

Finding low-hanging fruits vulnerabilities in a commercial AV

HackSecuReims – 7^e édition
29-03-2024

Madimodi DIAWARA

TABLE OF CONTENTS

- What is an AV again?
- AV attack surface
- Case study
- Conclusion

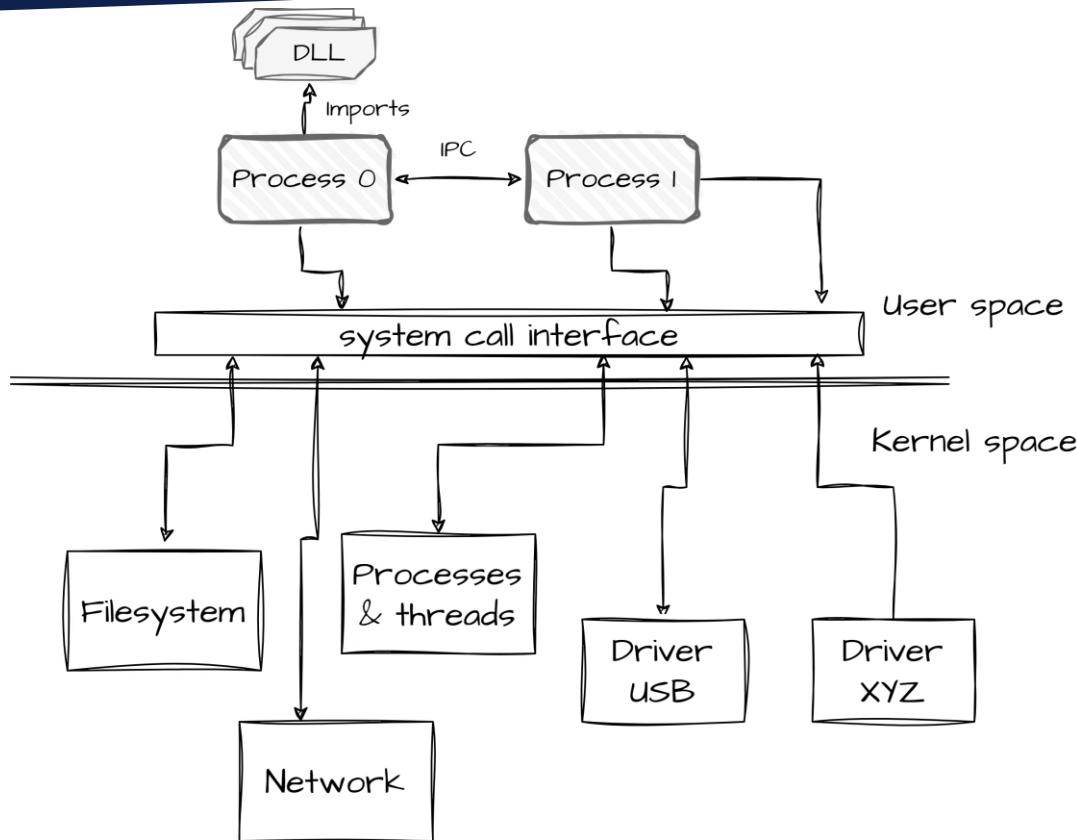
What is an AV again?

Its purpose, its workings and why it is a target of choice

WHAT IS AN AV AGAIN?

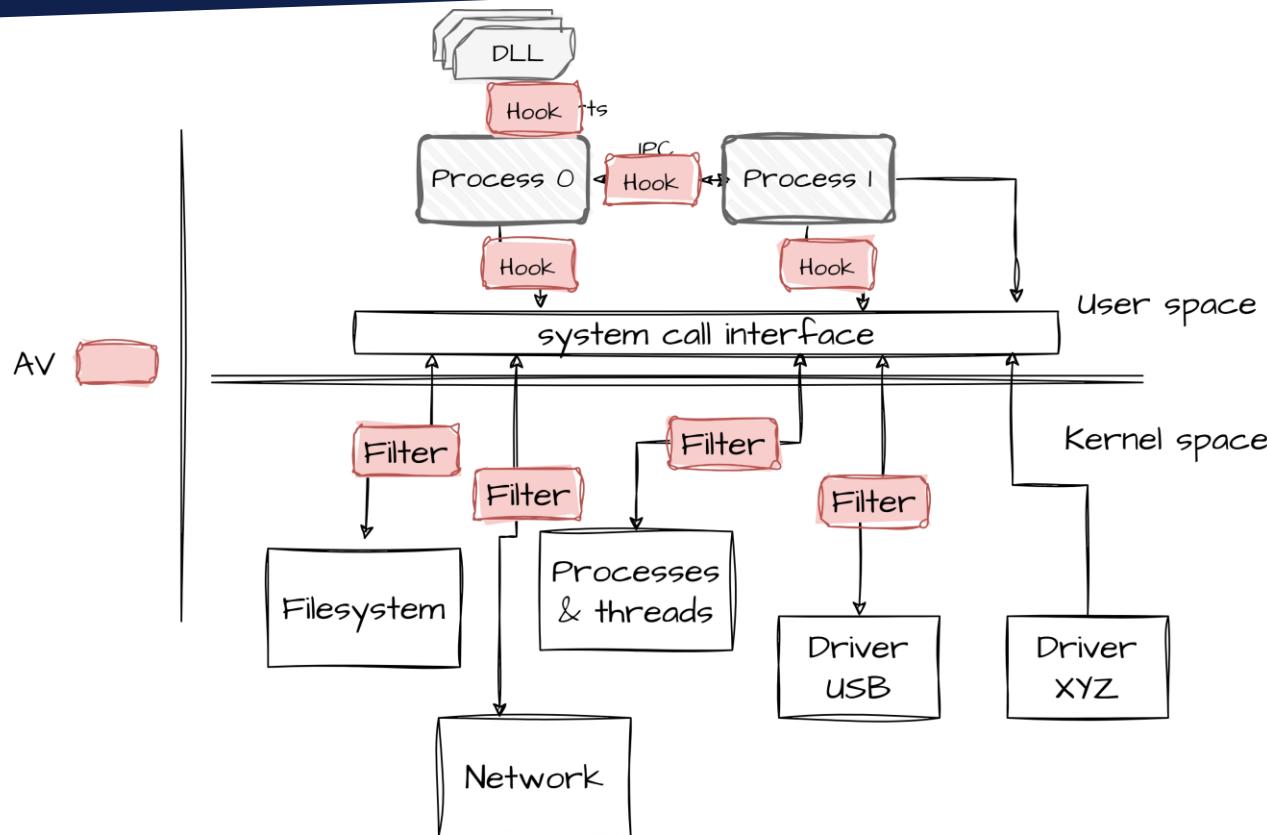
- Software enhancing an operating system (OS) security
- Protects against malware
- Not a software as any other
 - High privileges
 - Kernel components

EMBRACE OS COMPLEXITY



(Very) simplified view of an OS

EMBRACE AV COMPLEXITY



An AV shall fully embrace an OS to be efficient

THE LOGICAL MINUTE: A SYLLOGISM

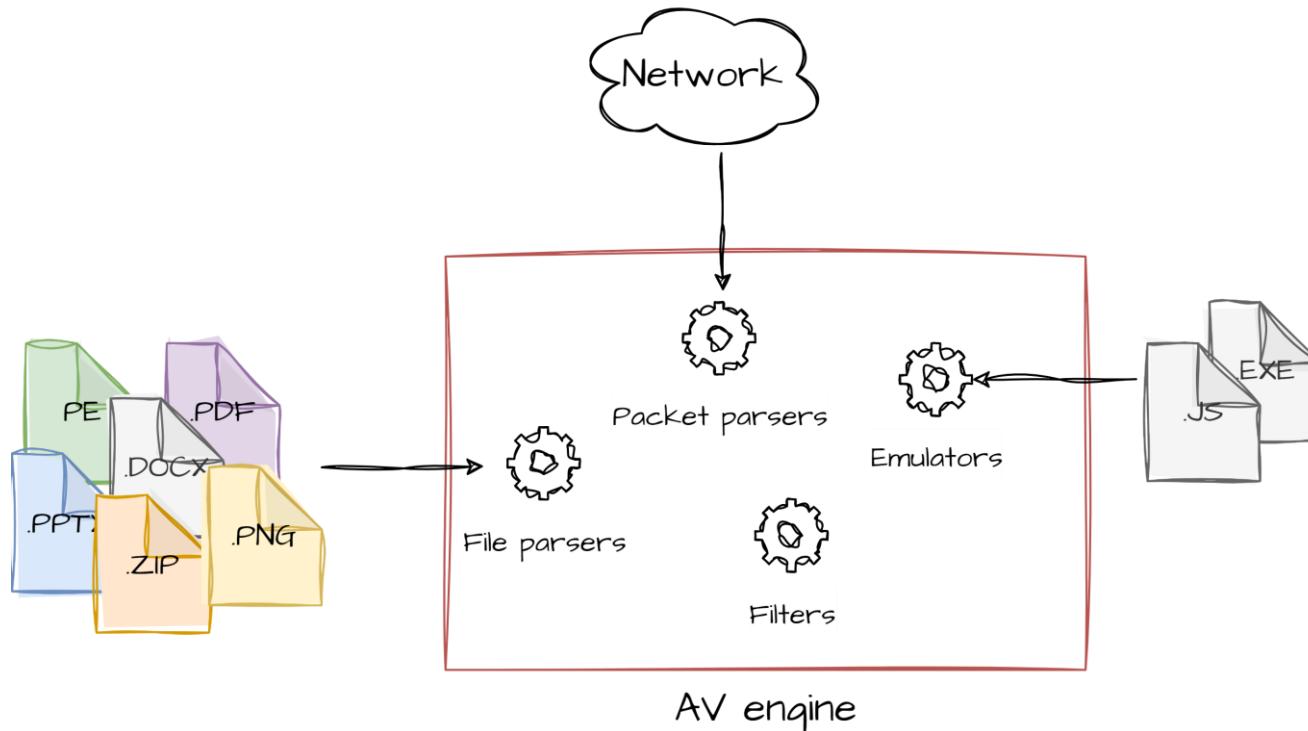


- A running software adds potential attack surface
- An antivirus is a **much more** complex software
- Therefore, an antivirus adds **a much higher** potential attack surface

AV attack surface

Complexity, cartography and analysis efficiency

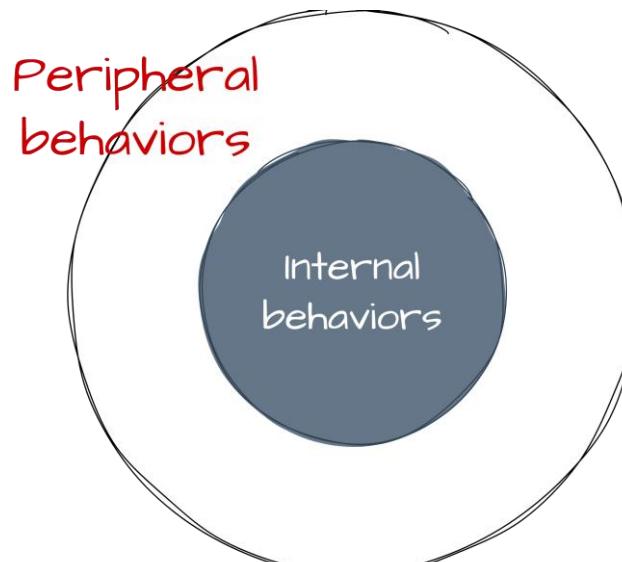
AV ATTACK SURFACE – THE AV ENGINE



A typical AV engine components

INTERNAL VS PERIPHERAL BEHAVIORS

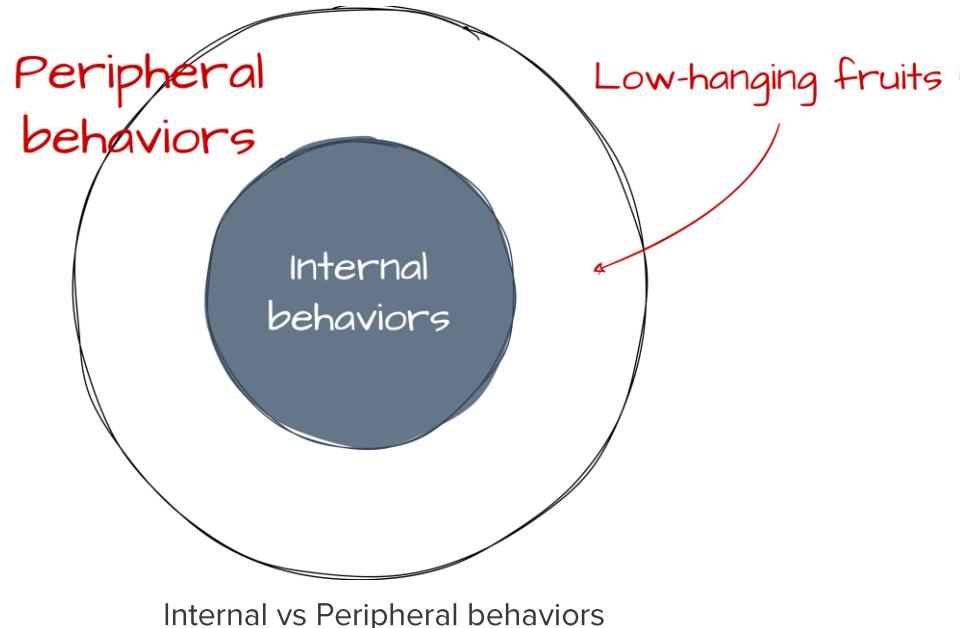
- Analyzing the AV Engine = too time-consuming!
- Let's focus on the ***peripheral behaviors***



Internal vs Peripheral behaviors

INTERNAL VS PERIPHERAL BEHAVIORS

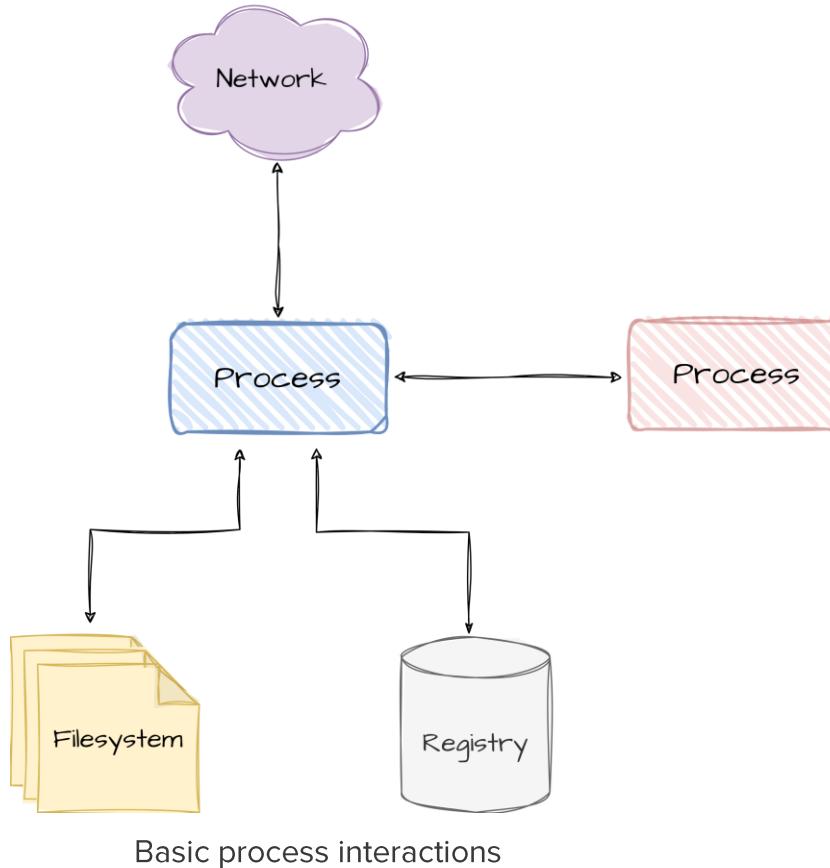
- Analyzing the AV Engine = too time-consuming!
- Let's focus on the ***peripheral behaviors***



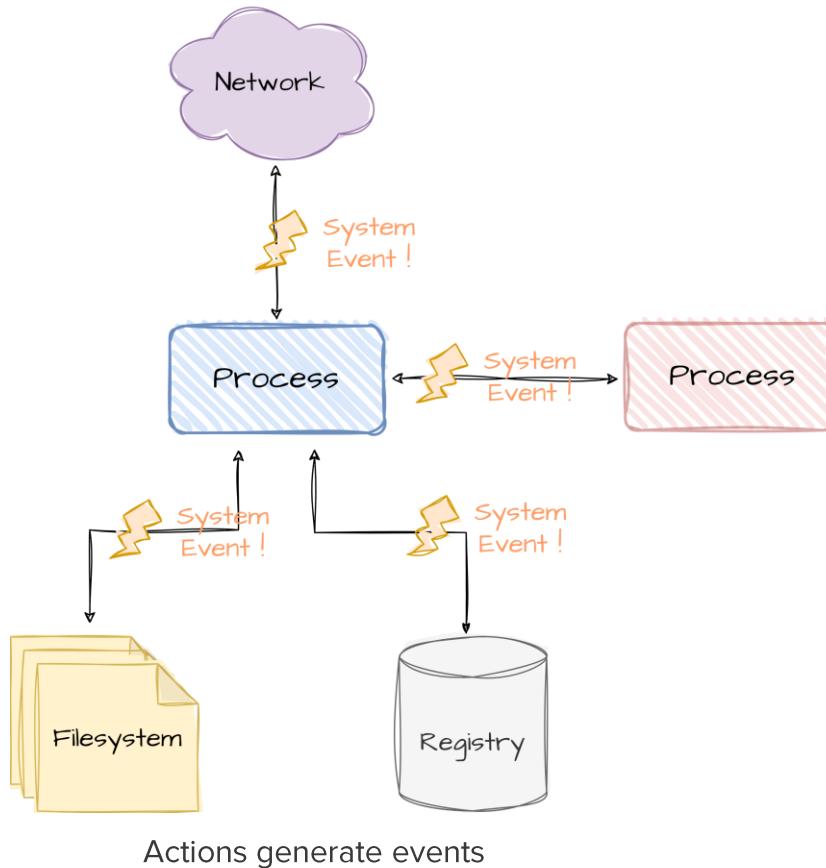
Cartography: Drawing a map

Making a basic cartography

DRAWING A MAP WITH PERIPHERAL BEHAVIORS



DRAWING A MAP WITH PERIPHERAL BEHAVIORS



DRAWING A MAP WITH PERIPHERAL BEHAVIORS

Prepare the artillery !



Procmon



WinObjEx64



DriverView



Procepx

DRAWING A MAP WITH PERIPHERAL BEHAVIORS

Date: 3/23/2022 6:00:47.9225818 AM

Thread: 8172

Class: File System

Operation: CreateFile

Result: SUCCESS

Path: C:\Users\user\AppData\Local\Temp\demo.txt

Duration: 0.0002486

What?

Why?

Desired Access:

Generic Read/Write

Disposition:

OpenIf

Options:

Synchronous IO Non-Alert, Non-Directory File

Attributes:

N

ShareMode:

Read, Write

AllocationSize:

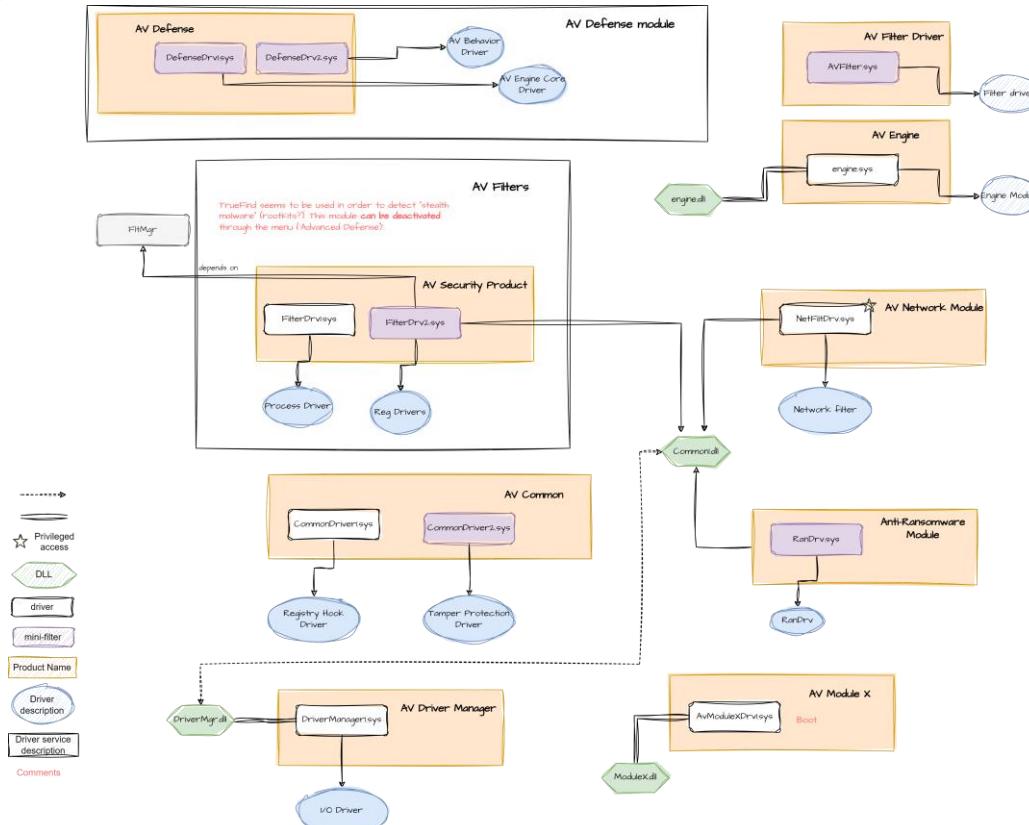
0

OpenResult:

Created

Event from ProcMon representing the creation of demo.txt

DRAWING A MAP WITH PERIPHERAL BEHAVIORS



Result of the cartography phase

Case study

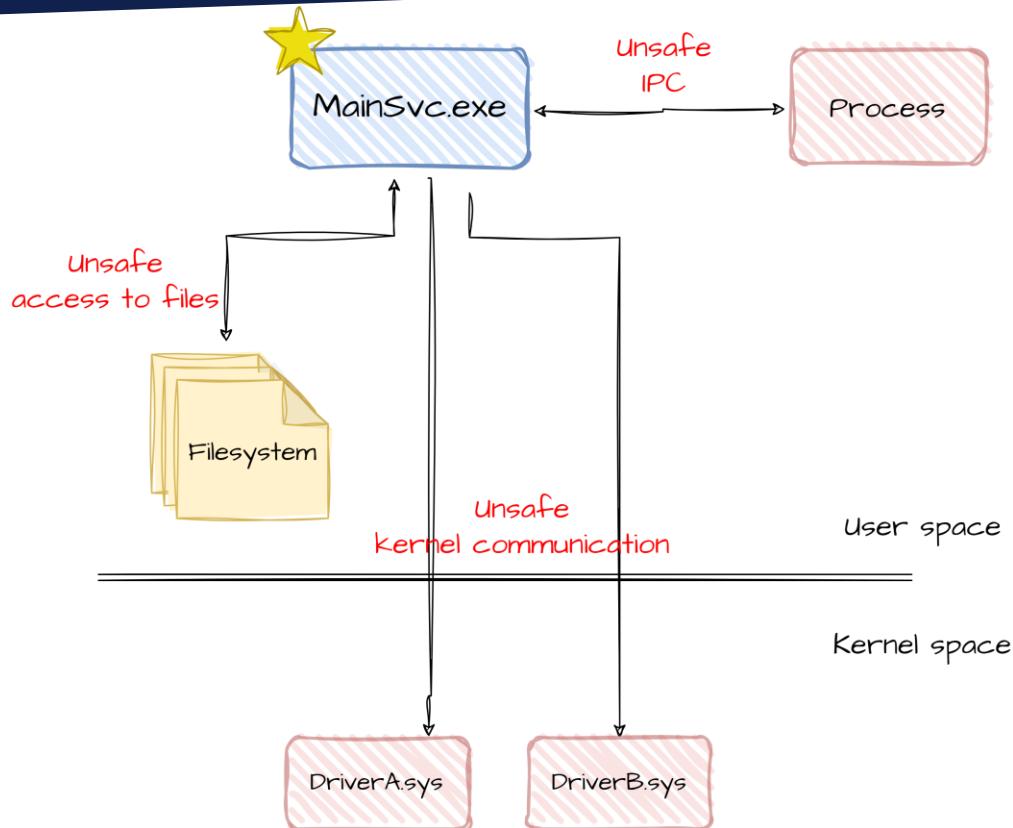
A tour of a commercial product

USE CASE – UNDISCLOSED COMMERCIAL AV PRODUCT



- Widely used antivirus product (occurs in Tops)
- Reporting process still **on-going** (started in August 2023)

USE CASE – CARTOGRAPHY FINDINGS



Cartography results on the use case

USE CASE – VULNERABILITIES SUMMARY

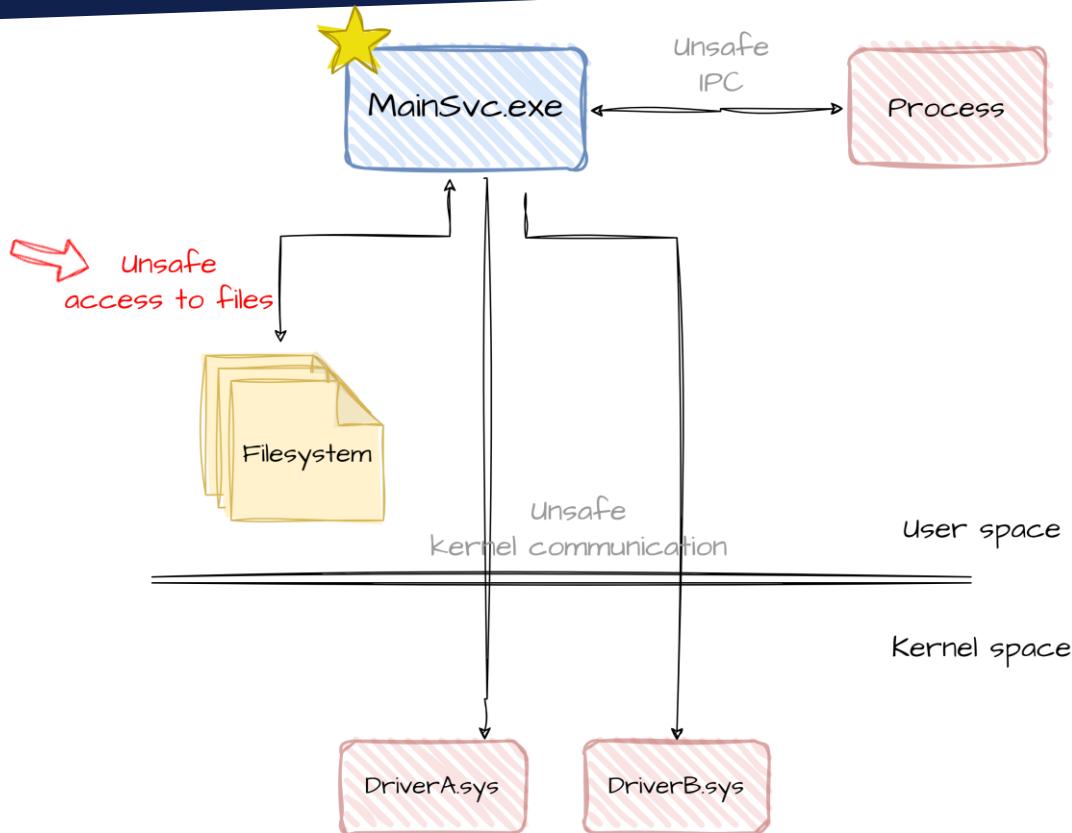
- 5 vulnerabilities
 - Elevation of privileges (x2)
 - Arbitrary kernel code execution
 - Denial of service (x2)
- And more
 - Bad crypto implementation
 - Service hijacking

Vulnerability #1

Arbitrary file write

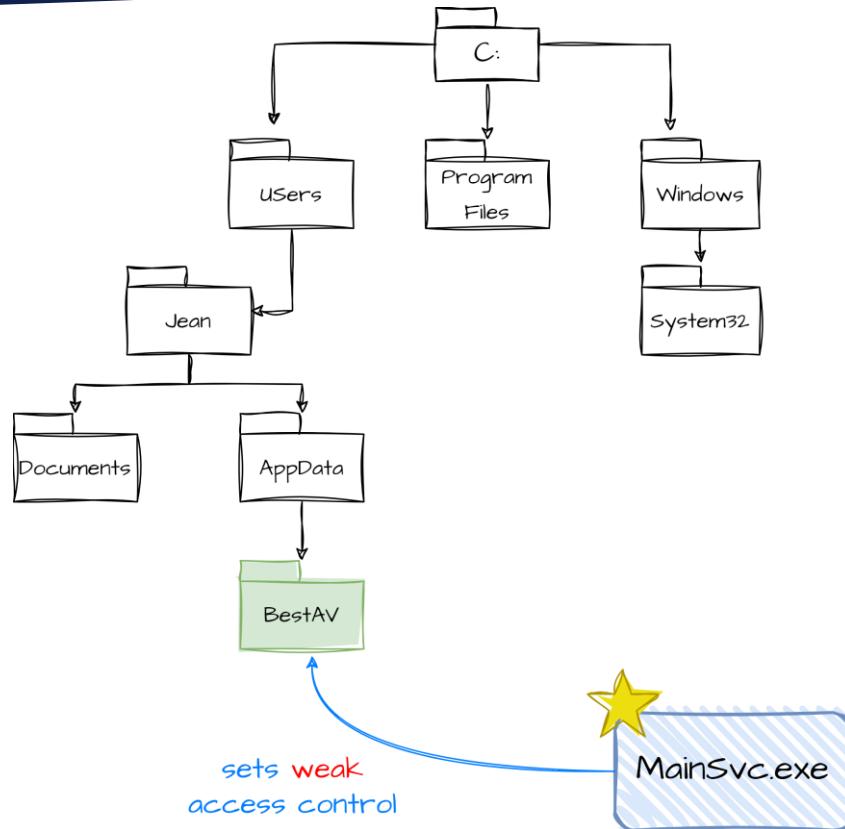
NTFS Junction + Object Manager symbolic link

ARBITRARY FILE WRITE – CARTOGRAPHY FINDING 1



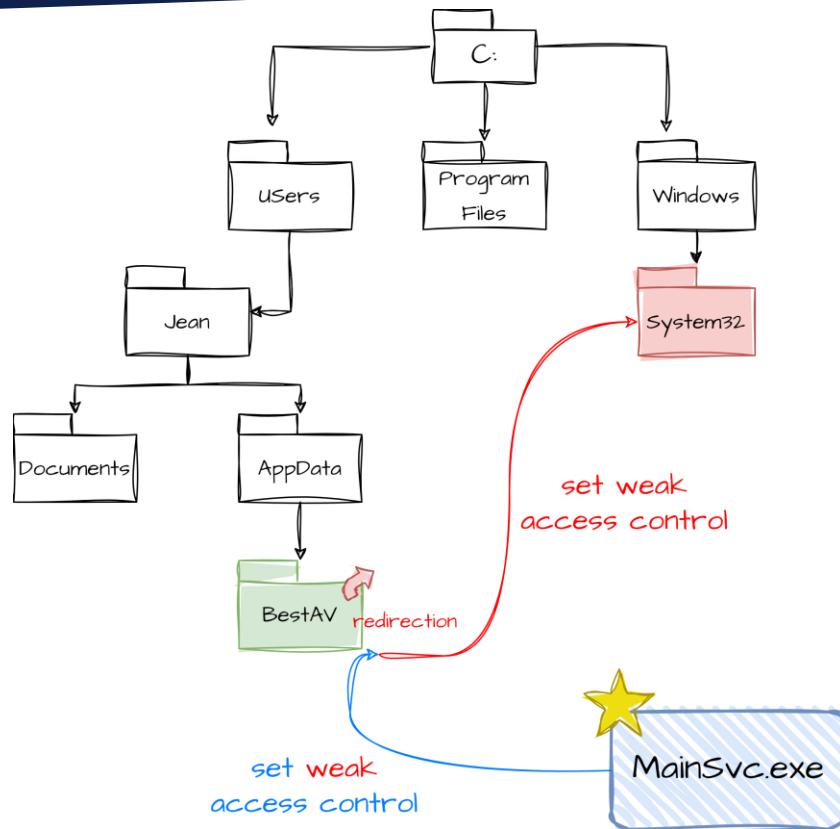
Cartography outlined unsafe access to files

ARBITRARY FILE WRITE



Too permissive access control

ARBITRARY FILE WRITE



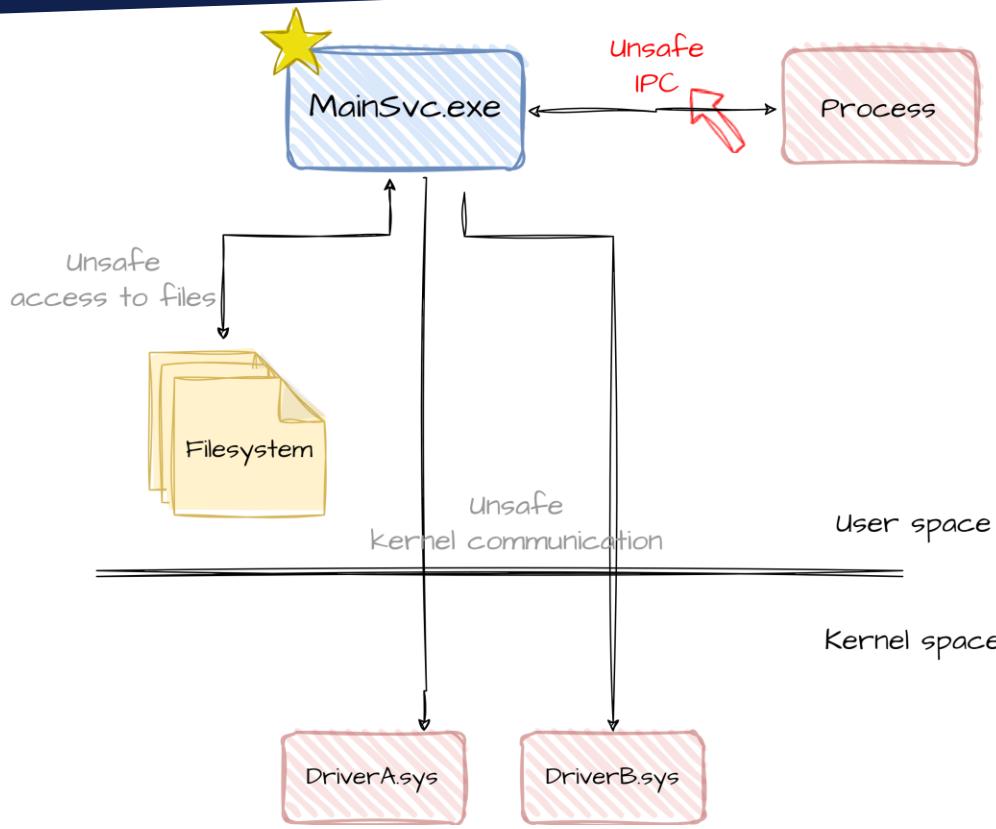
Redirection of access control rule

Vulnerability #2

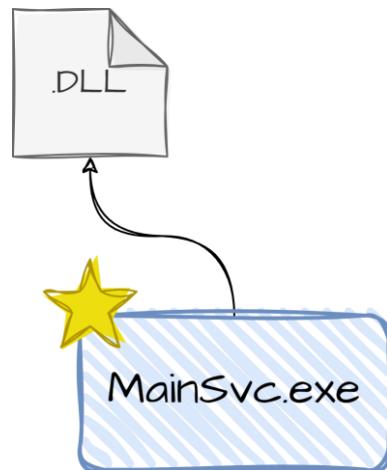
DLL hijacking

Bypassing AV kernel protections

DLL HIJACKING – CARTOGRAPHY FINDING 2

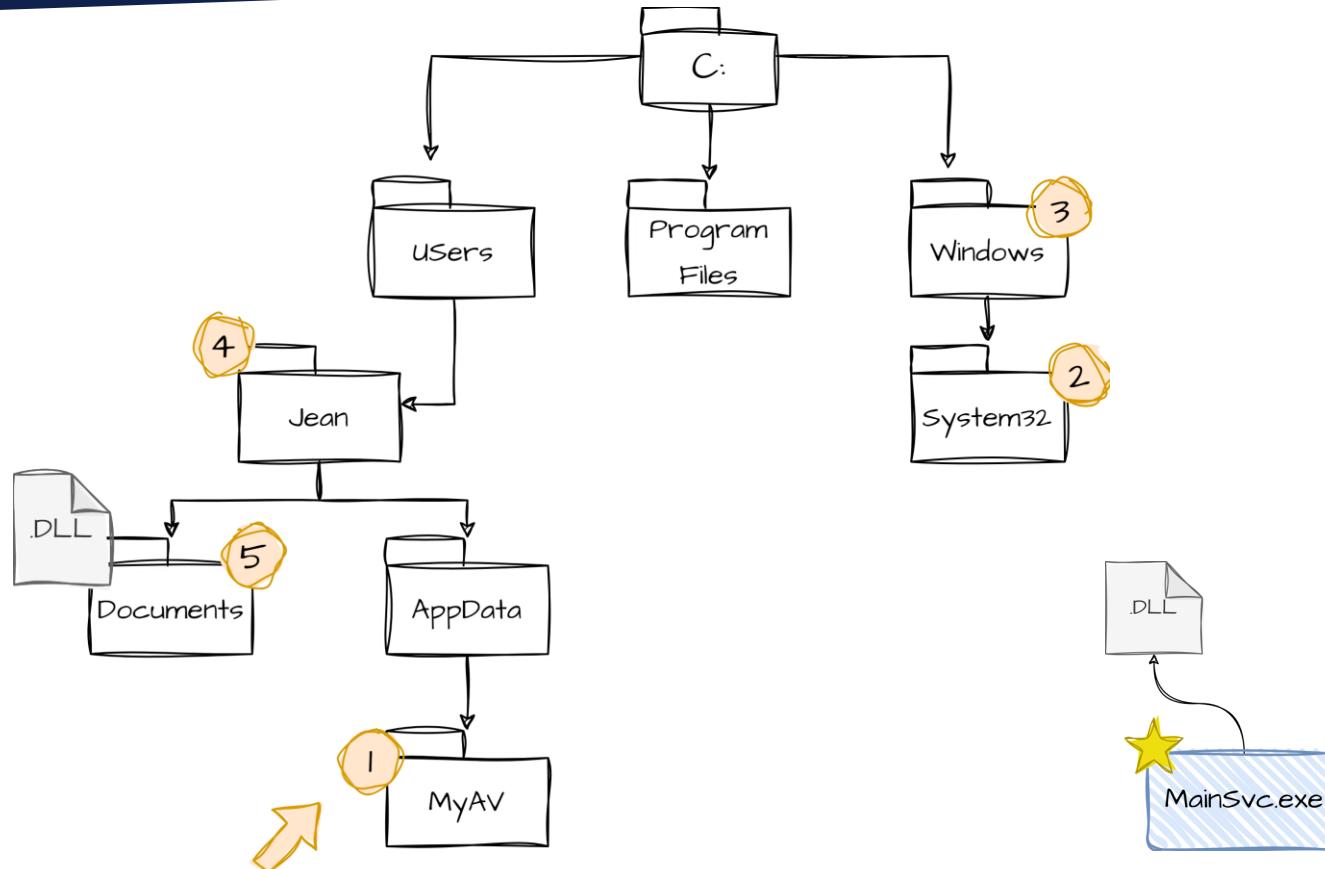


DLL HIJACKING – DLL LOADING

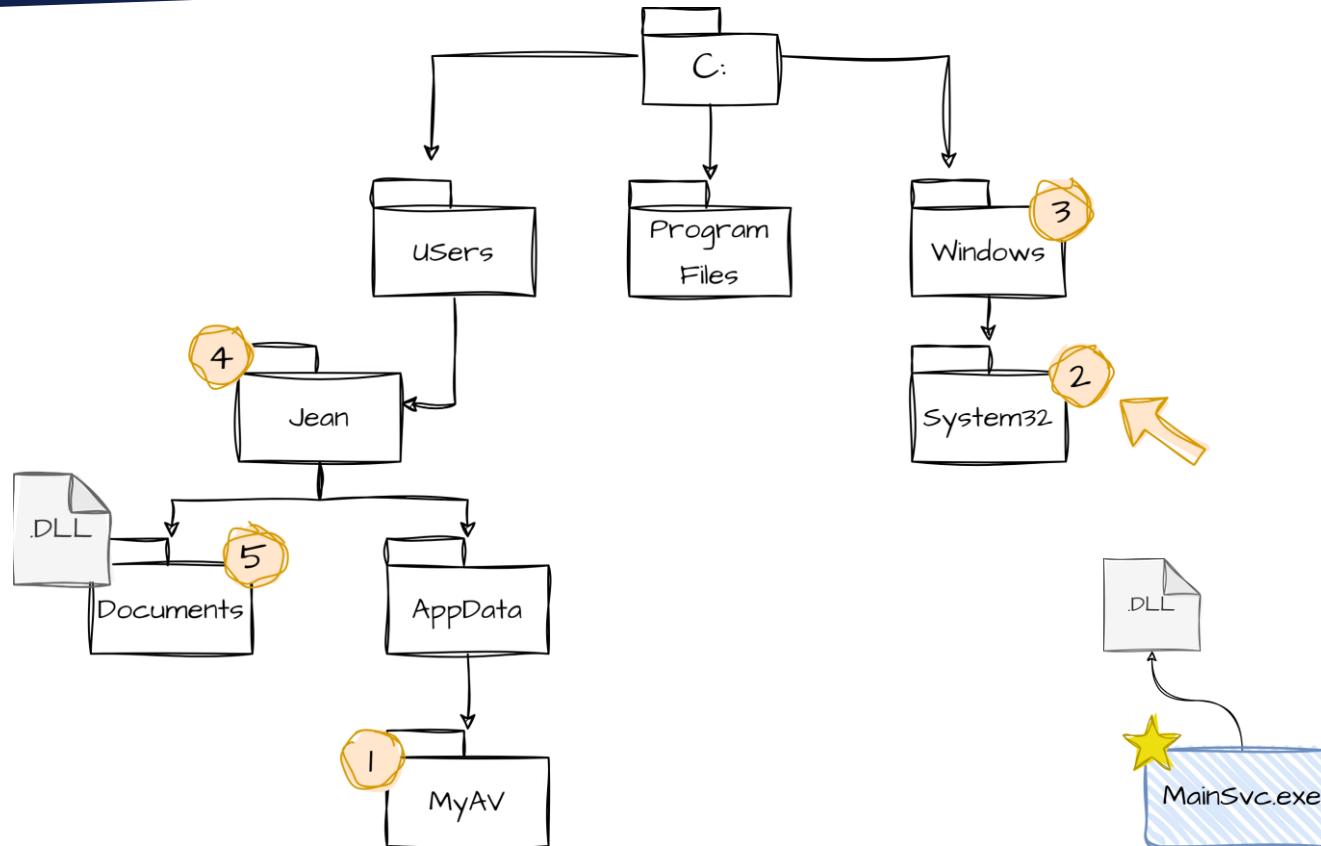


A process loading a DLL

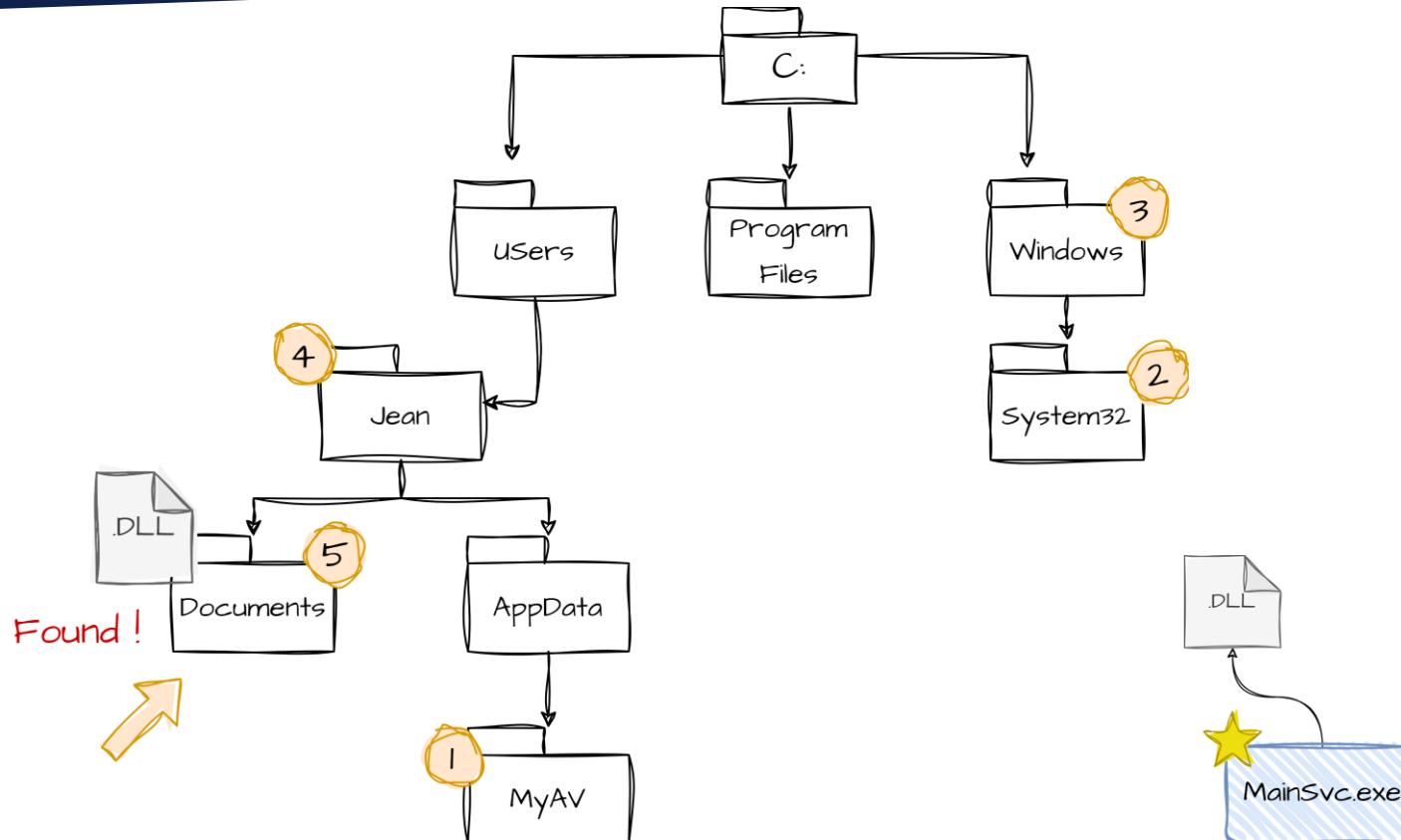
DLL HIJACKING – SEARCH ORDER



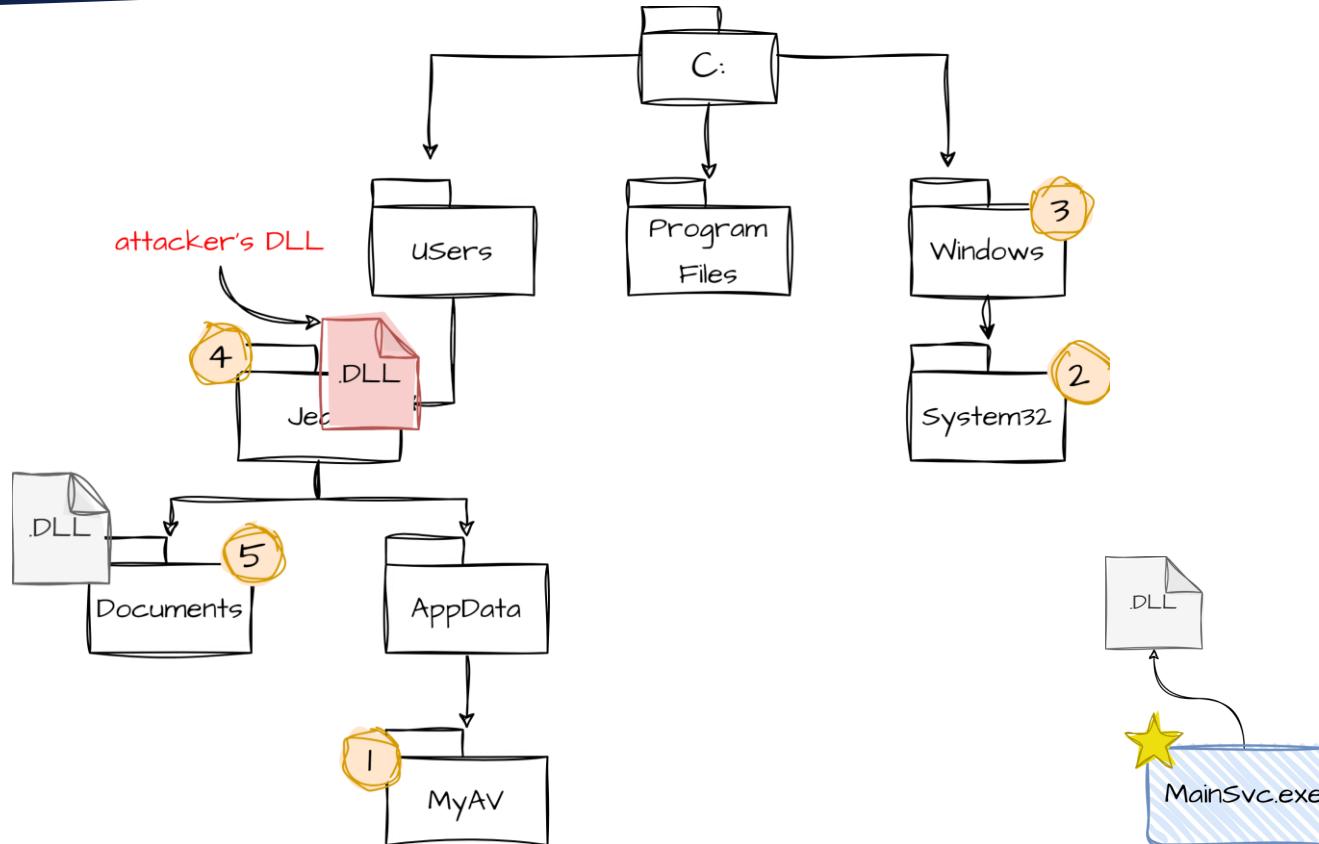
DLL HIJACKING – SEARCH ORDER



DLL HIJACKING – SEARCH ORDER



DLL HIJACKING – LOADING MALICIOUS FILE



DLL HIJACKING

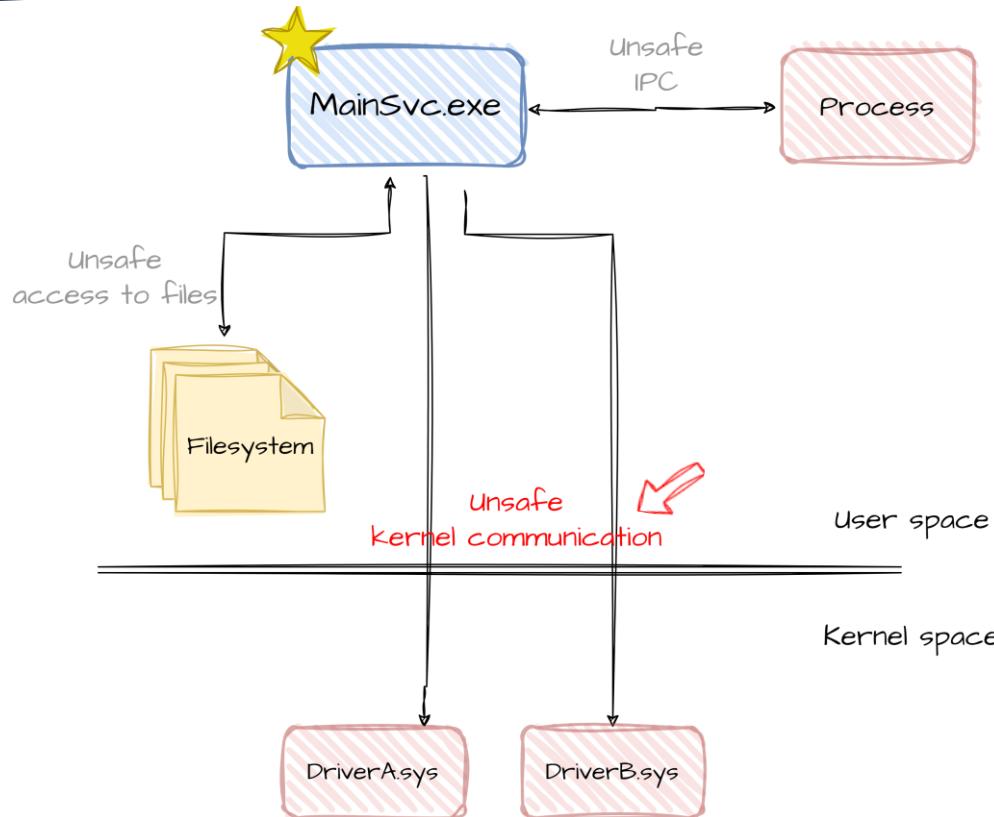
- Well known technique
- We also loaded drivers
 - **Foreign signed drivers**
- Bypass of driver protection
 - Mini-filter based

Vulnerability #3

Type confusion

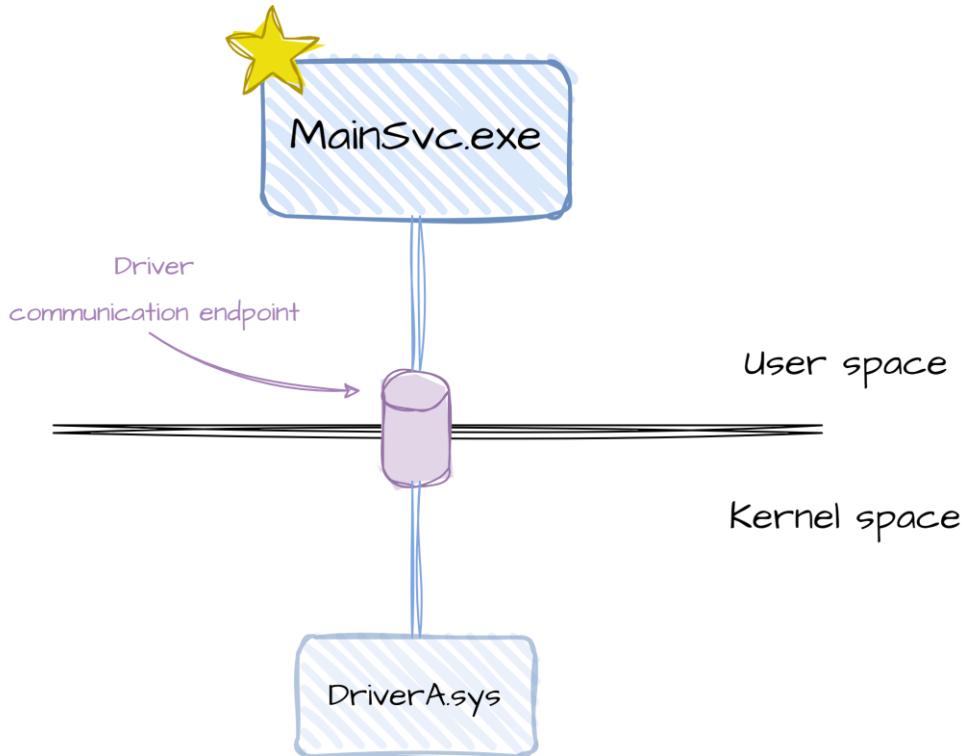
Denial of service

TYPE CONFUSION - CARTOGRAPHY FINDING 3

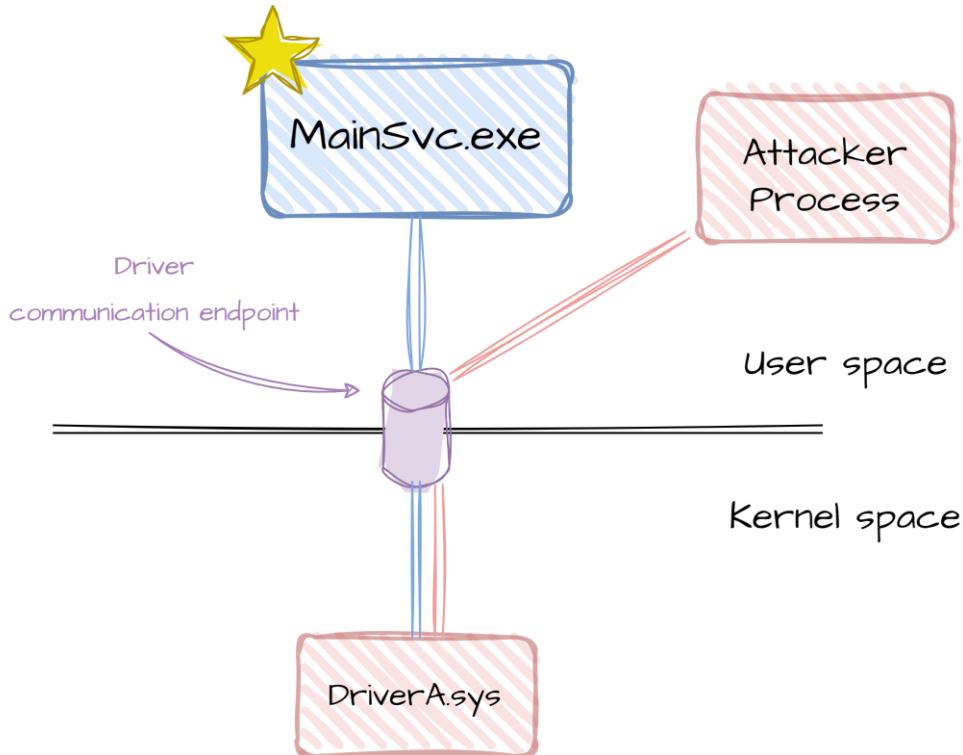


Cartography outlined unsafe kernel communication

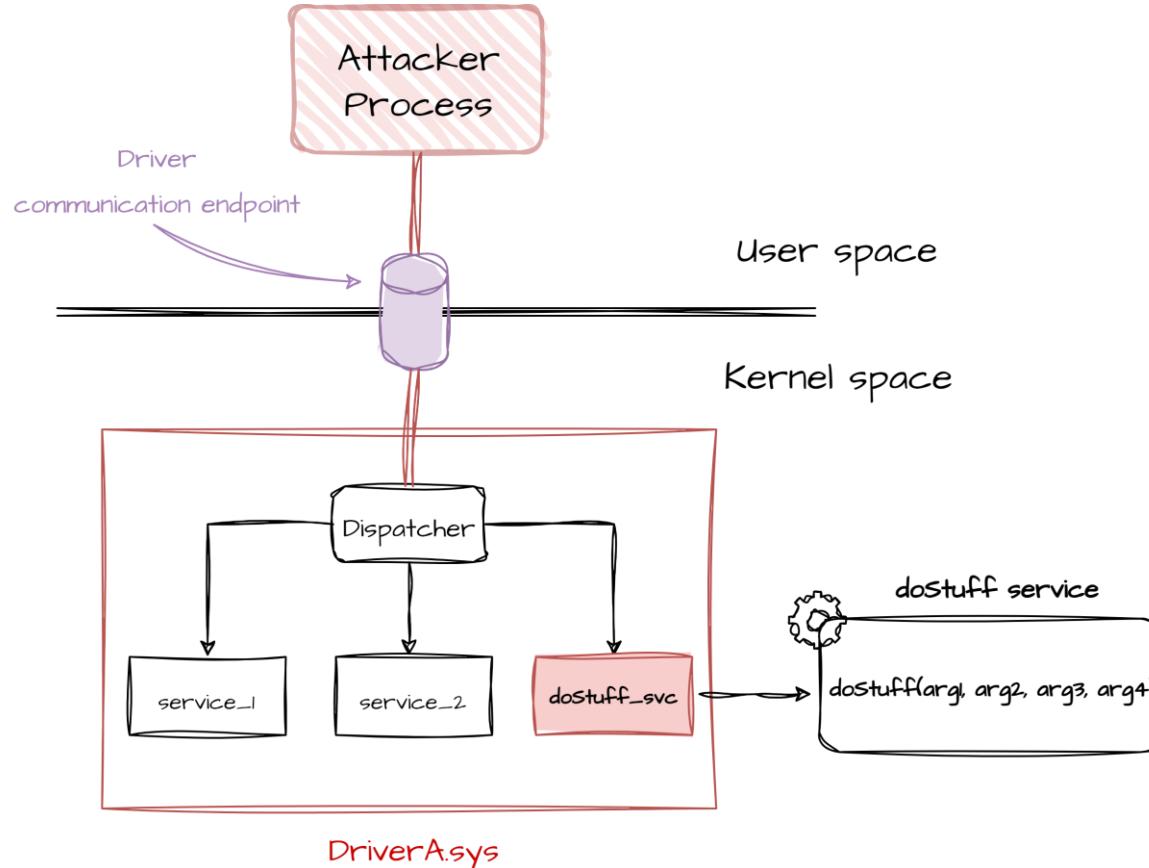
BRIEF ASIDE – DRIVER COMMUNICATION



BRIEF ASIDE – DRIVER COMMUNICATION



BRIEF ASIDE – DRIVER COMMUNICATION



TYPE CONFUSION - CARTOGRAPHY FINDING 3

- Found a IOCTL with those characteristics :

```
struct PACKET_1 {  
    HANDLE handle;  
    PVOID ptr;  
    DWORD size;  
    DWORD unknown0;  
    ULONGLONG unknown1;  
};
```

TYPE CONFUSION - CARTOGRAPHY FINDING 3

- Found a IOCTL with those characteristics :

- ptr is probed



```
struct PACKET_1 {  
    HANDLE handle;  
    PVOID ptr;  
    DWORD size;  
    DWORD unknown0;  
    ULONGLONG unknown1;  
};
```

TYPE CONFUSION - CARTOGRAPHY FINDING 3

- Found a IOCTL with those characteristics :

- **ptr** is probed X
- **size** is checked X

```
struct PACKET_1 {  
    HANDLE handle;  
    PVOID ptr;  
    DWORD size;  
    DWORD unknown0;  
    ULONGLONG unknown1;  
};
```

TYPE CONFUSION - CARTOGRAPHY FINDING 3

- Found a IOCTL with those characteristics :

- **ptr** is probed 
- **size** is checked 
- **handle** is not ! 

```
struct PACKET_1 {  
    HANDLE handle;  
    PVOID ptr;  
    DWORD size;  
    DWORD unknown0;  
    ULONGLONG unknown1;  
};
```

TYPE CONFUSION - CARTOGRAPHY FINDING 3

- Handle is used directly with **ObReferenceObjectByHandle**

```
if ((packet != 0 && out_val != 0))
{
    void* pObject;
    NTSTATUS status = ObReferenceObjectByHandle(packet->handle, 0, NULL, 1, &pObject, NULL);
}
```

- And ?

TYPE CONFUSION - CARTOGRAPHY FINDING 3

```
NTSTATUS ObReferenceObjectByHandle(  
    HANDLE Handle,  
    ACCESS_MASK DesiredAccess,  
    POBJECT_TYPE ObjectType,  
    KPROCESSOR_MODE AccessMode,  
    PVOID *Object,  
    POBJECT_HANDLE_INFORMATION HandleInformation  
) ;
```

- "*If ObjectType is not NULL, the operating system verifies that the supplied object type matches the object type of the object that Handle specifies.*"
- And if not ? **Arbitrary handle => Pointer to unexpected object**



TYPE CONFUSION - CARTOGRAPHY FINDING 3

Expects to receive a _FILE_OBJECT...

```
struct _FILE_OBJECT
{
    SHORT Type;
    SHORT Size;
    struct _DEVICE_OBJECT* DeviceObject;

    ...
}
```

```
int32_t _status = doStuff(arg1: a_int, PDEVICE_OBJECT: Object->_DEVICE_OBJECT, ...);
```



TYPE CONFUSION - CARTOGRAPHY FINDING 3

Expects to receive a _FILE_OBJECT... so we give it a _TOKEN !

```
struct _FILE_OBJECT                                struct _TOKEN
{                                                 {
    SHORT Type;                                     CHAR SourceName[8];           // 0x0
    SHORT Size;                                      struct _LUID SourceIdentifier; // 0x8
    struct _DEVICE_OBJECT* DeviceObject;             } _TOKEN_SOURCE ;
    ...                                              ...
}}
```

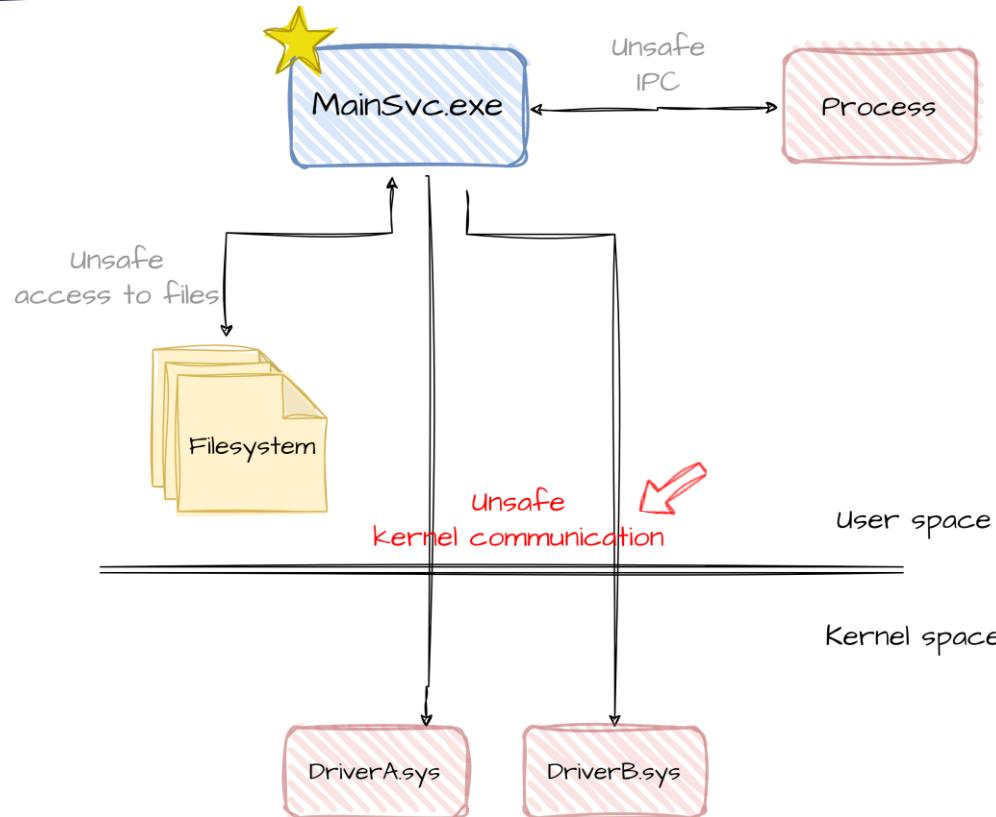
```
int32_t _status = doStuff(arg1: a_int, PDEVICE_OBJECT: Object->SourceIdentifier, ...);
```

Vulnerabilities #4

Kernel code execution

Write-What-Where

KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4



Cartography outlined unsafe kernel communication

KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4

```
struct PACKET2
{
    void* va_address_destination;
    void* va_address_source;
    unsigned int size;
}
```



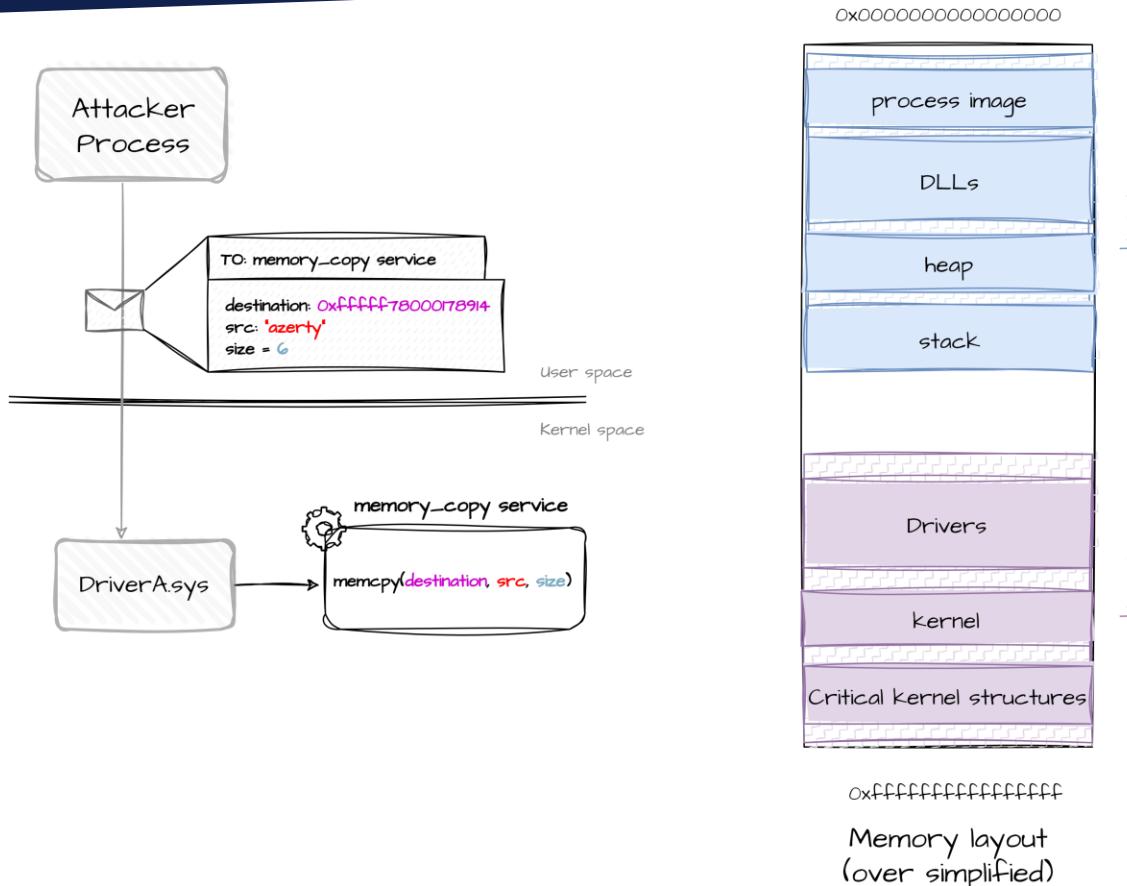


KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4

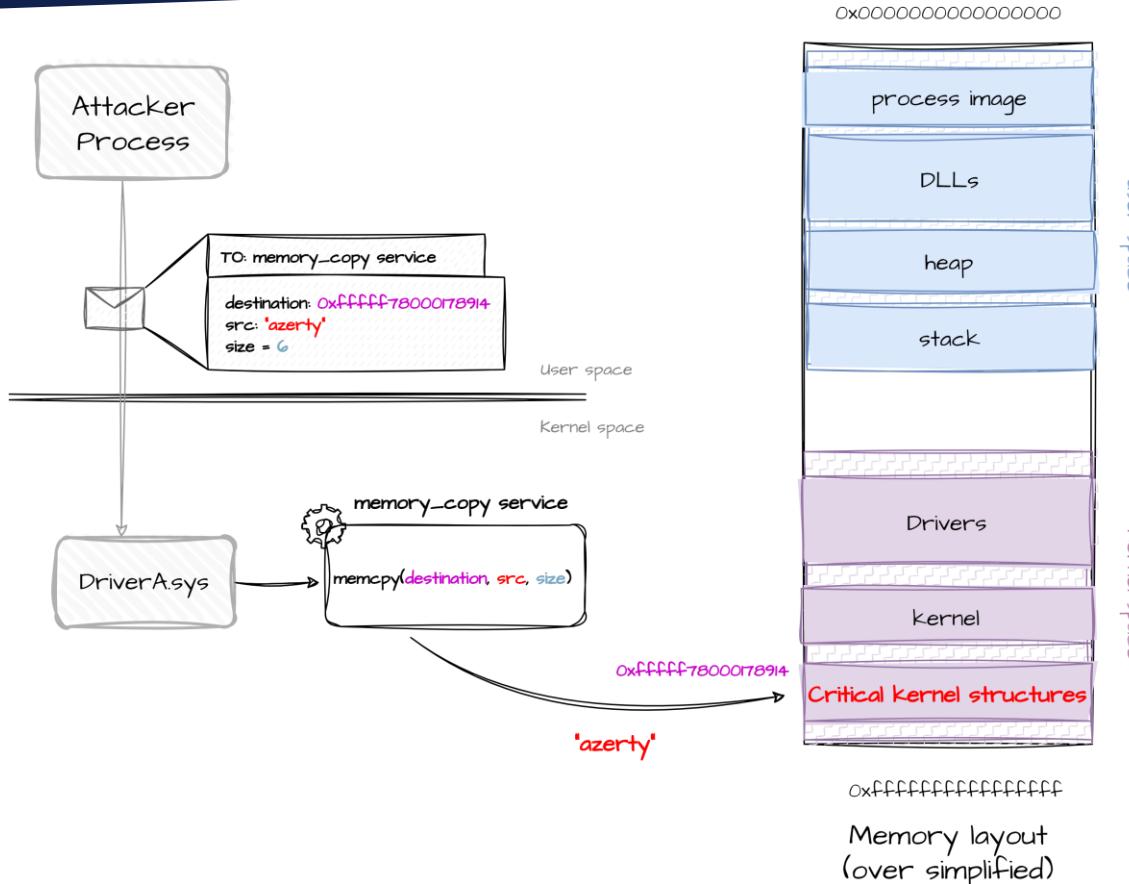
```
mdl_address_source = IoAllocateMdl(packet->va_address_source, packet->size, 0, AccessMode_UserMode, 0);
MmProbeAndLockPages(mdl_address_source, AccessMode_UserMode, 0);
void* va_system_address_source = MmMapLockedPagesSpecifyCache(mdl_address_source, ...);

int64_t phy_address_destination = MmGetPhysicalAddress(packet->va_address_destination);
if /* some conditions*/
{
    int64_t va_system_address_destination = MmMapIoSpace(phy_address_destination, packet->size, 0);
    __builtin_memcpy(va_system_address_destination, va_system_address_source, packet->size);
}
```

KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4



KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4



KERNEL CODE EXECUTION – CARTOGRAPHY FINDING 4

- **Veeeery** powerful primitive
- Privilege escalation : easy
- Code execution : not that hard but
 - `MmMapIoSpace()` can't map page tables
→ Can't get rid of DEP nor SMEP
 - One solution: get a **new write primitive** from the first one

Conclusion

CONCLUSION

- AVs are huge and complex by nature
 - ➡ **Attack surface is really important
(can even lower overall security)**
- Loads of stuff to check
 - ➡ **Audits can be tedious and time-consuming**

CONCLUSION

- Find the entry points and prioritize them
 - ➡ **Perform a thorough cartography**
- Not necessarily over-complicated!
 - ➡ **Being smart brings results**

Thank you

Contact information:

Email: contact@quarkslab.com

Phone: +33 1 58 30 81 51

Website: quarkslab.com



@quarkslab