MemoryRanger Prevents Hijacking FILE_OBJECT structures in Windows Kernel

Igor Korkin

2019 ADFSL Conference

WHOAMI

- MEPhI Alumni, PhD in Cyber Security
- Area of interest is Windows Kernel security:
  - Memory Forensics
  - Rootkits Detection
  - Bare-Metal Hypervisors
- Fan of cross-disciplinary research - igorkorkin.blogspot.com
- Love traveling and powerlifting - igor.korkin

AGENDA

AGENDA

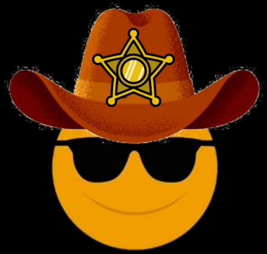
- FILE_OBJECT hijacking: details and demo
-
-

AGENDA

- FILE_OBJECT hijacking: details and demo
- A history of related OS components and memory protection issues
-

AGENDA

- FILE_OBJECT hijacking: details and demo
- A history of related OS components and memory protection issues
- MemoryRanger hypervisor protects sensitive kernel memory

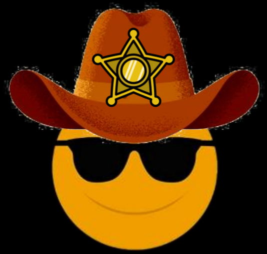


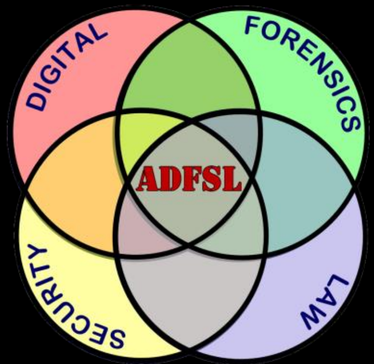
AGENDA

- FILE_OBJECT hijacking: details and demo

- A history of related OS components and memory protection issues

- MemoryRanger hypervisor protects sensitive kernel memory





File Manager in Kernel Mode

COMPANY BUDGET 2019

	Budget
Team A	$3,000,000
Team B	$7,000,000

Page 1 of 1 10 words English (United States) 100%

ENG 9:43 PM


Document1 - Word
File
Share
COMPANY BUDGET 2019
Budget
Team A $3,000,000
Team B $7,000,000
Page 1 of 1
10 words
English (United States)
100%
ENG
9:43 PM

ZWCREATEFILE ROUTINE

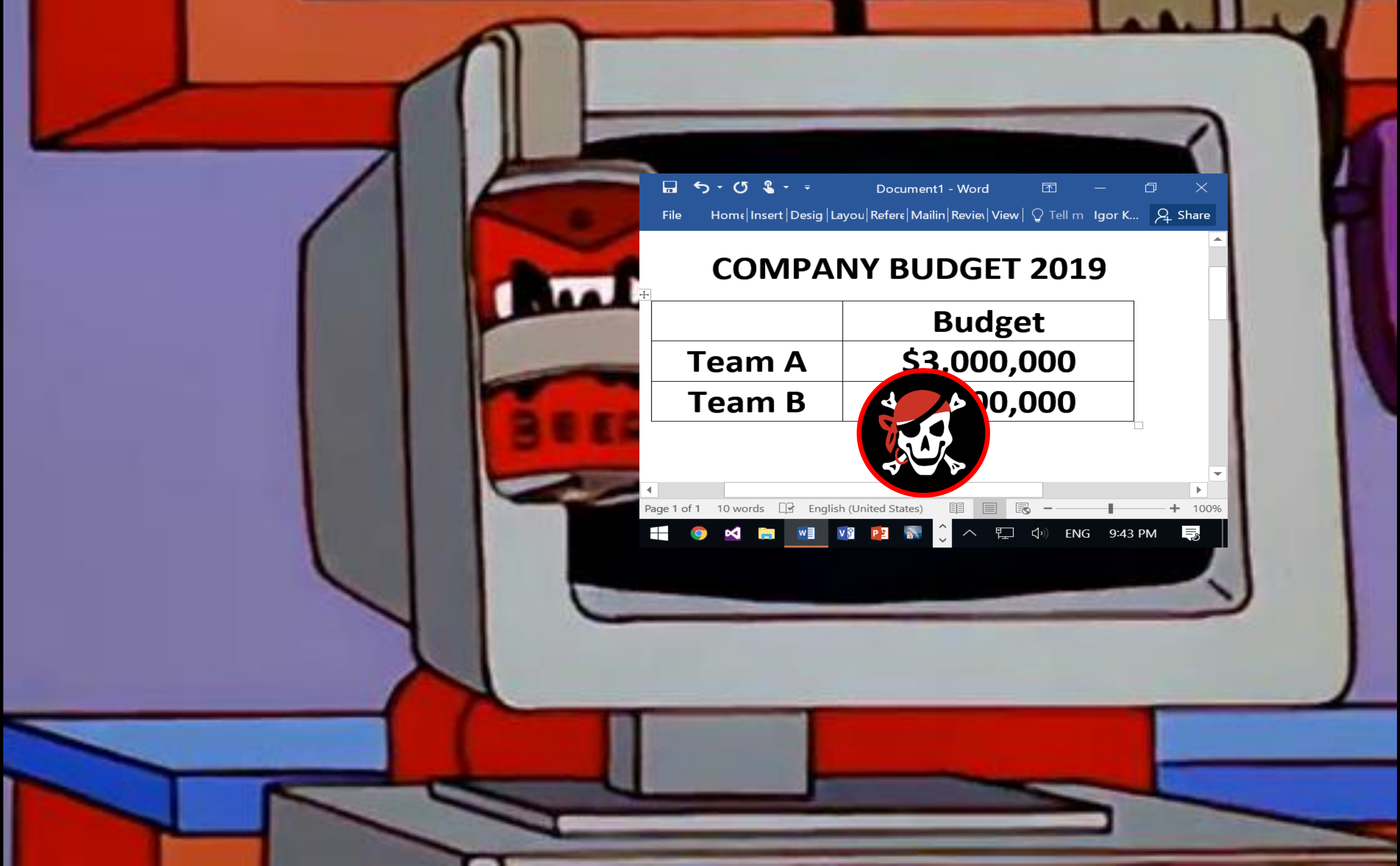
```
NTSTATUS ZwCreateFile(..., ShareAccess, ...);
```

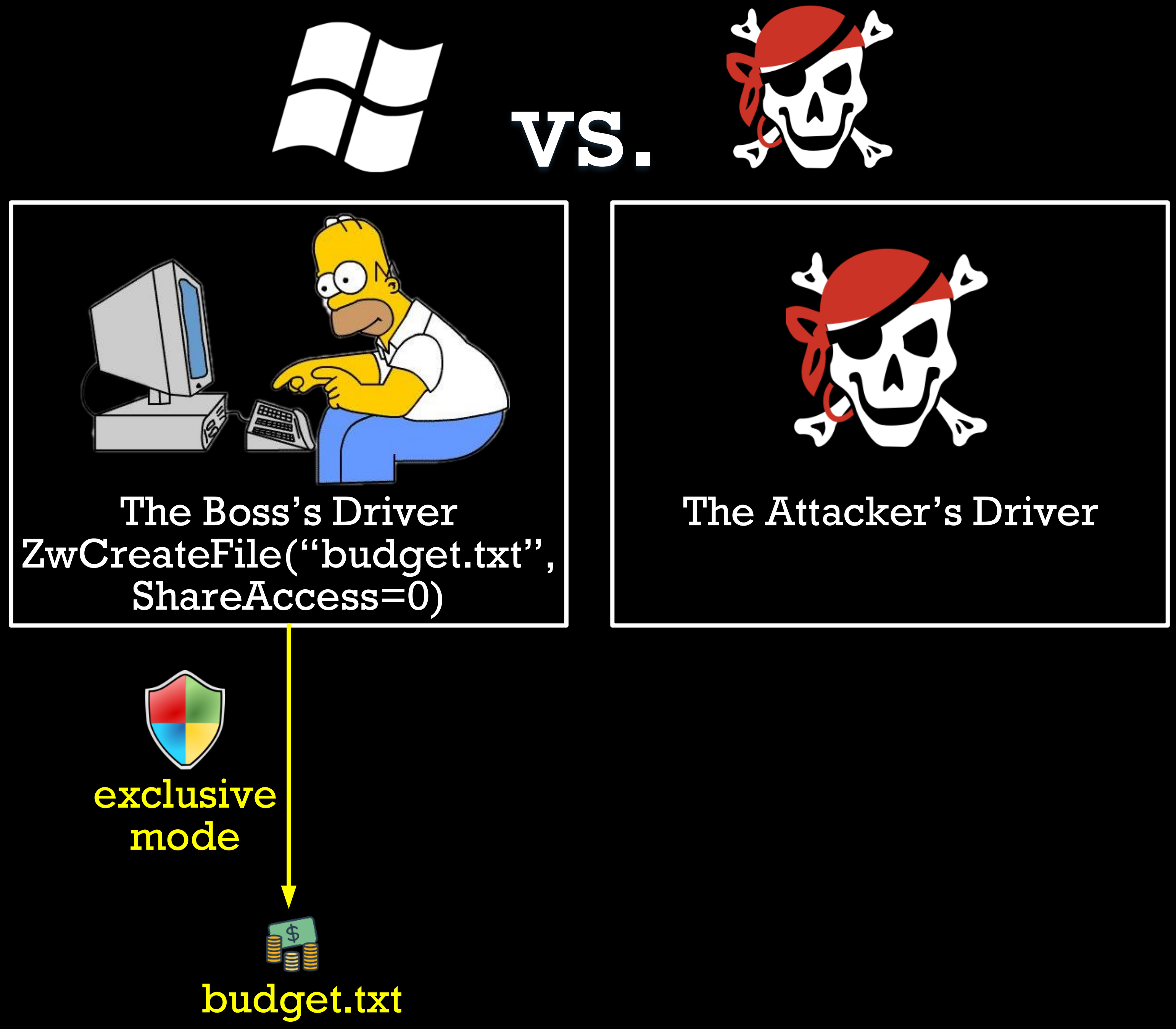
ZWCREATEFILE ROUTINE

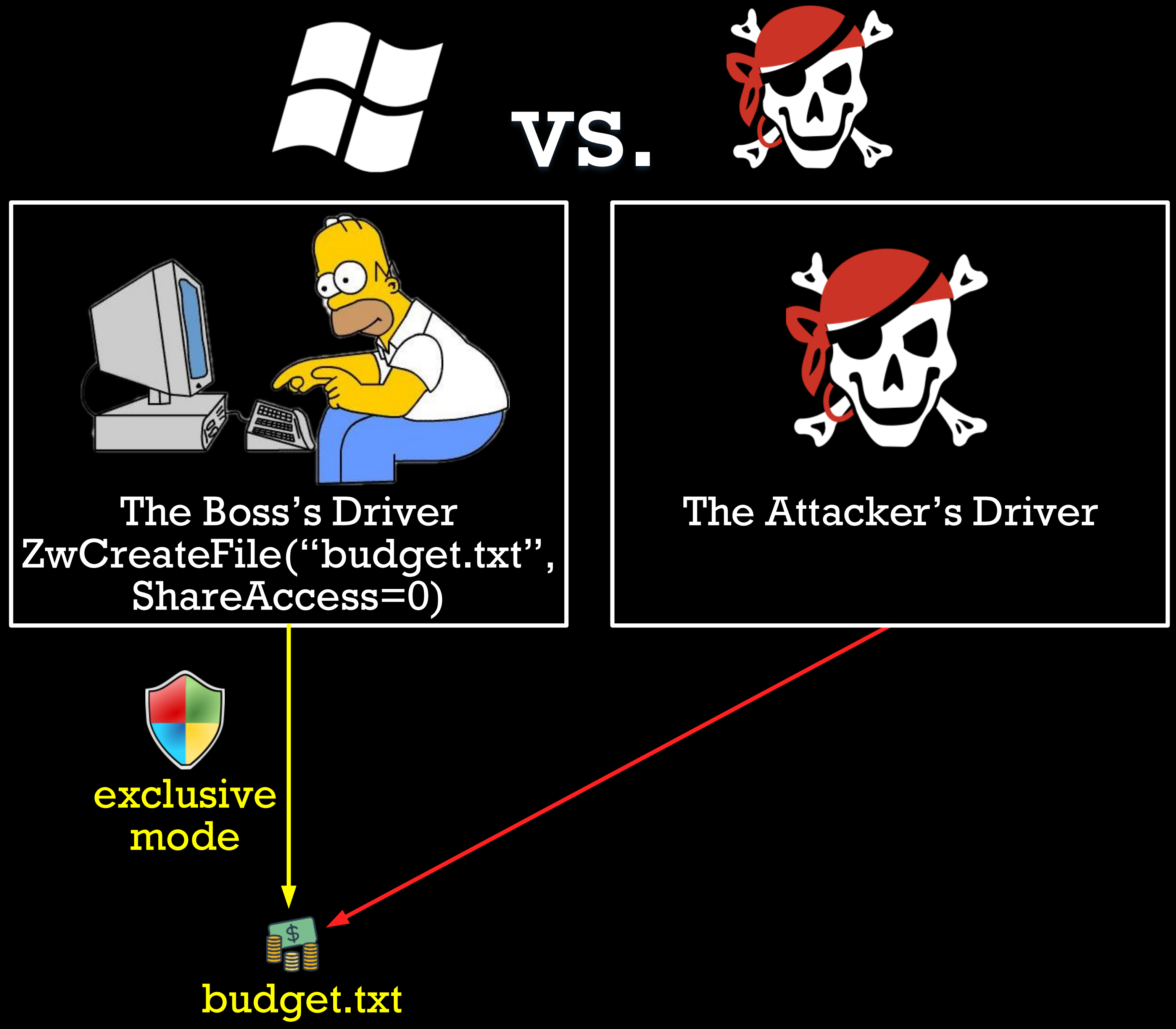
```
NTSTATUS ZwCreateFile(..., ShareAccess, ...);
```

`ShareAccess`

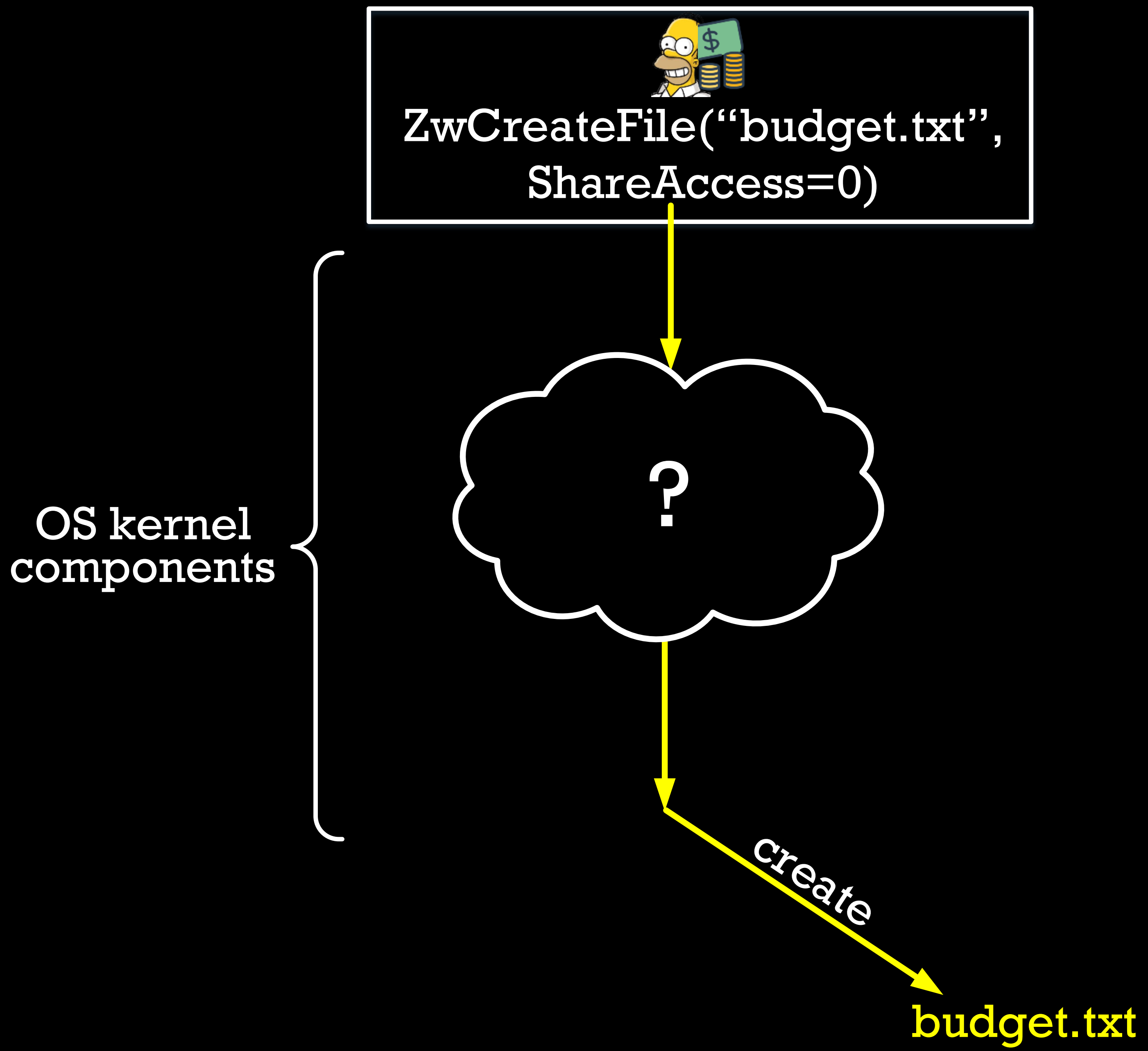
- ShareAccess flag determines whether other drivers can access the opened file.
- Calling ZwCreateFile with ShareAccess=0 gives the caller exclusive access to the file.


Document1 - Word
File
Home Insert Desig Layou Refere Mailin Reviev View Tell m Igor K... Share
COMPANY BUDGET 2019
Budget
Team A
$3,000,000
Team B
00,000
Page 1 of 1
10 words
English (United States)
100%
ENG
9:43 PM


vs.
The Boss's Driver
ZwCreateFile("budget.txt", ShareAccess=0)
The Attacker's Driver
exclusive mode
budget.txt


vs.
The Boss's Driver
ZwCreateFile("budget.txt",
ShareAccess=0)
The Attacker's Driver
exclusive
mode
budget.txt

FILE SYSTEM ROUTINES IN WINDOWS KERNEL



FILE SYSTEM ROUTINES IN WINDOWS KERNEL

ZwCreateFile("budget.txt", ShareAccess=0)

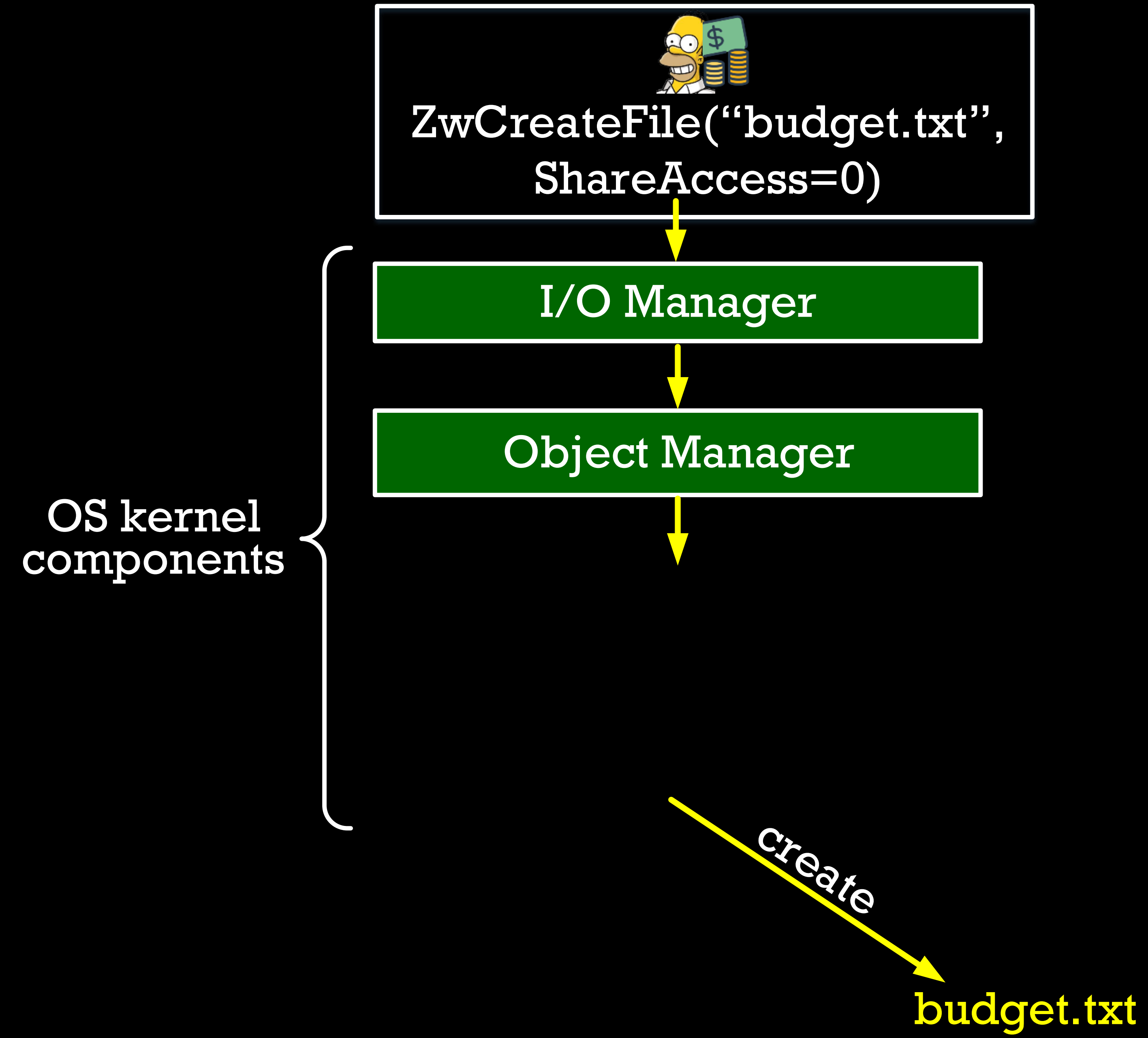
I/O Manager

OS kernel components

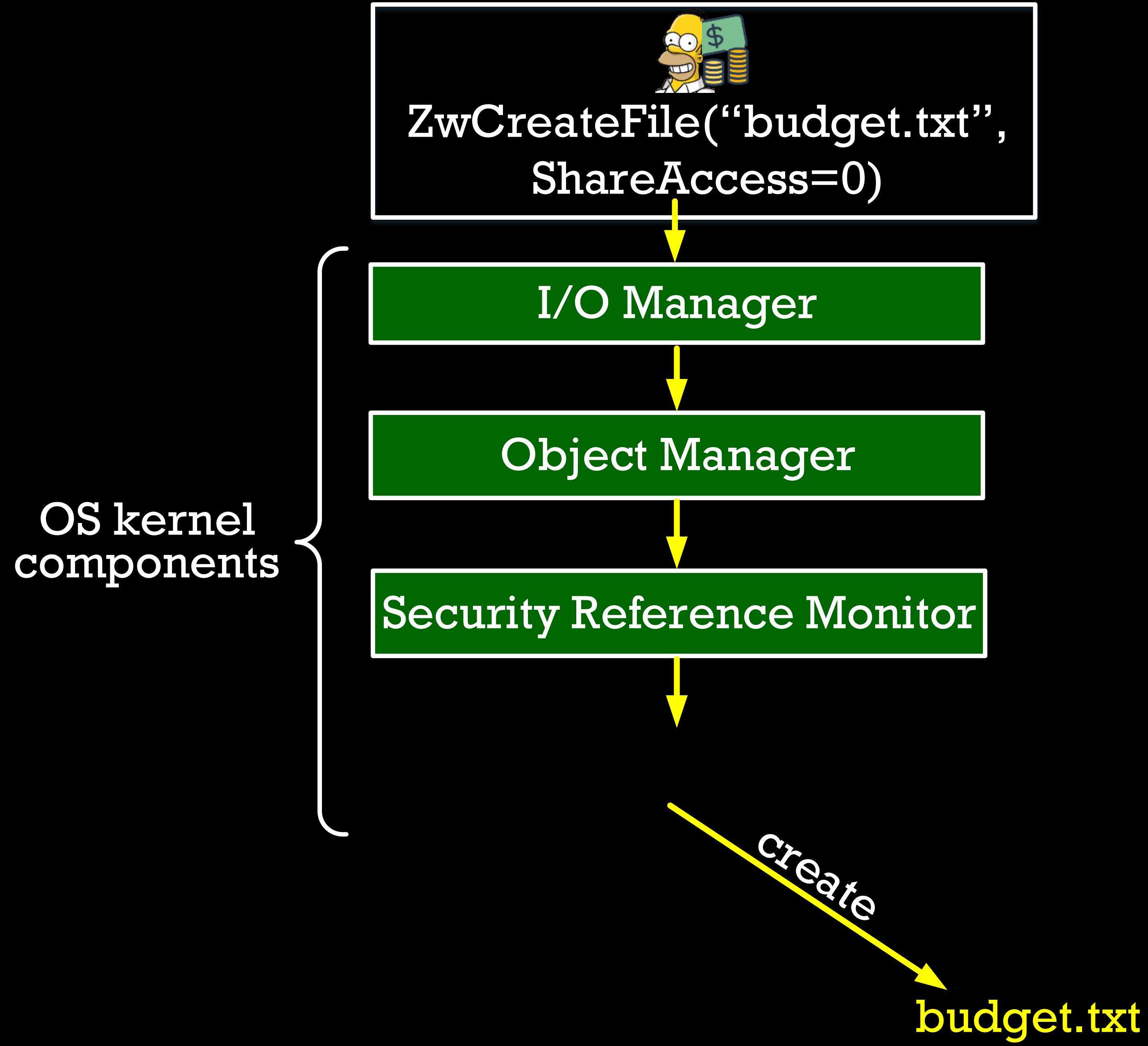
create

budget.txt

FILE SYSTEM ROUTINES IN WINDOWS KERNEL



FILE SYSTEM ROUTINES IN WINDOWS KERNEL



FILE SYSTEM ROUTINES IN WINDOWS KERNEL

ZwCreateFile("budget.txt", ShareAccess=0)

↓

I/O Manager

↓

Object Manager

↓

Security Reference Monitor

↓

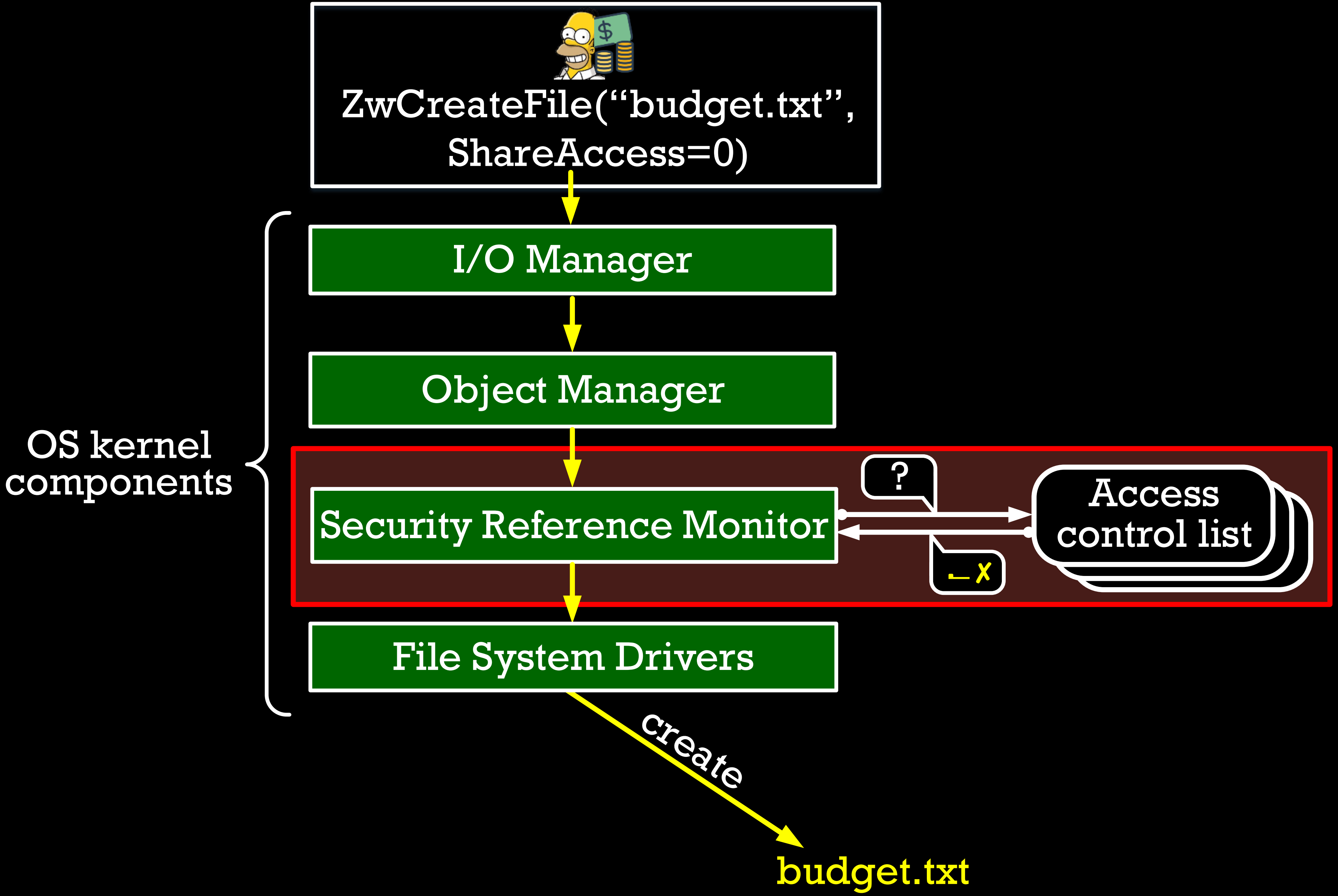
File System Drivers

OS kernel components

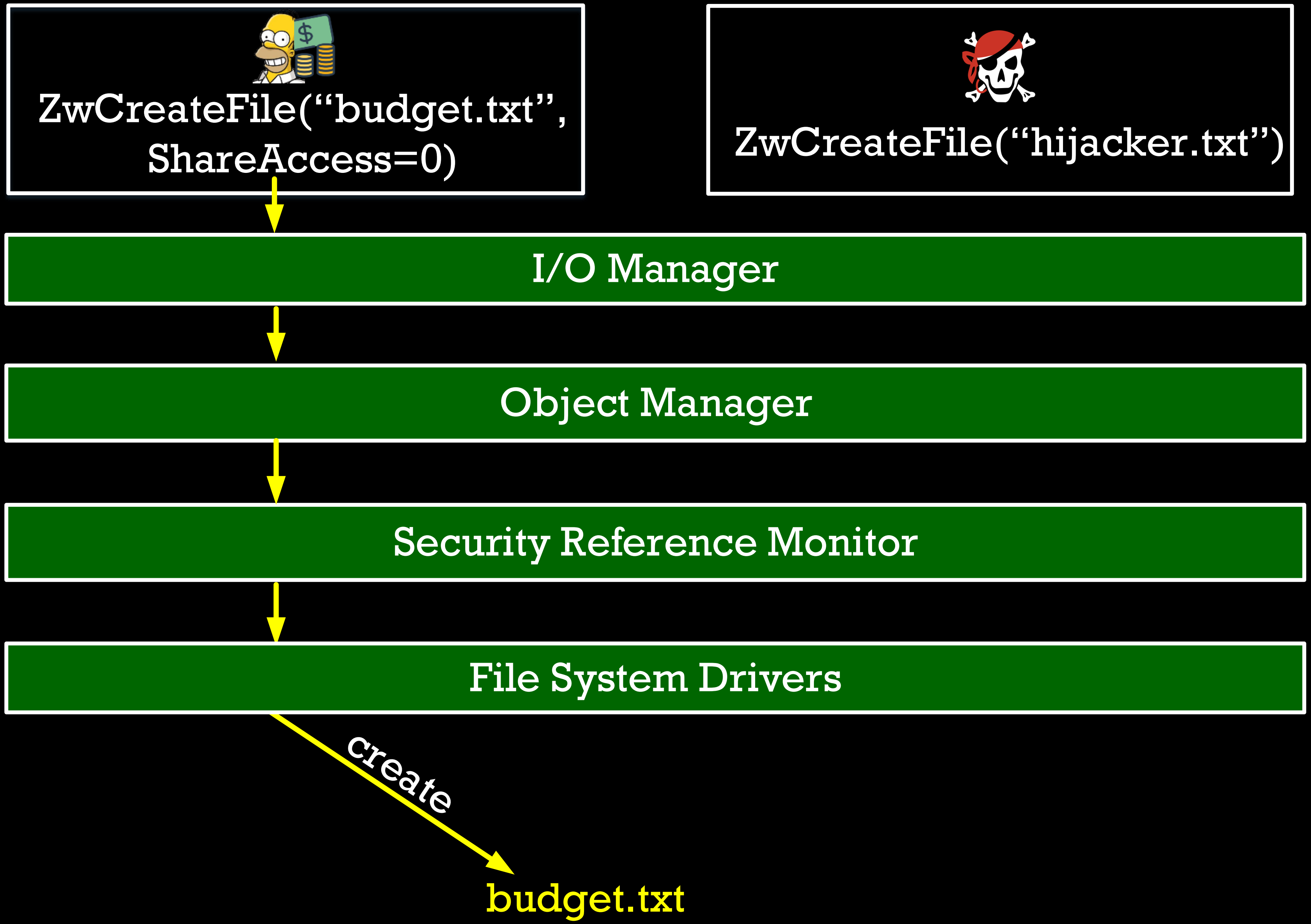
create

budget.txt

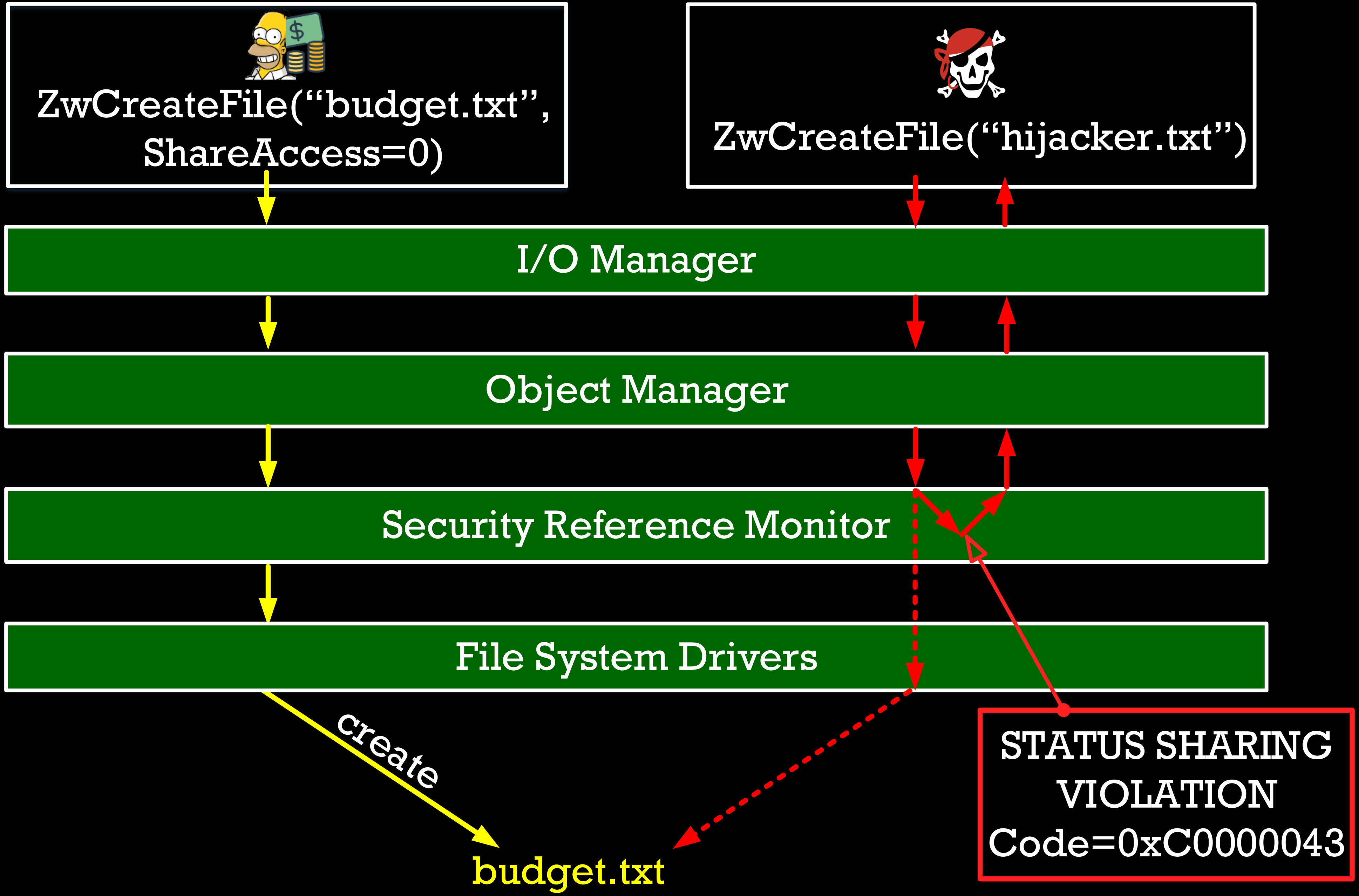
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



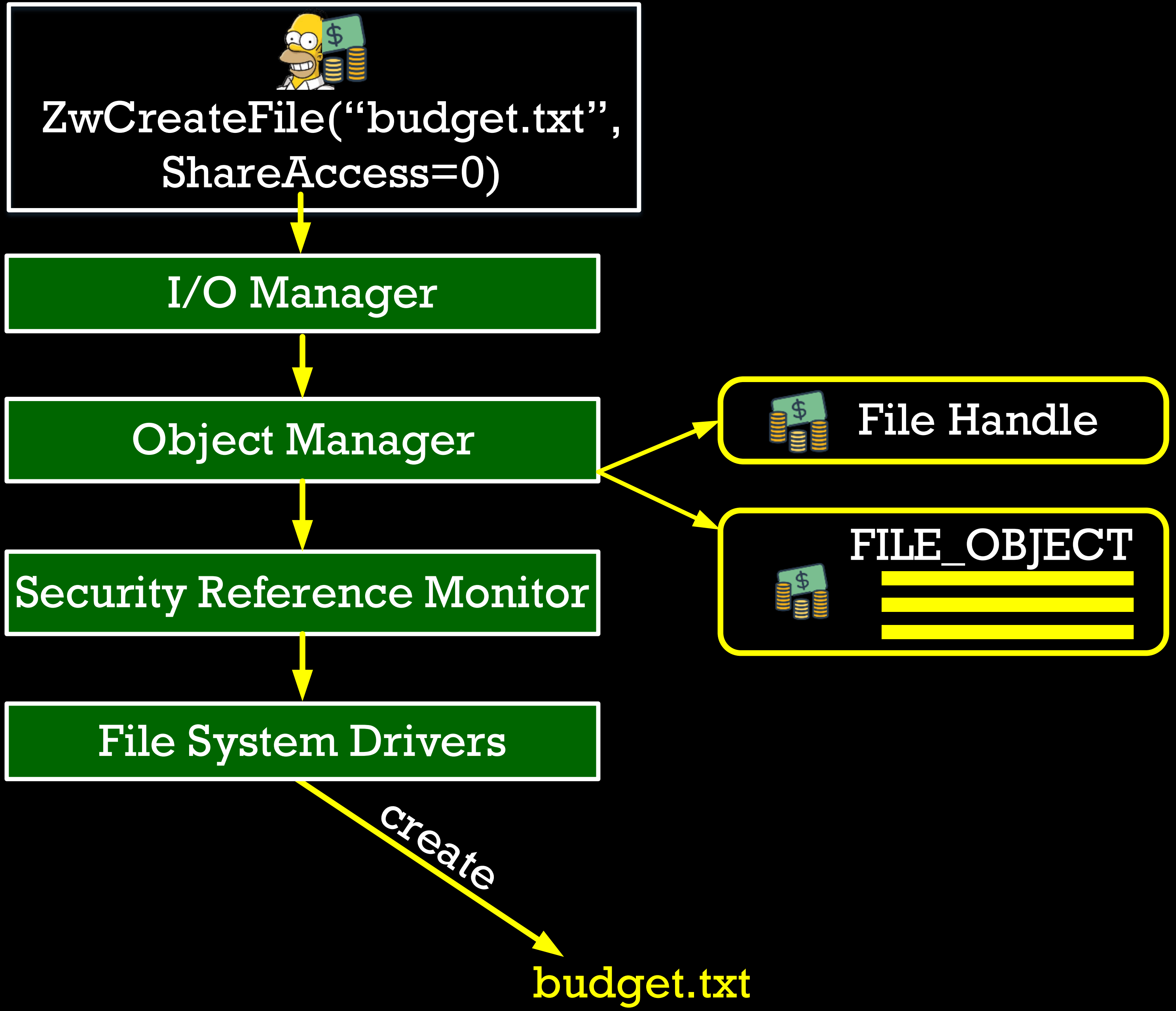
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



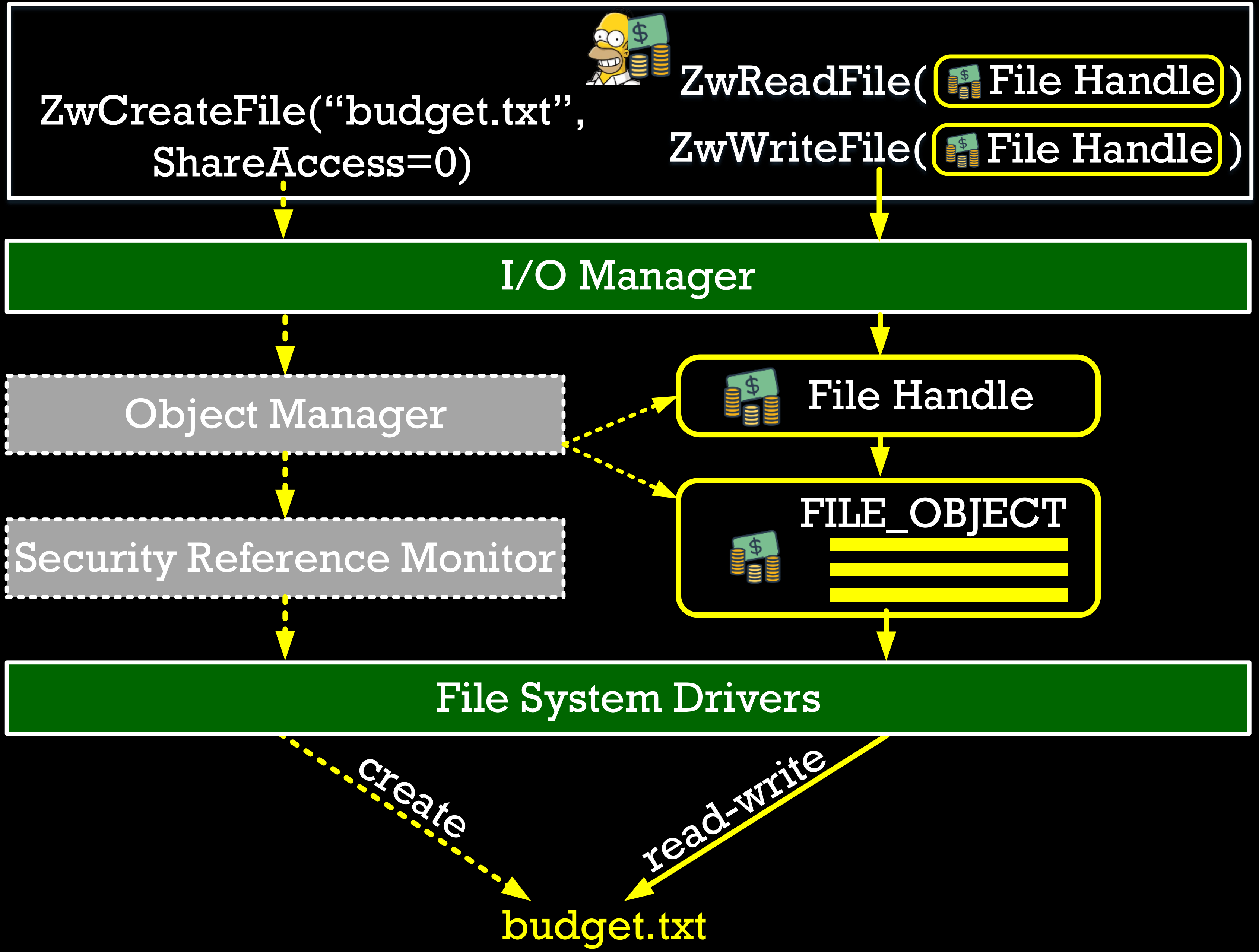
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



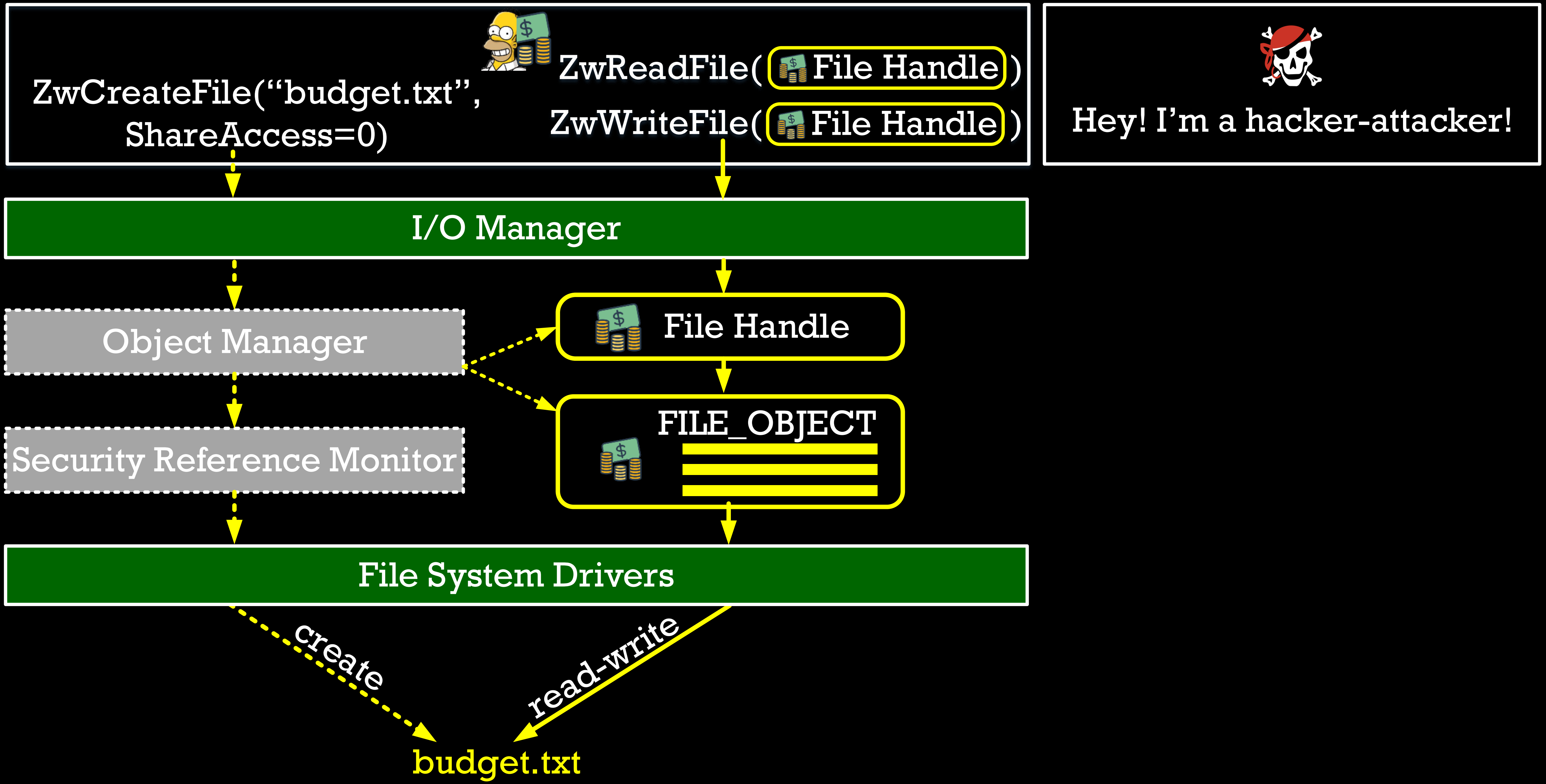
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



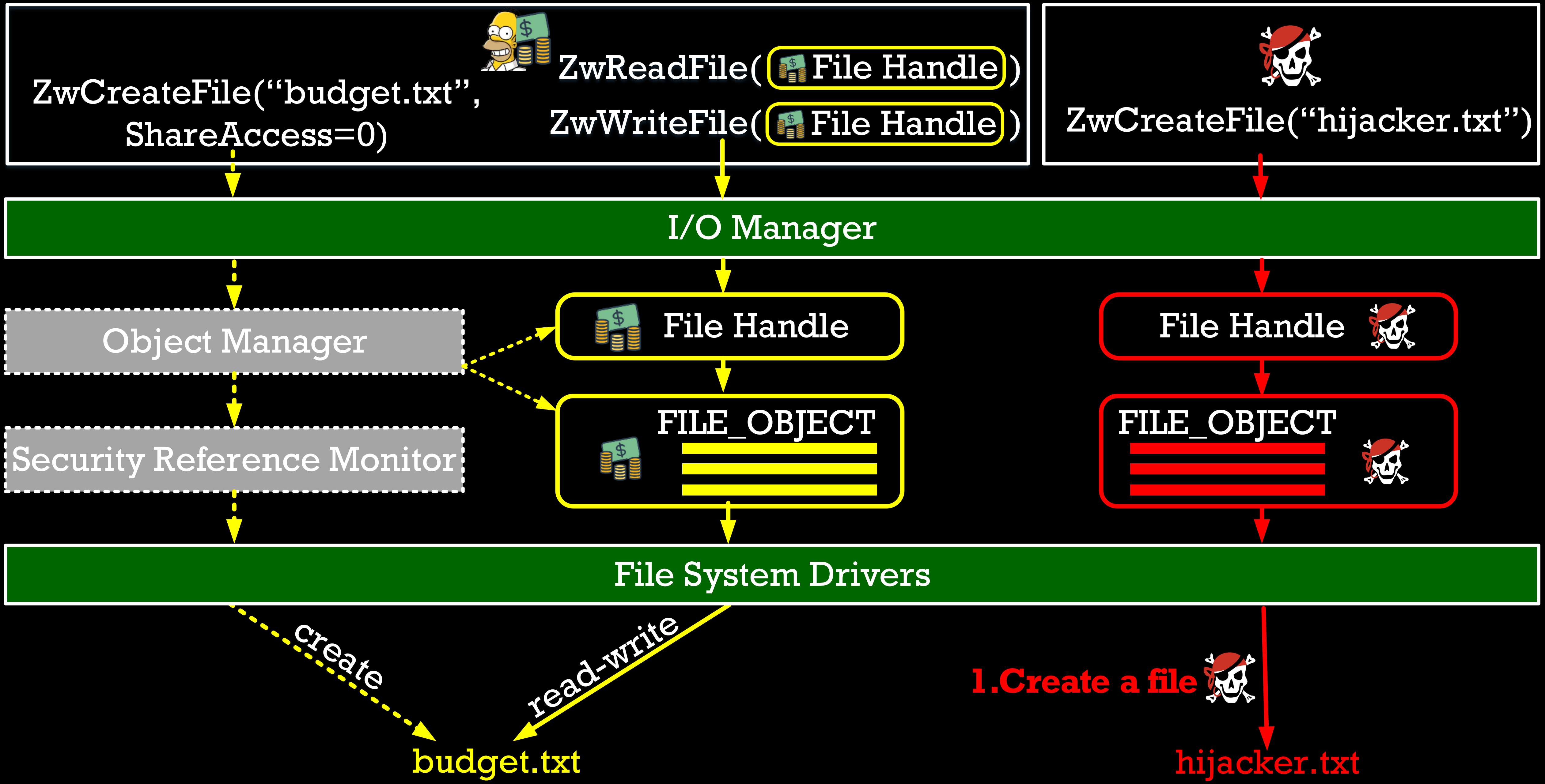
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



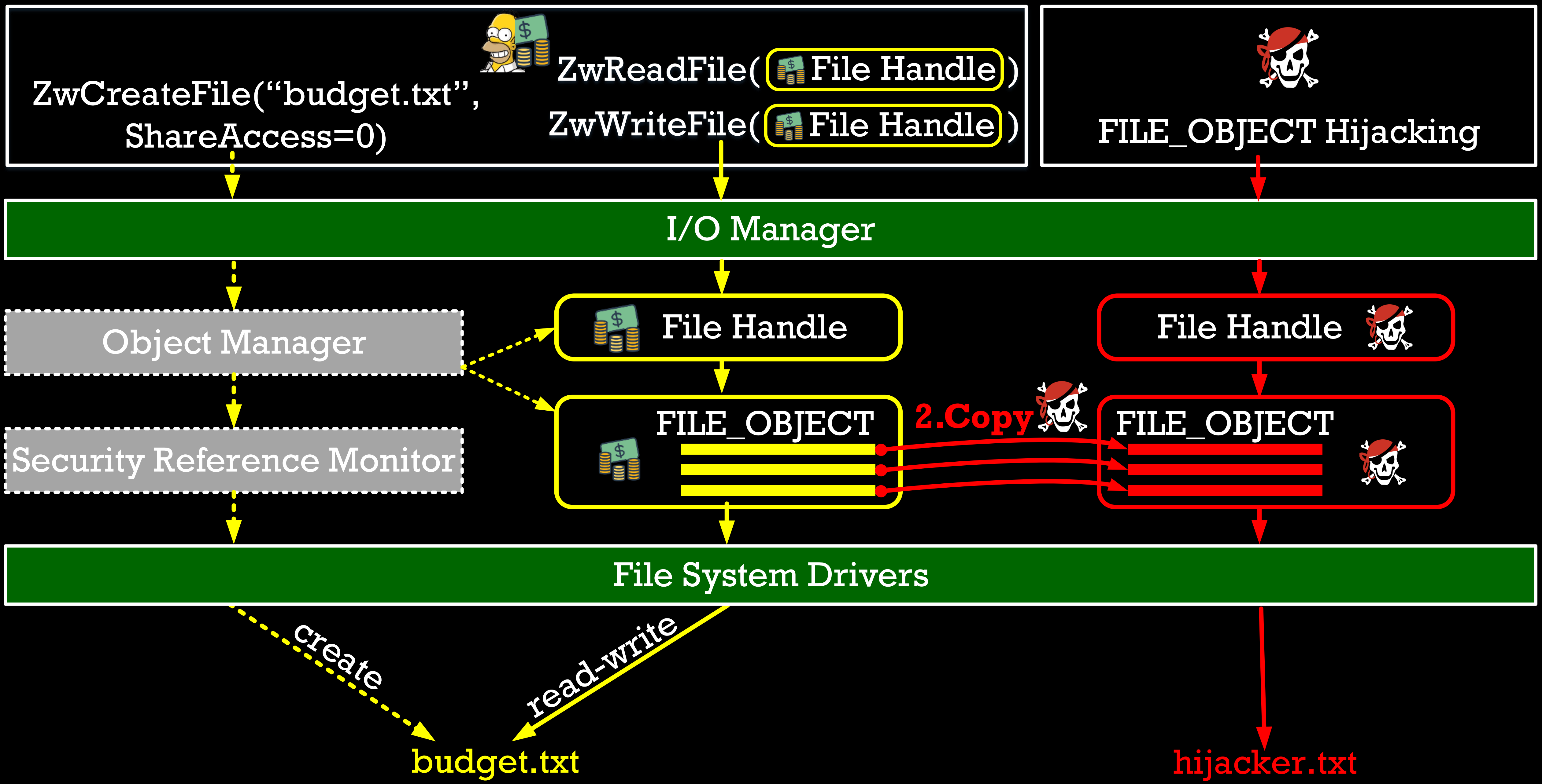
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



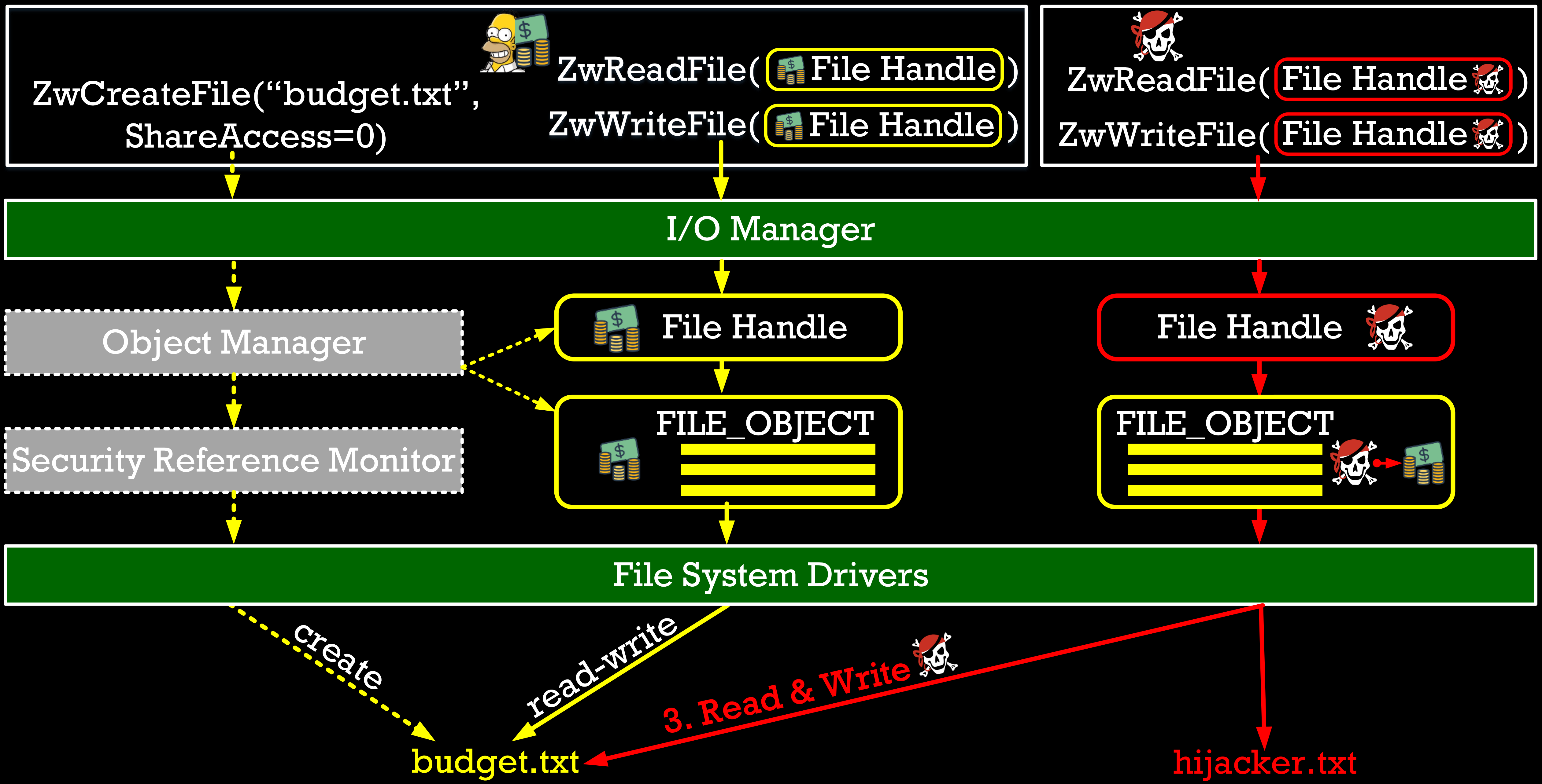
FILE SYSTEM ROUTINES IN WINDOWS KERNEL



FILE SYSTEM ROUTINES IN WINDOWS KERNEL



FILE SYSTEM ROUTINES IN WINDOWS KERNEL

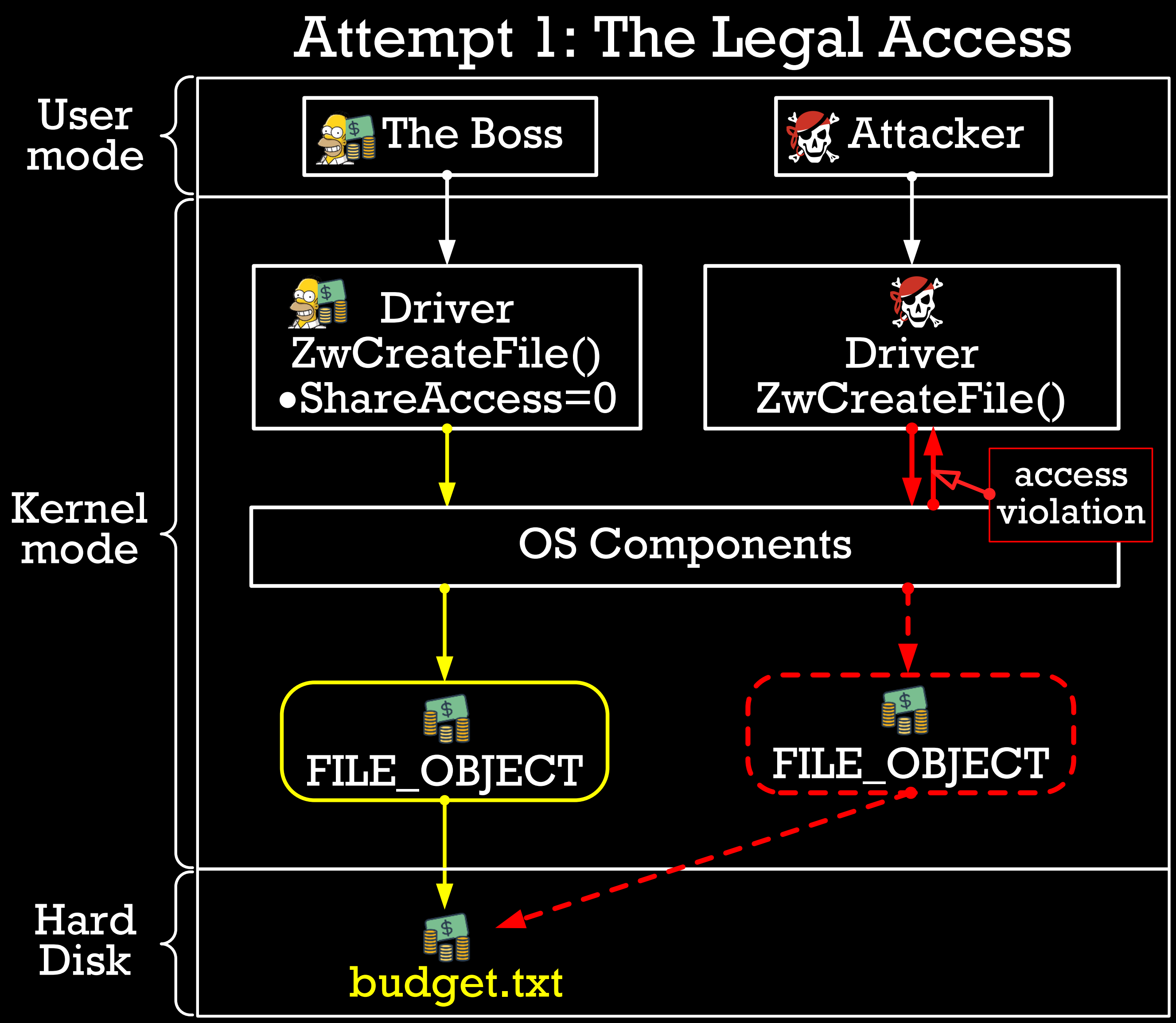


JUST 4 CRUCIAL FIELDS FOR FILES HIJACKING

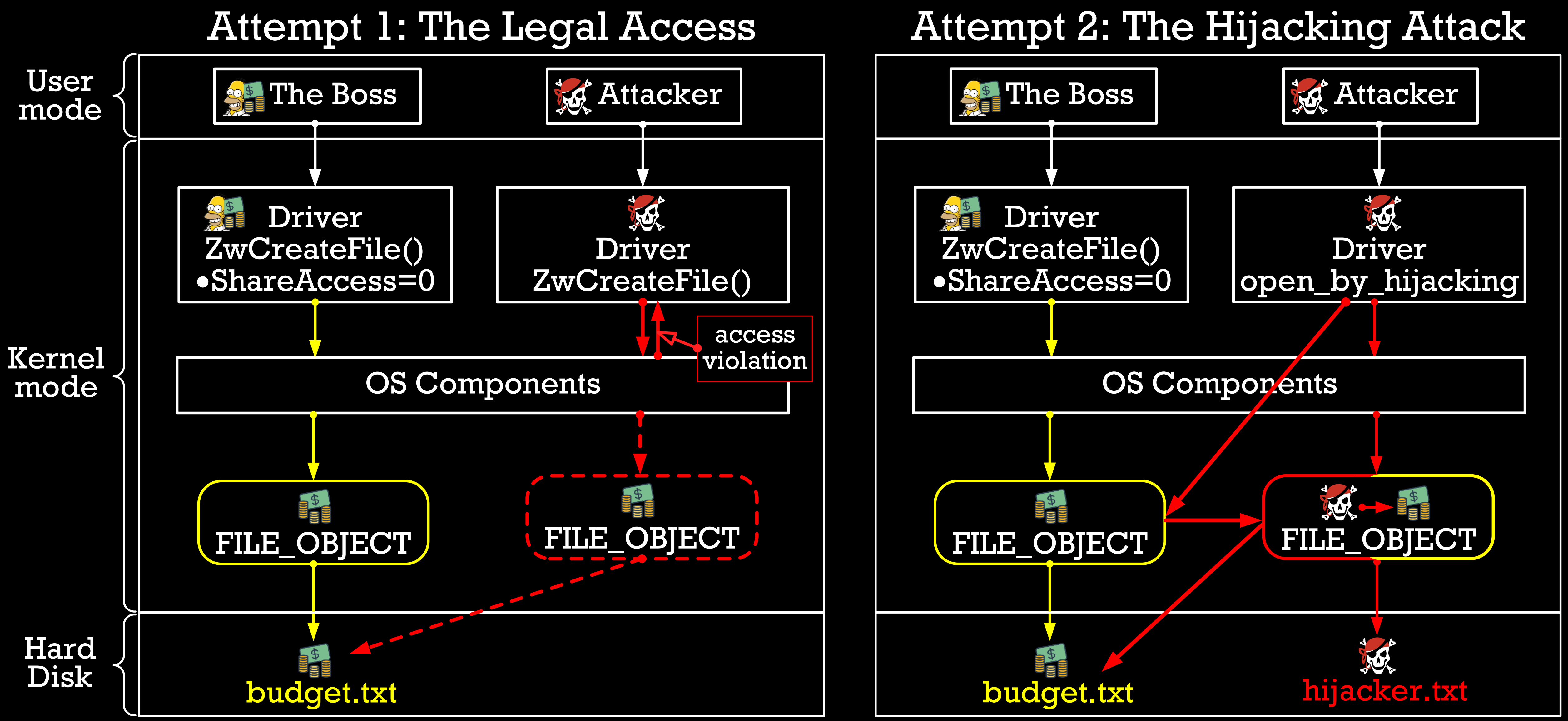
```
typedef struct _FILE_OBJECT {
    …
PVPB Vpb;
PVOID FsContext;
PVOID FsContext2;
PSECTION_OBJECT_POINTERS SectionObjectPointer;
    …
} FILE_OBJECT;
```

- The Vpb field points to a mounted Volume Parameter Block (VPB), associated with the target device object.
- FsContext points to the FSRTL_COMMON_FCB_HEADER structure, which has to be allocated by the file driver.
- FsContext2 field refers to the Context Control Block (CBB) associated with the file object
- SectionObjectPointer stores file-mapping and caching-related information for a file stream.

THE ATTACK


Attempt 1: The Legal Access
User mode
The Boss
Attacker
Kernel mode
Driver
ZwCreateFile()
•ShareAccess=0
Driver
ZwCreateFile()
access violation
OS Components
FILE_OBJECT
FILE_OBJECT
Hard Disk
budget.txt

THE ATTACK

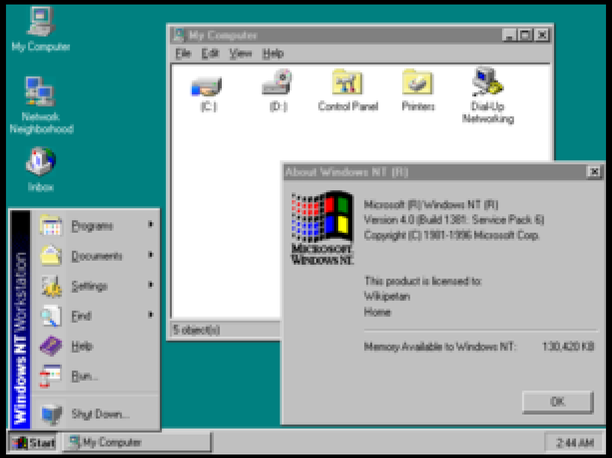

Attempt 1: The Legal Access
Attempt 2: The Hijacking Attack
User mode
Kernel mode
Hard Disk
The Boss
Attacker
Driver
ZwCreateFile()
•ShareAccess=0
Driver
ZwCreateFile()
access violation
OS Components
FILE_OBJECT
FILE_OBJECT
budget.txt
The Boss
Attacker
Driver
ZwCreateFile()
•ShareAccess=0
Driver
open_by_hijacking
OS Components
FILE_OBJECT
FILE_OBJECT
budget.txt
hijacker.txt

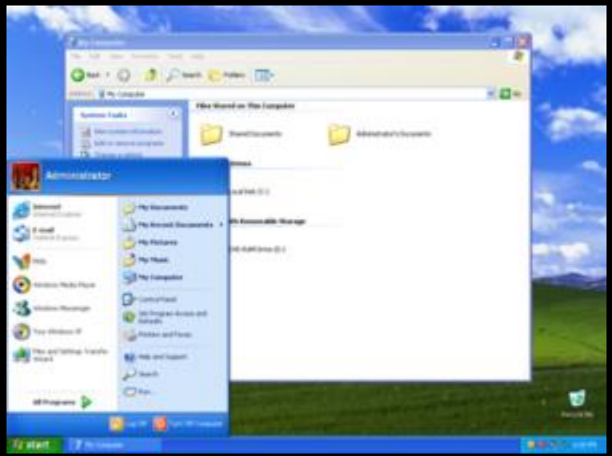
DEMO: THE ATTACK

The online version is here –
https://www.youtube.com/watch?v=2mU85RluOSA?vq=hd1080

THE ANALYSIS OF THE ATTACK

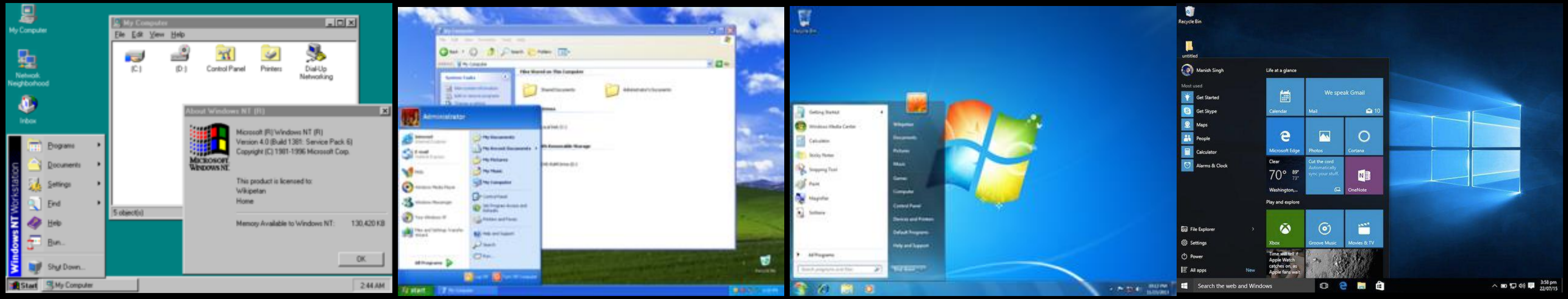
- All Windows OSes since NT 4.0 are vulnerable for FILE_OBJECT hijacking:



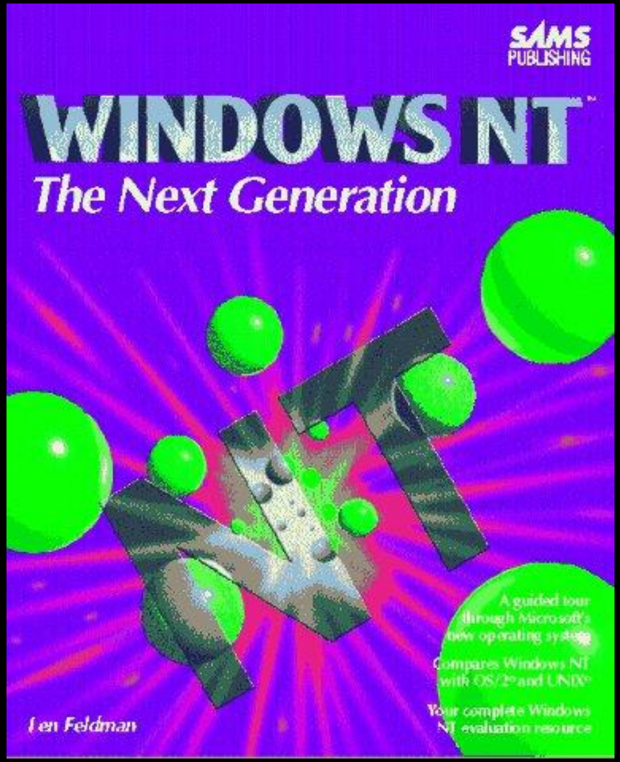


THE ANALYSIS OF THE ATTACK

- All Windows OSes since NT 4.0 are vulnerable for FILE_OBJECT hijacking:



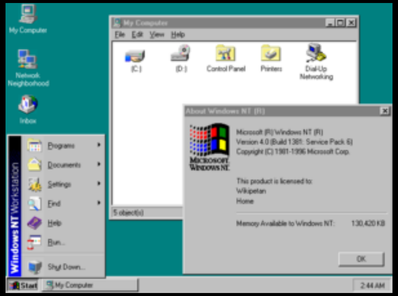
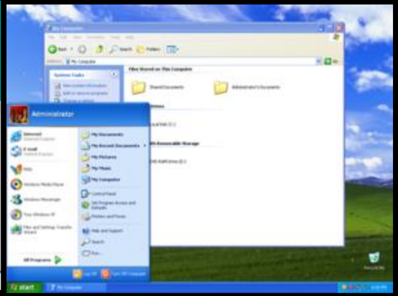
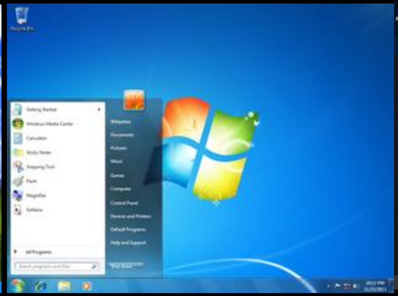
- **1993** - the first mention of Object Manager and Security Reference Monitor



Windows NT: The Next Generation by Len Feldman, March 1, **1993**

THE ANALYSIS OF THE ATTACK

- All Windows OSes since NT 4.0 are vulnerable for FILE_OBJECT hijacking:


- **1993** - the first mention of Object Manager and Security Reference Monitor

- **1965** – the first memory isolation concept Multics* was developed for General Electric 645 mainframe. Multics joined to the ARPANet and gave rise to the Unix.

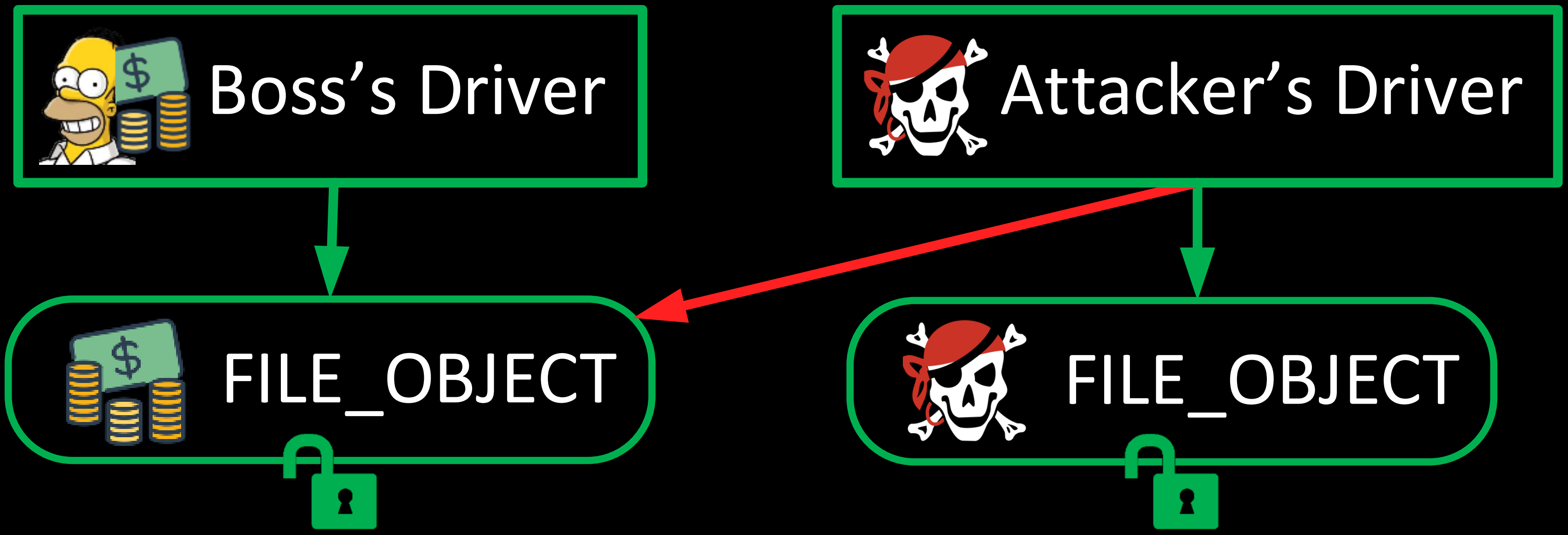
Two Fathers of Multics

Fernando Corbato

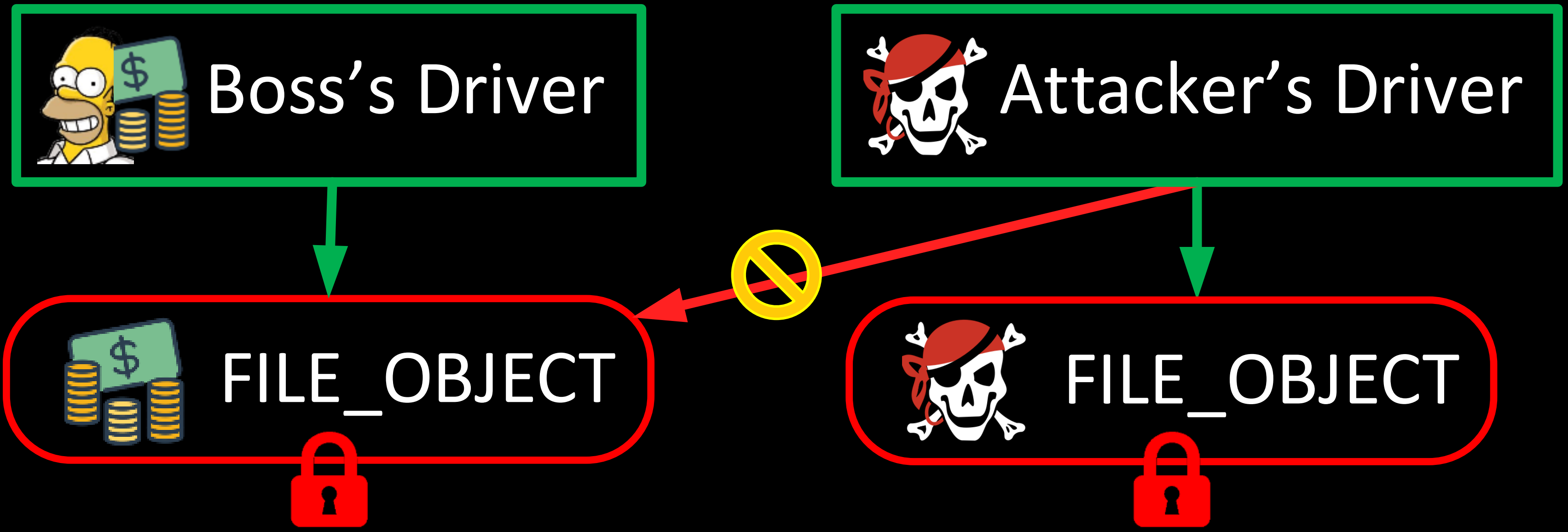
Victor Vyssotsky

*DOI: http://dx.doi.org/10.1145/1463891.1463912

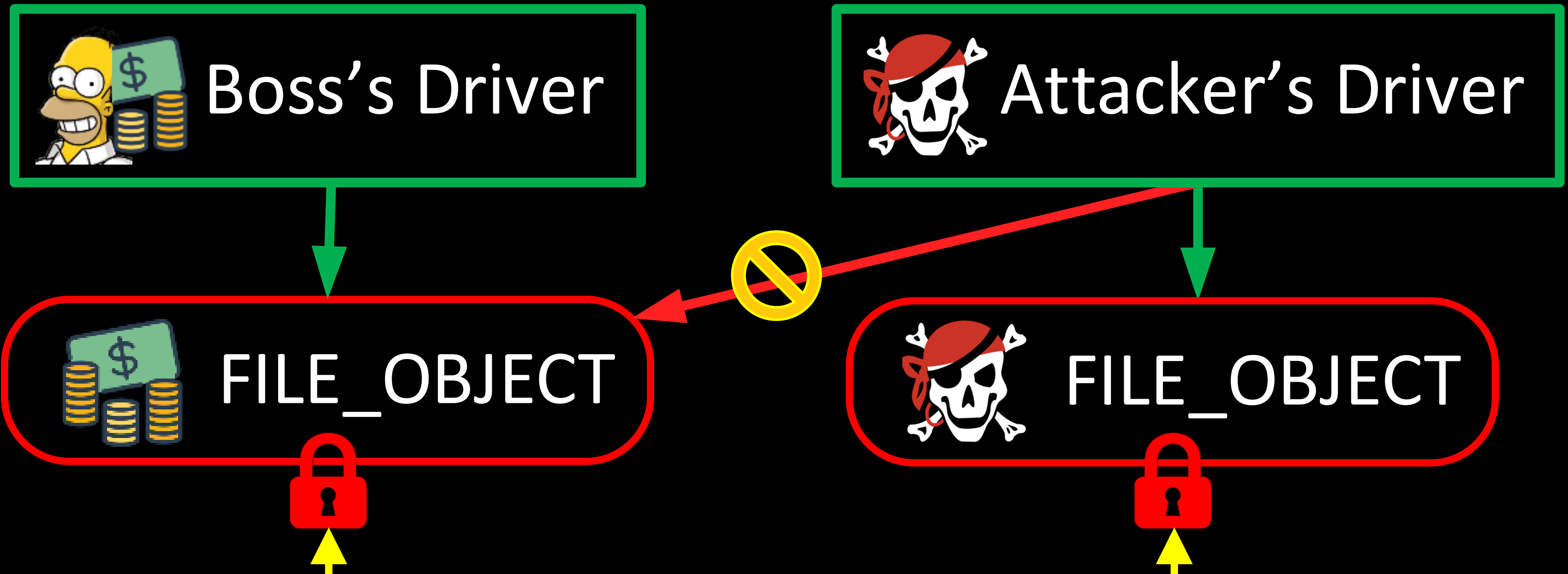
THE FILE_OBJECT PROTECTION VIA ENCRYPTION



THE FILE_OBJECT PROTECTION VIA ENCRYPTION

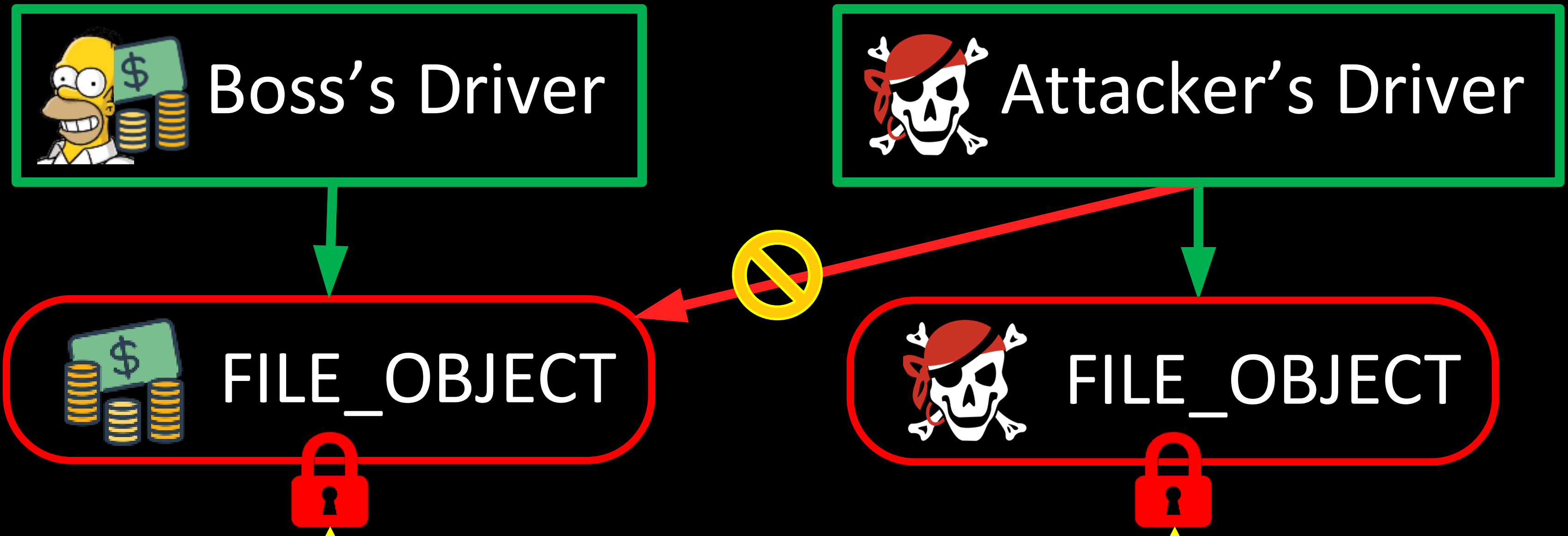


THE FILE_OBJECT PROTECTION VIA ENCRYPTION



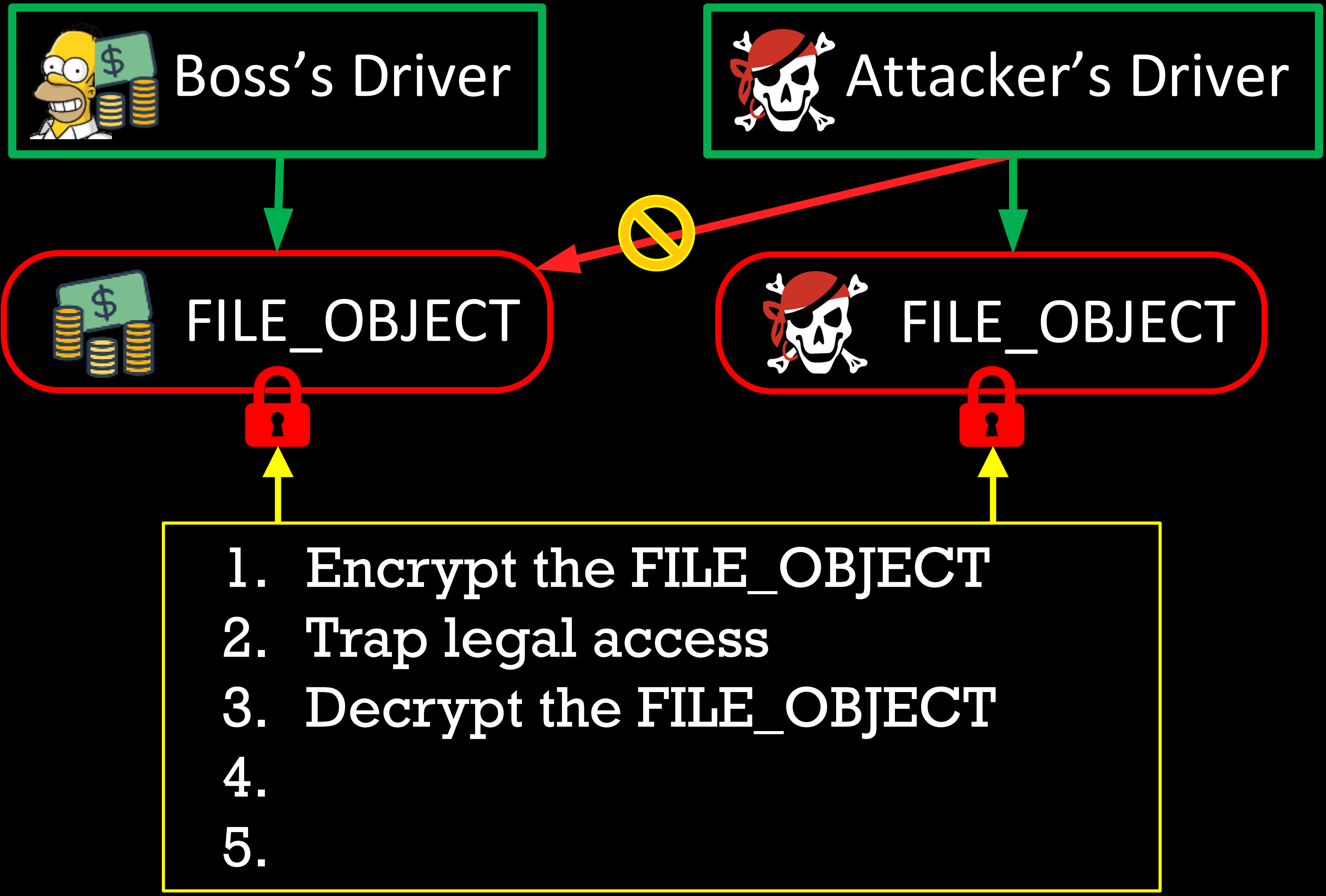
1. Encrypt the FILE_OBJECT
2.
3.
4.
5.

THE FILE_OBJECT PROTECTION VIA ENCRYPTION

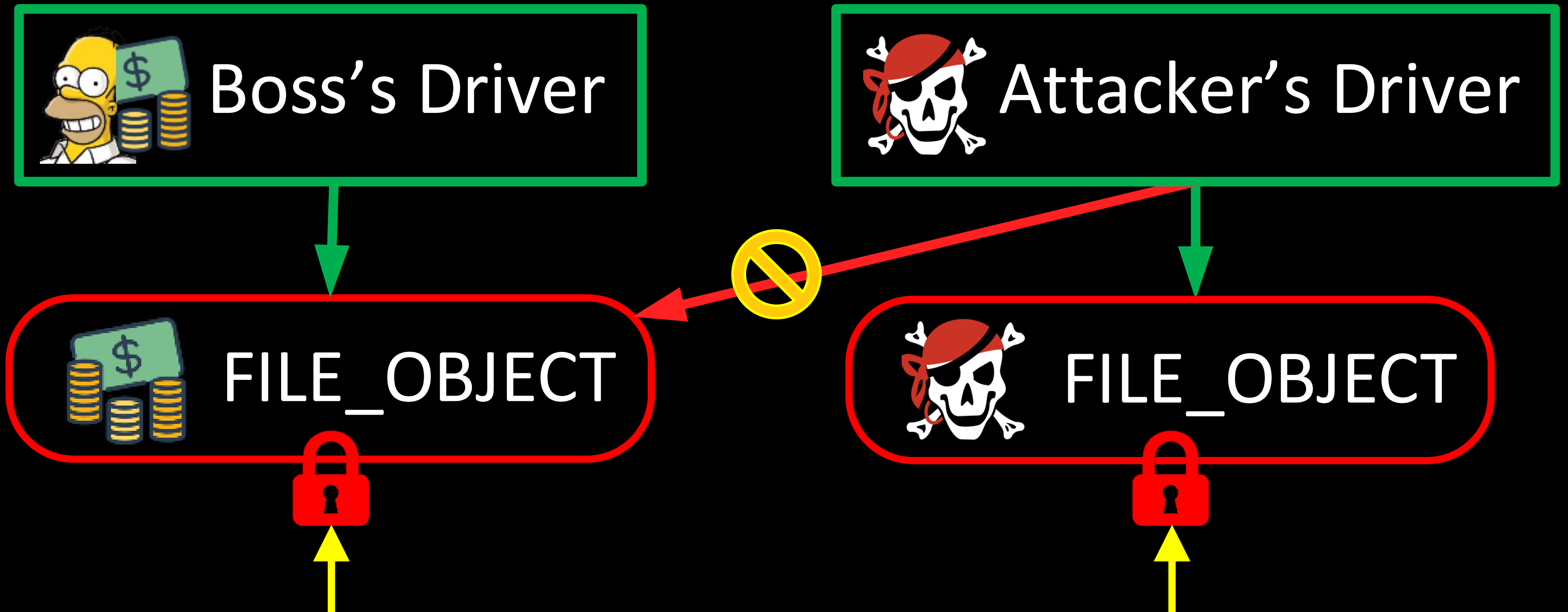


1. Encrypt the FILE_OBJECT
2. Trap legal access
3.
4.
5.

THE FILE_OBJECT PROTECTION VIA ENCRYPTION

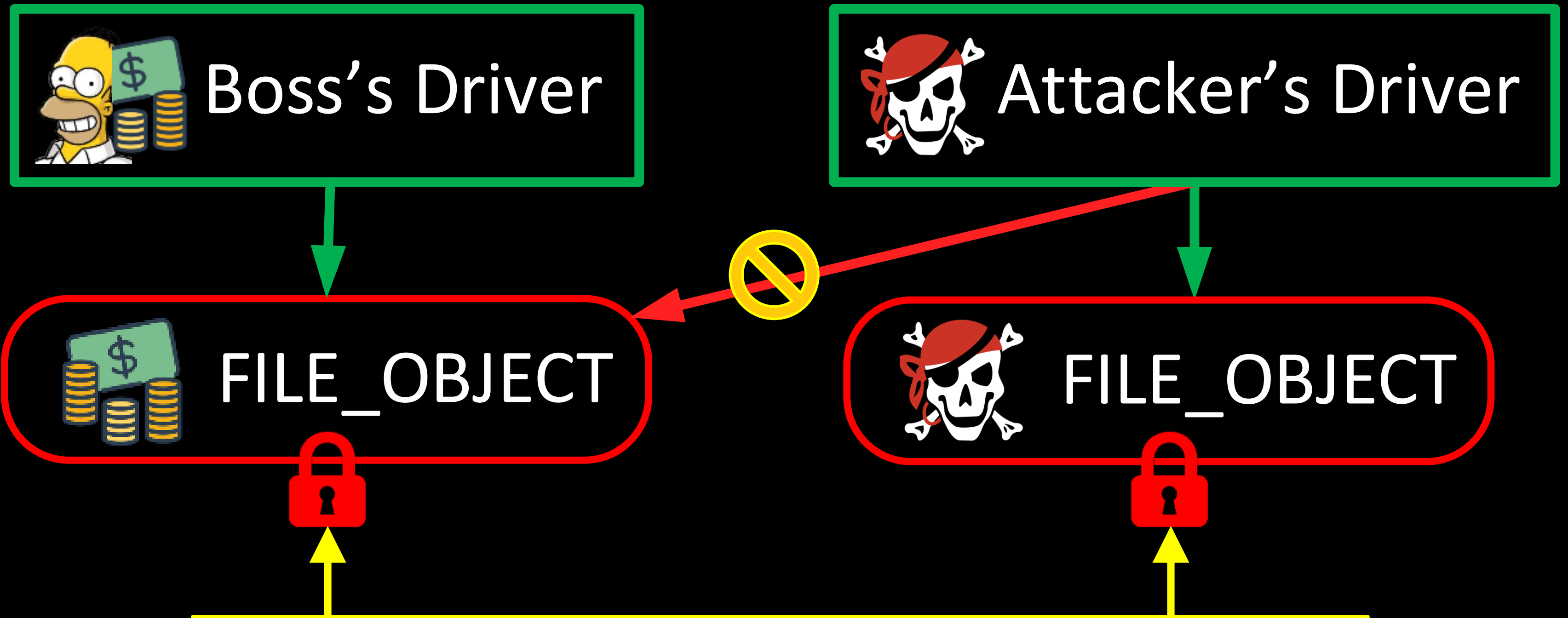


THE FILE_OBJECT PROTECTION VIA ENCRYPTION



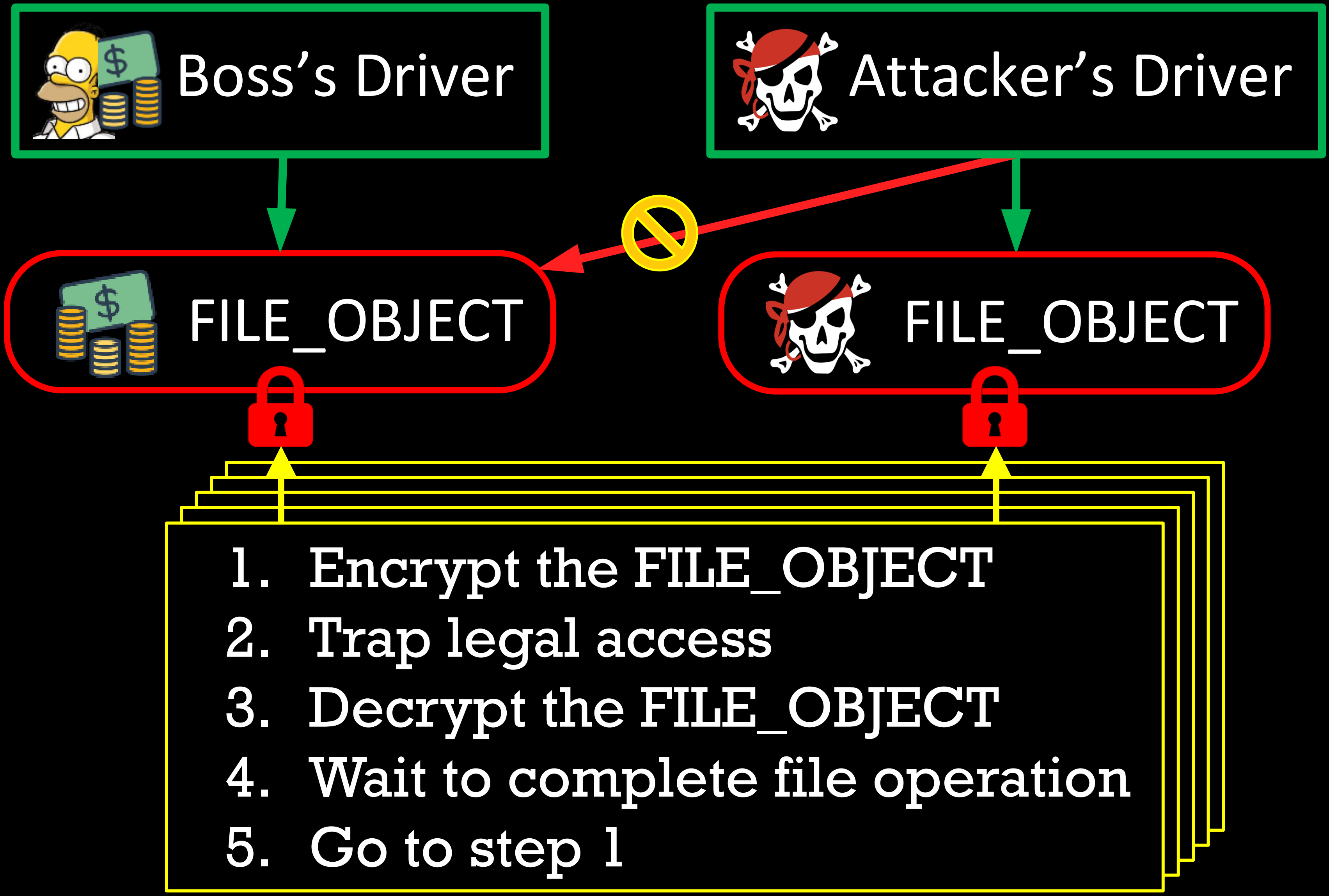
1. Encrypt the FILE_OBJECT
2. Trap legal access
3. Decrypt the FILE_OBJECT
4. Wait to complete file operation
5.

THE FILE_OBJECT PROTECTION VIA ENCRYPTION

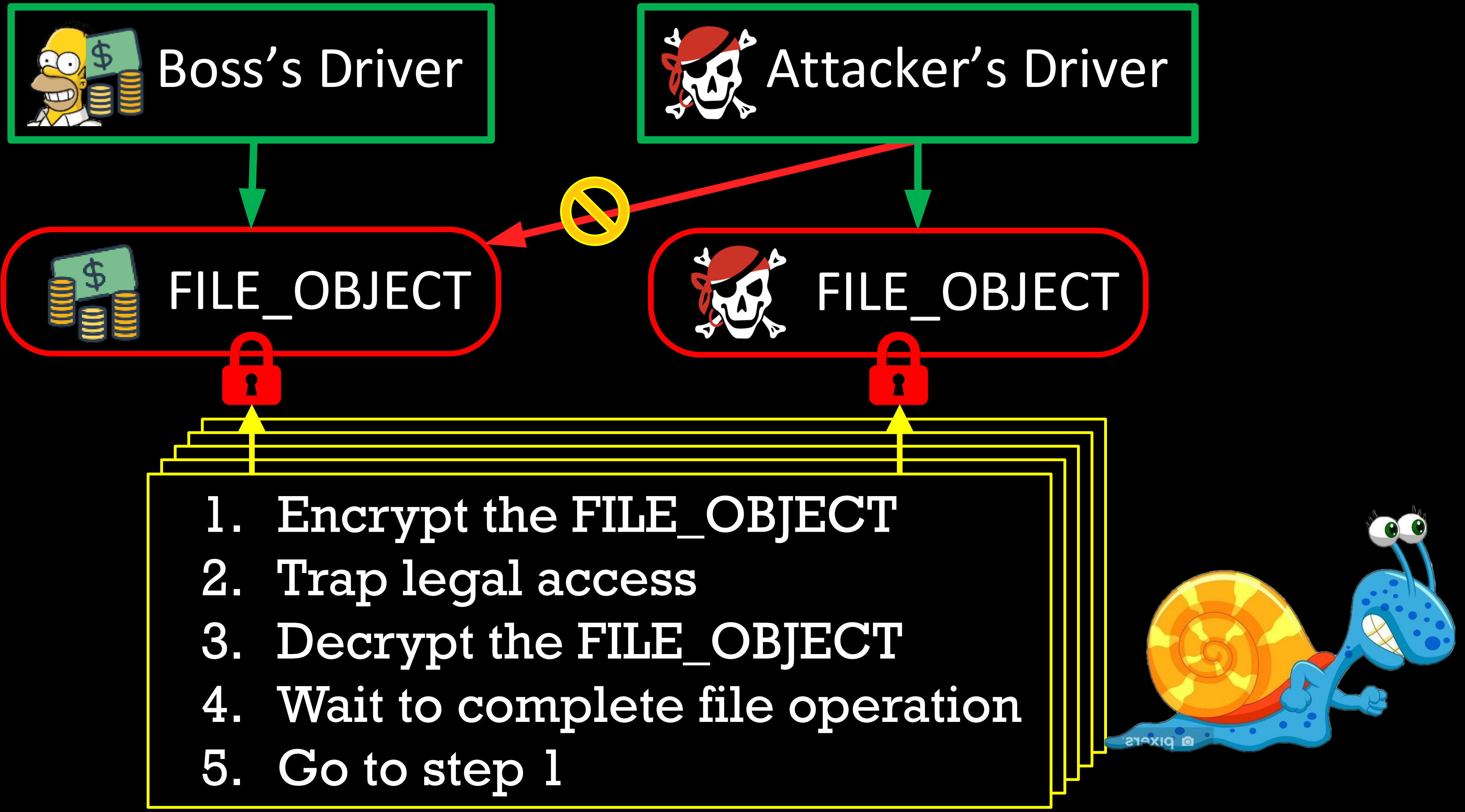


1. Encrypt the FILE_OBJECT
2. Trap legal access
3. Decrypt the FILE_OBJECT
4. Wait to complete file operation
5. Go to step 1

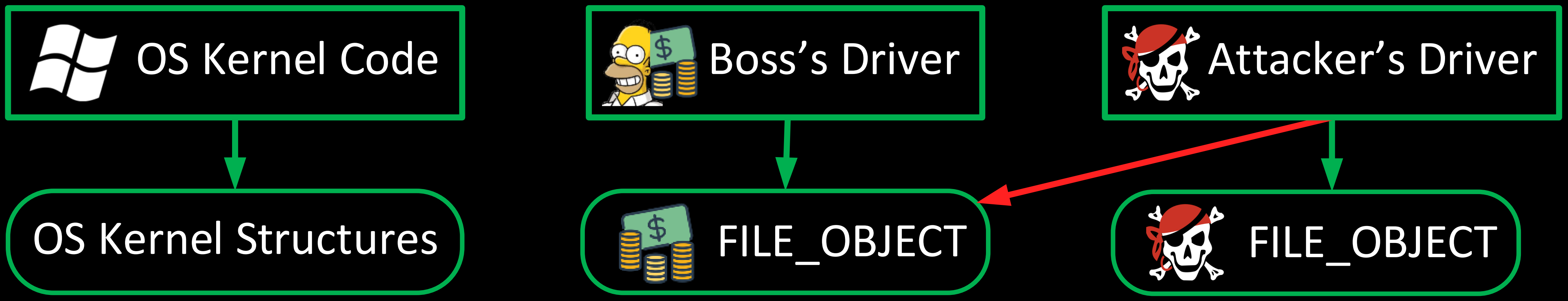
THE FILE_OBJECT PROTECTION VIA ENCRYPTION



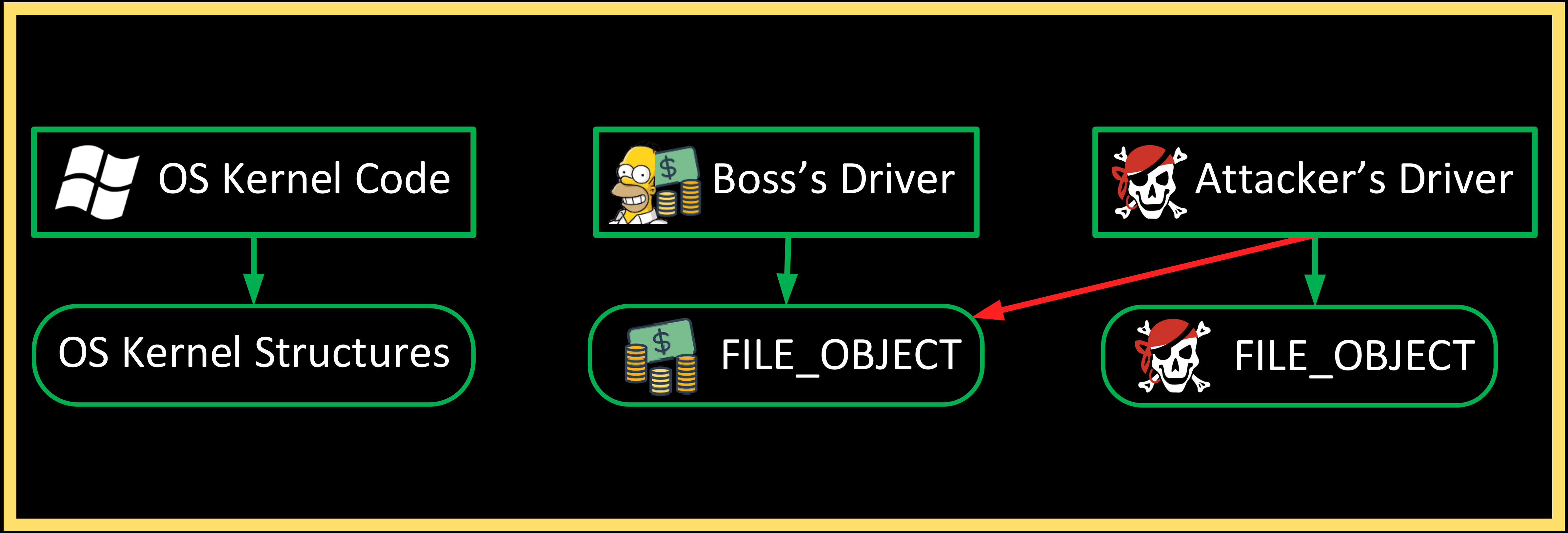
THE FILE_OBJECT PROTECTION VIA ENCRYPTION



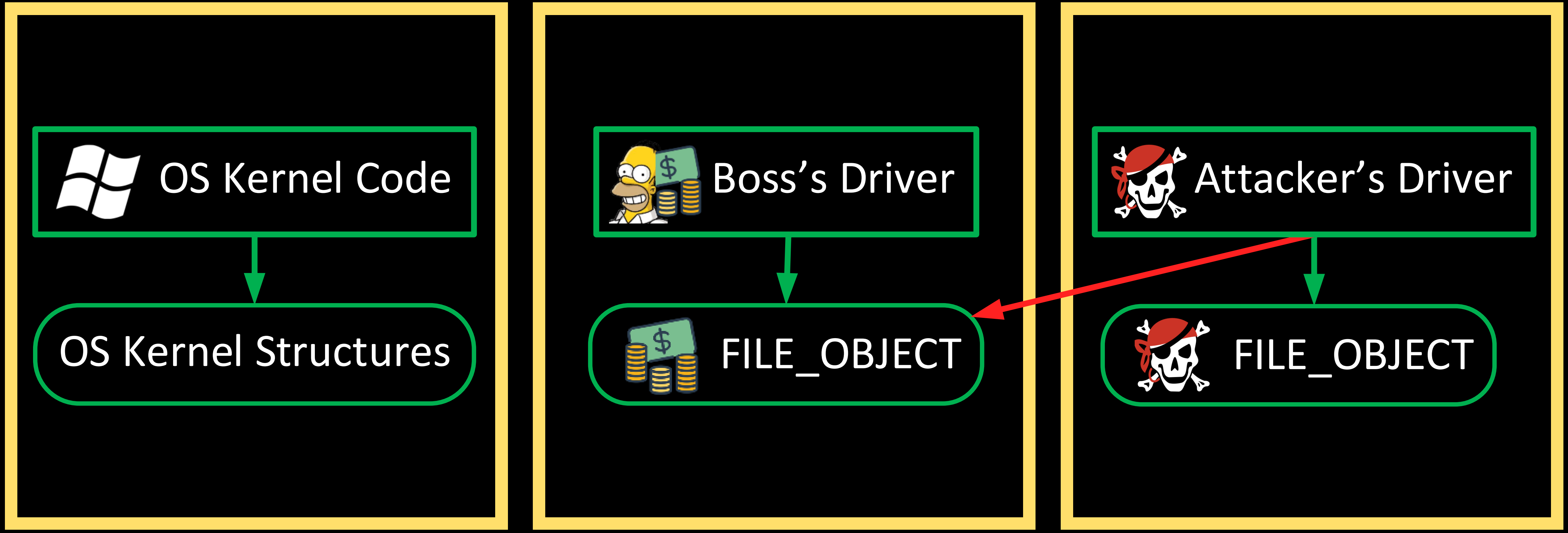
WINDOWS KERNEL MEMORY



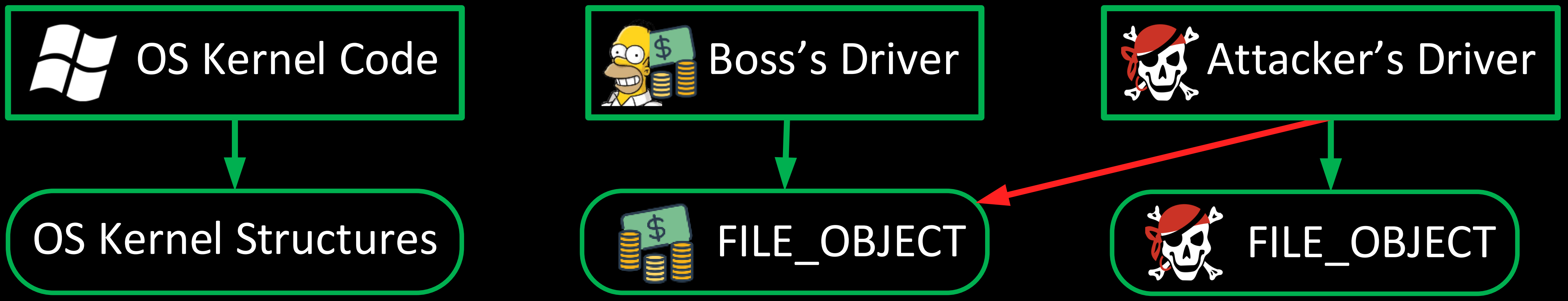
WINDOWS KERNEL MEMORY



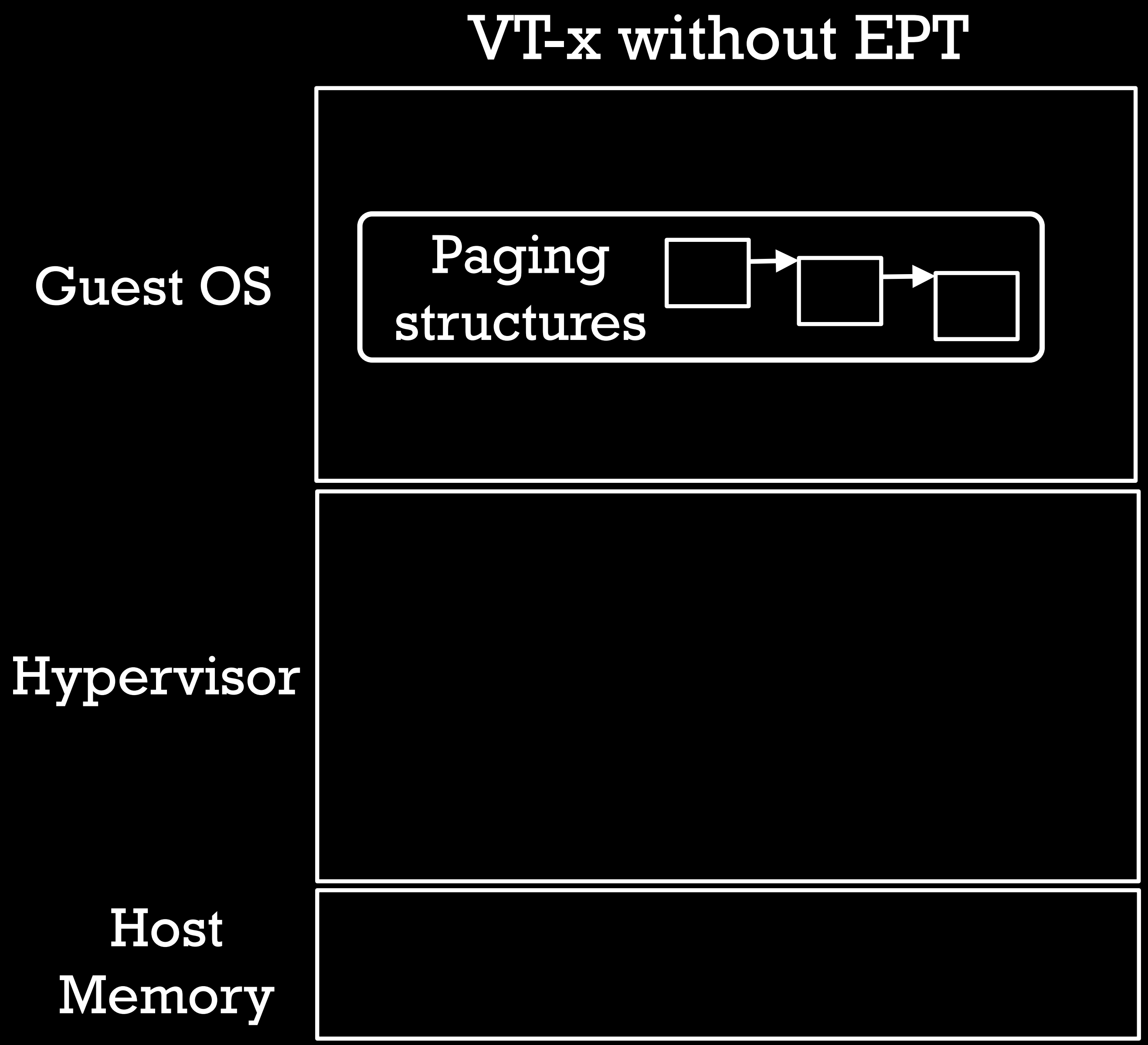
WINDOWS KERNEL MEMORY



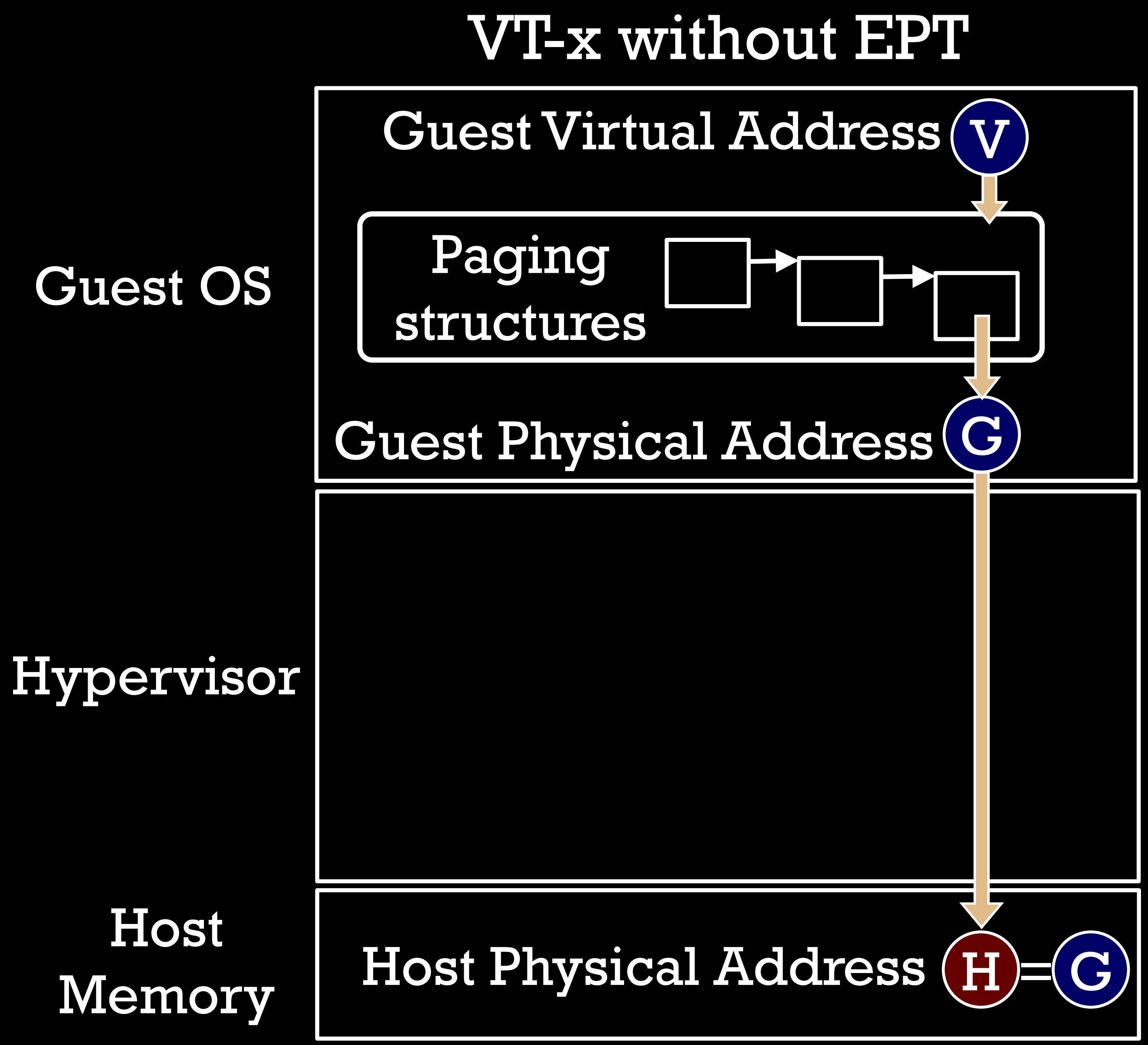
WINDOWS KERNEL MEMORY



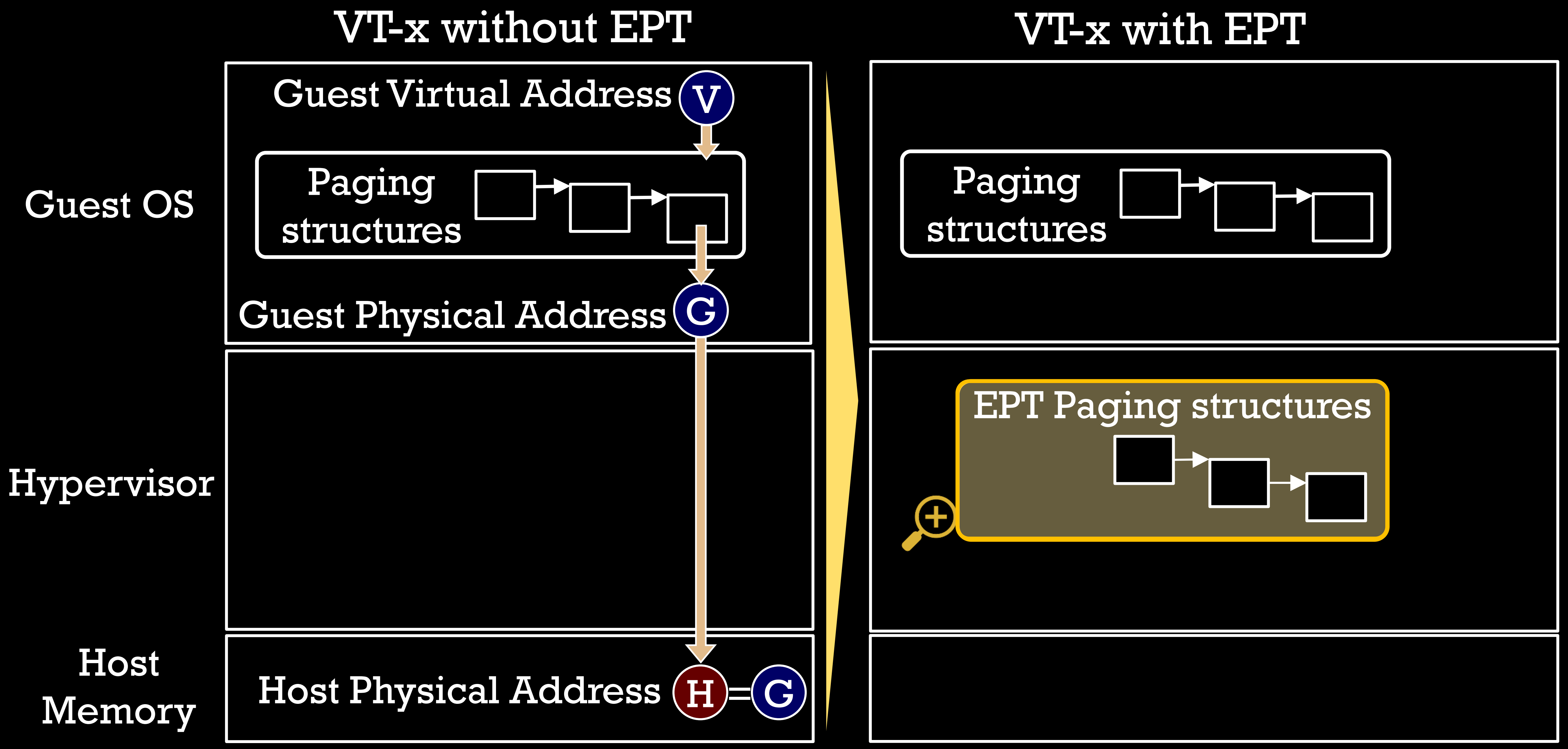
PROCESSING MEMORY ACCESS: EPT FEATURE



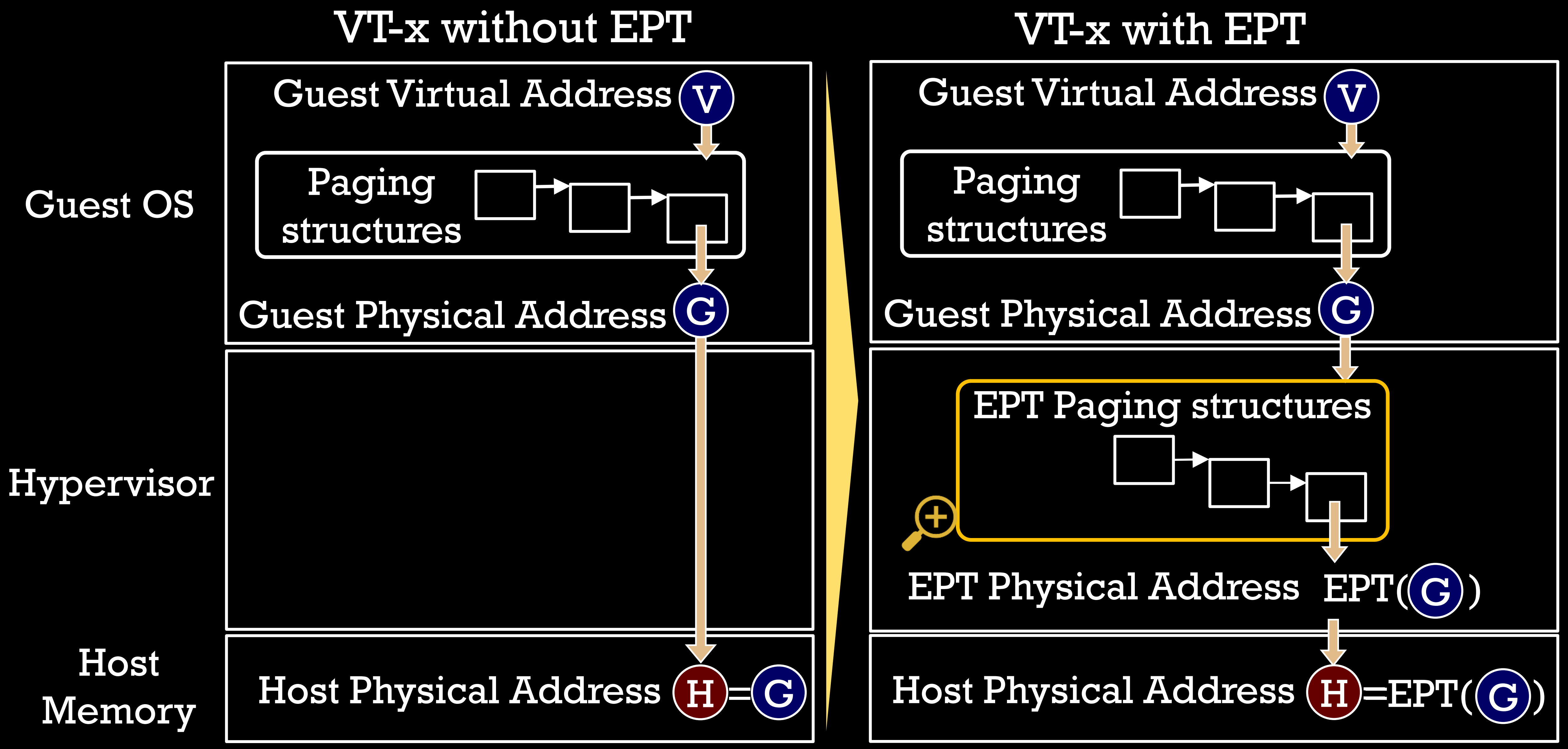
PROCESSING MEMORY ACCESS: EPT FEATURE



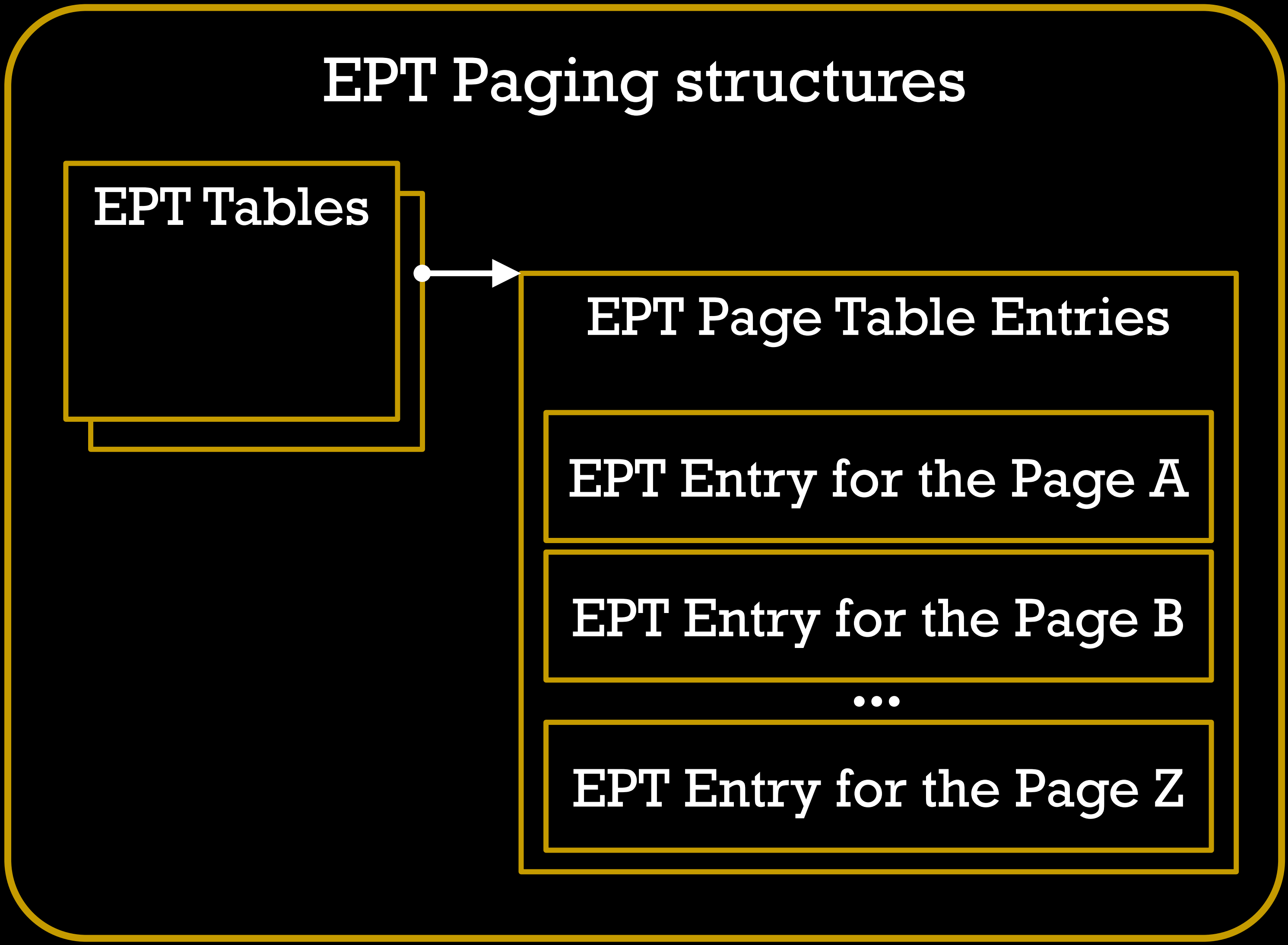
PROCESSING MEMORY ACCESS: EPT FEATURE



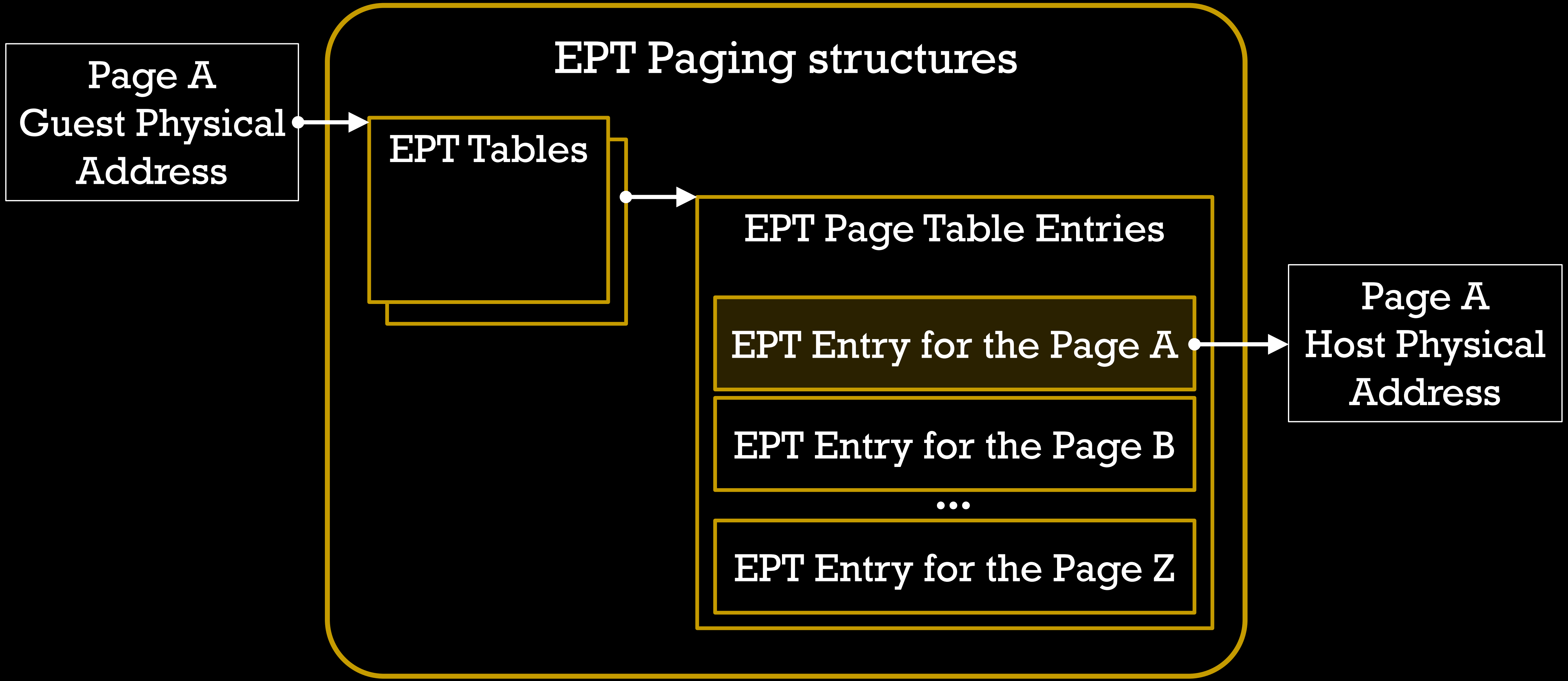
PROCESSING MEMORY ACCESS: EPT FEATURE



EPT PAGING STRUCTURES



EPT PAGING STRUCTURES



EPT MAIN FEATURES

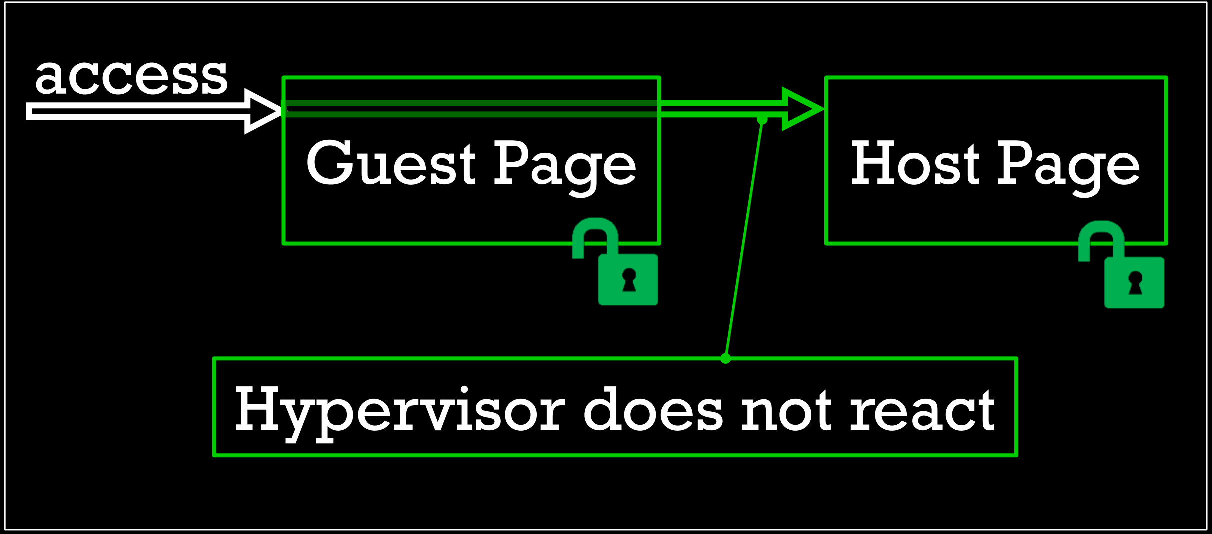
1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

2.

3.

EPT MAIN FEATURES

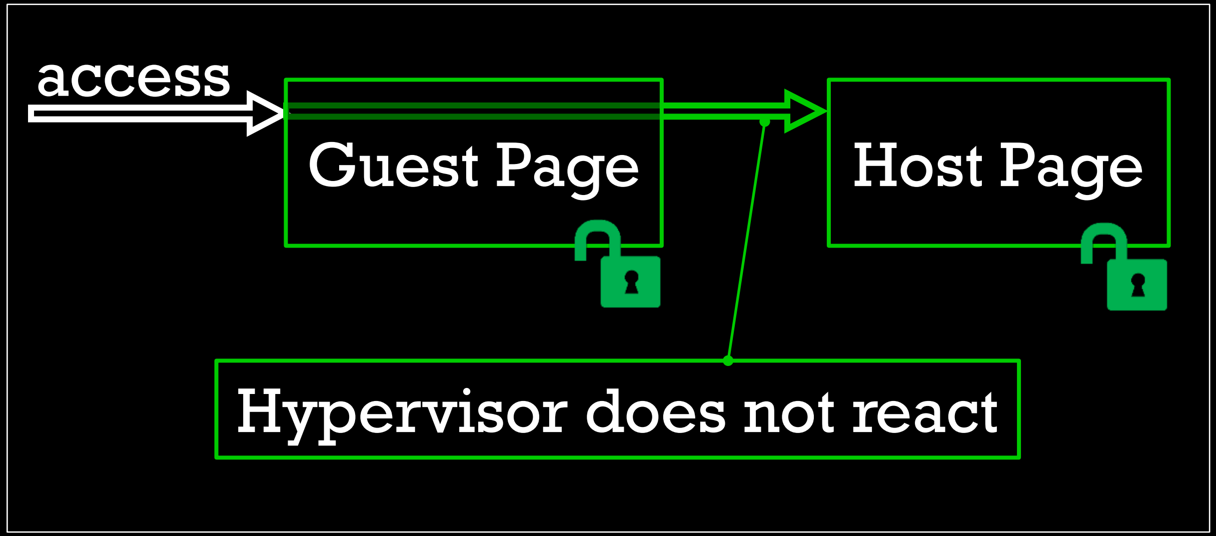
1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

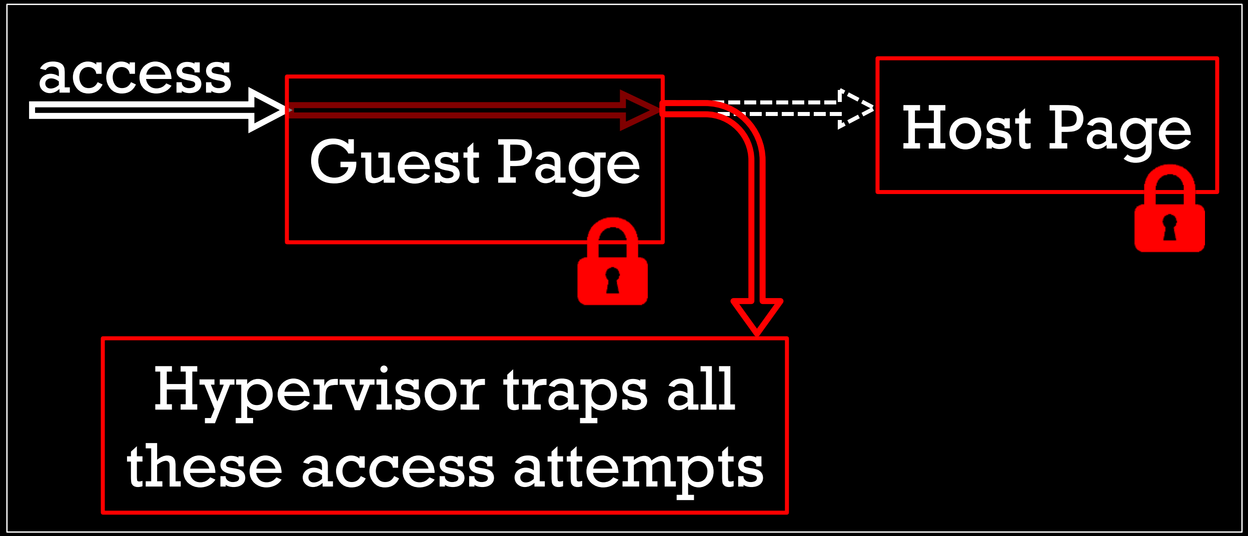


2.
3.

EPT MAIN FEATURES

1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:



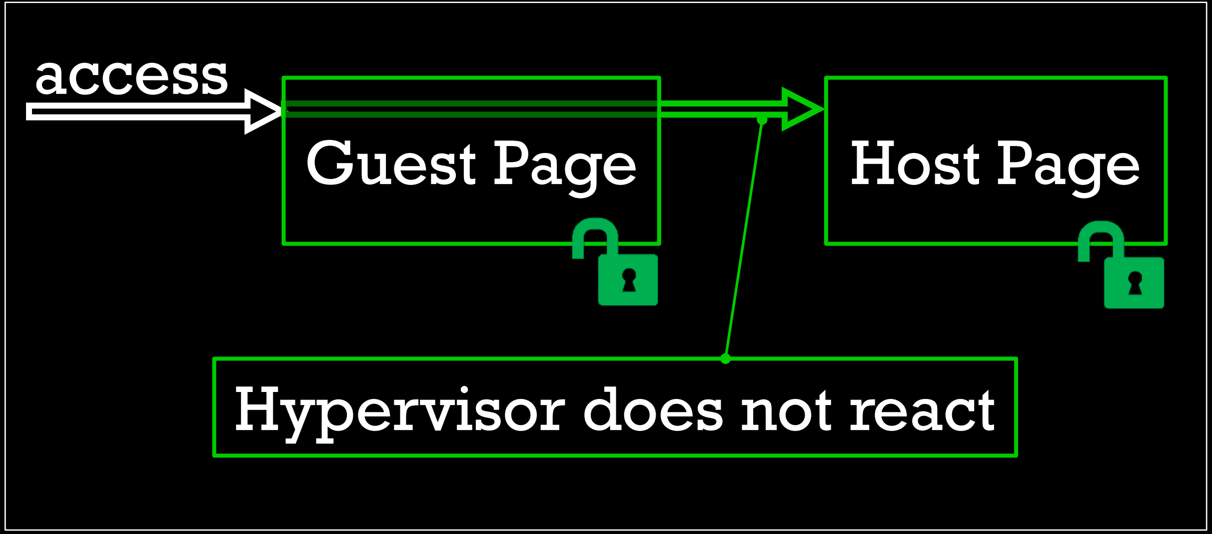


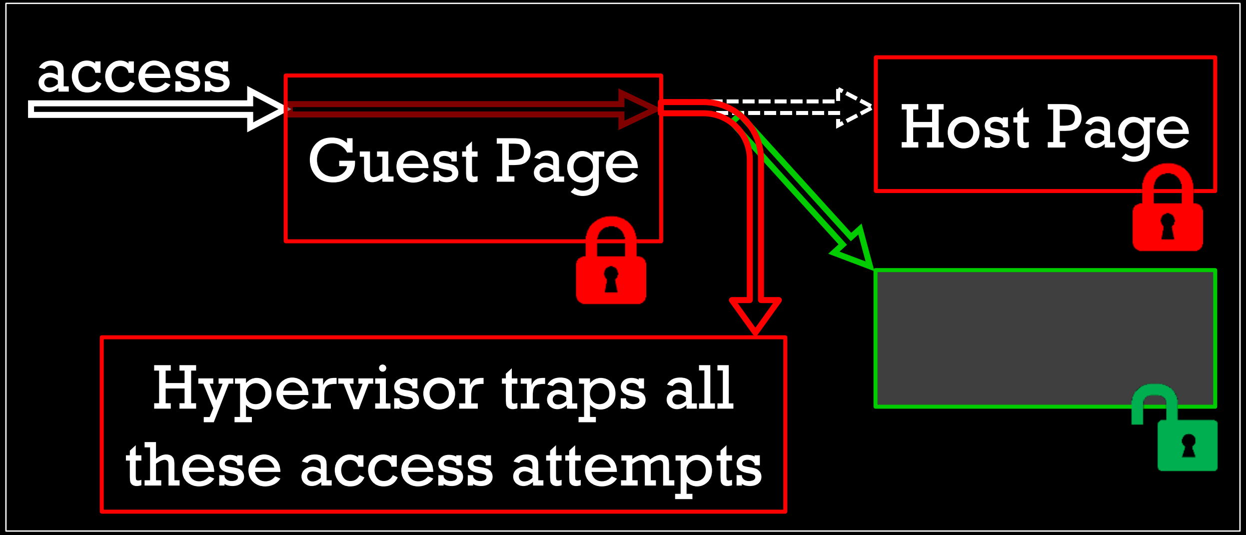
2.

3.

EPT MAIN FEATURES

1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

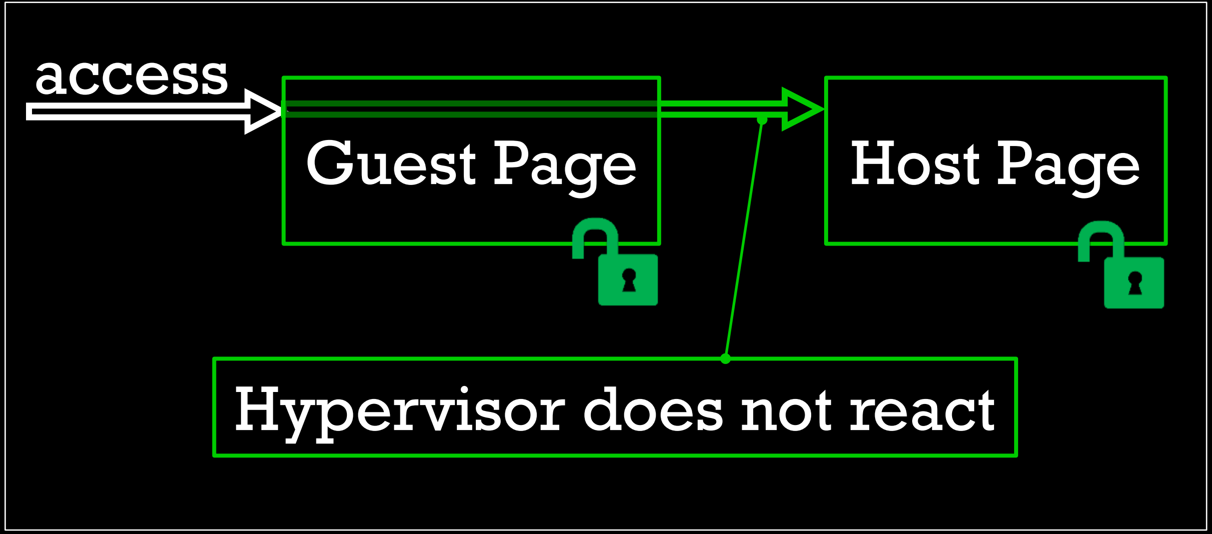


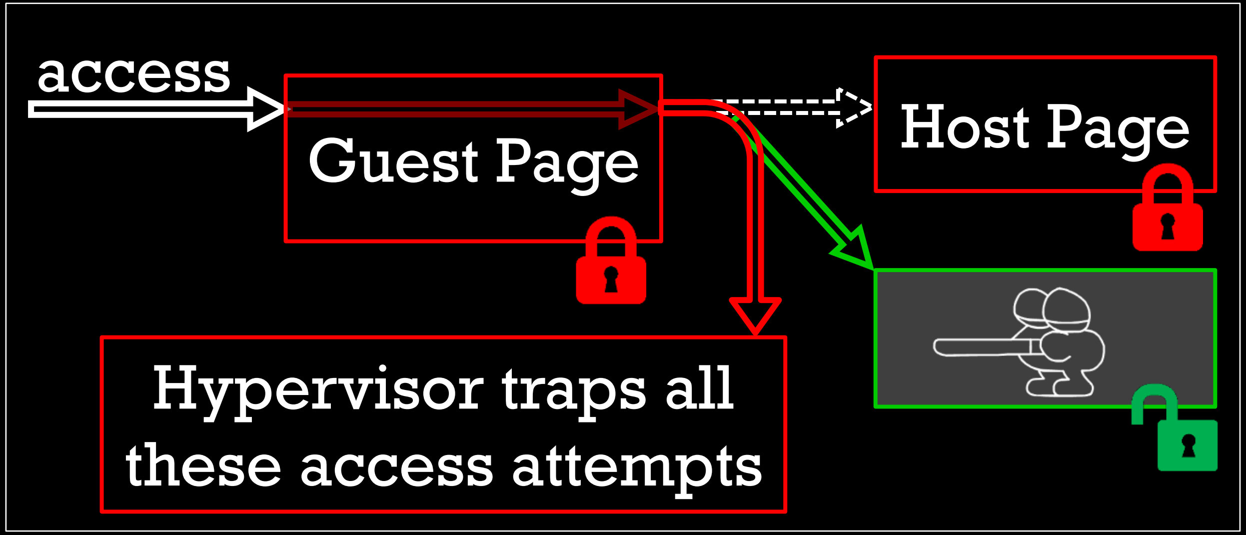


2.
3.

EPT MAIN FEATURES

1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

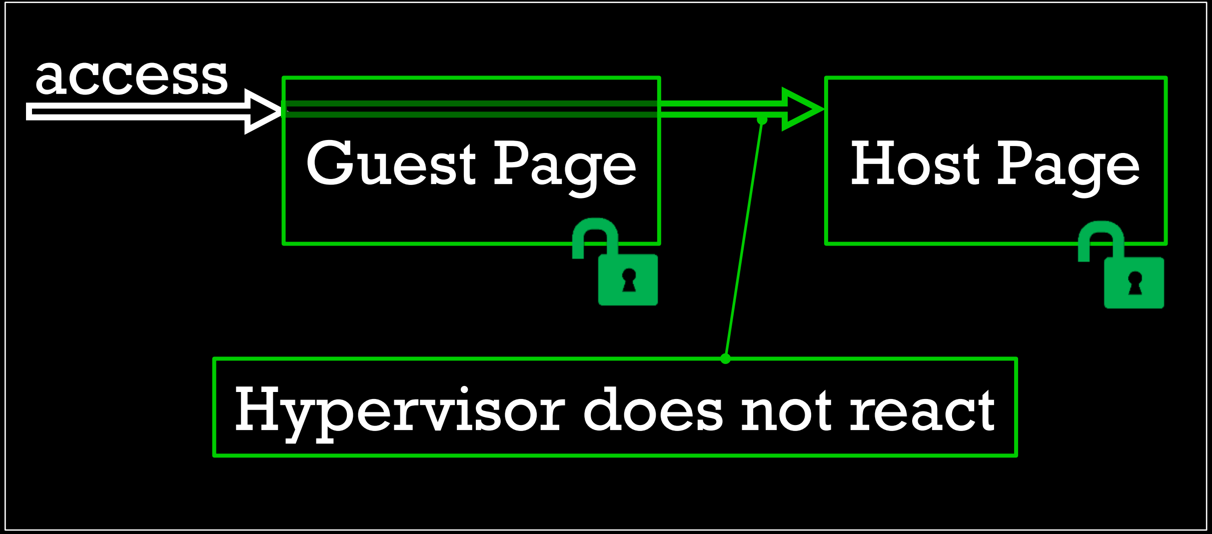


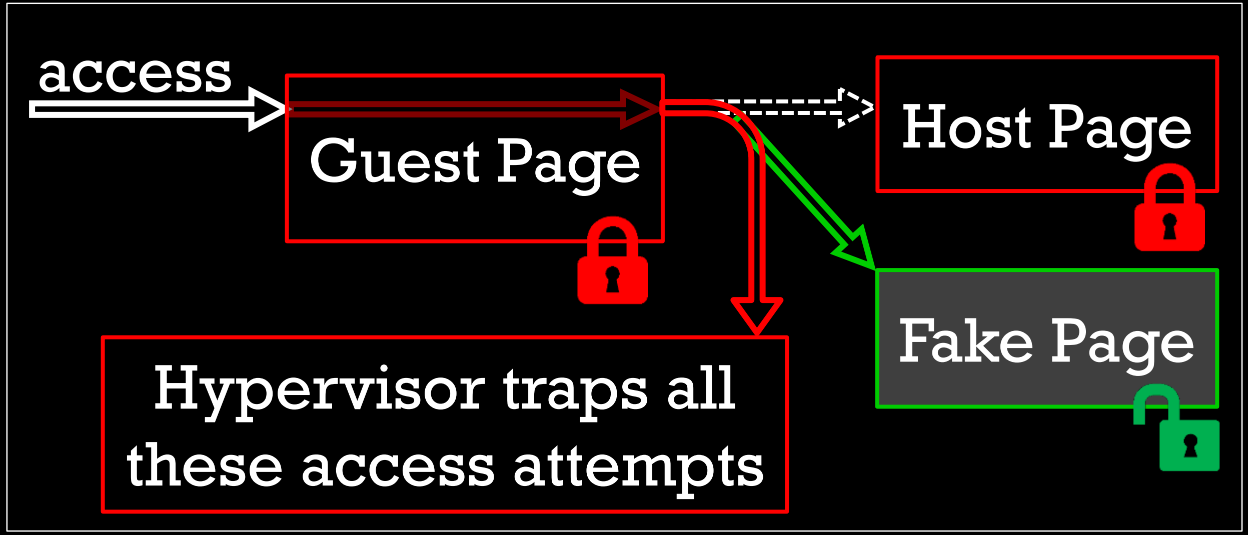


2.
3.

EPT MAIN FEATURES

1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

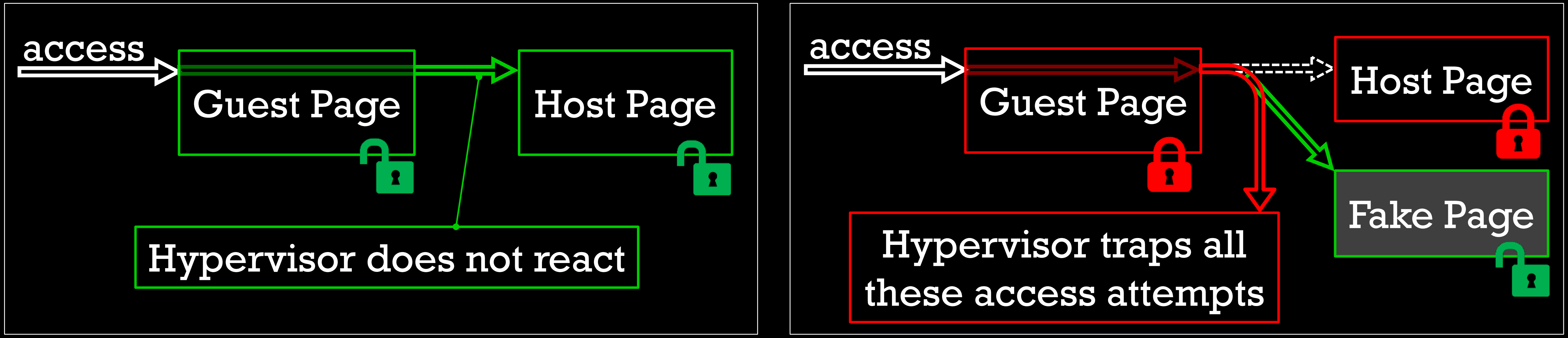




2. EPT memory settings can be updated in the real time
3.

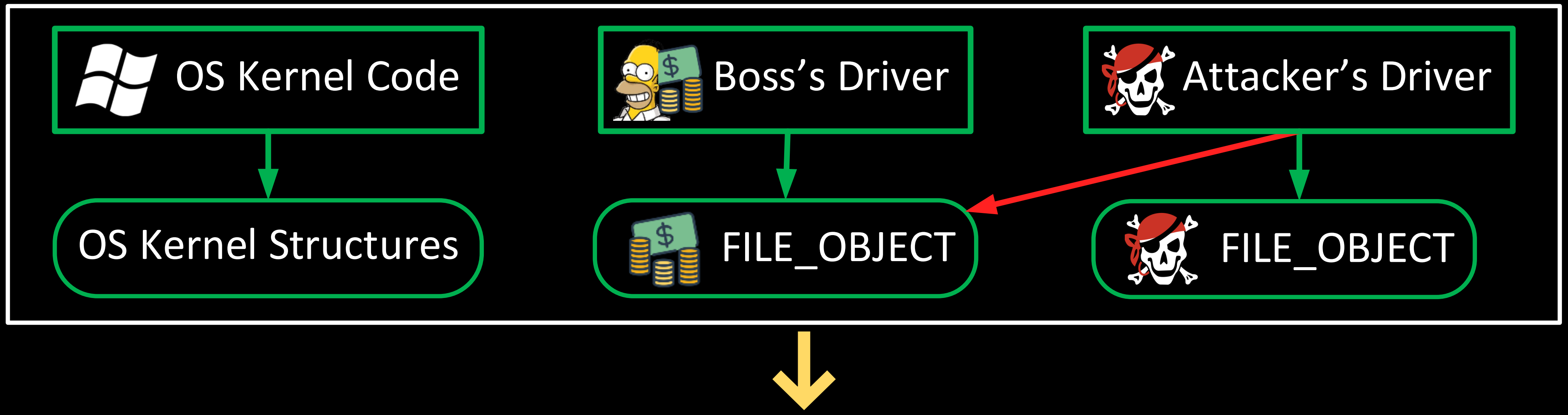
EPT MAIN FEATURES

1. Using EPT we can trap read/write/execute access attempts and redirect them from the secret page to the fake one:

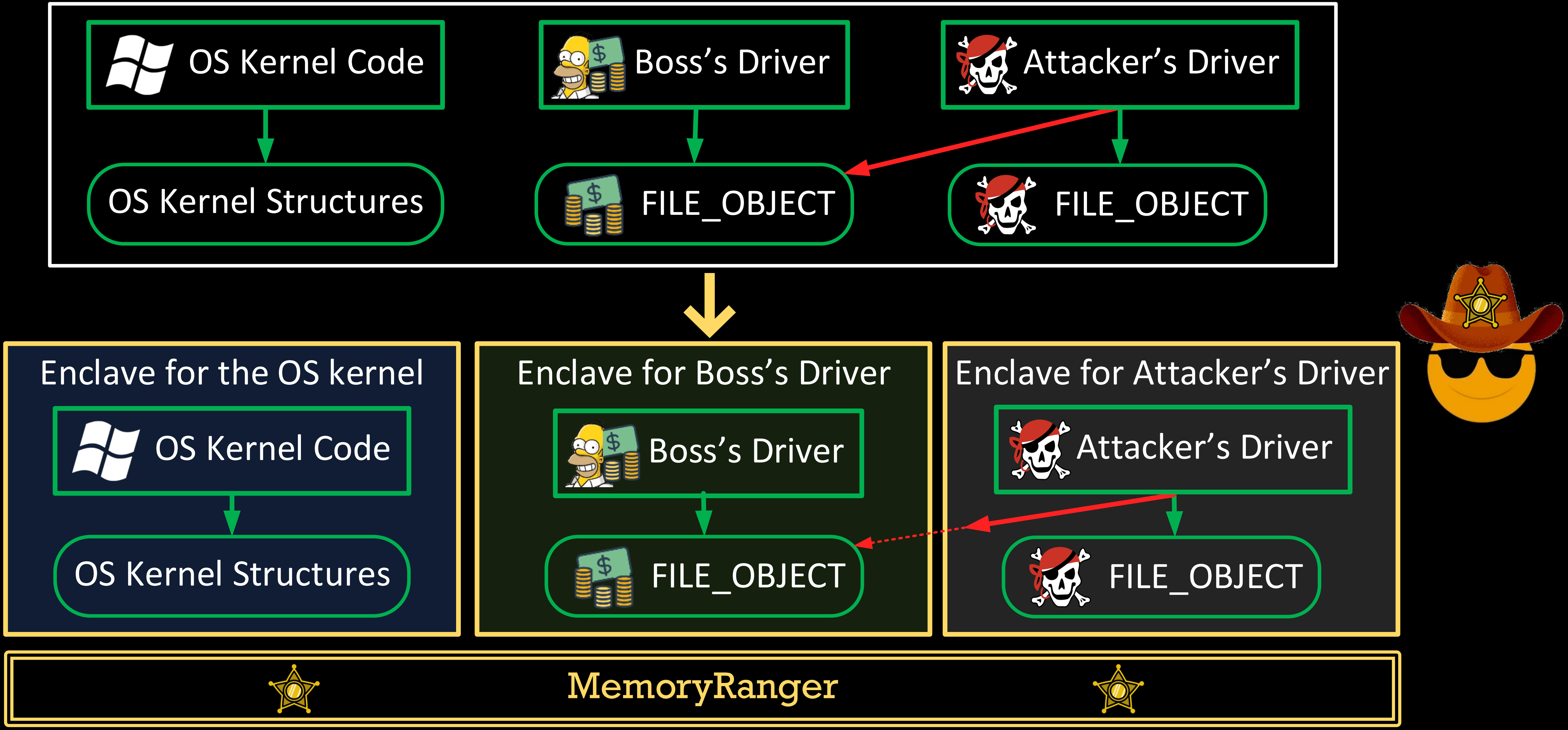


2. EPT memory settings can be updated in the real time
3. We can dynamically allocate several EPTs with different memory setting and switch between them in the real time

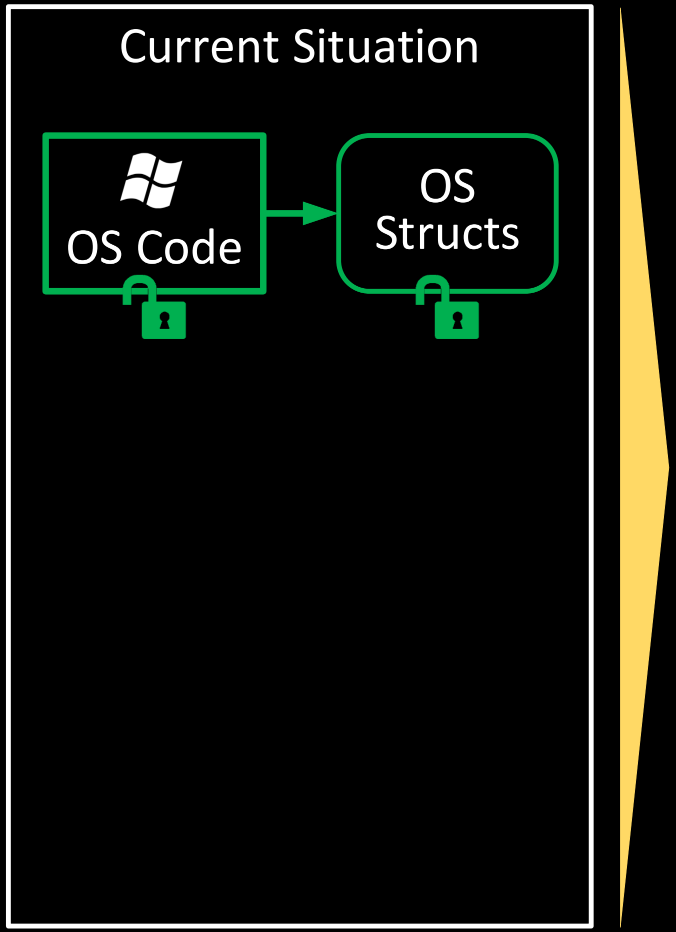
WINDOWS KERNEL MEMORY



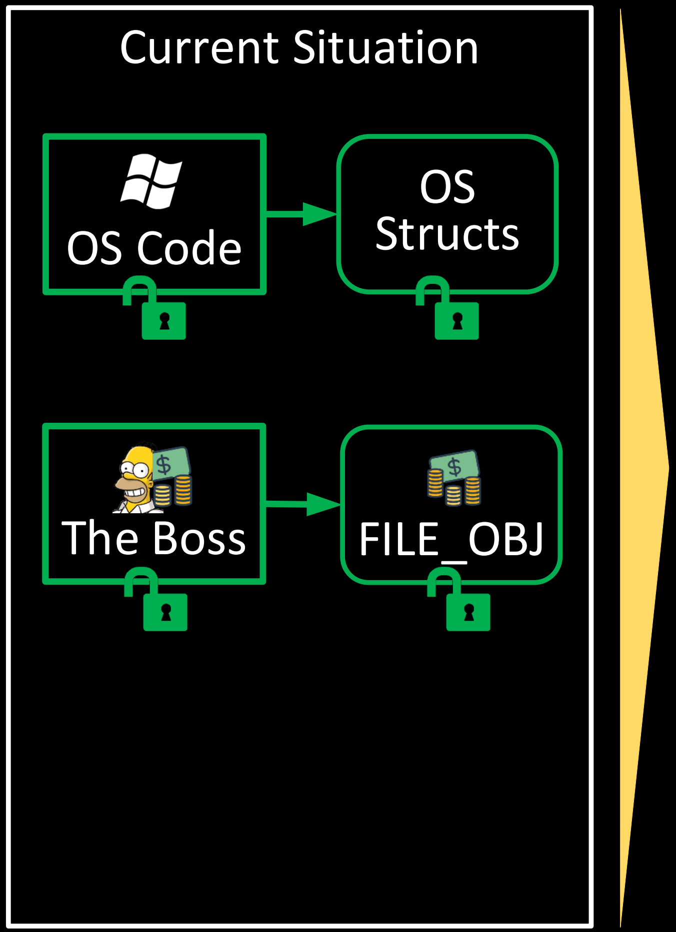
WINDOWS KERNEL MEMORY



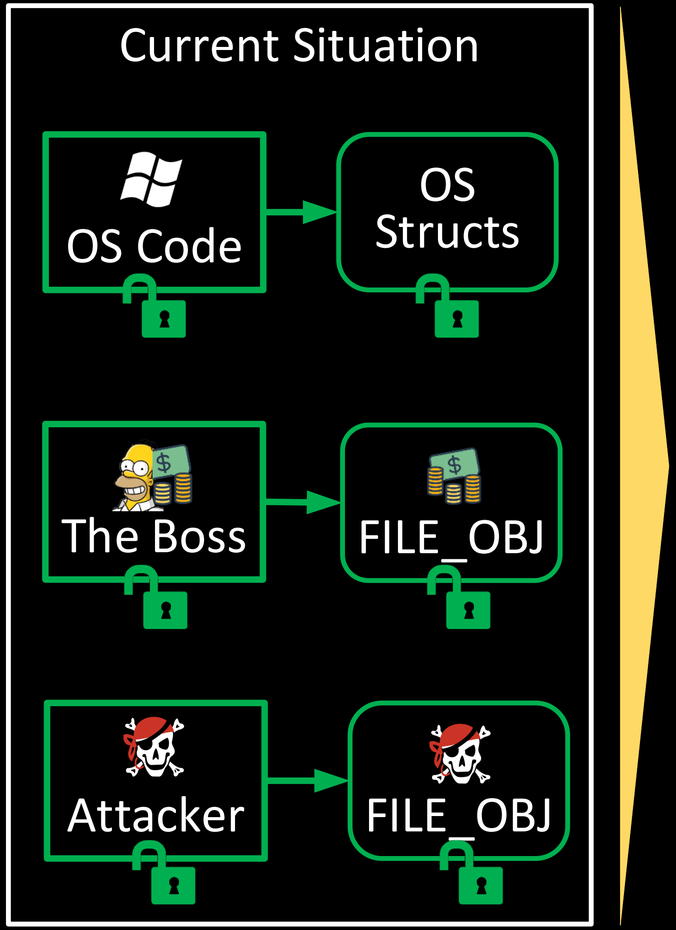
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



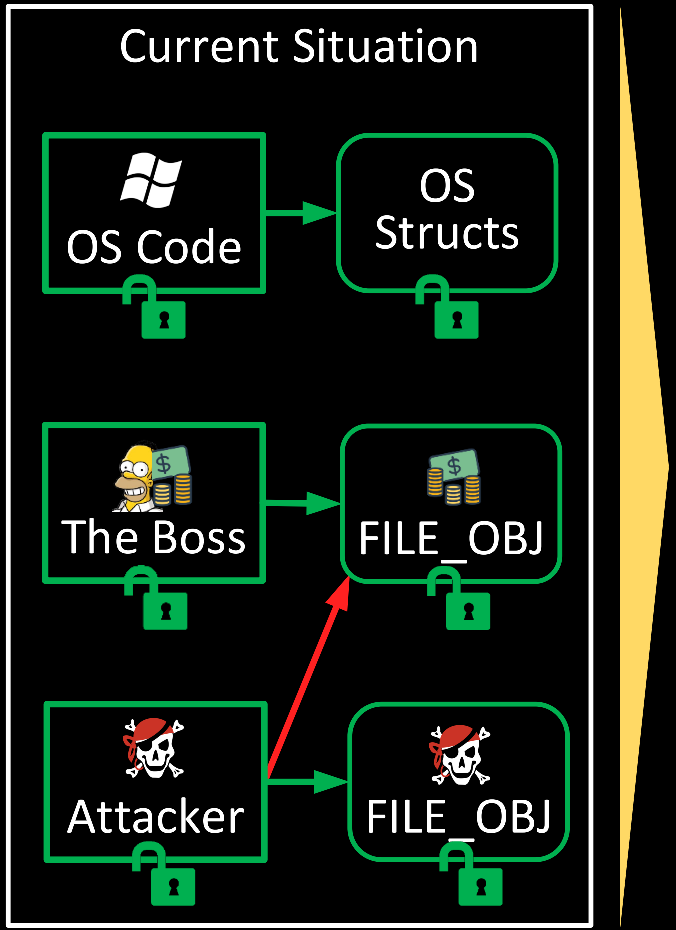
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



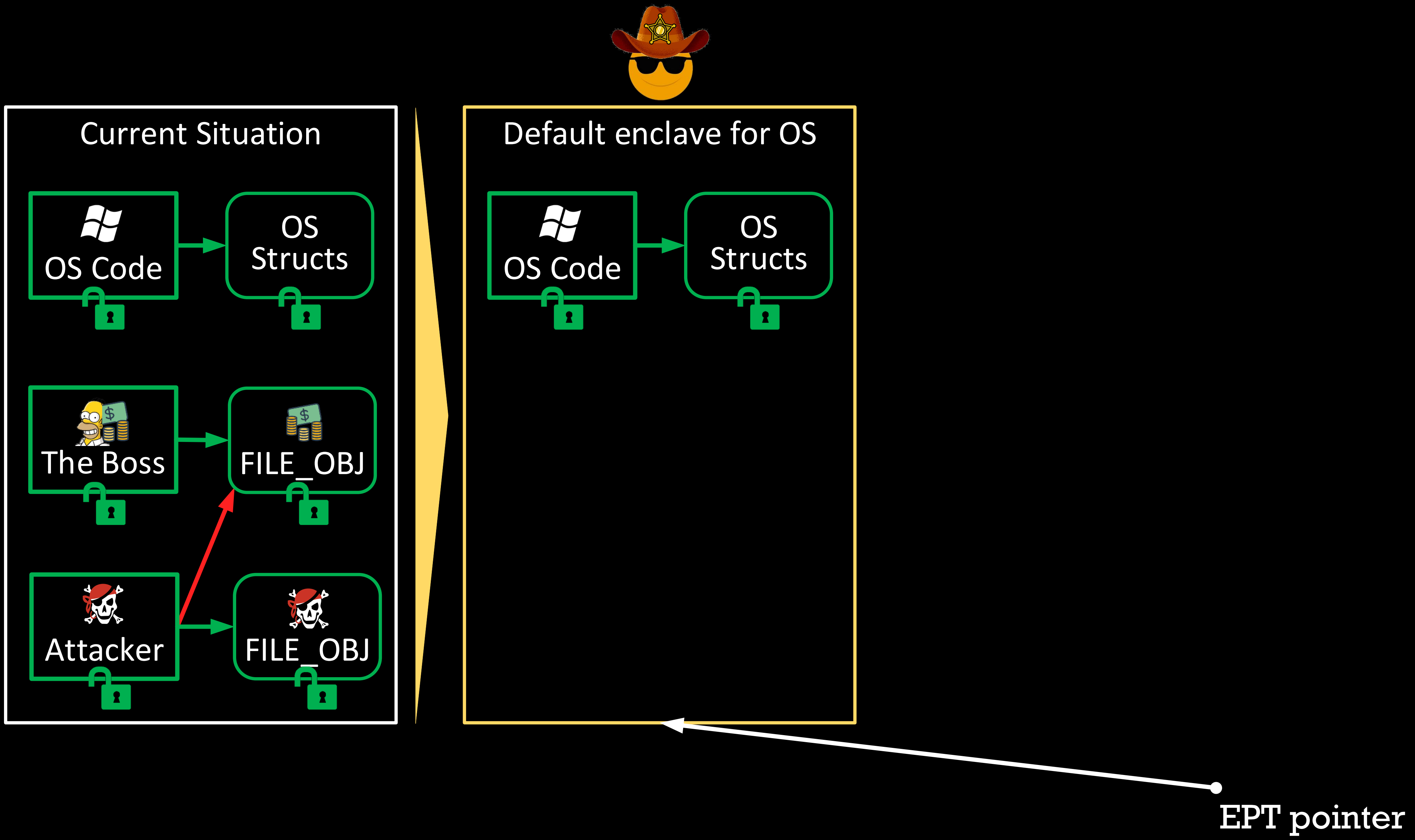
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



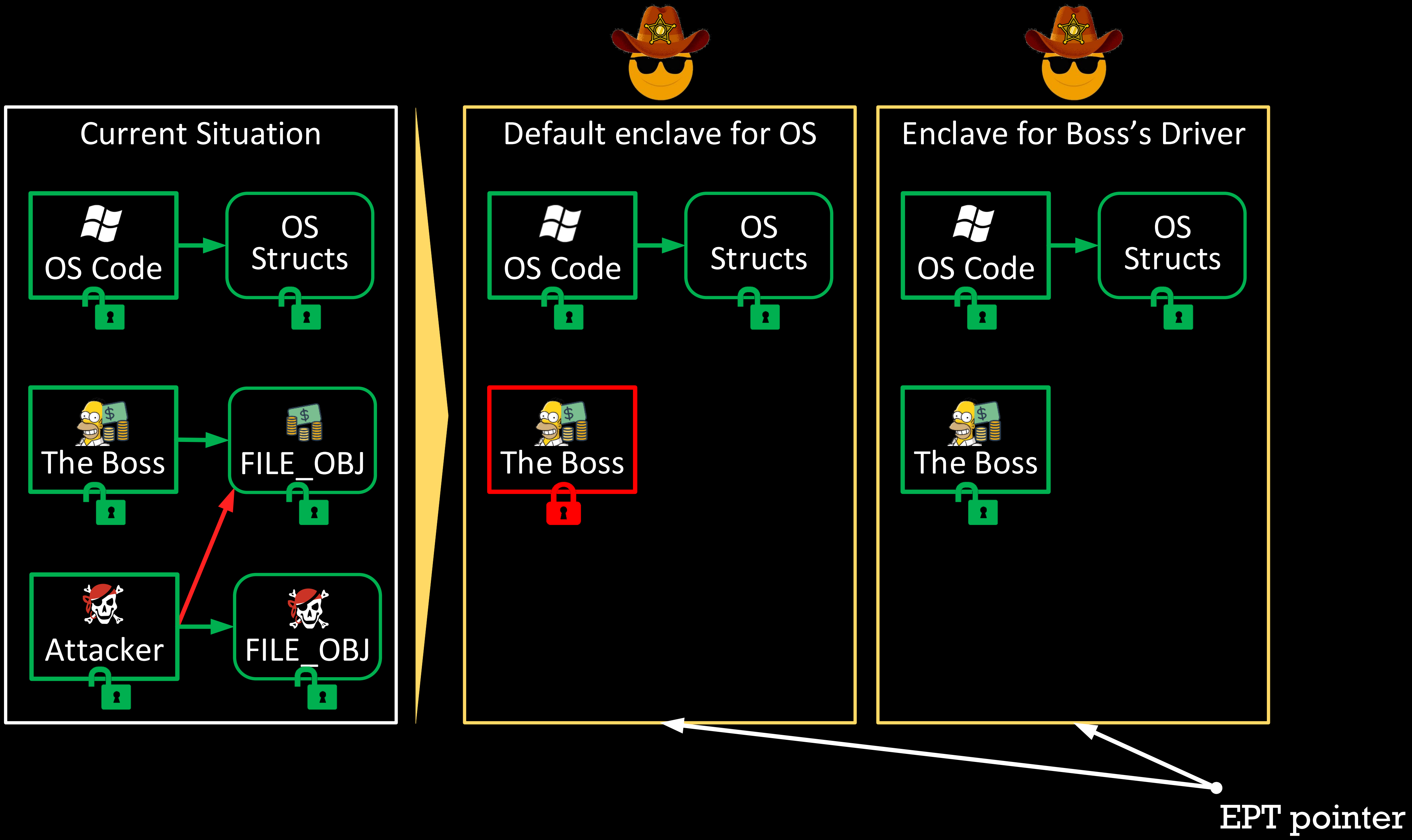
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



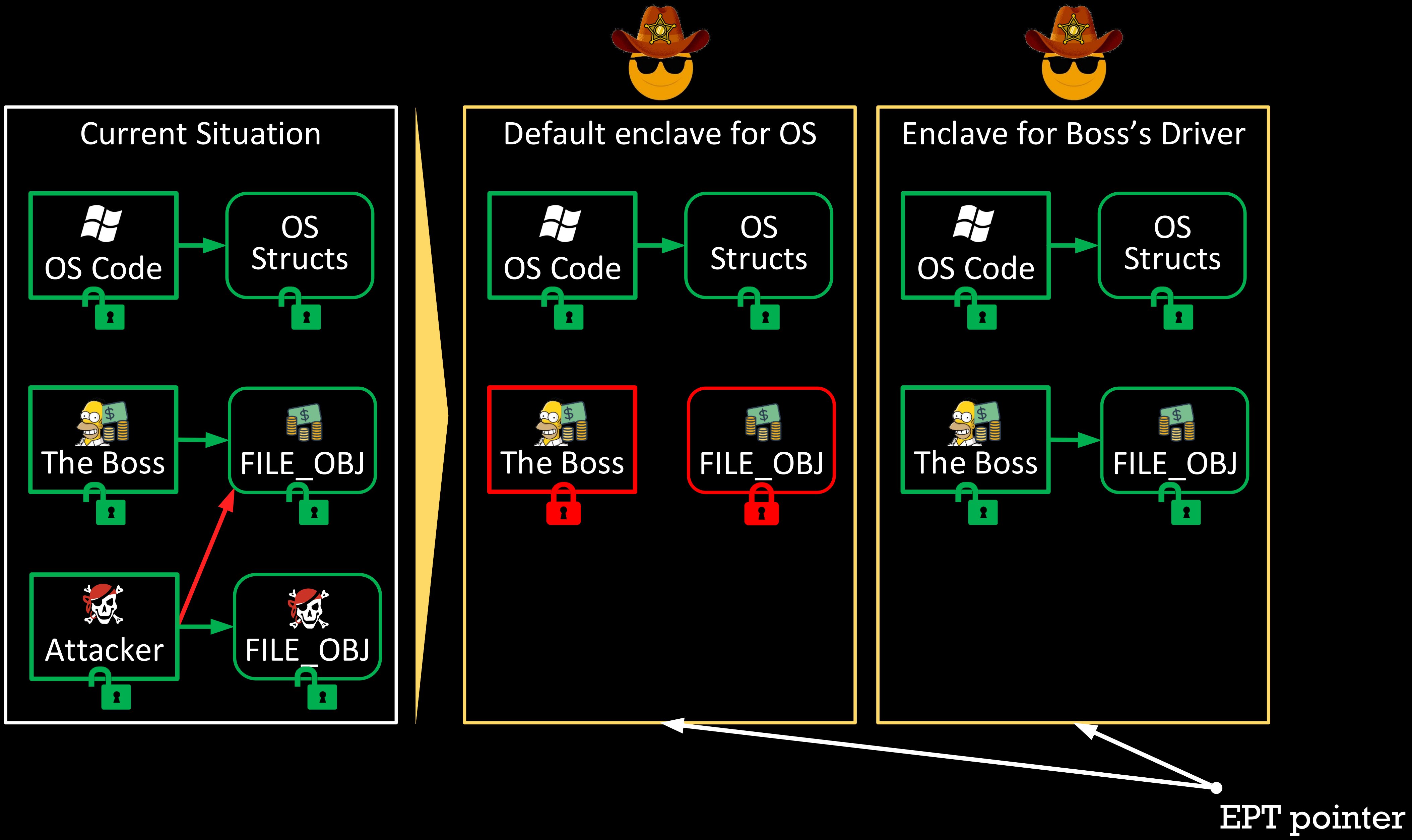
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



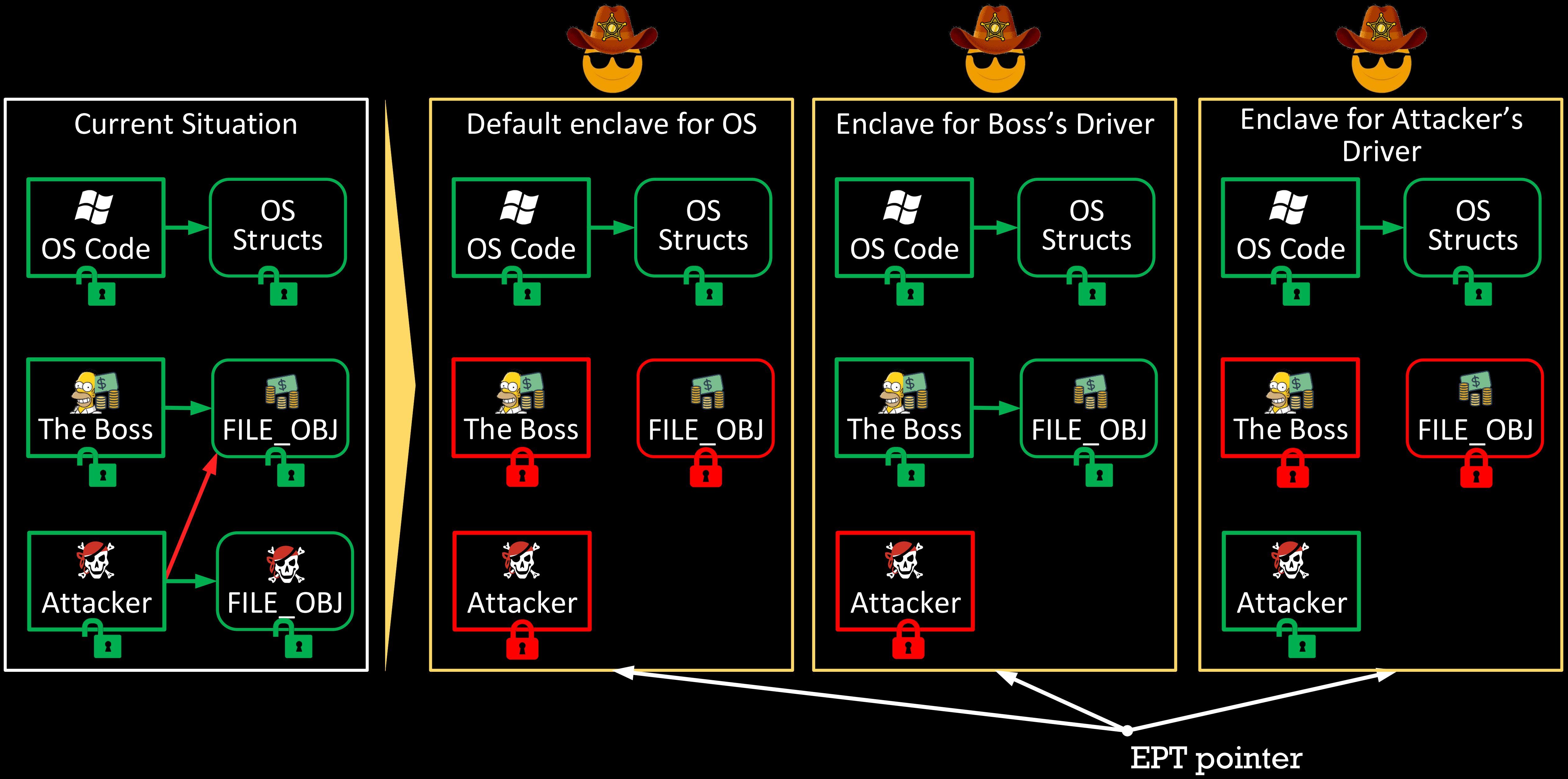
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



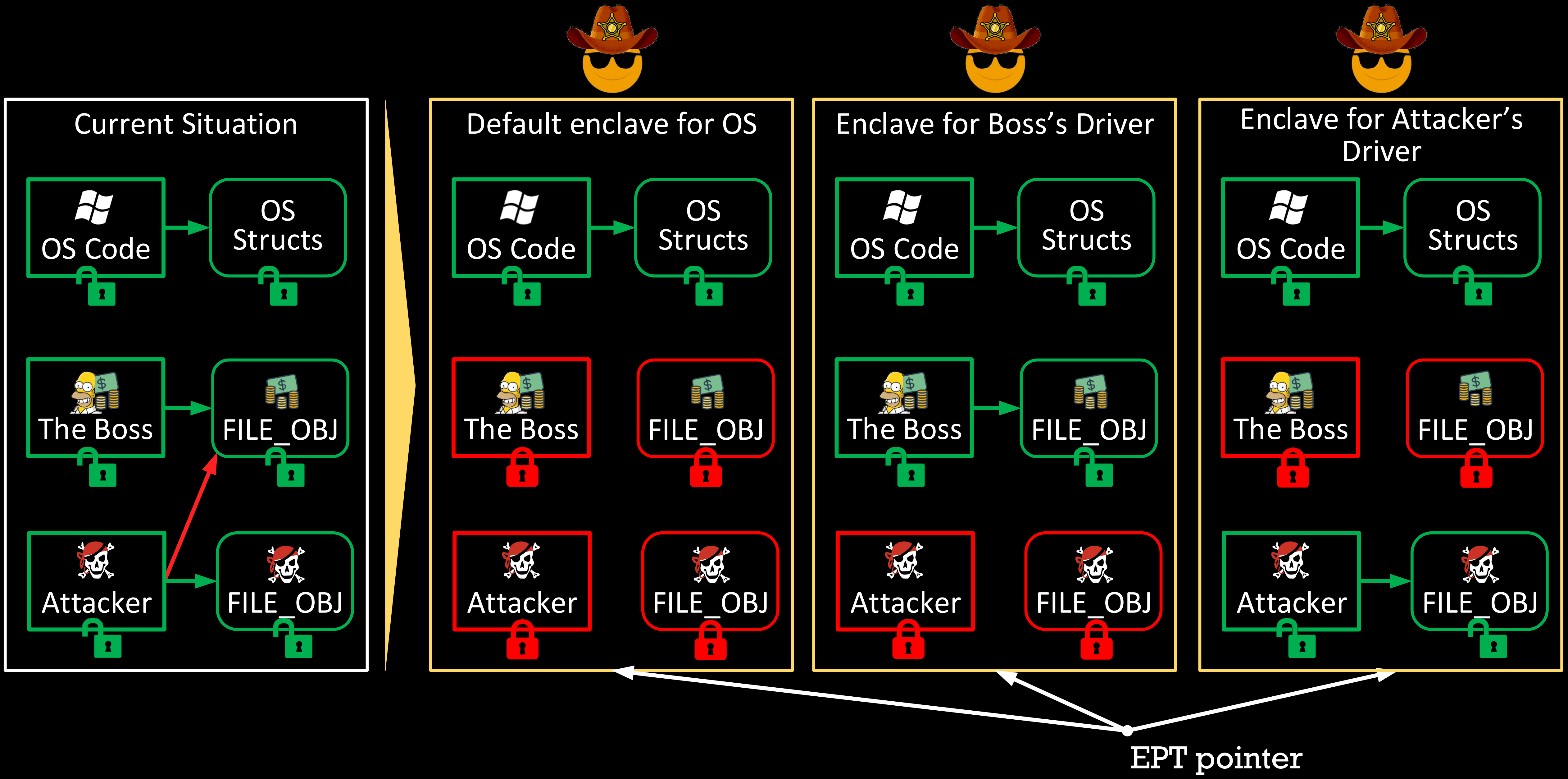
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



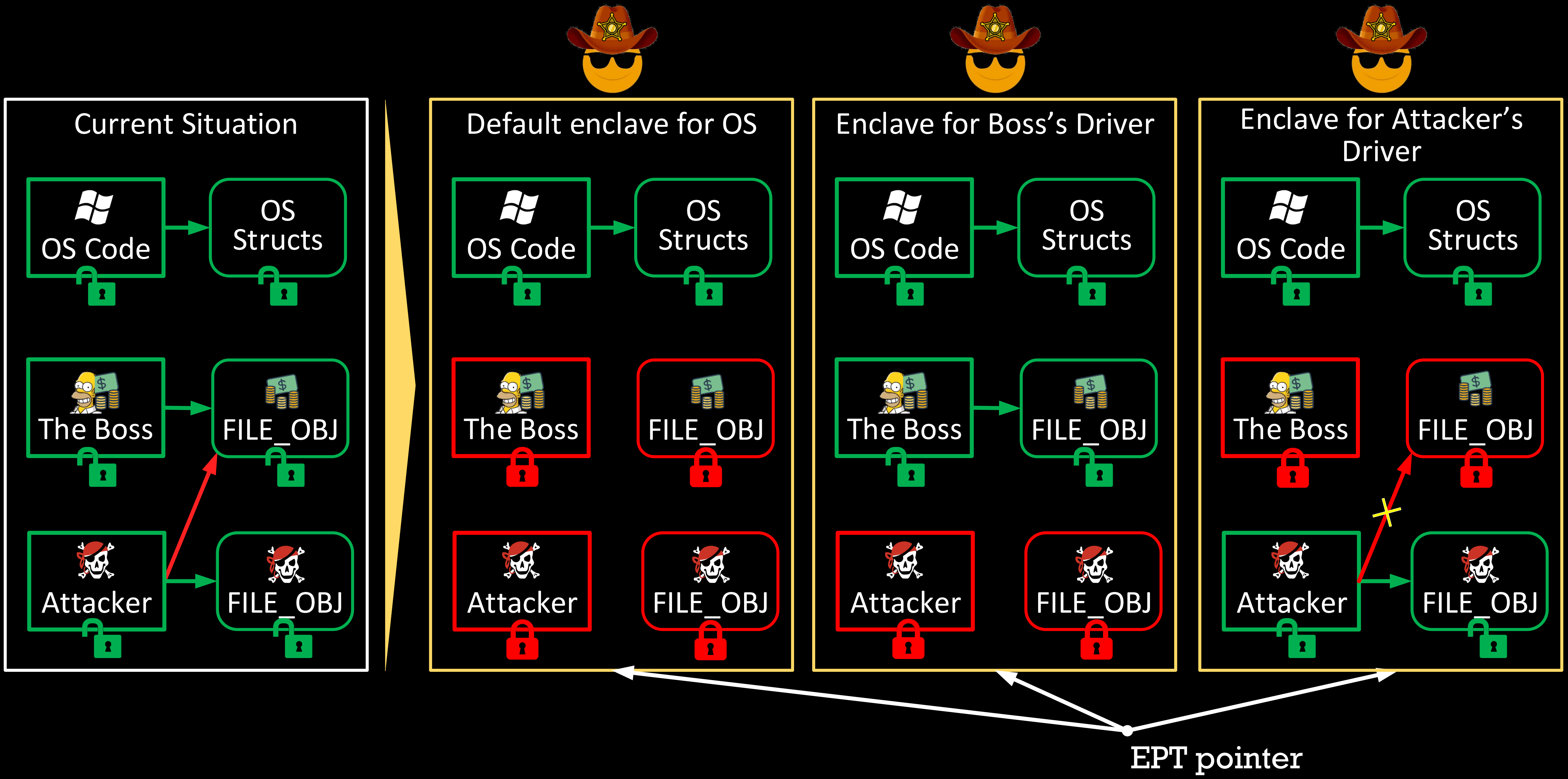
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



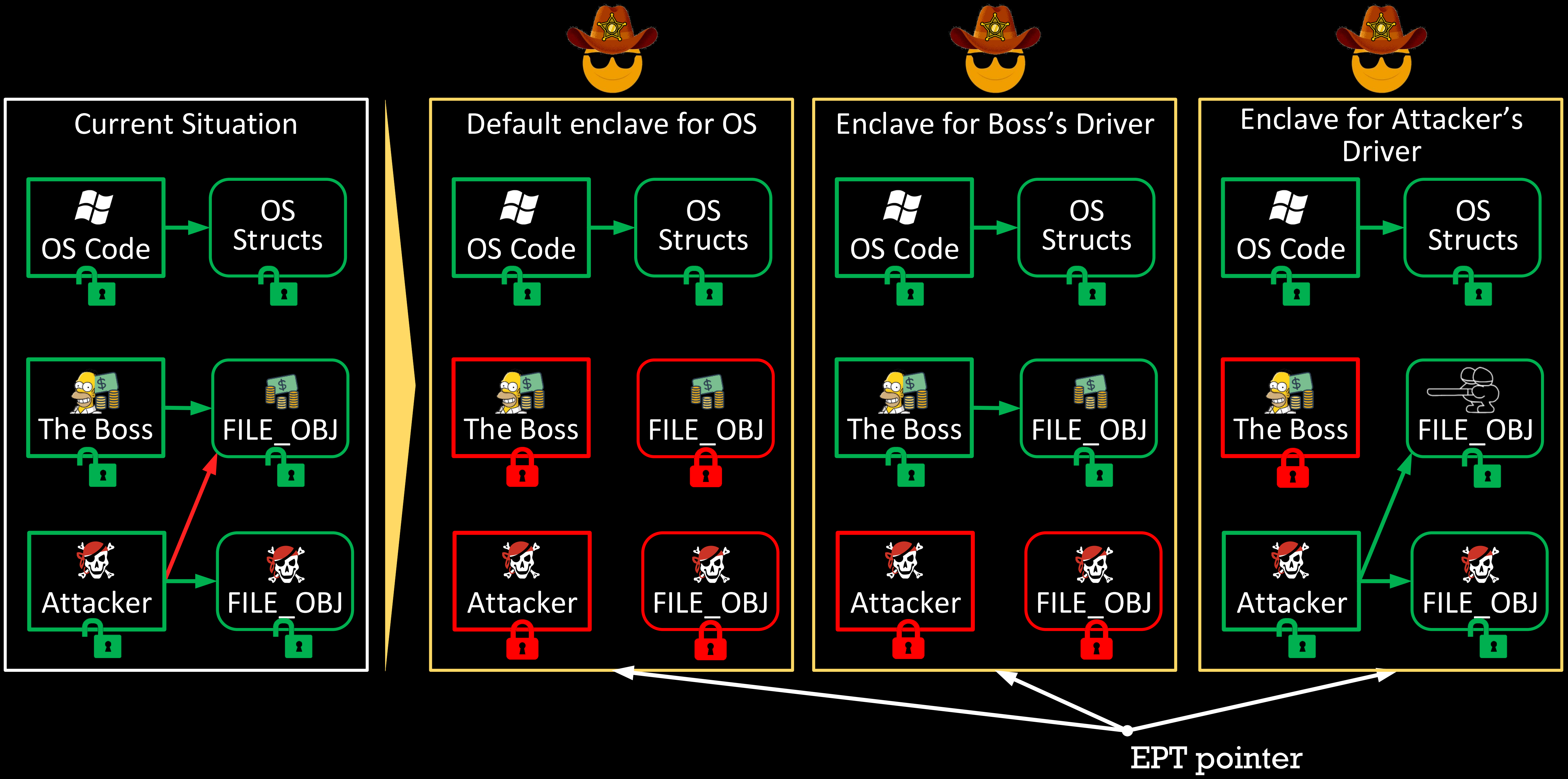
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



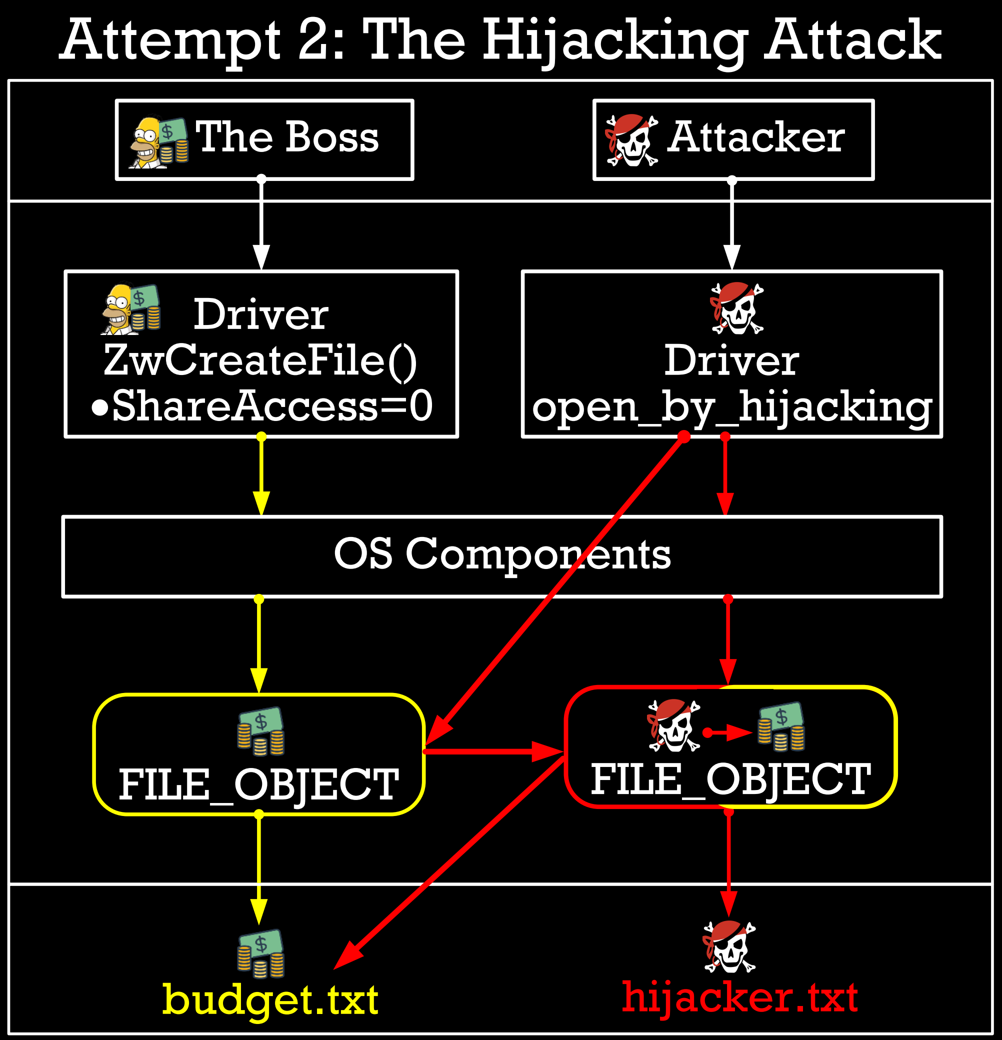
MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING



MEMORY RANGER PREVENTS FILE_OBJECT HIJACKING

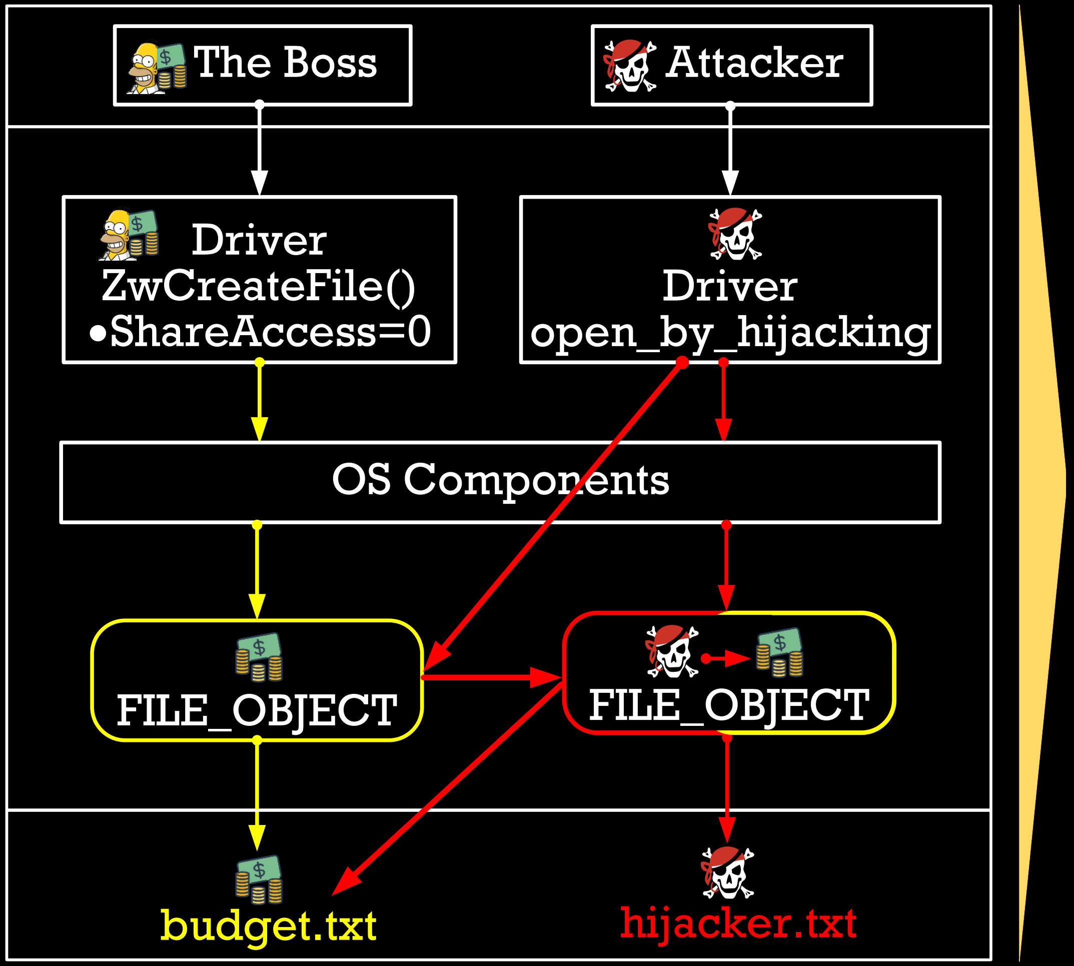


DEMO: PREVENTING THE HIJACKING



DEMO: PREVENTING THE HIJACKING

Attempt 2: The Hijacking Attack

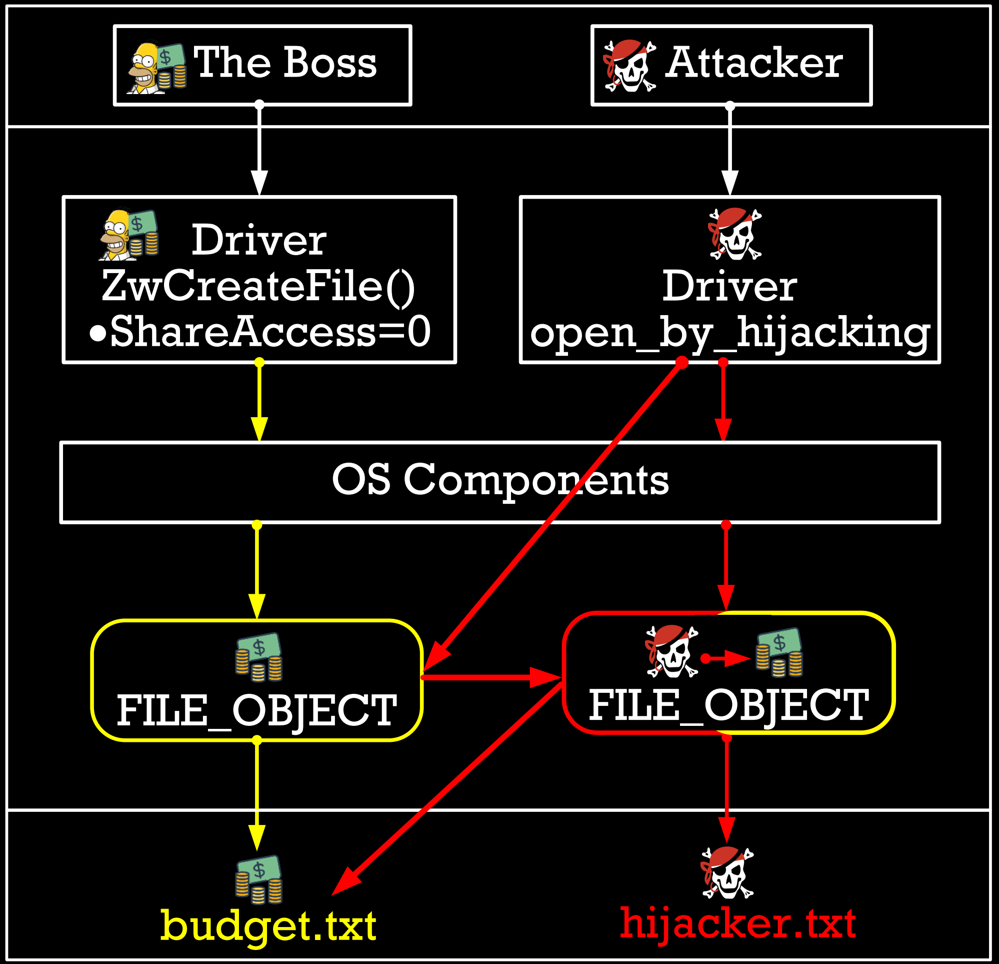


DEMO: THE ATTACK PREVENTION

The online version is here –
https://www.youtube.com/watch?v=8ONmC5Do4I4?vq=hd1080

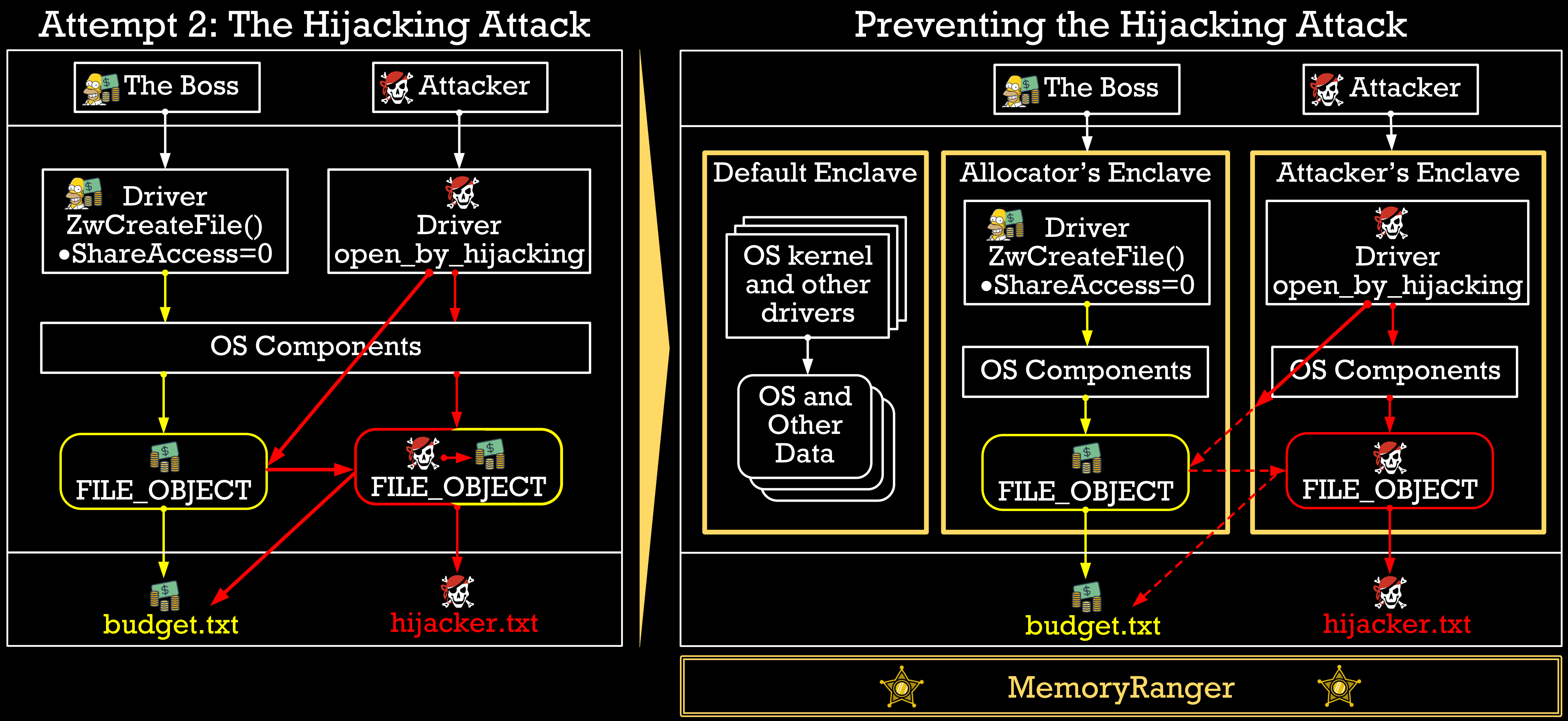
DEMO: PREVENTING THE HIJACKING

Attempt 2: The Hijacking Attack

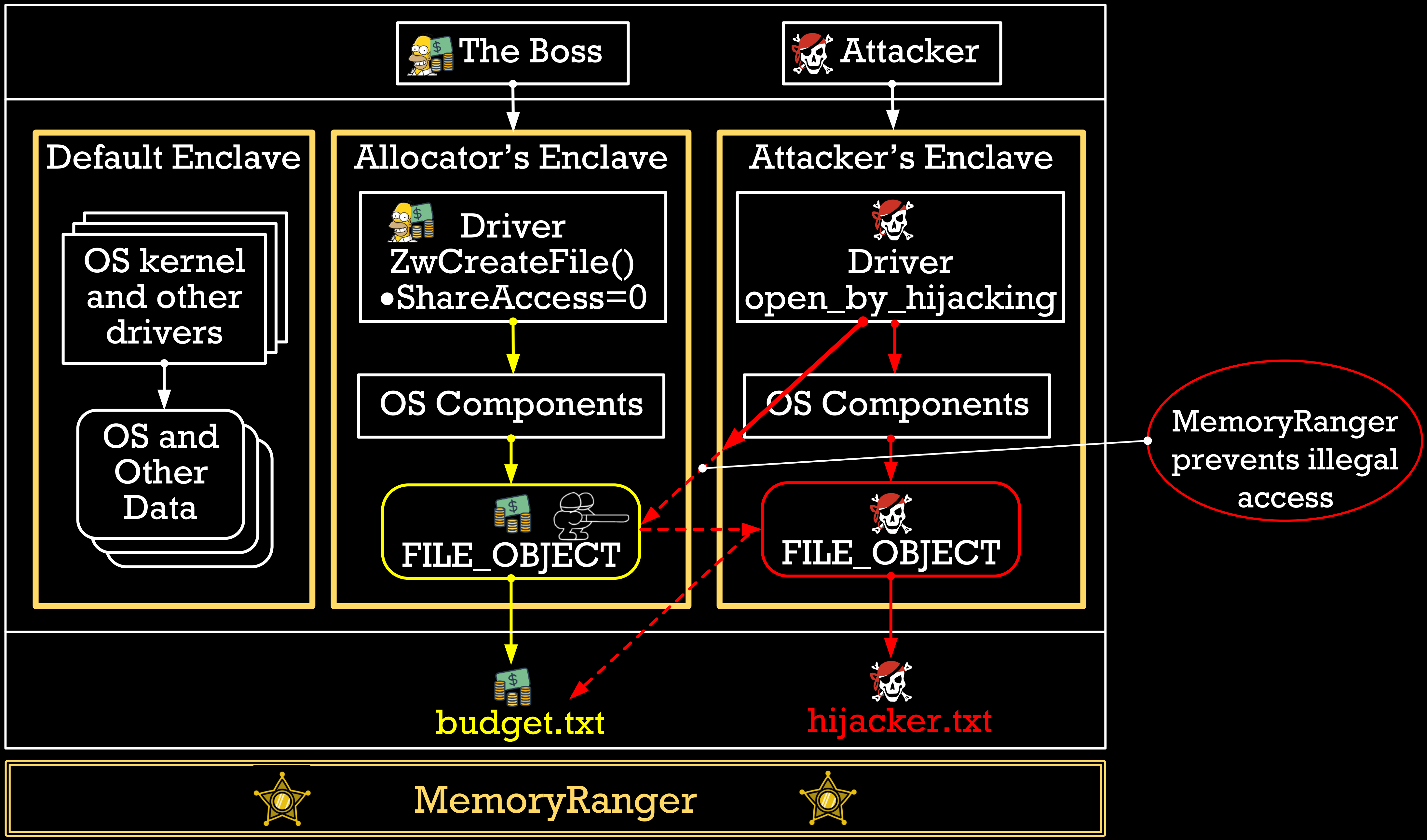


MemoryRanger

DEMO: PREVENTING THE HIJACKING



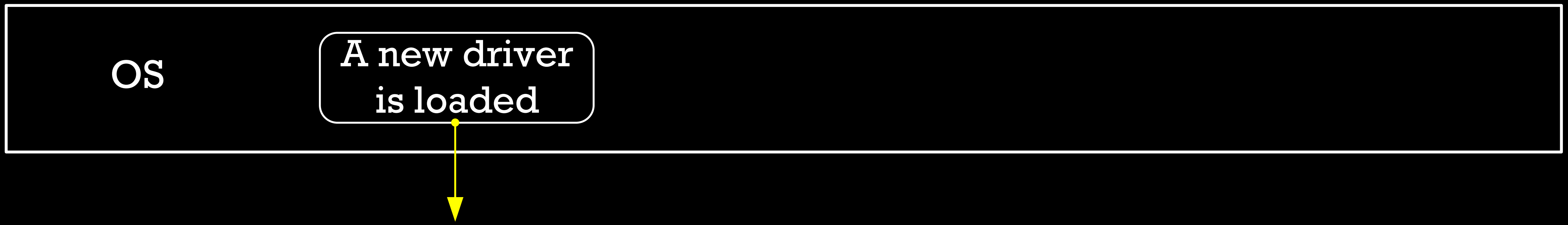
Preventing the Hijacking Attack



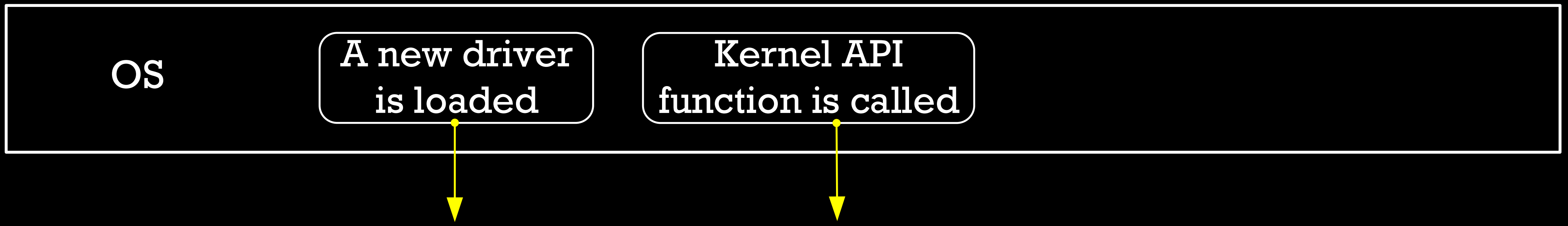
MEMORY RANGER ARCHITECTURE

OS

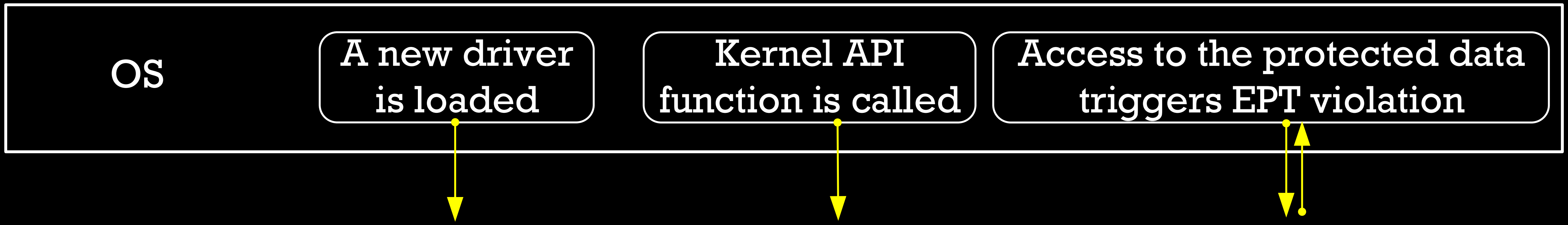
MEMORY RANGER ARCHITECTURE



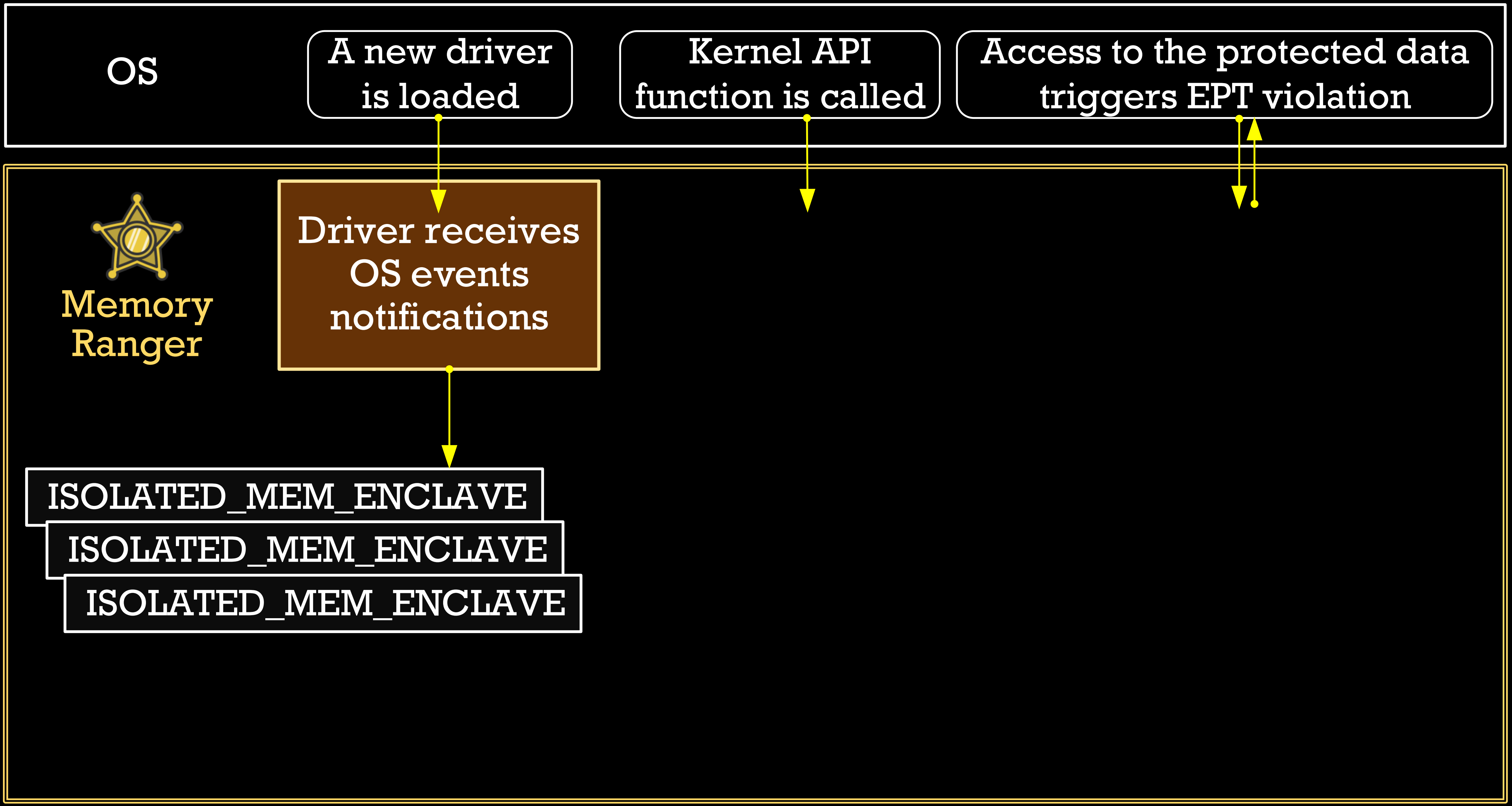
MEMORY RANGER ARCHITECTURE



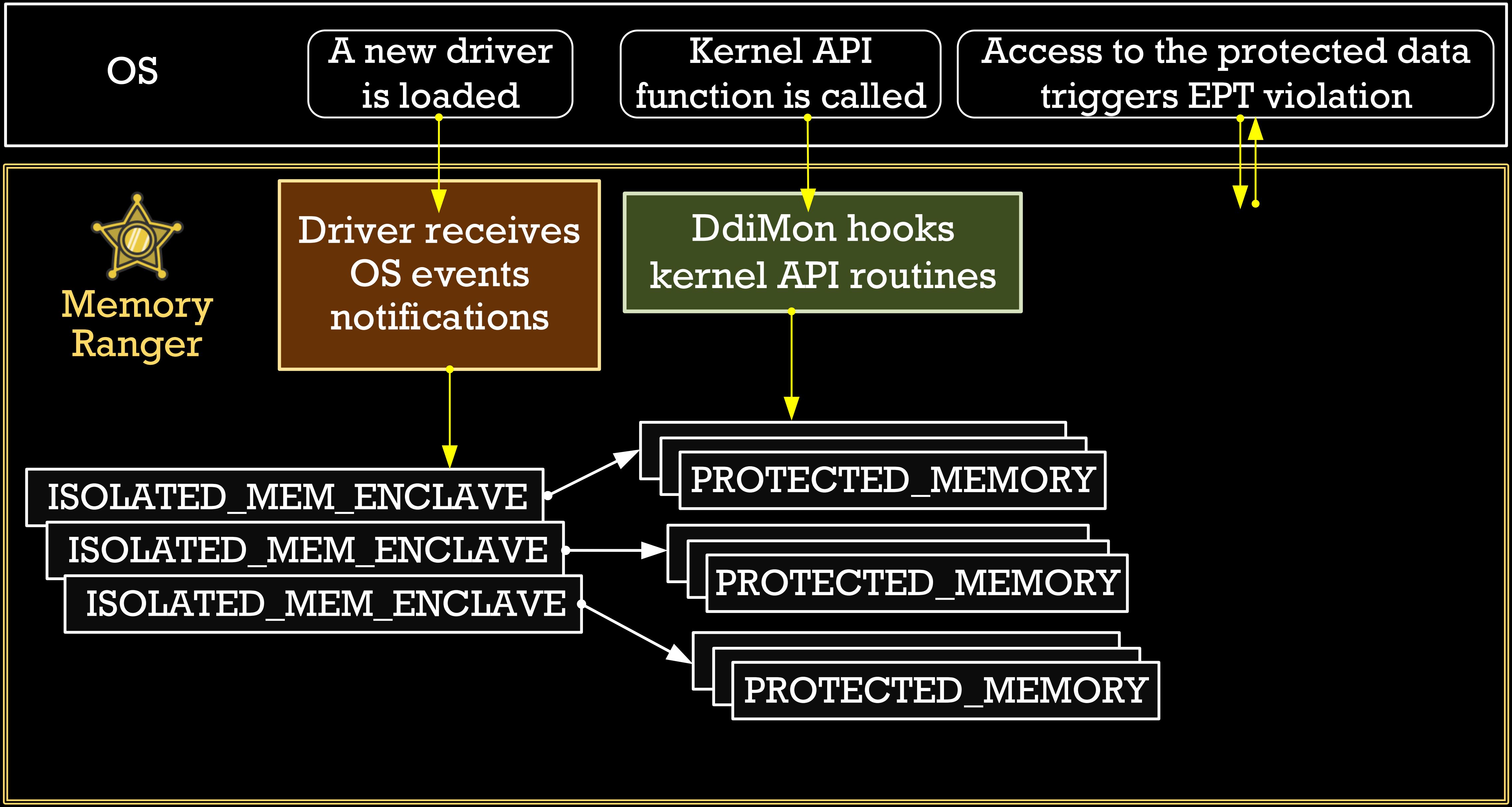
MEMORY RANGER ARCHITECTURE



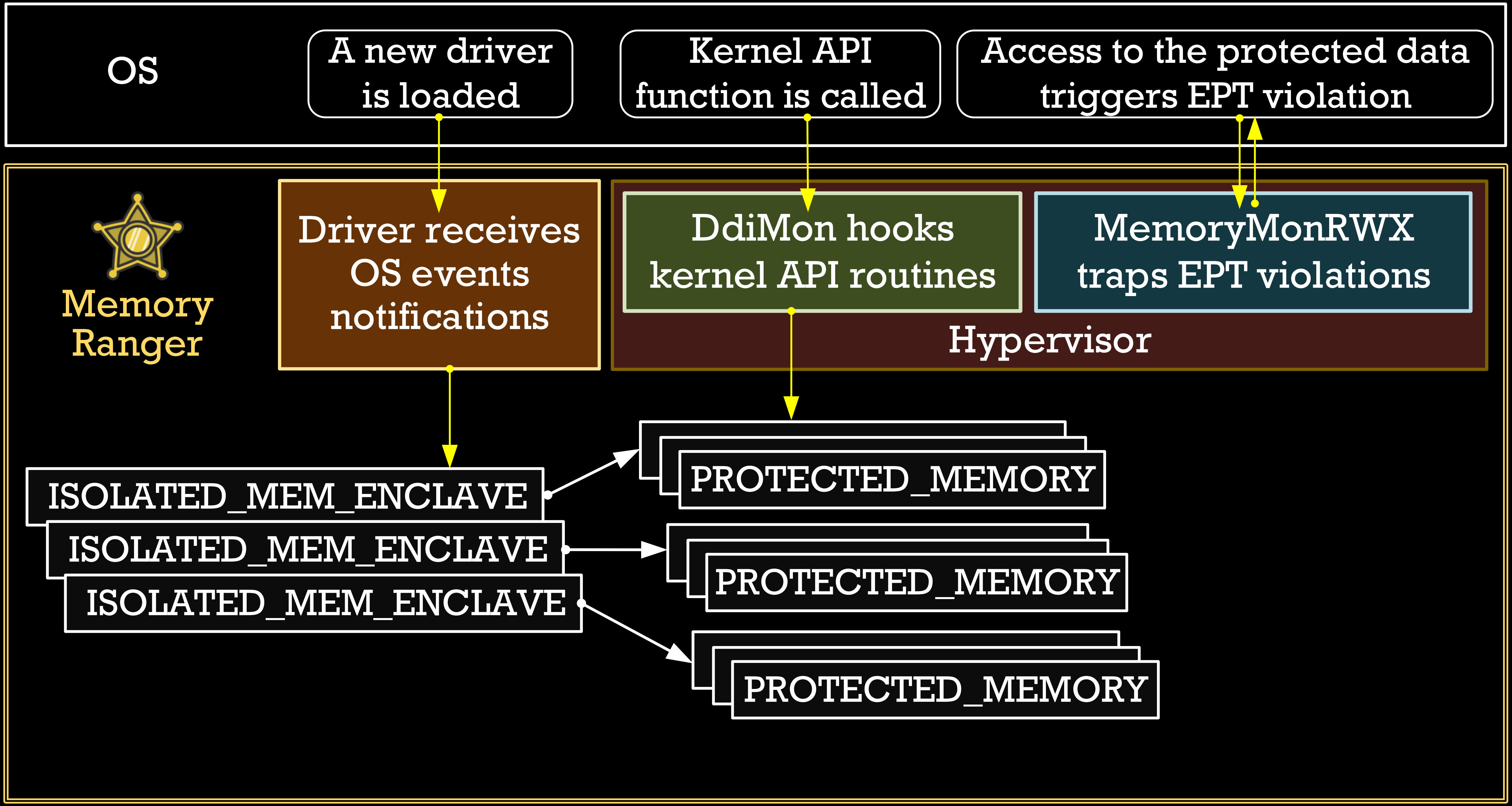
MEMORY RANGER ARCHITECTURE



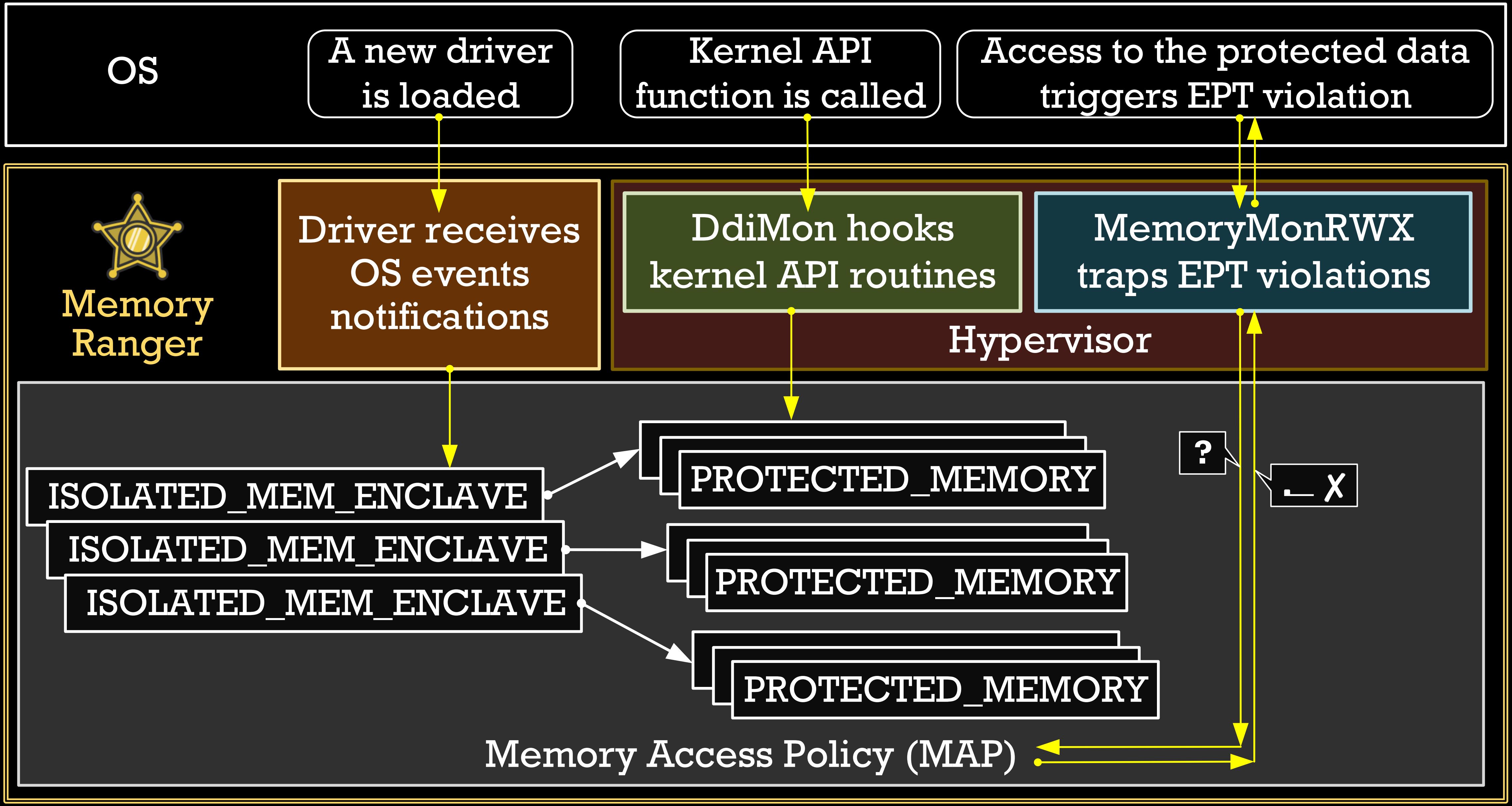
MEMORY RANGER ARCHITECTURE



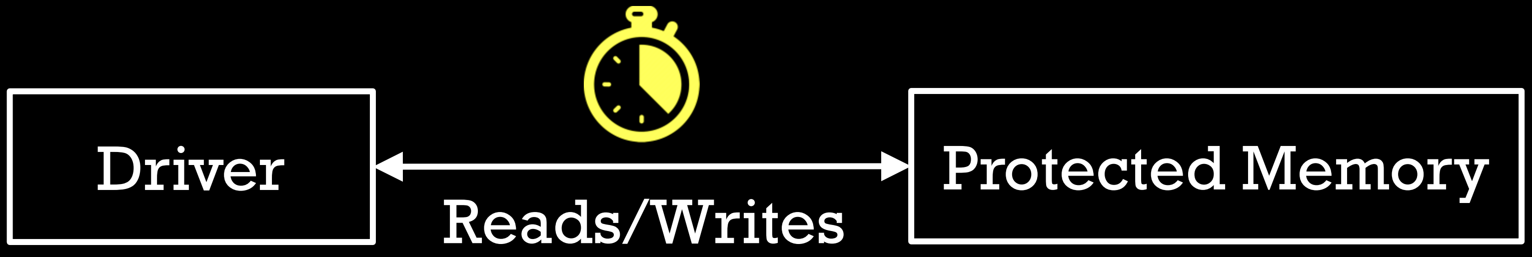
MEMORY RANGER ARCHITECTURE



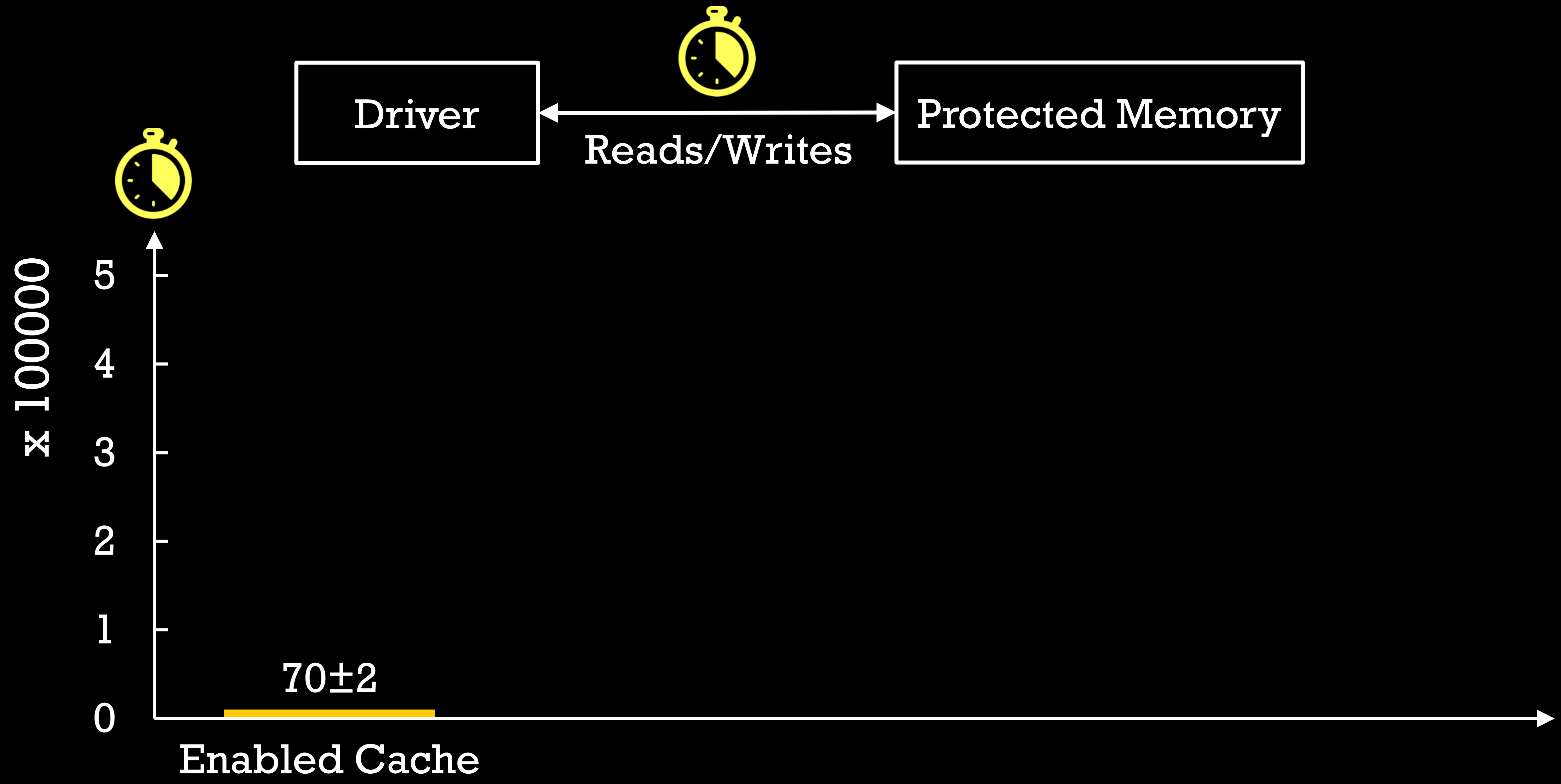
MEMORY RANGER ARCHITECTURE



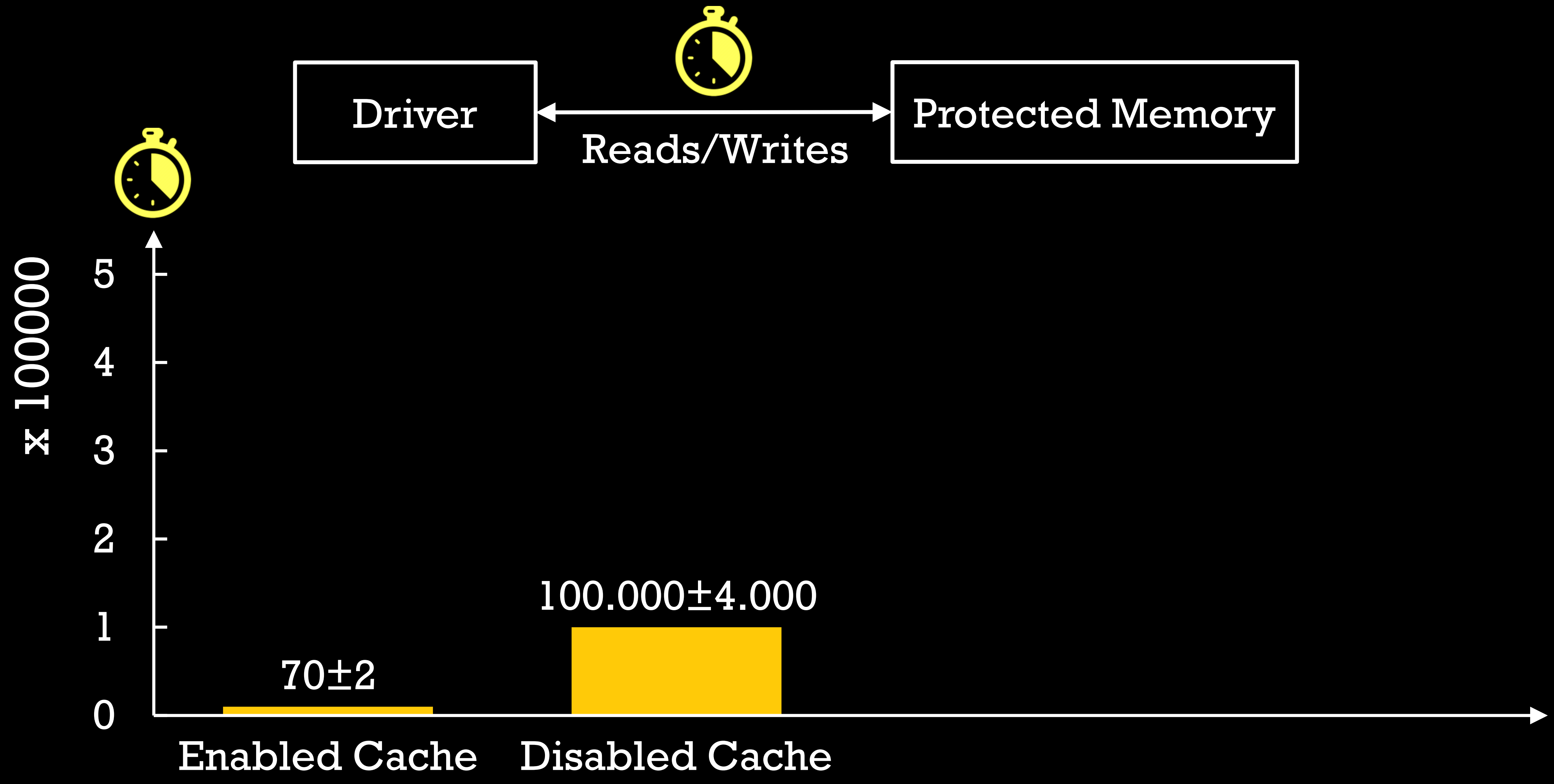
MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



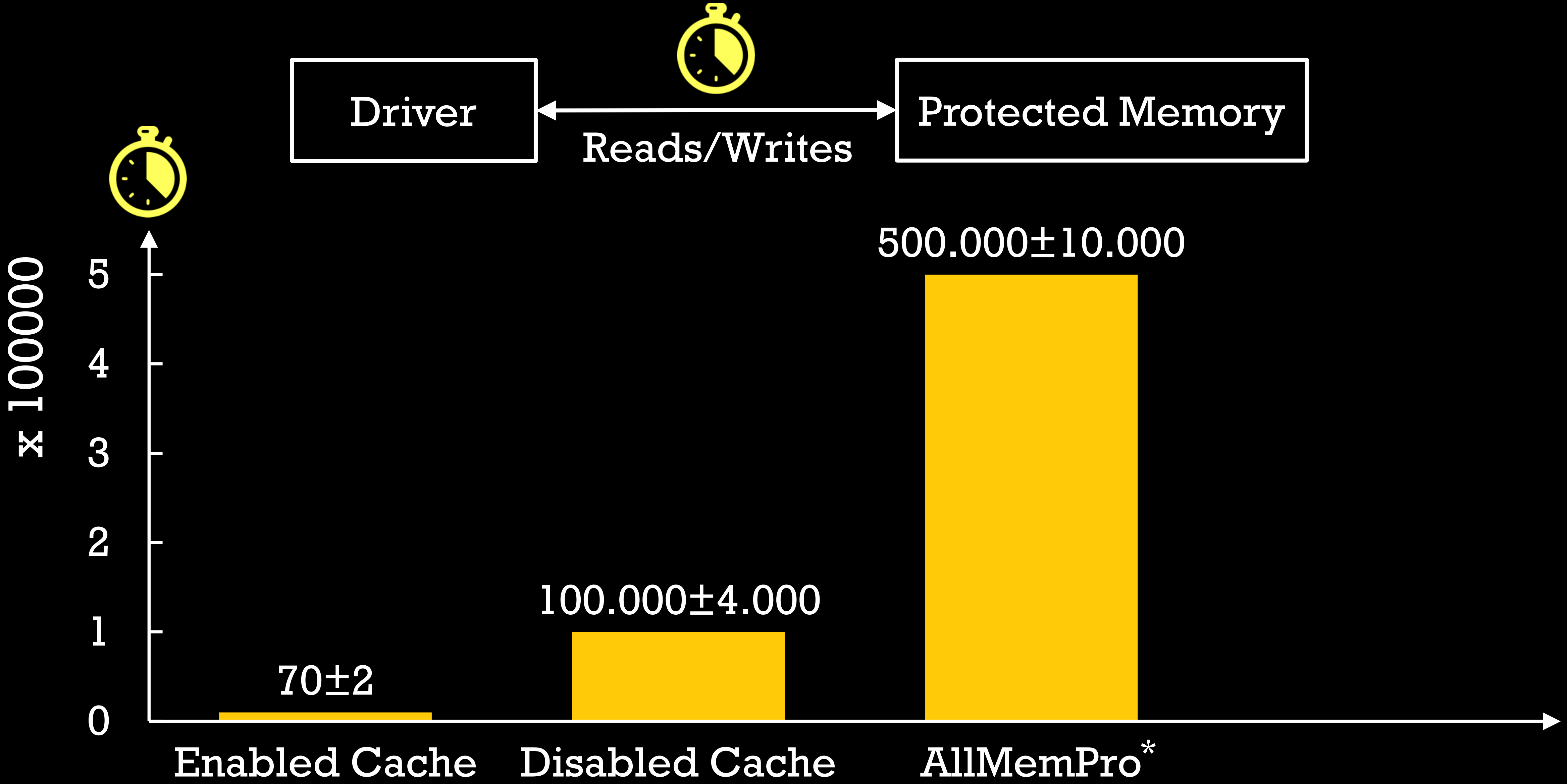
MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME

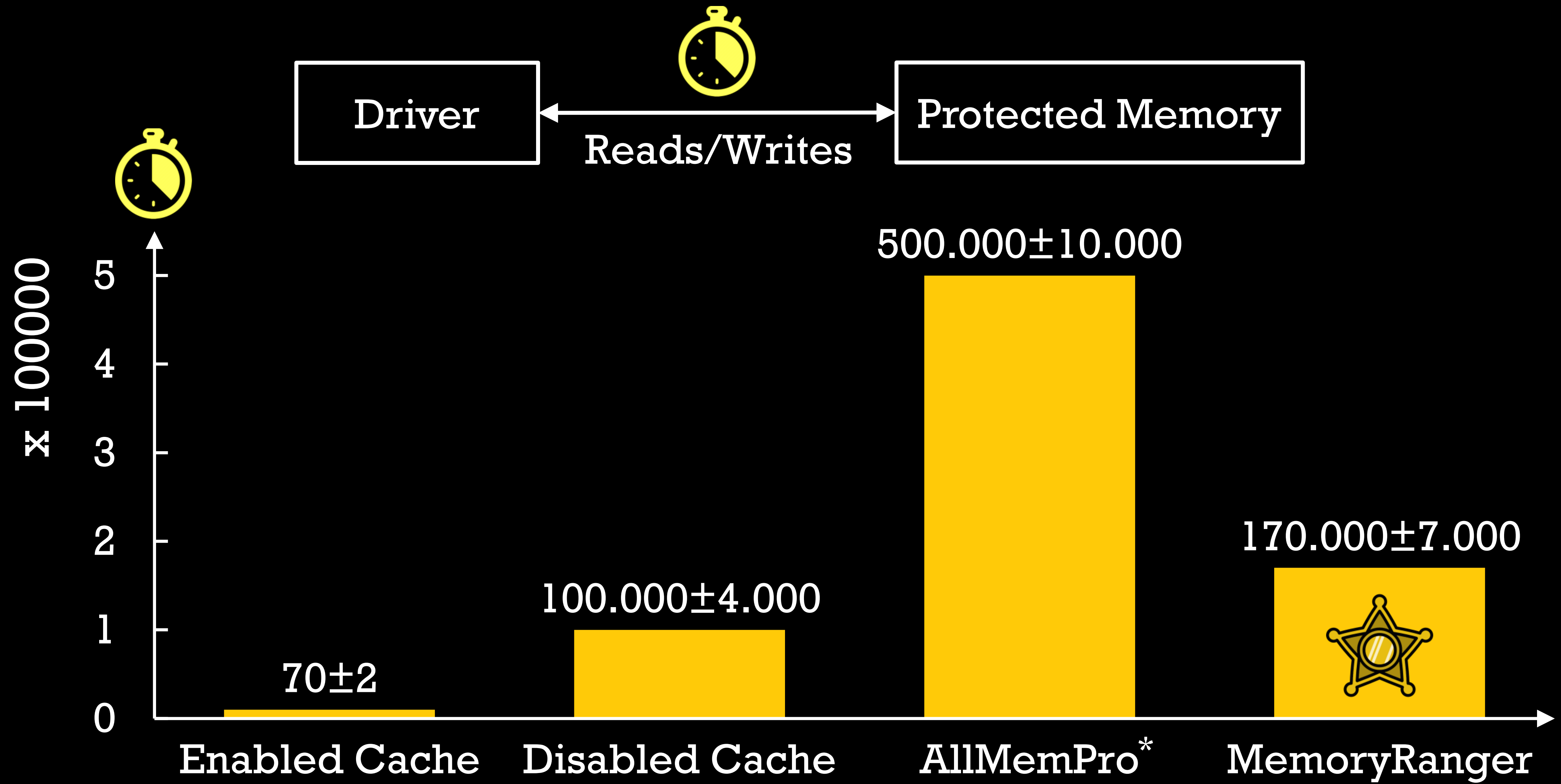


MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



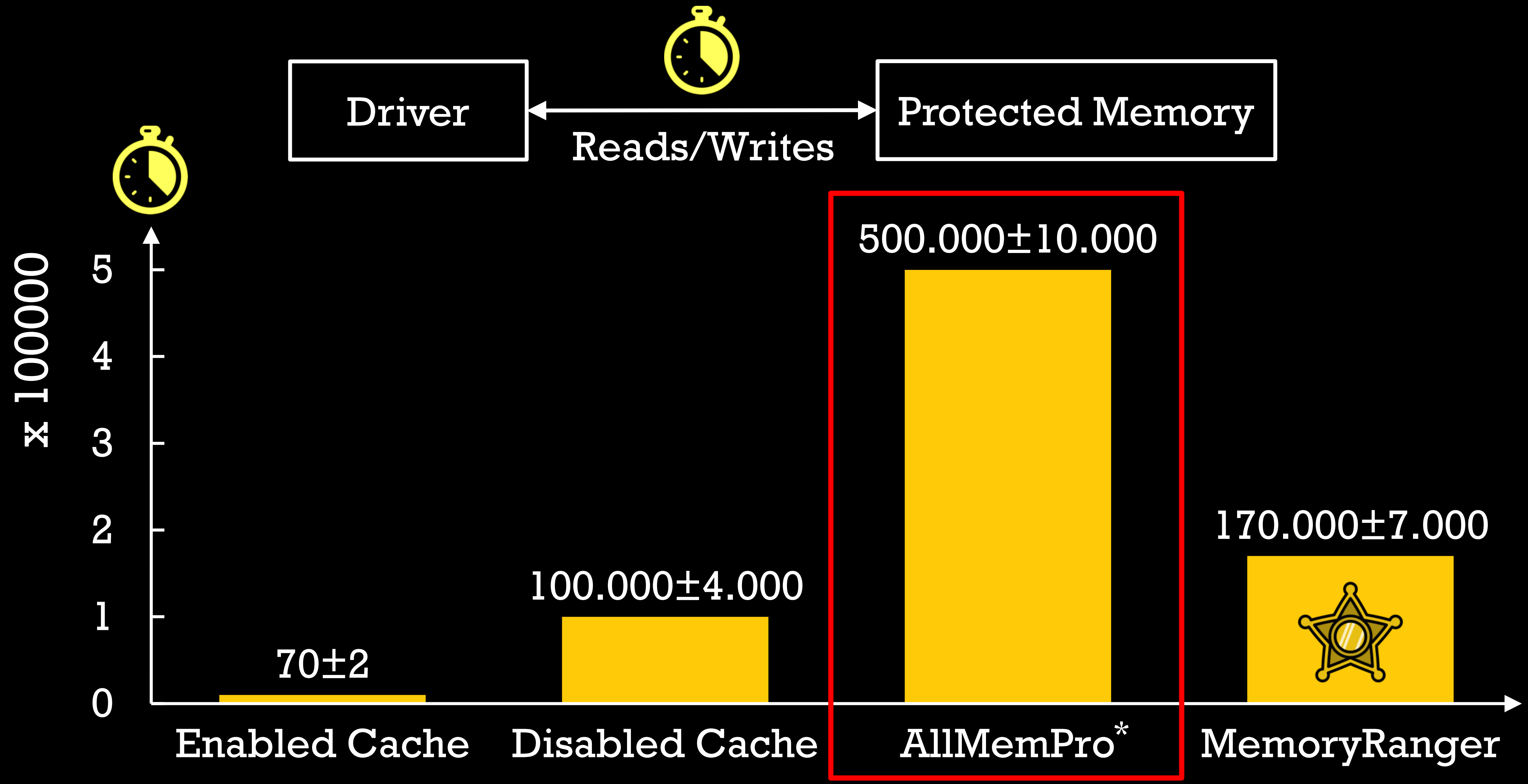
* AllMemPro details - http://bit.ly/AllMemPro

MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



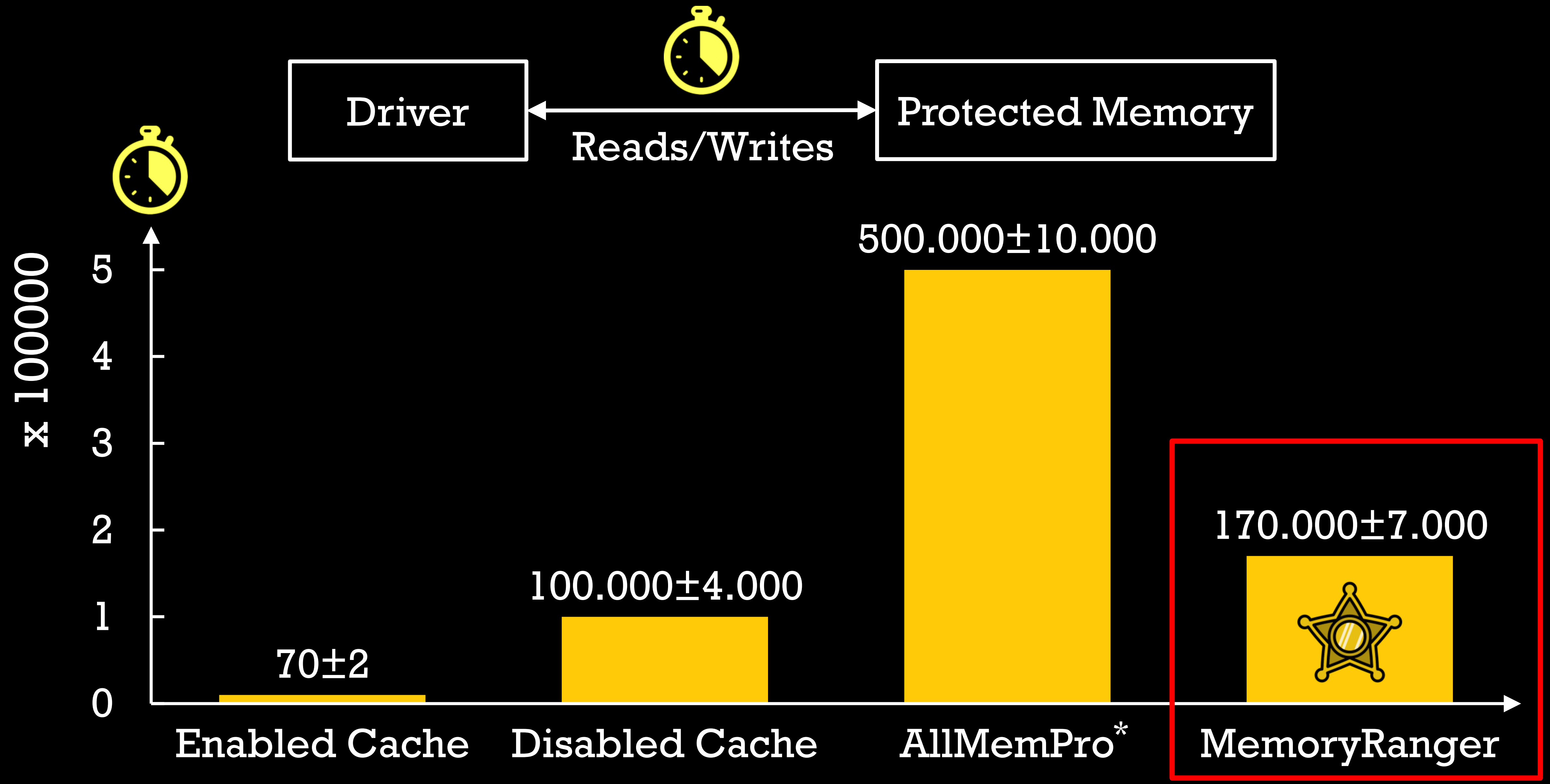
* AllMemPro details - http://bit.ly/AllMemPro

MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



* AllMemPro details - http://bit.ly/AllMemPro

MEMORY RANGER BENCHMARKS: MEMORY ACCESS TIME



* AllMemPro details - http://bit.ly/AllMemPro

THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions					
Integrity					
Confidentiality					

THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code				
Integrity					
Confidentiality					

THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code OS Code				
Integrity					
Confidentiality					

THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code OS Code				
Integrity	Device Guard				
Confidentiality					

THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code OS Code				
Integrity	Device Guard				
Confidentiality					

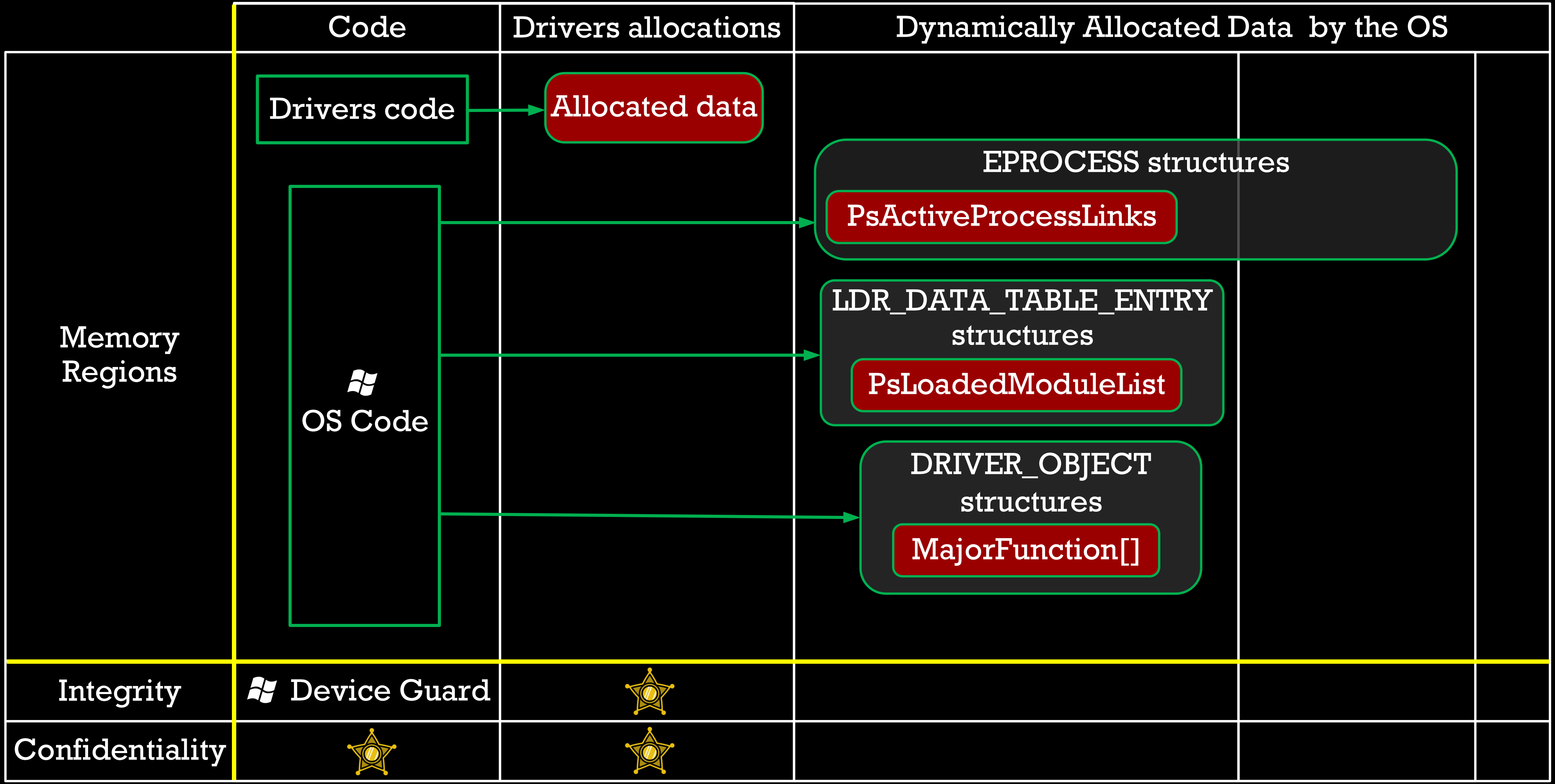
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code OS Code	Allocated data			
Integrity	Device Guard				
Confidentiality					

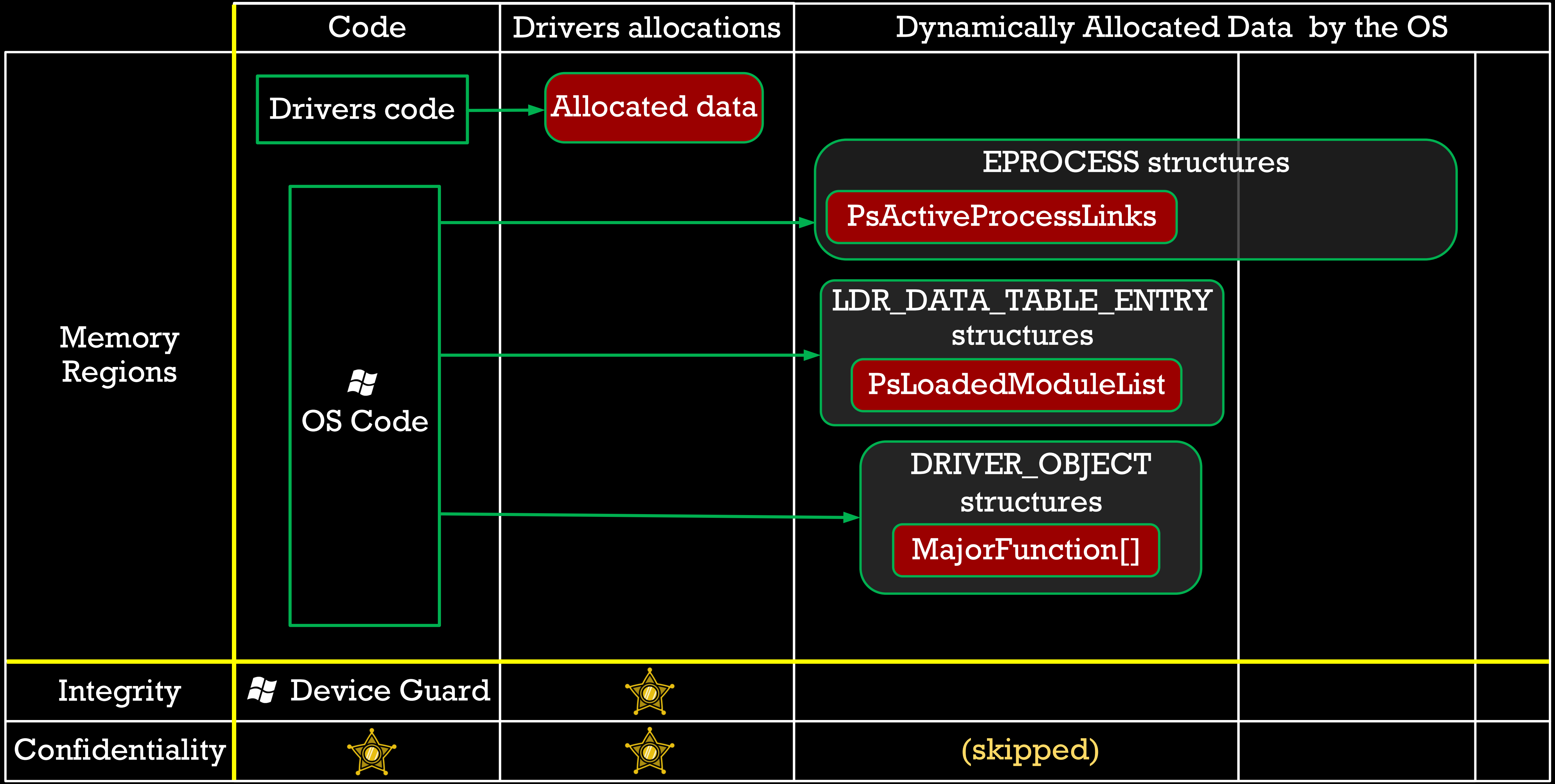
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions	Drivers code → OS Code	Allocated data			
Integrity	Device Guard				
Confidentiality					

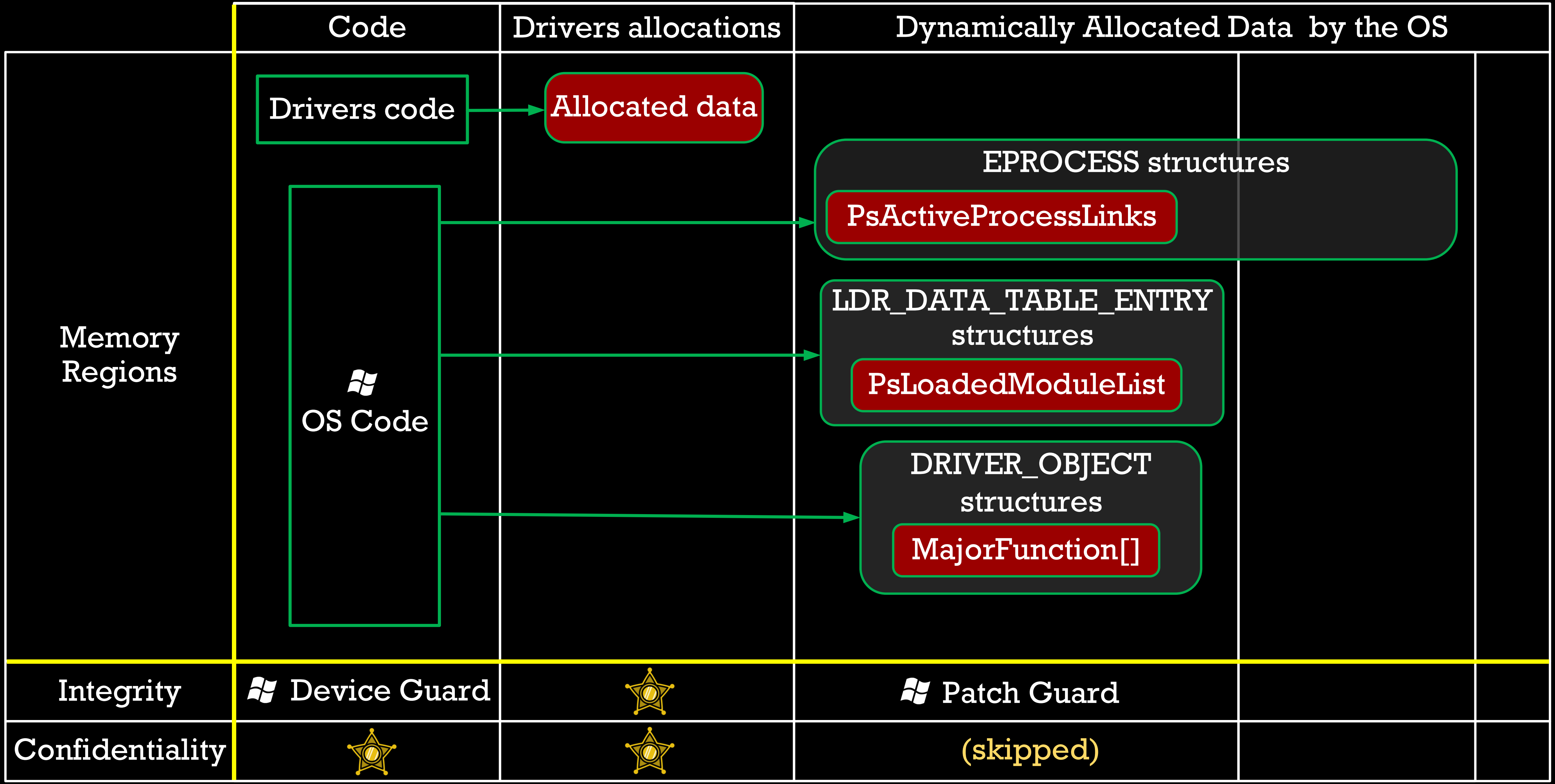
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



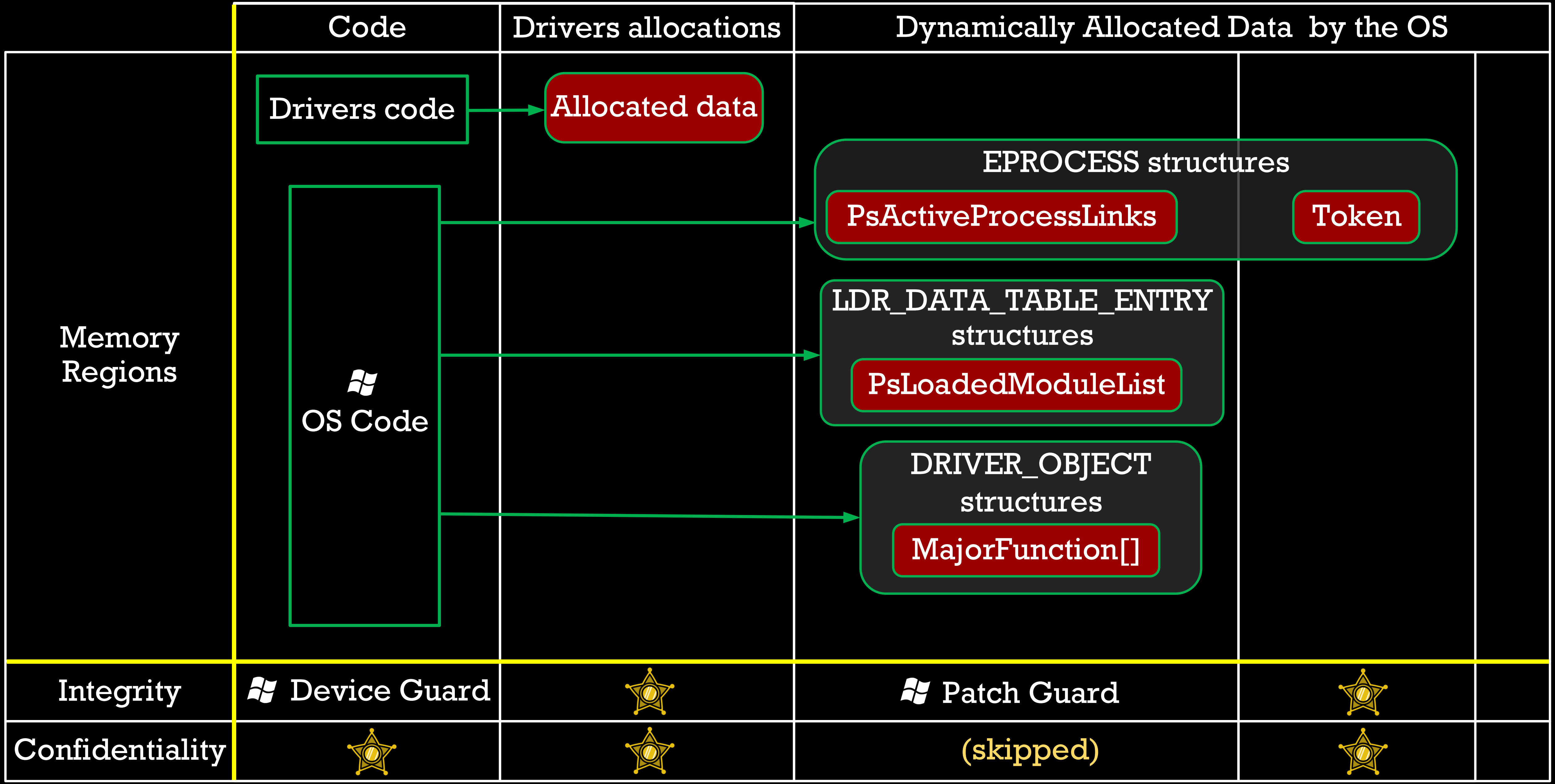
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



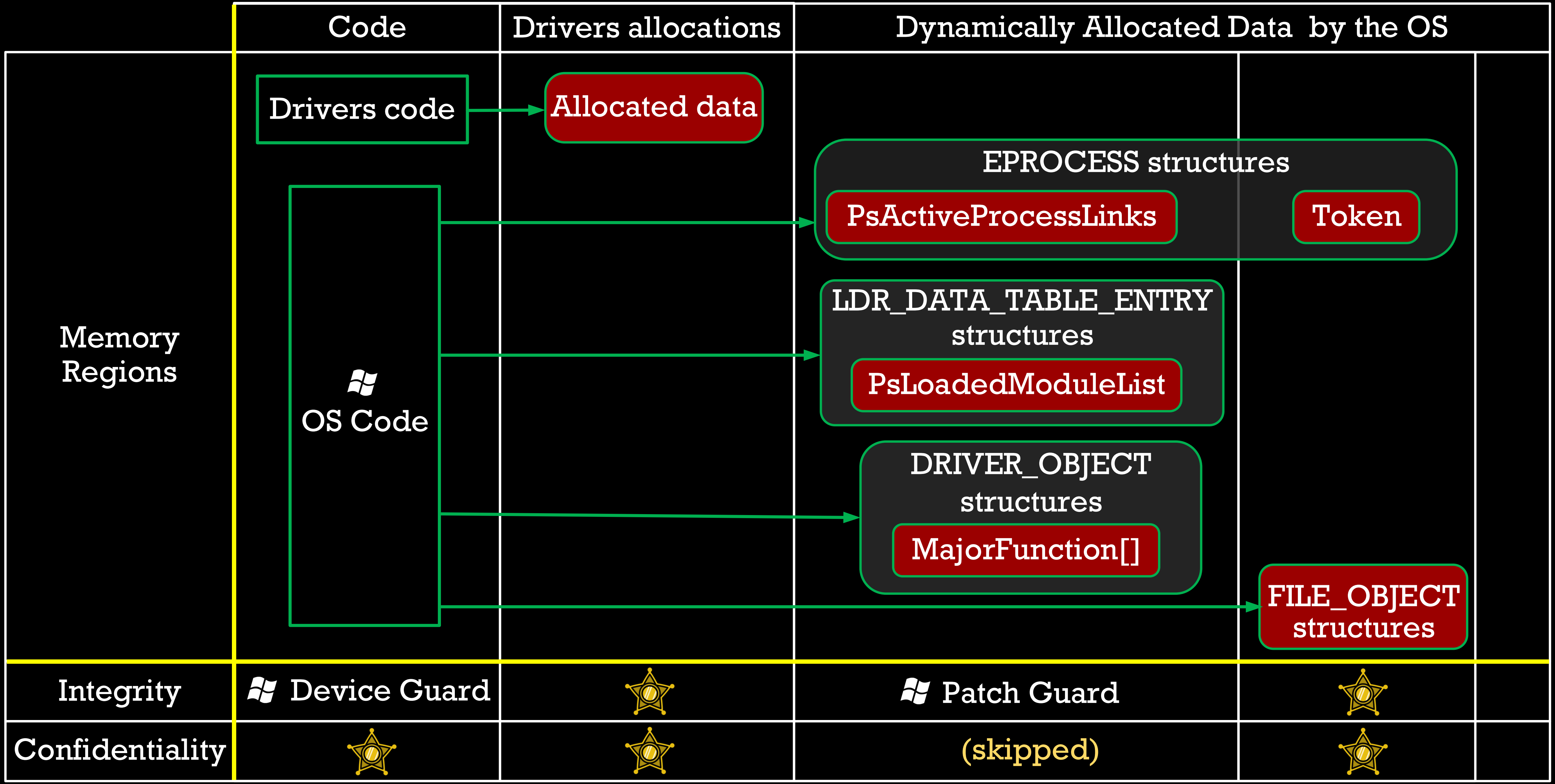
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



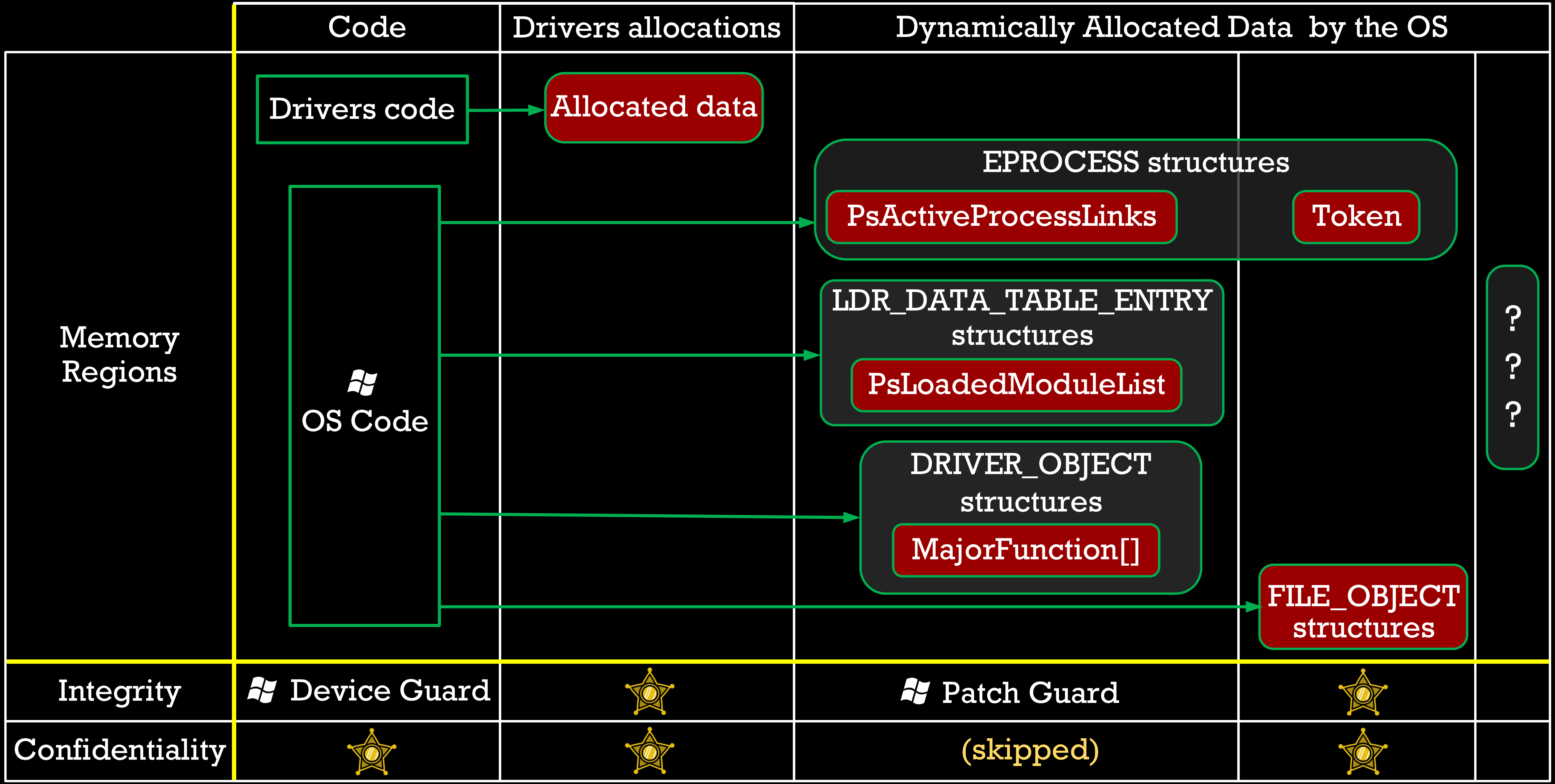
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



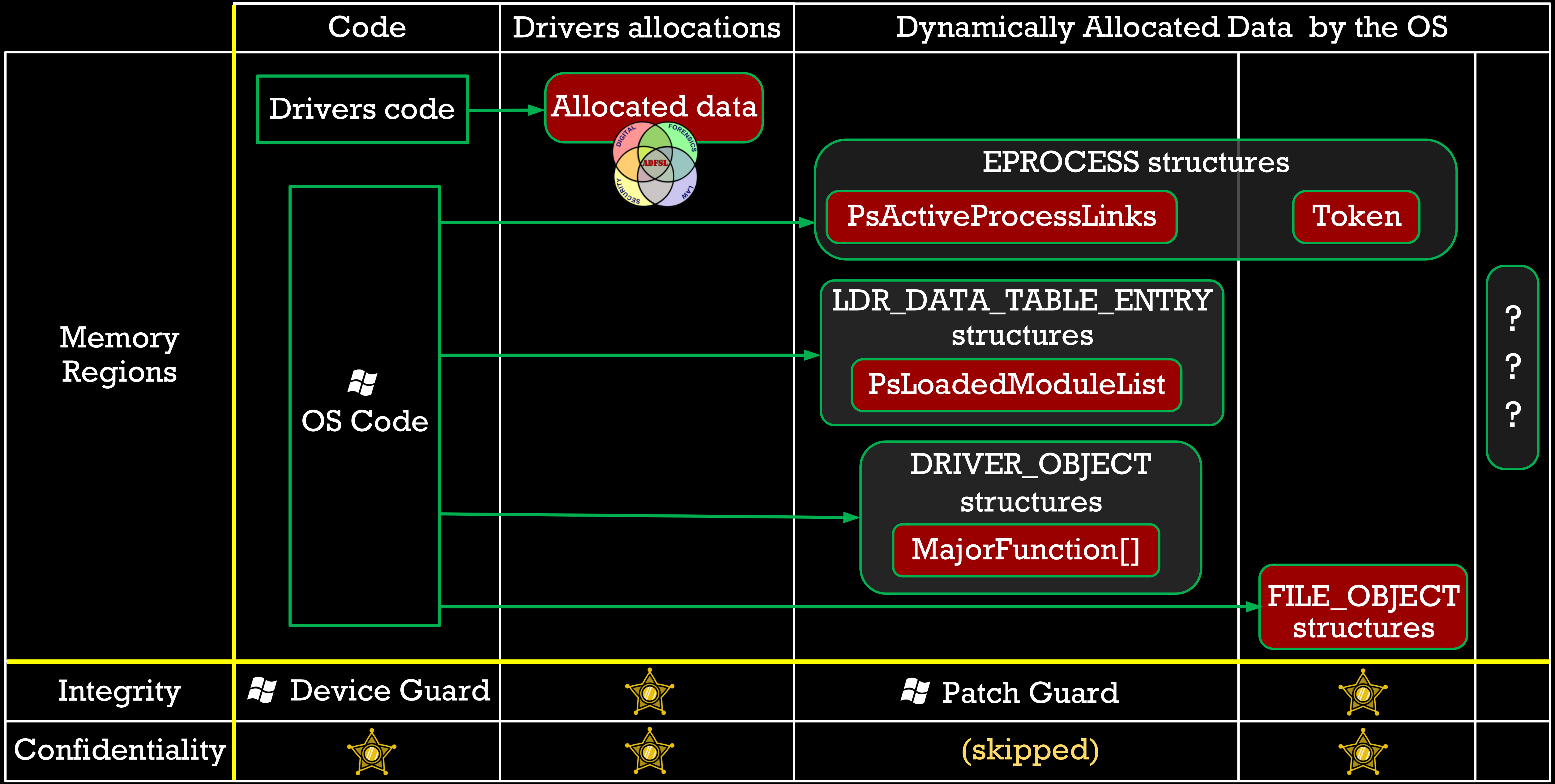
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



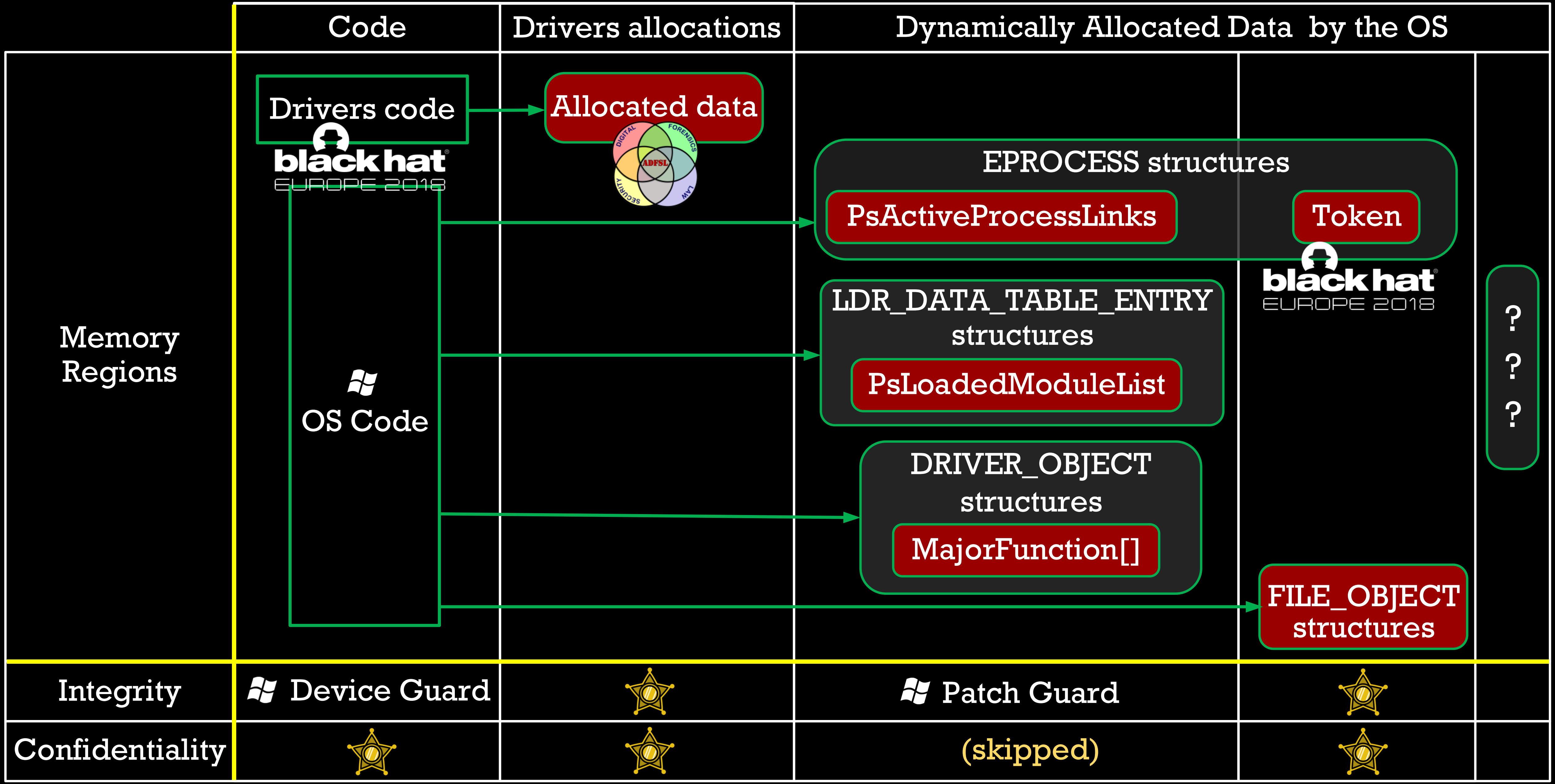
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



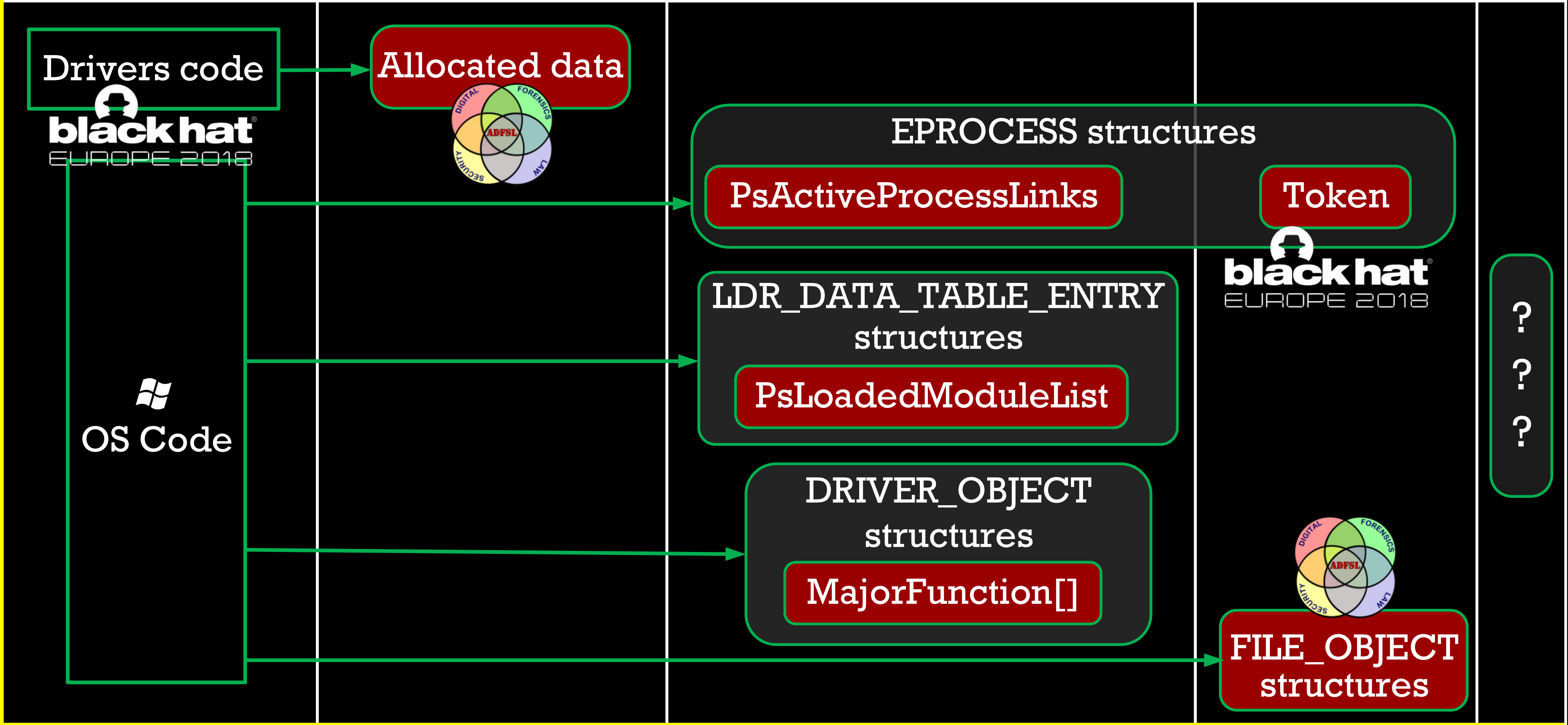
THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY



THE CURRENT SITUATION WITH ATTACKS ON WINDOWS MEMORY

	Code	Drivers allocations	Dynamically Allocated Data by the OS		
Memory Regions					
Integrity	Device Guard		Patch Guard		
Confidentiality			(skipped)		

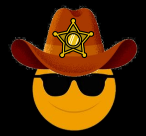
CONCLUSION

- All modern Windows OSes are vulnerable to FILE_OBJECT hijacking
- MemoryRanger prevents the hijacking attack by running drivers into isolated memory enclaves
- Research is ongoing

Thank you!

Igor Korkin igor.korkin@gmail.com

All the details & my CV are here igorkorkin.blogspot.com

MEMORY RANGER HISTORY

Step 1

HyperPlatform

REcon

Step 2

MemoryMonRWX
- HyperPlatform

ADFSL

Step 3

AllMemPro
- MemoryMonRWX
  - HyperPlatform

ADFSL

Step 4

MemoryRanger
- AllMemPro
  - MemoryMonRWX
    - HyperPlatform

black hat EUROPE 2018

Step 5

MemoryRanger with a new feature

Prevention of the FILE_OBJECT attack

ADFSL

1. Korkin, I., & Tanda, S. (2016). Monitoring & controlling kernel-mode events by HyperPlatform. Recon, Canada.
2. Korkin, I., & Tanda, S. (2017). Detect Kernel-Mode Rootkits via Real Time Logging & Controlling Memory Access. ADFSL, USA.
3. Korkin, I. (2018). Hypervisor-Based Active Data Protection for Integrity and Confidentiality of Dynamically Allocated Memory in Windows Kernel. ADFSL, USA.
4. Korkin, I. (2018). Divide et Impera: MemoryRanger Runs Drivers in Isolated Kernel Spaces. BlackHat, UK
5. Korkin, I. (2019). MemoryRanger Prevents Hijacking FILE_OBJECT structures in Windows Kernel. ADFSL, USA.