kernel API routines
  - Redirects illegal access to the sensitive data to the fake content
  - Supports newest Windows 11 x64 and it is open-source

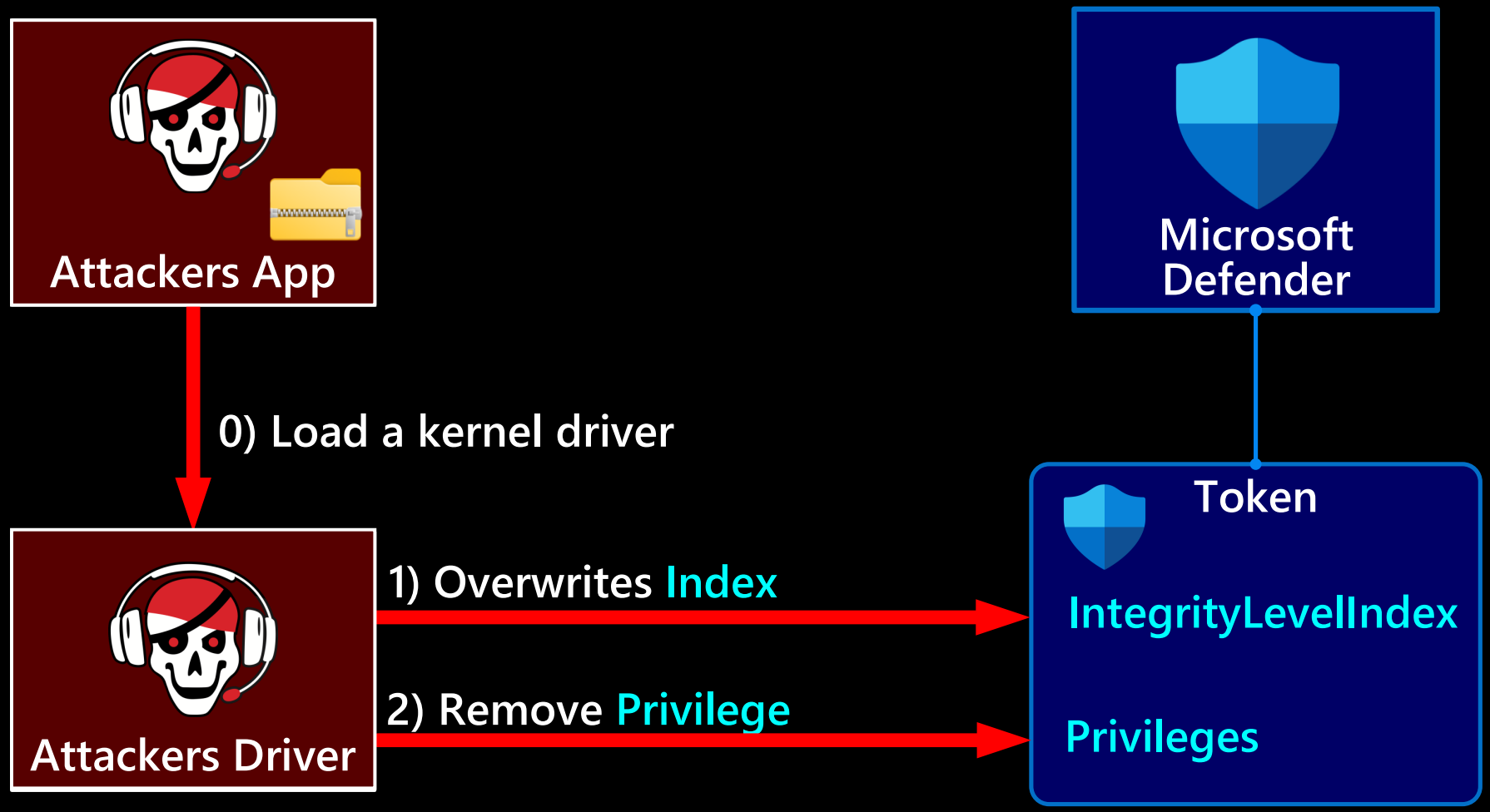
github.com/IgorKorkin/MemoryRanger

MemoryRanger was in US, UK, and Asia and triple at BlackHat

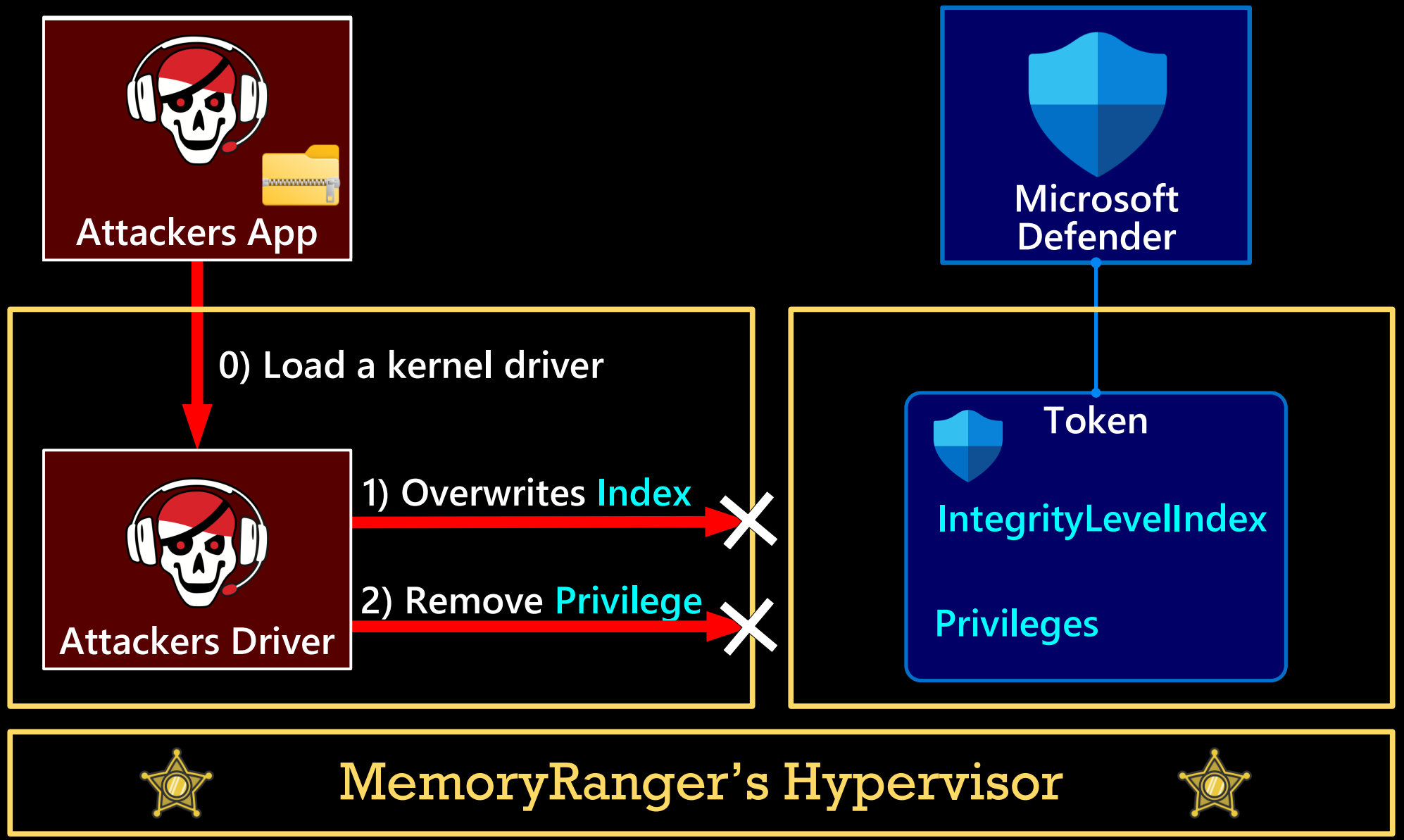
2018 2019 2020 2021 2022



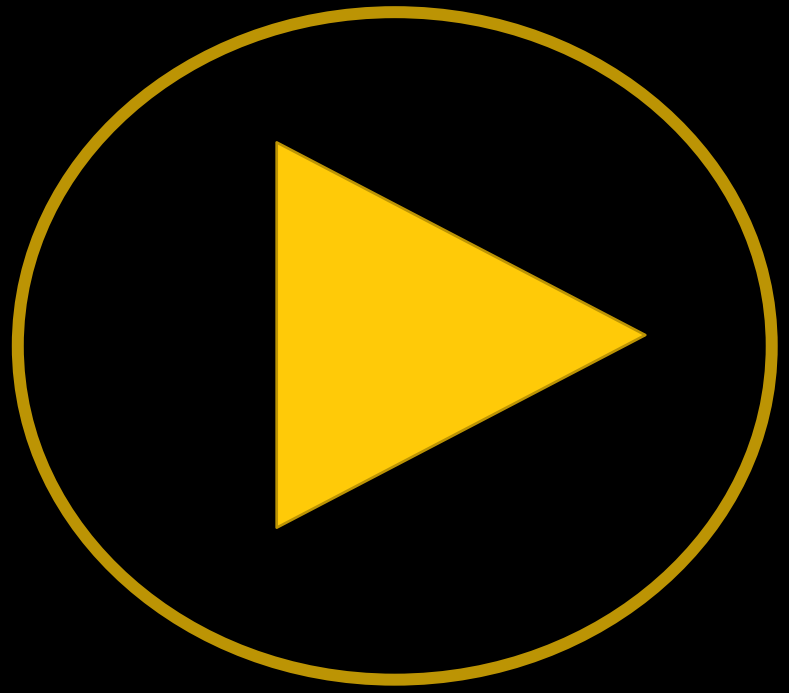
MemoryRanger Customization protects Microsoft Defender



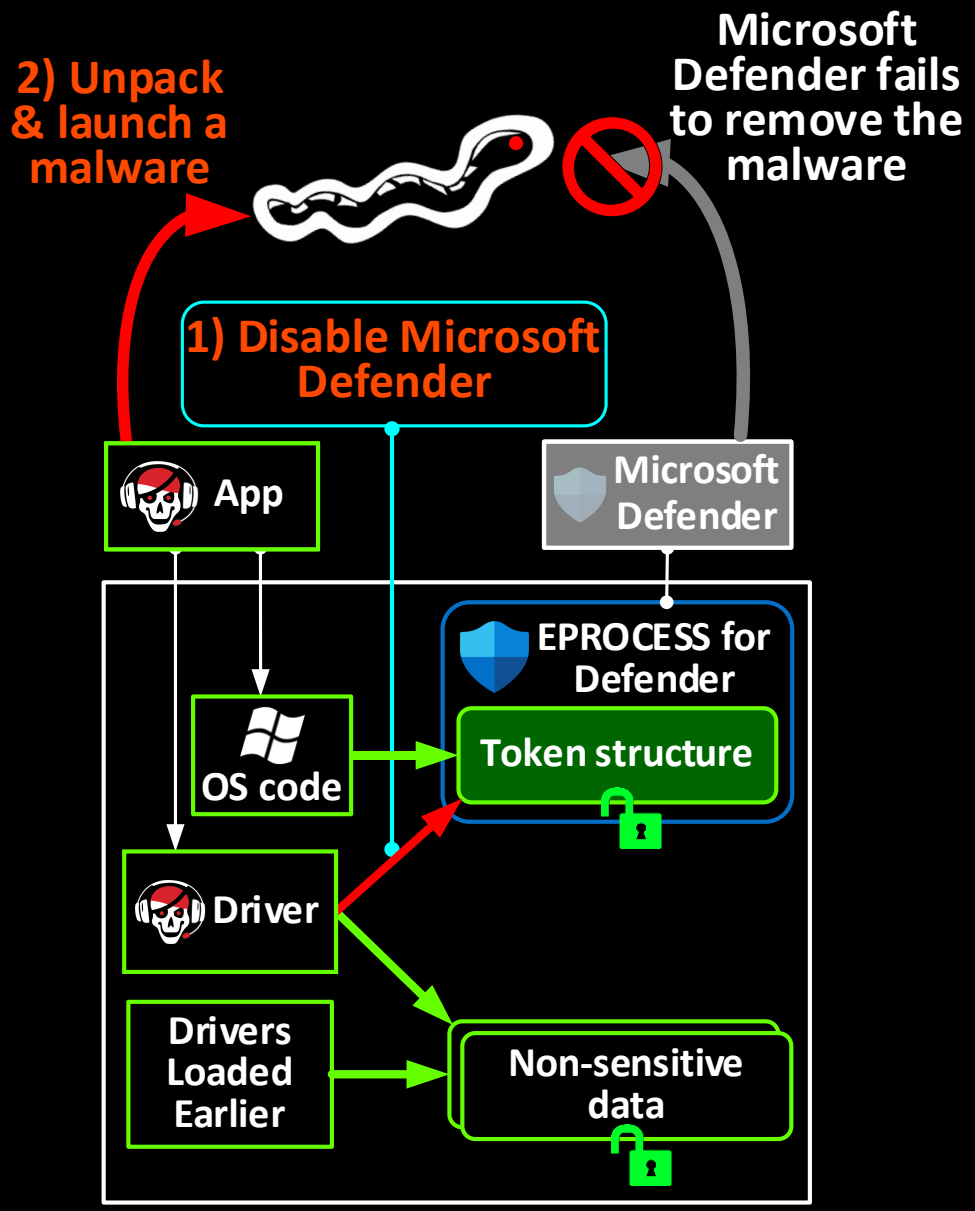
MemoryRanger Customization protects Microsoft Defender

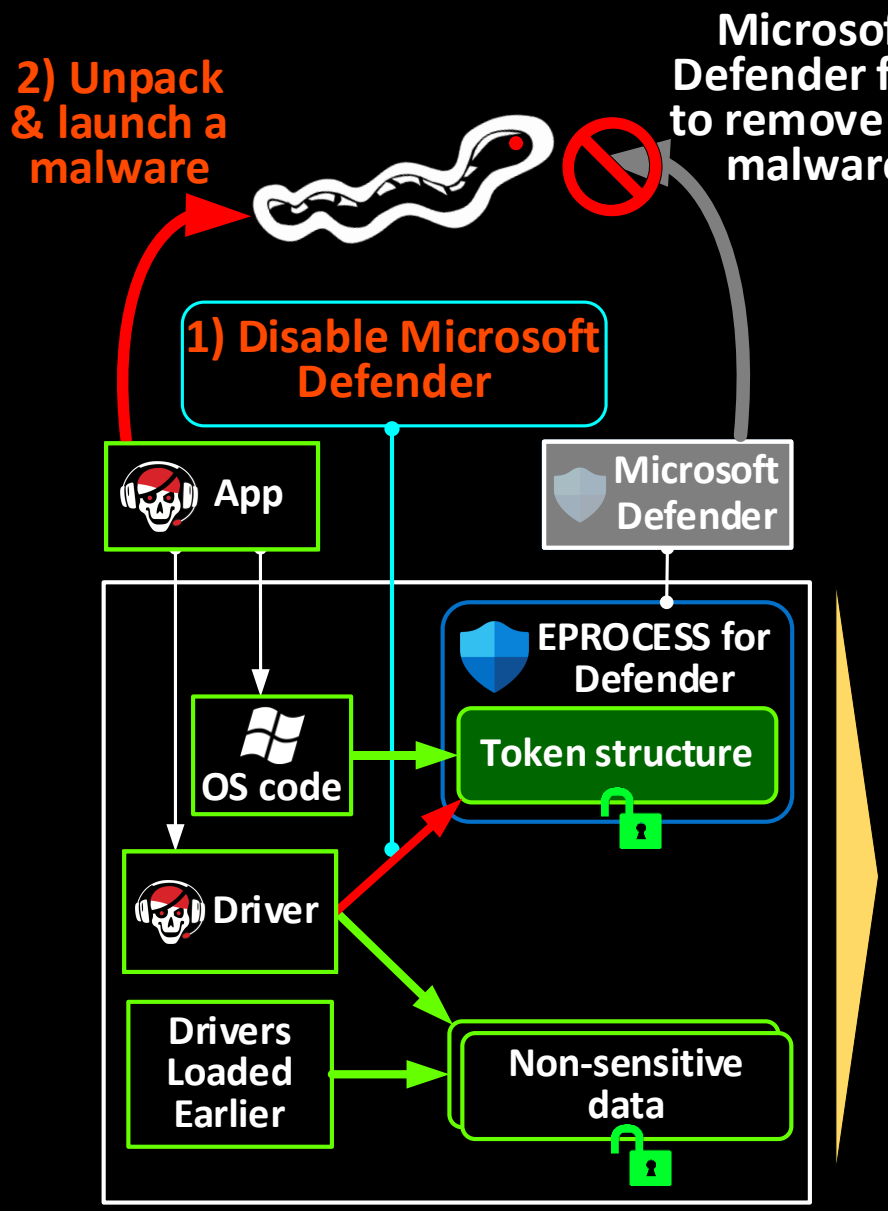


Demo: MemoryRanger Defends Microsoft Defender

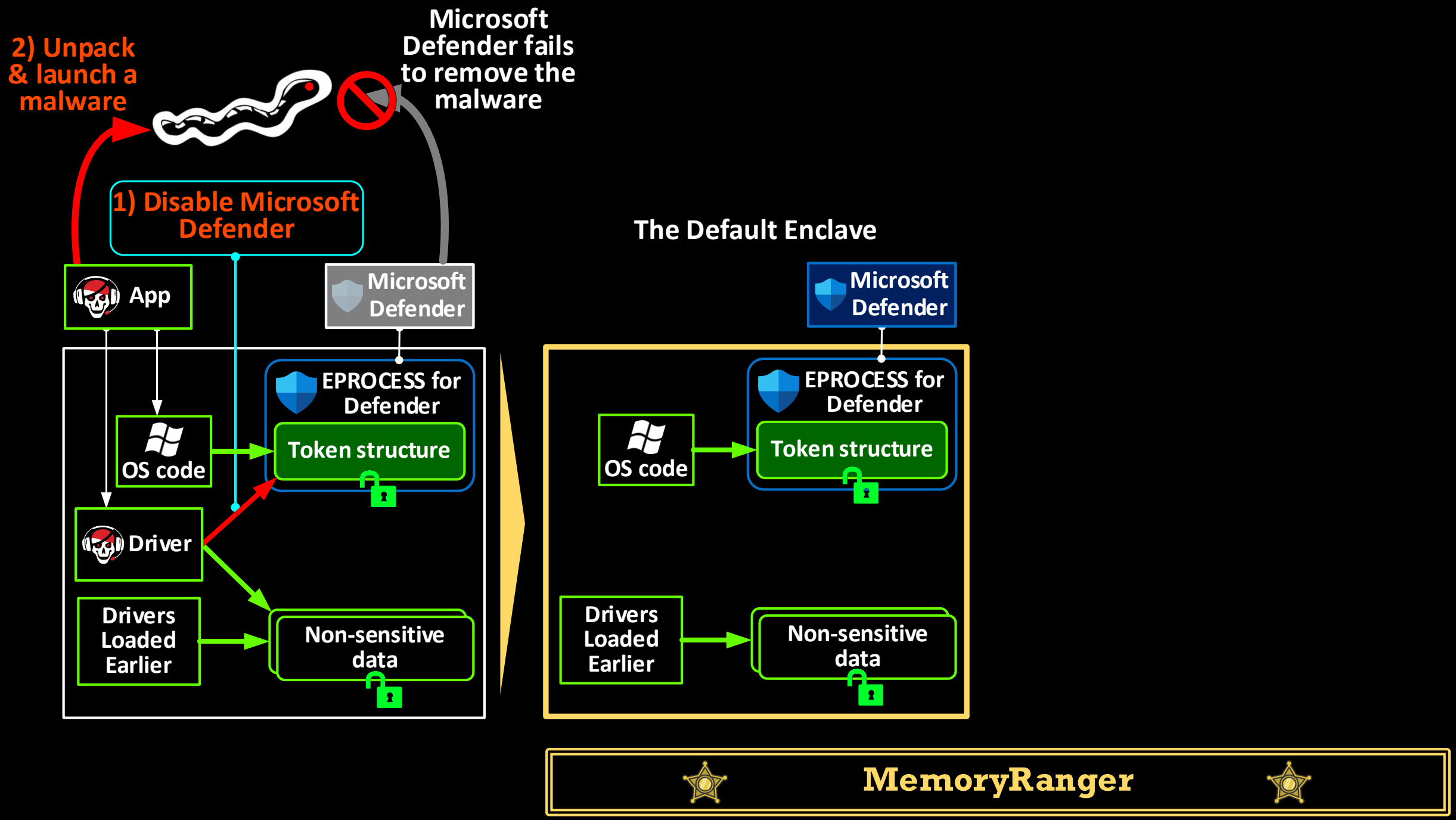


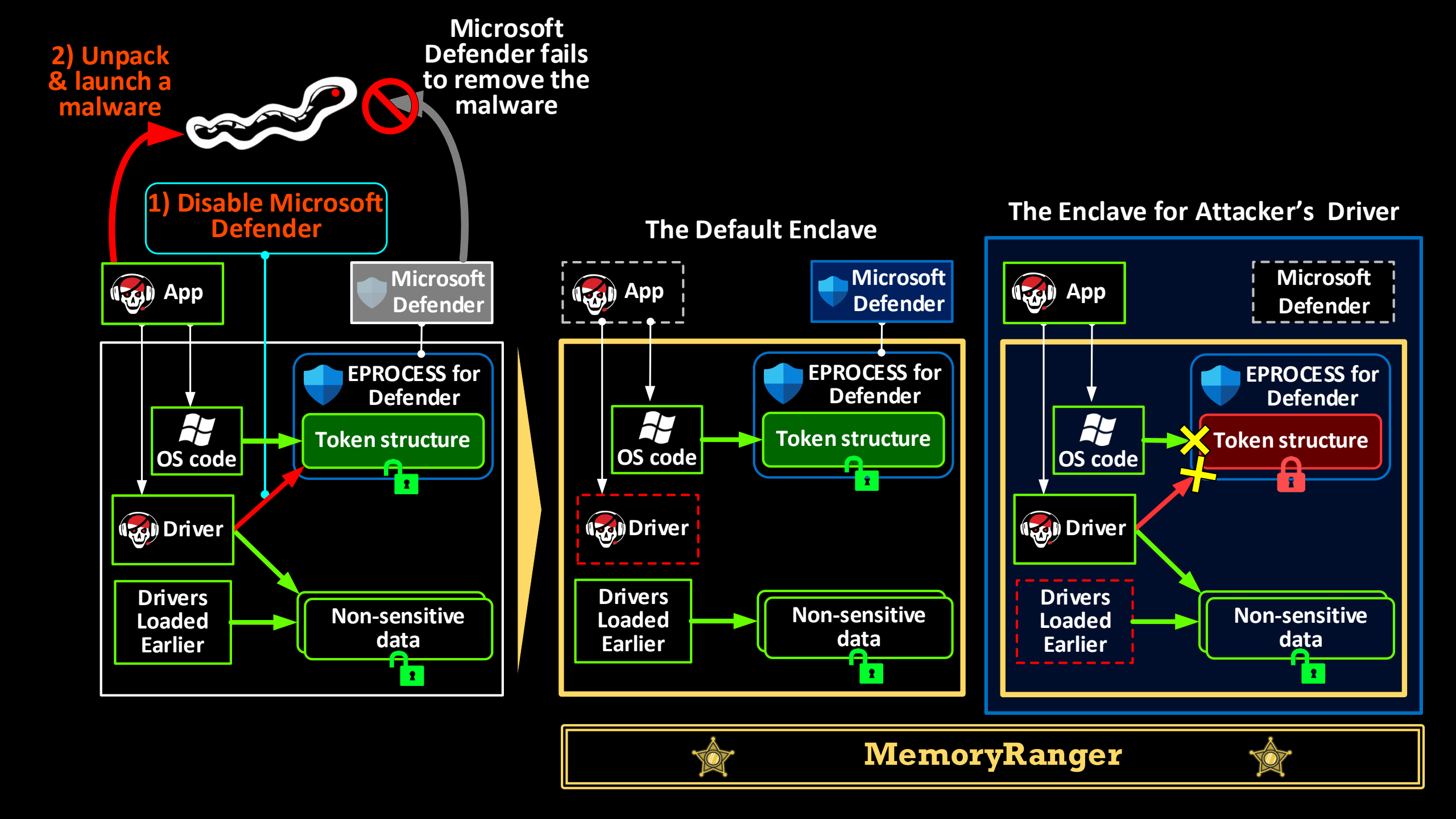
The online version is here –
https://www.youtube.com/embed/Ohqhq50wVjI?vq=hd1440

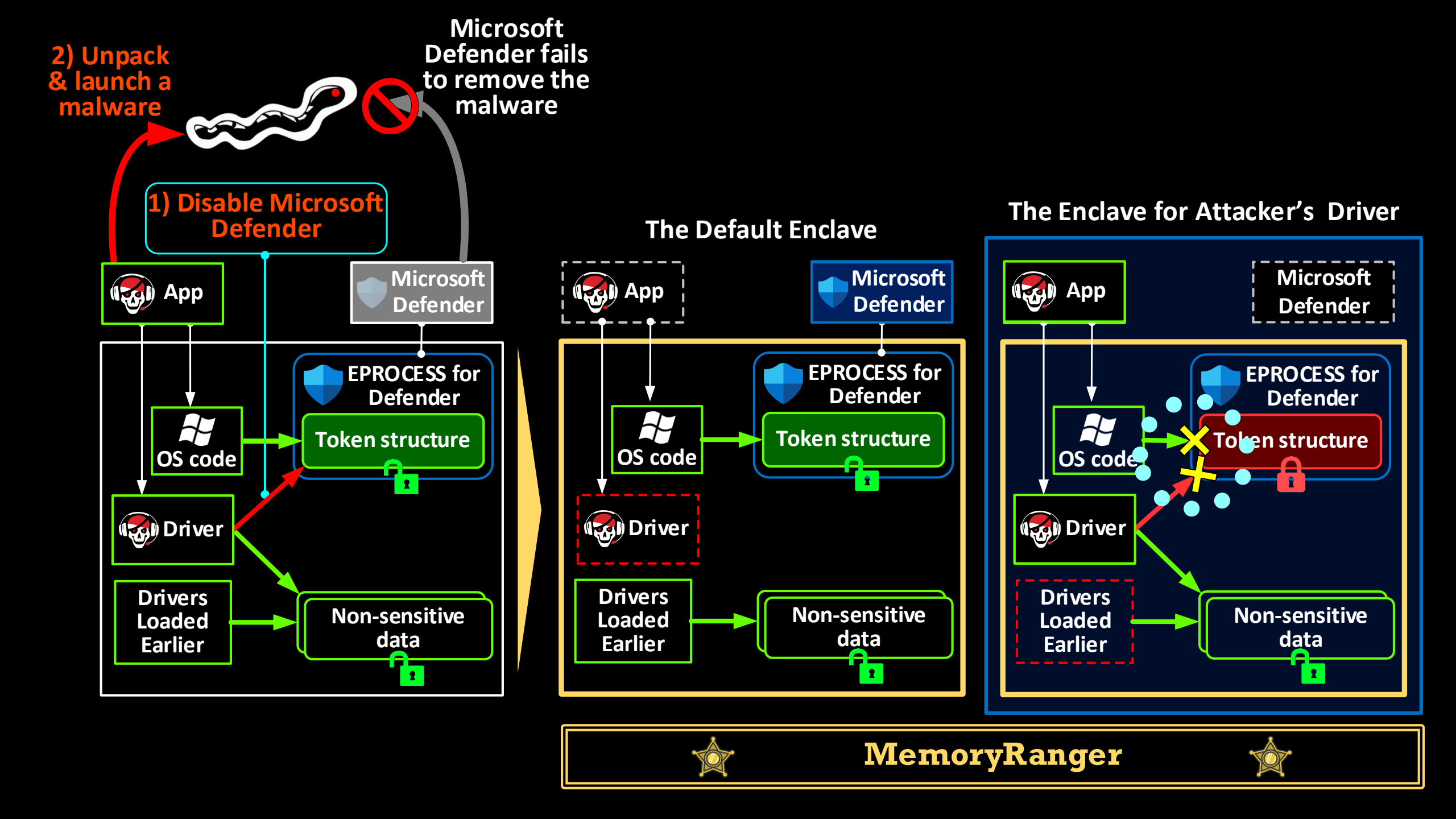

2) Unpack & launch a malware
Microsoft Defender fails to remove the malware
1) Disable Microsoft Defender
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data

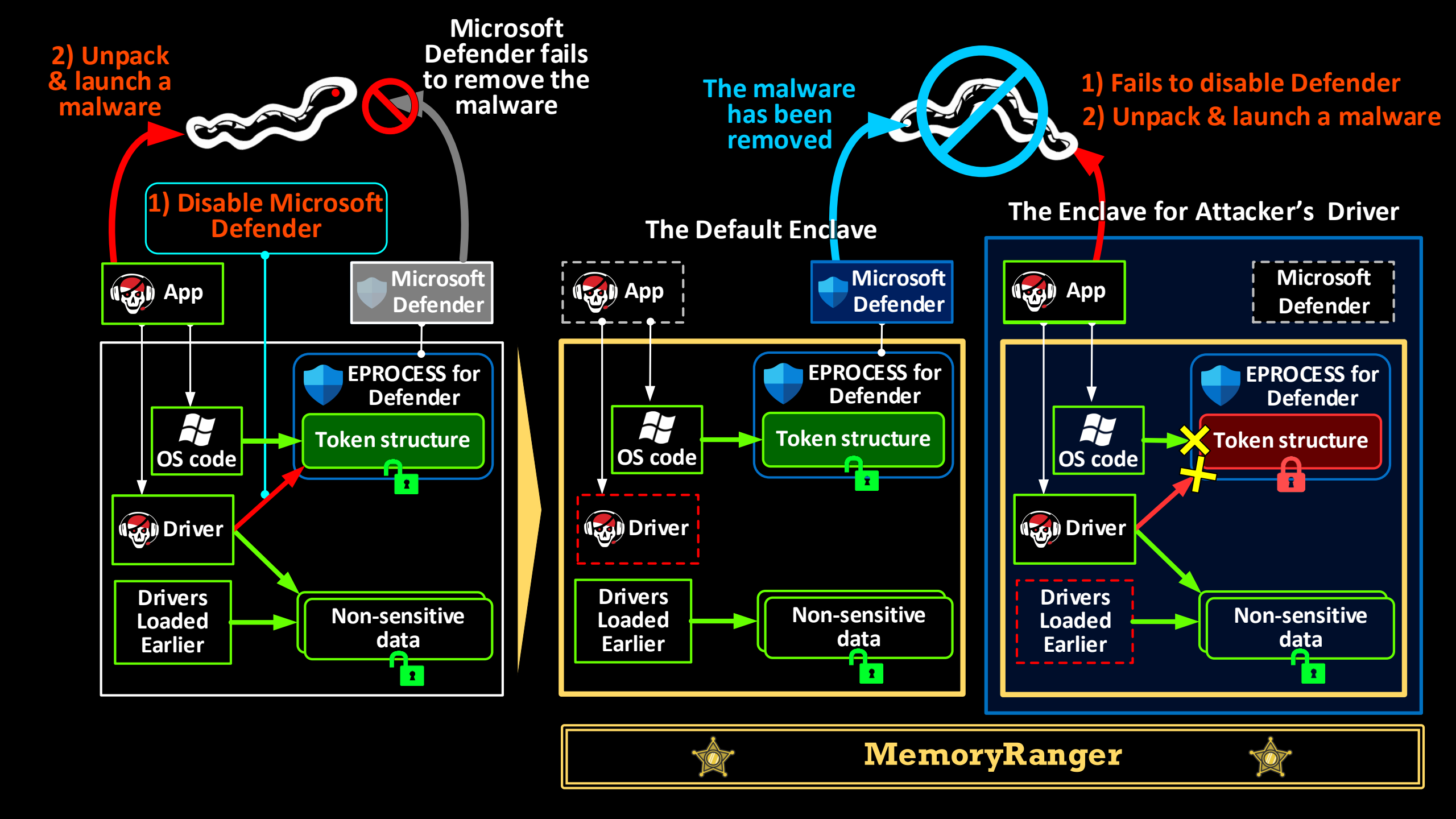


MemoryRanger


2) Unpack & launch a malware
Microsoft Defender fails to remove the malware
1) Disable Microsoft Defender
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Default Enclave
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Drivers Loaded Earlier
Non-sensitive data
MemoryRanger


2) Unpack & launch a malware
Microsoft Defender fails to remove the malware
1) Disable Microsoft Defender
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Default Enclave
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Enclave for Attacker's Driver
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
MemoryRanger

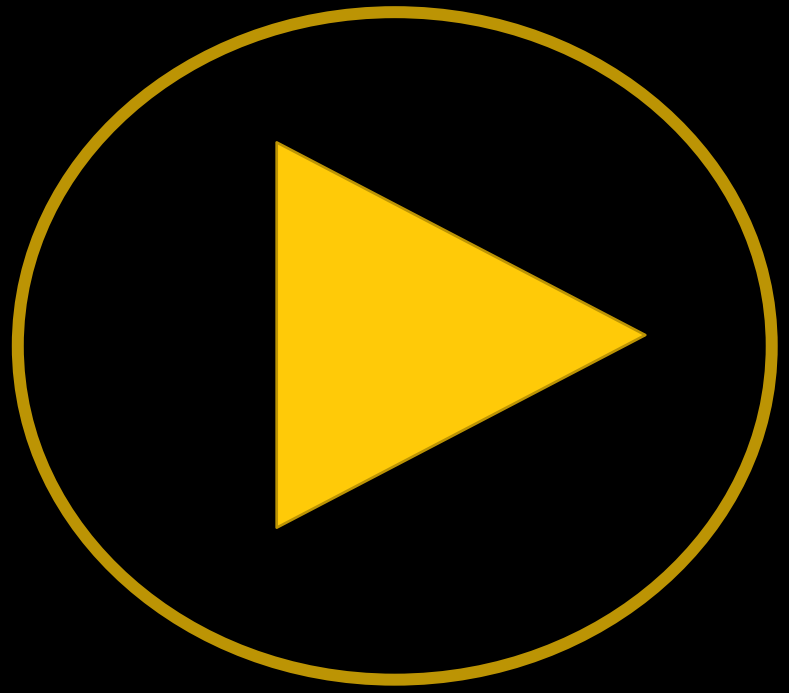

2) Unpack & launch a malware
Microsoft Defender fails to remove the malware
1) Disable Microsoft Defender
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Default Enclave
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Enclave for Attacker's Driver
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
MemoryRanger


2) Unpack & launch a malware
Microsoft Defender fails to remove the malware
1) Disable Microsoft Defender
App
Microsoft Defender
EPROCESS for Defender
Token structure
OS code
Driver
Drivers Loaded Earlier
Non-sensitive data
The Default Enclave
The malware has been removed
1) Fails to disable Defender
2) Unpack & launch a malware
The Enclave for Attacker's Driver
MemoryRanger

MemoryRanger requires the hardware virtualization support.

How to protect PCs without using hypervisor?

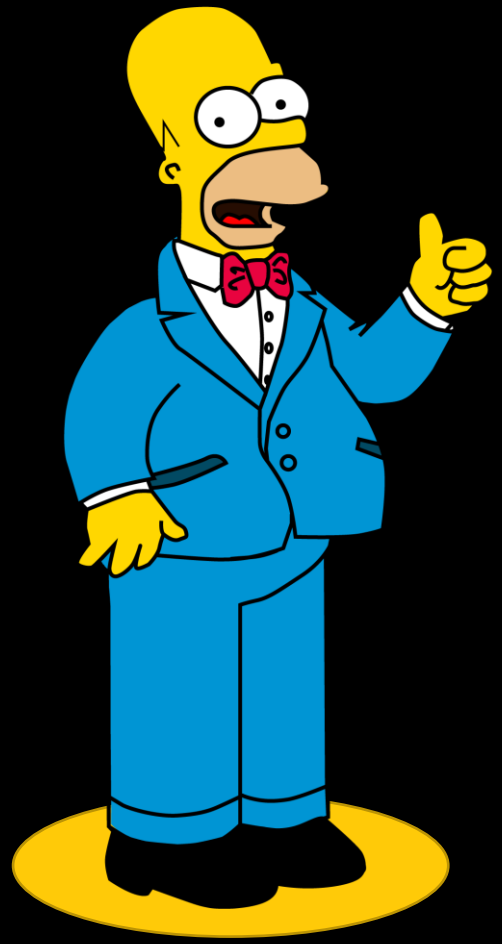
The Part of Denis



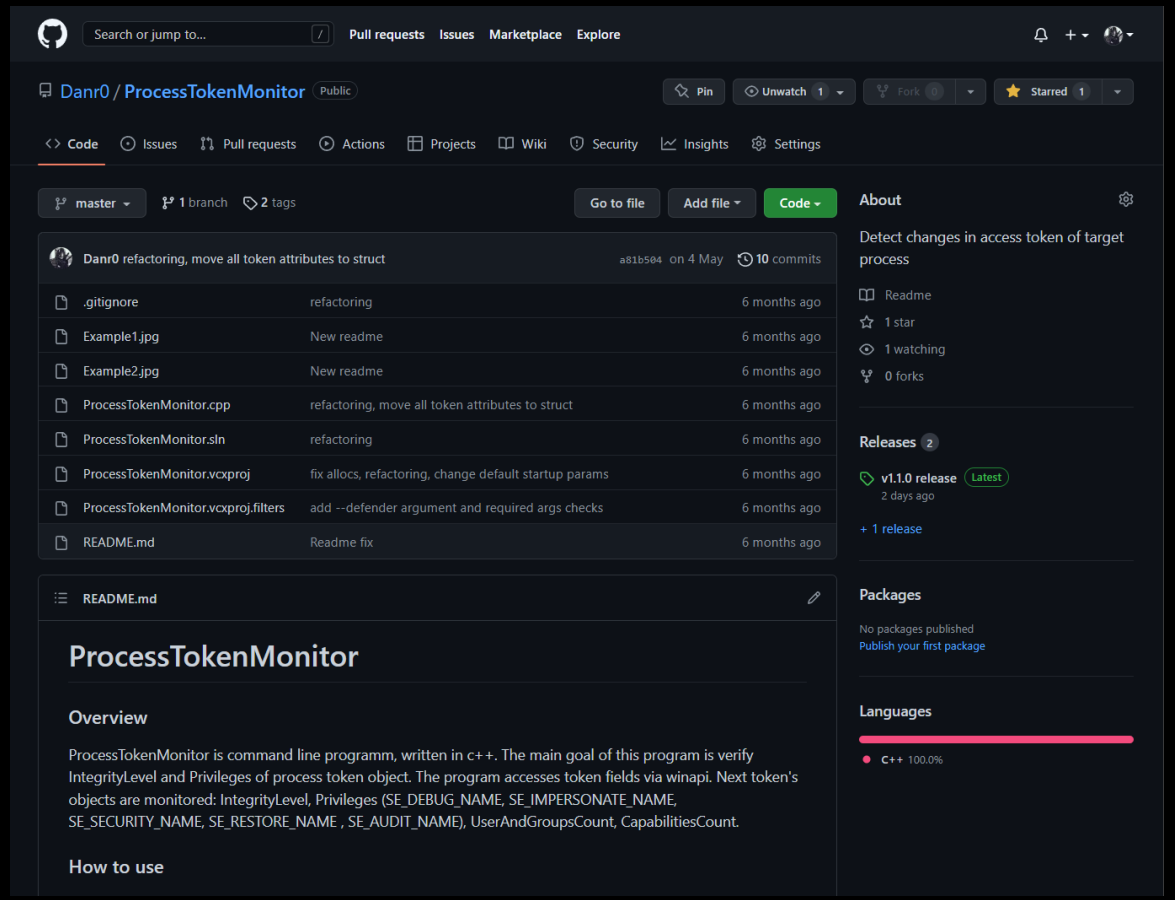
PROCESSTOKENMONITOR OVERVIEW

If the computer doesn't support hardware virtualization, what can I use for a defense?

PROCESSTOKENMONITOR DETECTS THE ATTACK

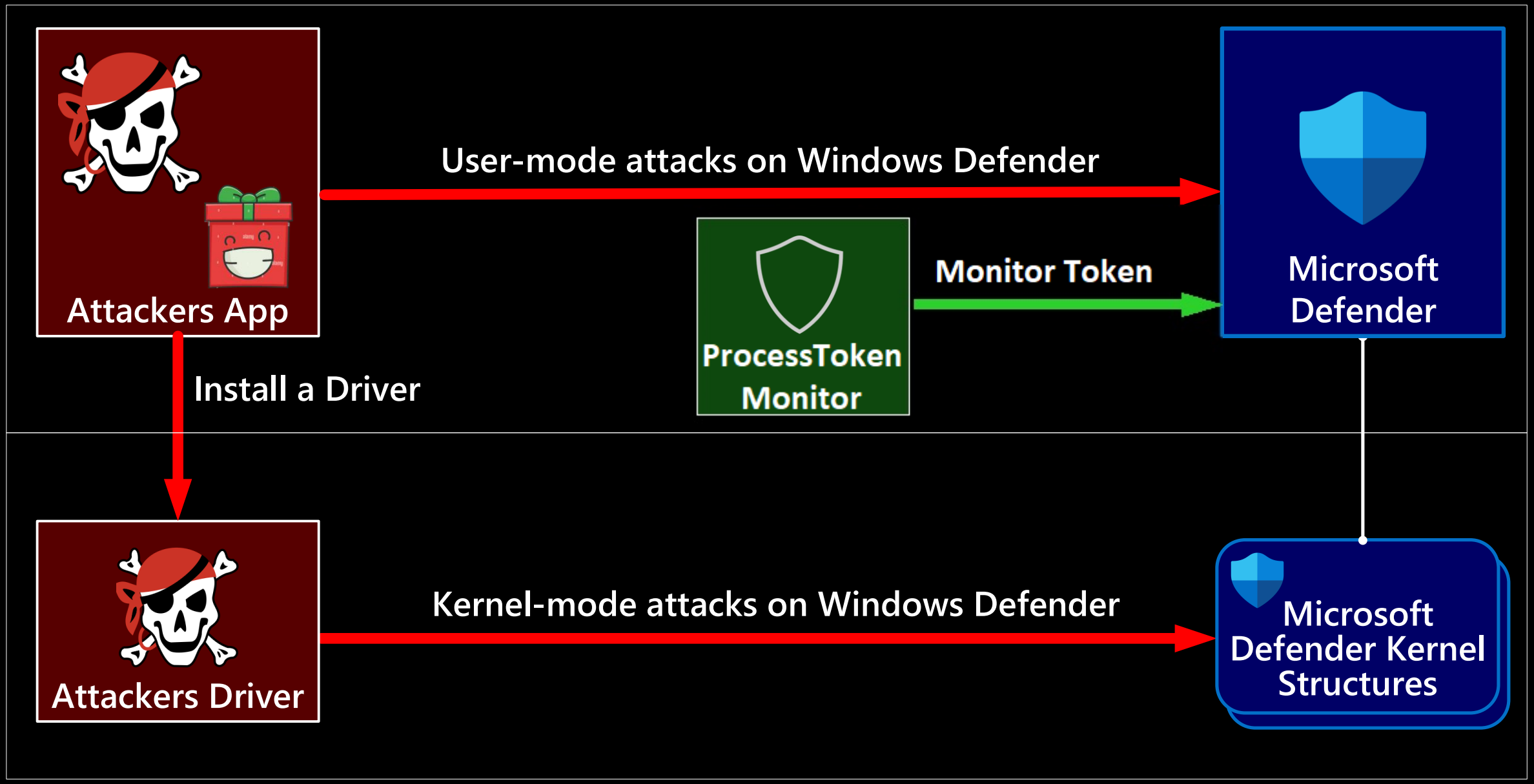


PROCESSTOKENMONITOR

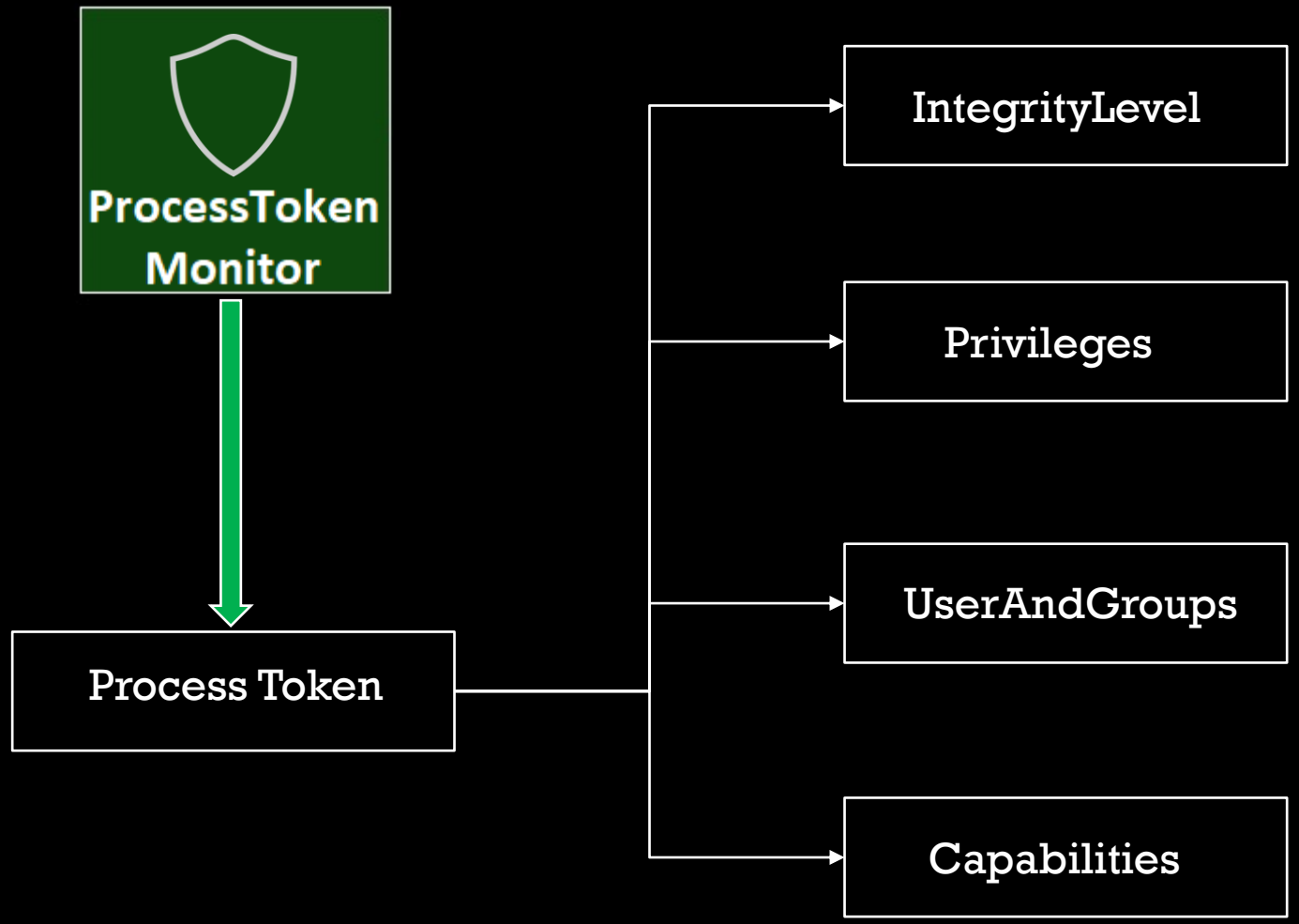


GitHub: https://github.com/Danr0/ProcessTokenMonitor

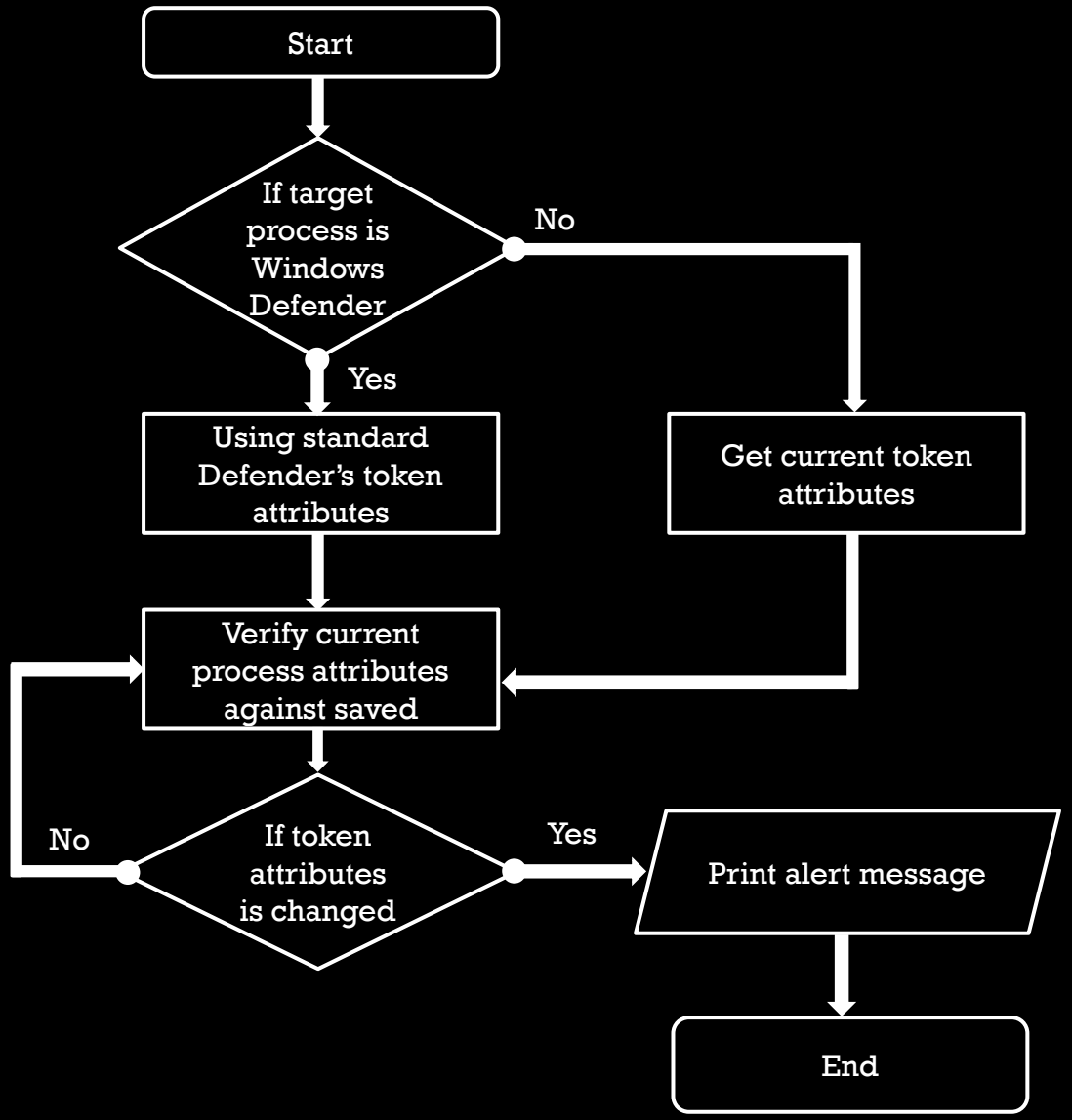
Program monitors the process token



MONITORED ATTRIBUTES OF TOKEN



ALGORITHM SCHEME



PROCESSTOKENMONITOR DEMO

ProcessTokenMonitor
detects the atttack

2022

CONCLUSION

1. Kernel-mode threats are very dangerous even for Windows 11 x64
2. The global malware trend is to bypass or disable security products without terminating the AV/EDR apps
3. Microsoft Defender is the most desired goal for attackers
4. Mandatory Integrity Control (MIC) is designed to sandbox untrusted apps, but attackers can abuse MIC to sandbox Microsoft Defender and other AVs
5. MemoryRanger can block attacks on kernel data including attacks on MIC
6. ProcessTokenMonitor can reveal the presented kernel sandboxing attack.

Thank you!

Igor Korkin igor.korkin@gmail.com
Denis Pogonin denpog00@gmail.com

All the details are here igorkorkin.blogspot.com