Bypassing Anti-Cheats & Hacking Competitive Games

Bypassing Anti-Cheats & Hacking Competitive Games

~~Destroying your opponents~~


8:40
306
7
100
25

Hello! I'm...

Rohan Aggarwal @nahoragg

Founder at DefCore Security

Bug Bounties

Live Hacking Event(H1 & Intigriti)

Love playing games(Valorant, Apex Legends, etc)

Agenda

- Get started in game hacking
- Demonstrate working methods of bypassing AC
- Encourage more research in this field.
- Encourage AC makers to produce more powerful AC system.

What kind of game hacking?

Outline

01 - Cheats vs Anti-Cheats

02 - Game Hacking Basics

03 - Cheat Development

04 - Demonstration & Release

01

Cheats vs Anti-Cheats

History of game hacking

```
00000000
00000000 B8 C0 07 8E D8 B8 00 90 8E C0 B9 00 01 29 F6 29 FF FC F3 A5 EA 19  .............).).......
00000016 00 00 90 8C C8 BA F4 3F 8E D8 8E C0 8E D0 89 D4 6A 00 0F A1 BB 78  .......?........j....x
0000002C 00 64 0F B5 37 89 D7 B9 06 00 FC 65 F3 A5 89 D7 C6 45 04 12 64 89  .d..7......e.....E..d.
00000042 3F 64 8C 47 02 8C C8 8E E0 8E E8 30 E4 30 D2 CD 13 31 D2 B9 02 00  ?d.G.......0.0...1....
00000058 BB 00 02 B8 04 02 CD 13 73 12 50 E8 2F 01 89 E5 E8 34 01 58 30 D2  ........s.P./....4.X0.
0000006E 30 E4 CD 13 EB DF 31 D2 B9 12 00 BB 00 0A B8 01 02 CD 13 73 0B B1  0.....1............s..
00000084 0F B8 01 02 CD 13 73 02 B1 09 2E 89 0E CE 01 B8 00 90 8E C0 B4 03  ......s.............
0000009A 30 FF CD 10 B9 09 00 BB 07 00 BD D0 01 B8 01 13 CD 10 B8 00 10 8E  0.......................
000000B0 C0 E8 14 00 E8 04 01 E8 0A 01 E8 D8 00 EA 00 00 20 90 05 00 00 00  ................ .....
000000C6 00 00 8C C0 A9 FF 0F 75 FE 31 DB 8C C0 2D 00 10 3B 06 F4 01 76 01  .......u.1...-..;...v.
000000DC C3 2E A1 CE 01 2B 06 C2 00 89 C1 C1 E1 09 01 D9 73 09 74 07 31 C0  .....+..........s.t.1.
000000F2 29 D8 C1 E8 09 E8 34 00 89 C1 03 06 C2 00 2E 3B 06 CE 01 75 12 B8  ).....4........;...u..
00000108 01 00 2B 06 C4 00 75 04 FF 06 C6 00 A3 C4 00 31 C0 A3 C2 00 C1 E1  ..+...u........1......
0000011E 09 01 CB 73 AE 8C C0 80 C4 10 8E C0 31 DB EB A3 60 60 B8 2E 0E BB  ...s........1...``....
00000134 07 00 CD 10 61 8B 16 C6 00 8B 0E C2 00 41 88 D5 8B 16 C4 00 88 D6  ....a........A........
0000014A 81 E2 00 01 B4 02 52 51 53 50 CD 13 72 05 83 C4 08 61 C3 50 E8 0C  ......RQSP..r....a.P..
00000160 00 30 E4 30 D2 CD 13 83 C4 0A 61 EB C1 B9 05 00 89 E5 51 E8 1F 00  .0.0......a.......Q...
00000176 3A 0E 05 00 73 0F B8 45 0E 28 C8 CD 10 B0 58 CD 10 B0 3A CD 10 83  :...s..E.(....X...:...
0000018C C5 02 E8 0E 00 59 E2 DE C3 B8 0D 0E CD 10 B0 0A CD 10 C3 B9 04 00  .....Y..................
000001A2 8B 56 00 C1 C2 04 B4 0E 88 D0 24 0F 04 30 3C 39 76 02 04 07 CD 10  .V........$..0<9v......
000001B8 E2 EB C3 52 BA F2 03 30 C0 EE 5A C3 B4 01 B7 00 B9 00 20 CD 10 C3  ...R...0..Z....... ....
000001CE 00 00 0D 0A 4C 6F 61 64 69 6E 67 00 00 00 00 00 00 00 00 00 00 00  ....Loading............
000001E4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 00 00 FB 2B 00 00 00  ....................+....
000001FA 00 00 00 00 55 AA FA B0 80 E6 70 B8 00 90 8E D8 8E C0 8E E0 8E D0  ....U.....p............
00000210 89 D4 0E 1F 0F 01 1E A2 00 0F 01 16 A8 00 BA 92 00 EC 3C FF 74 12  ....................<.t.
00000226 67 8A 64 24 04 84 E4 74 04 0C 02 EB 02 24 FD 24 FE EE E8 2B 00 B0  g.d$...t.....$.$...+..
0000023C D1 E6 64 E8 24 00 B0 DF E6 60 E8 1D 00 B8 01 00 0F 01 F0 EB 00 B8  ..d.$....`..............
00000252 18 00 8E D8 8E C0 8E D0 8E E0 8E E8 66 EA 00 00 01 00 10 00 E8 16  ............f.........
00000268 00 E4 64 3C FF 74 0F A8 01 74 07 E8 09 00 E4 60 EB EC A8 02 75 E8  ..d<.t...t.....`....u.
0000027E C3 EB 00 C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF 7F  ......................
```

Help | Save | Open | Goto | Find | Hex Addr | Hex Edit | Quit


RTM
BOT
DOOM

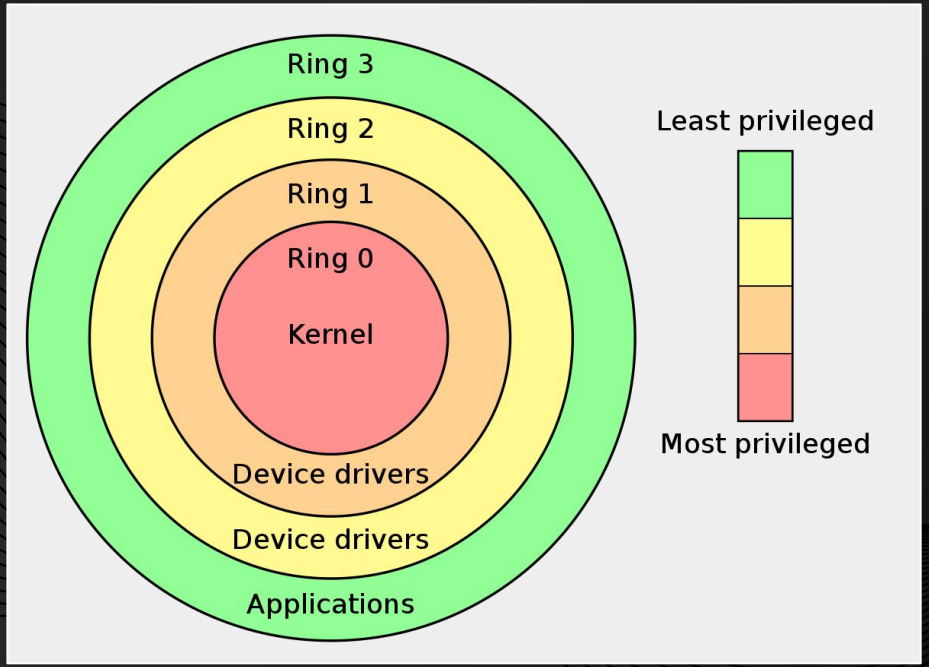

Finn
usp
Zane
glock18
Dave
glock18


PunkBuster
Online Countermeasures


VAC

easy™
ANTI-CHEAT

User Mode	Kernel Mode
Restricted access to system resources	direct and unrestricted access to system resources
Private virtual space for each process	Single virtual space for whole kernel



Features of Kernel AC

- Blocking / stripping of process handles in UM
- Detection of test signing
- Detection of usermode hooks
- Detection of injected modules
- Detection of manually mapped modules
- Detection of kernel drivers
- Detecting of traces of manually mapped drivers
- Detection of virtual machines and emulation

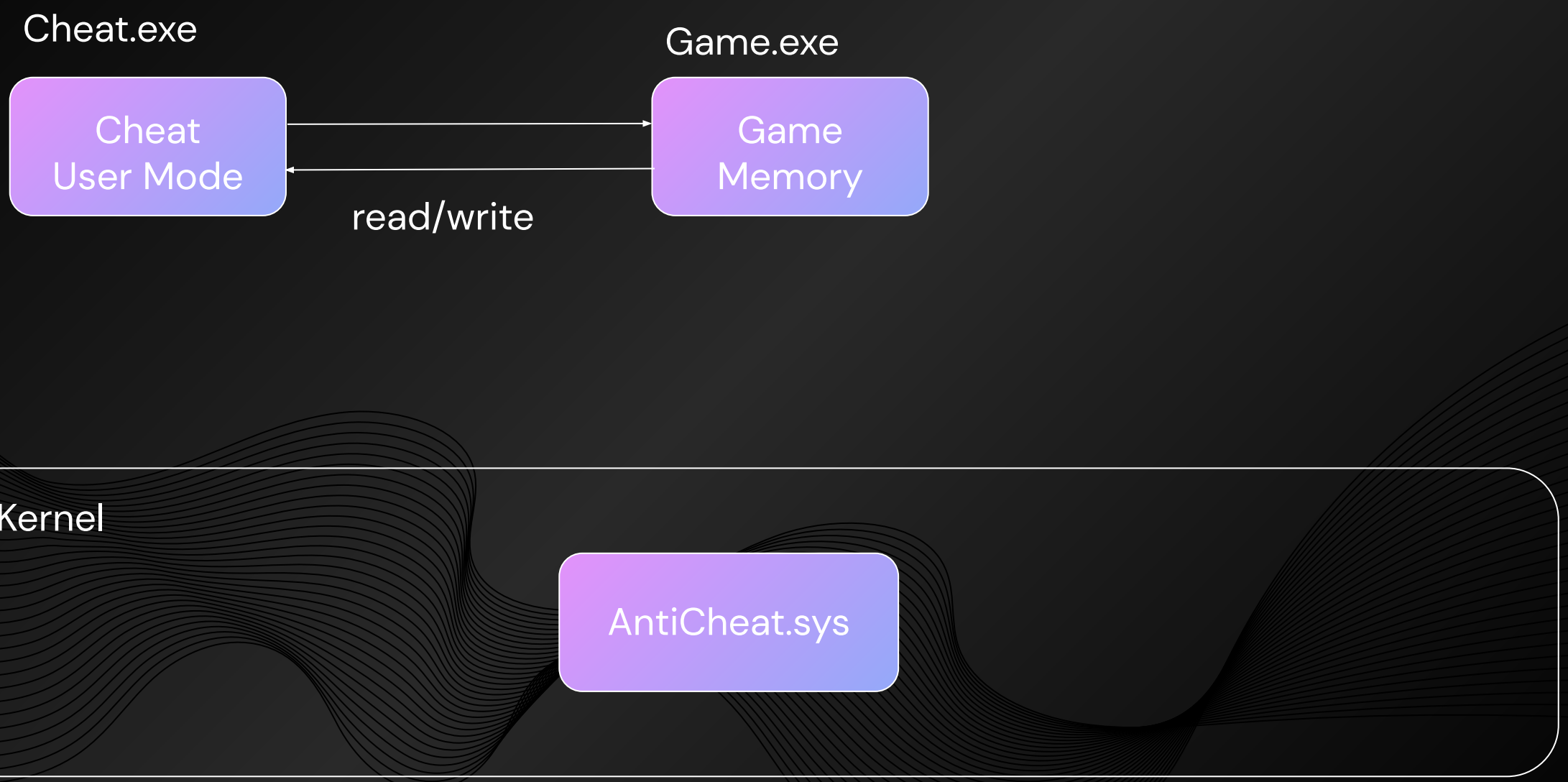
02

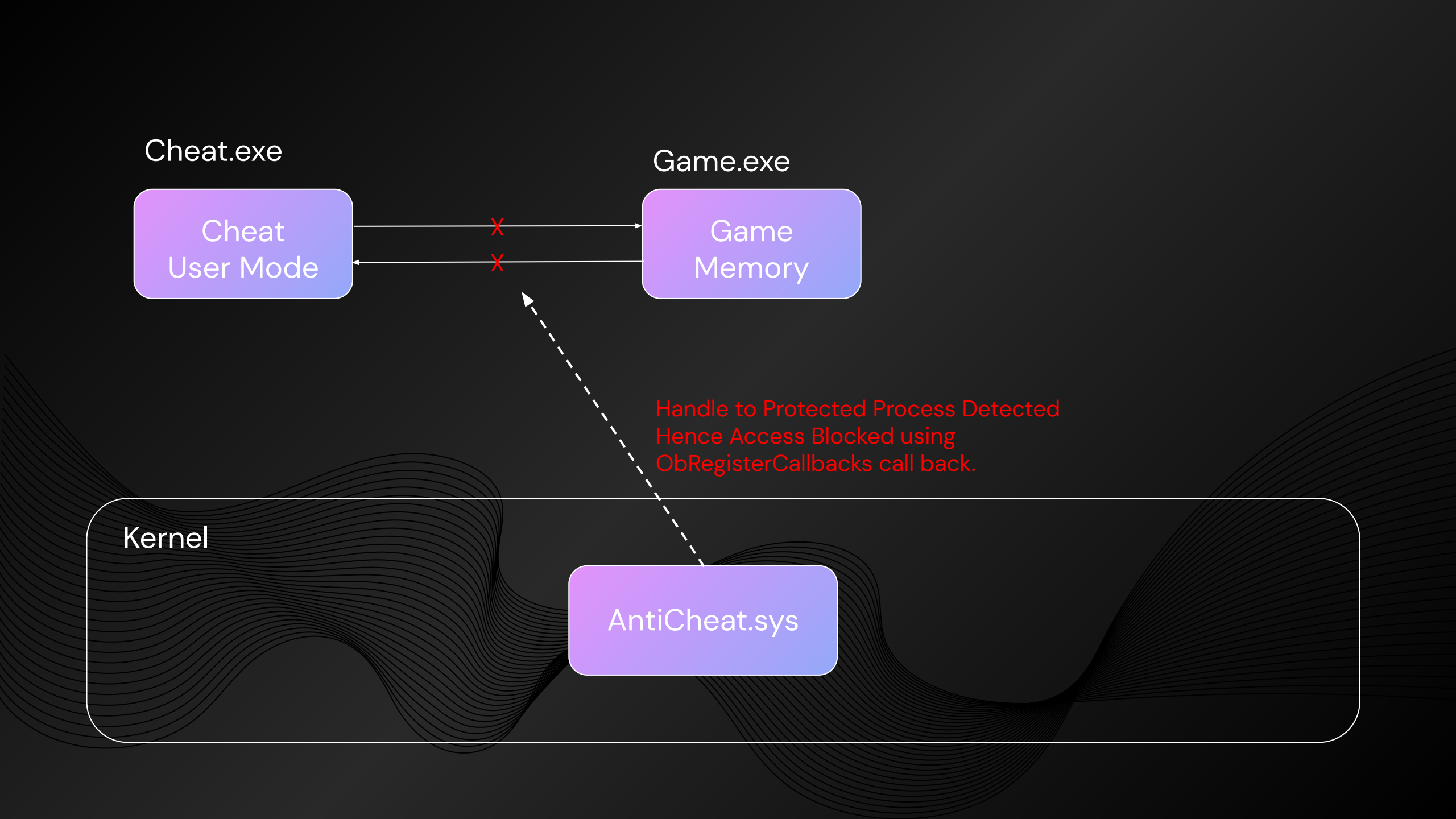
Game Hacking Basics

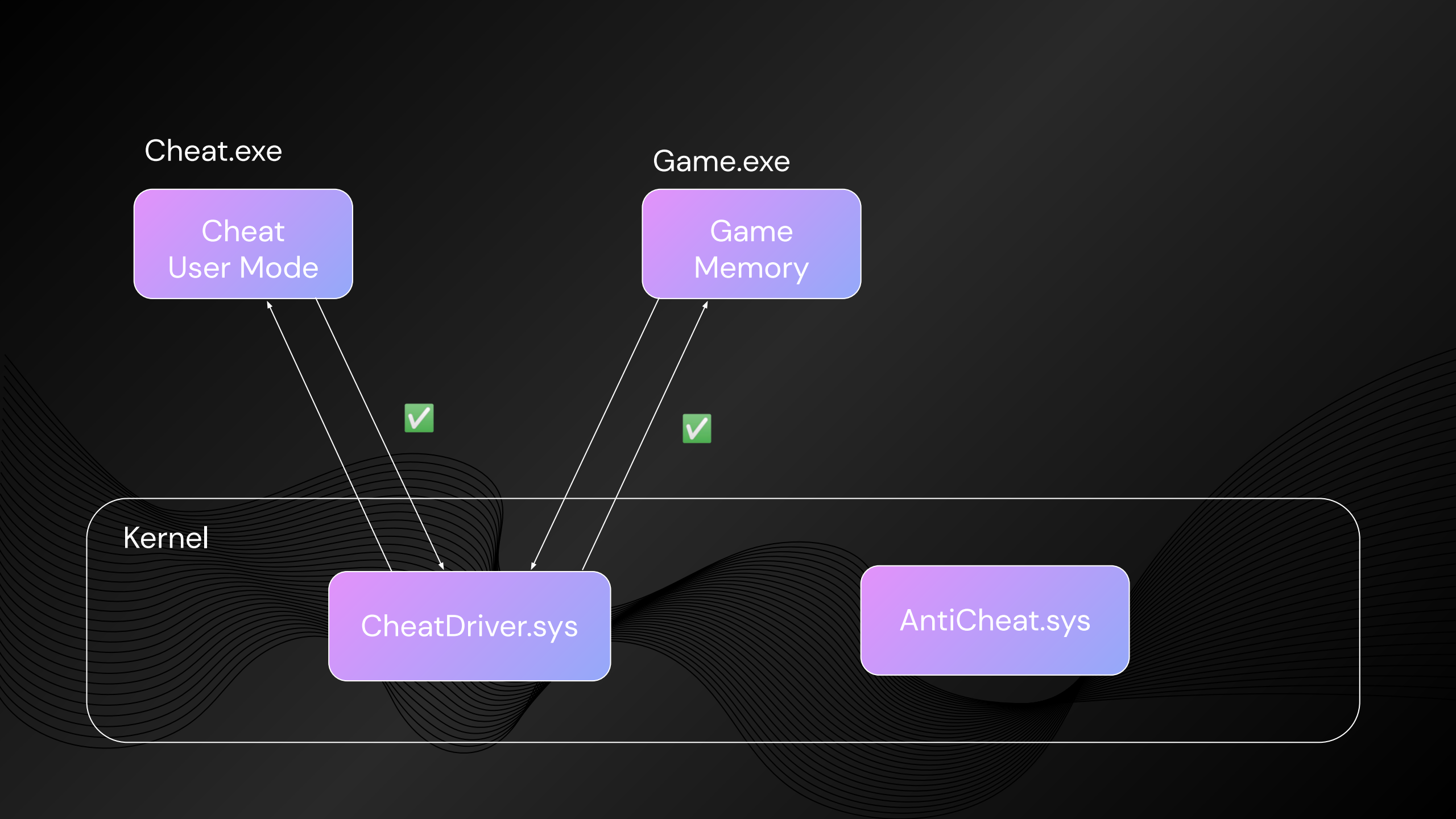
Types of Cheats

Internal	External
Injected into the target process itself.	Have their own process that manipulates target process
Complex to make(depends on engine)	Easy to make(Any lang.)
Much Flexible and great performance	Enough Flexibility and performance
Prefered for Games with low-level or No AC	Prefered for Games with Strong AC

General method of bypassing AC


Cheat.exe
Game.exe
Cheat
User Mode
Game
Memory
read/write
Kernel
AntiCheat.sys


Cheat.exe
Game.exe
Cheat
User Mode
Game
Memory
Handle to Protected Process Detected
Hence Access Blocked using
ObRegisterCallbacks call back.
Kernel
AntiCheat.sys


Cheat.exe
Cheat
User Mode
Game.exe
Game
Memory
Kernel
CheatDriver.sys
AntiCheat.sys

Challenges

1. How to create driver?
2. How to load the driver into kernel?
3. How to communicate from user mode to kernel mode?
4. How I can make my driver Undetected?

03

Cheat Development

Let’s develop

1. Reversing

Reversing Game to find the offsets for required cheat.

2. Creating Driver

Creating Custom Driver that read/write to game memory from kernel.

3. Hooking

Hooking to system call function and placing our shell code for establishing communication b/w UM & KM.

4. Loading Driver

Loading the Driver into Kernel.

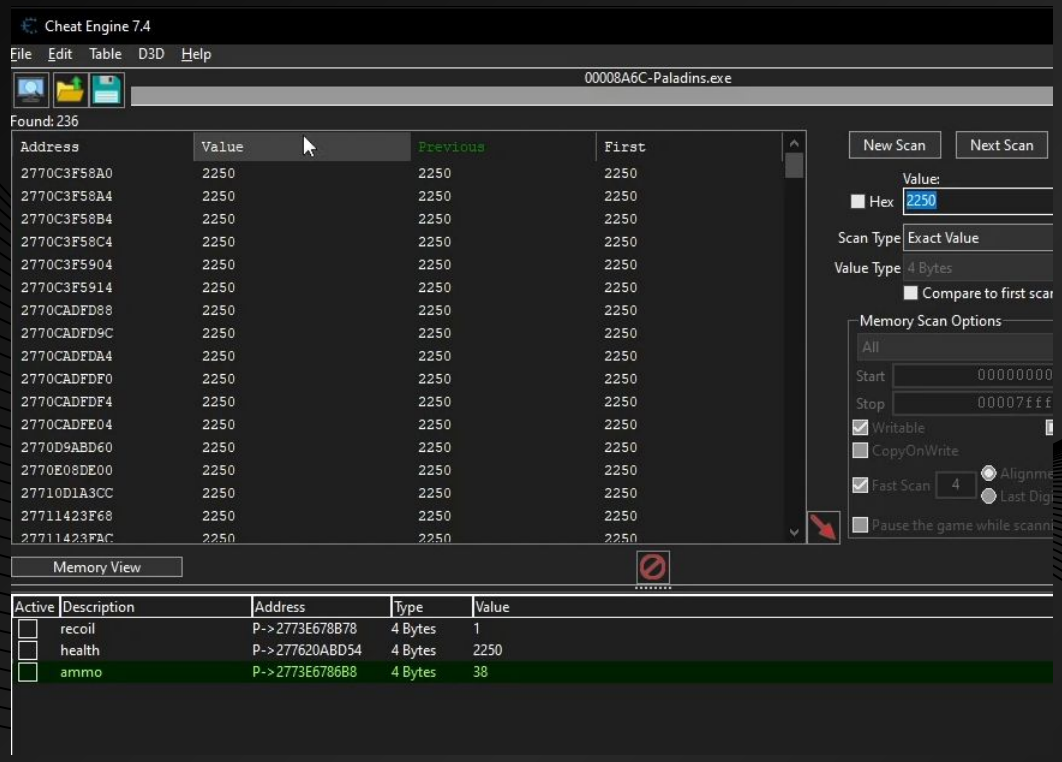
5. Creating User Mode

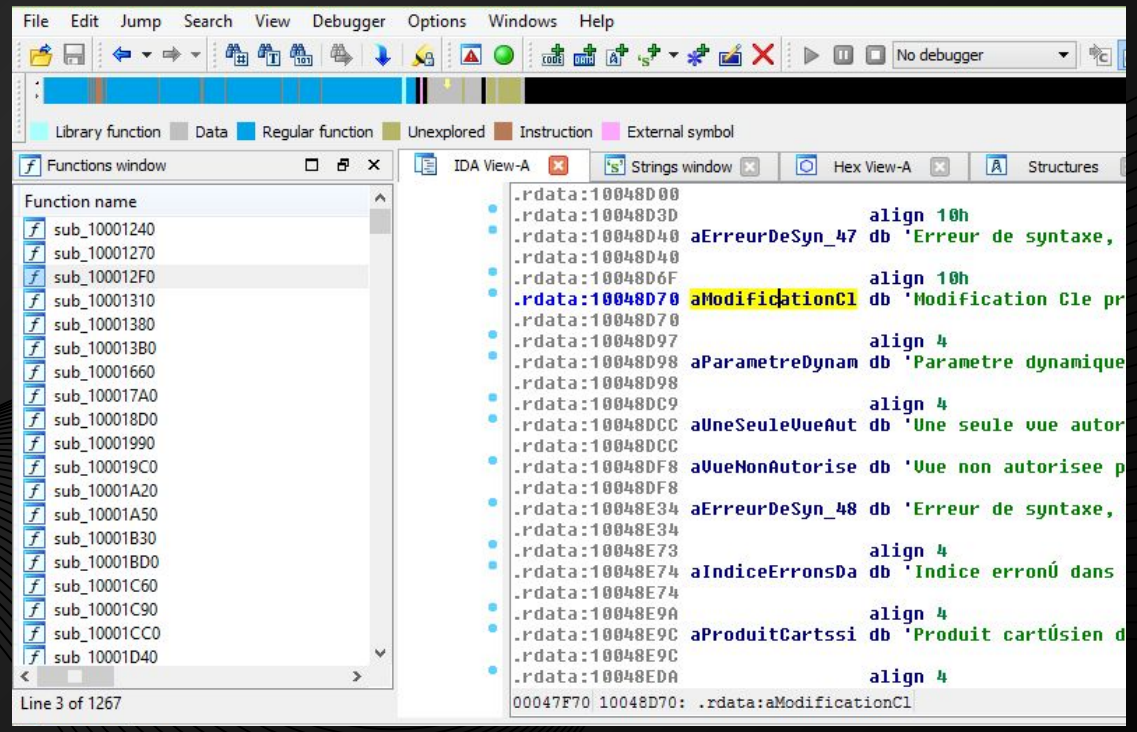
Creating User Mode that sends read/write request to kernel mode.

1. Reversing Game

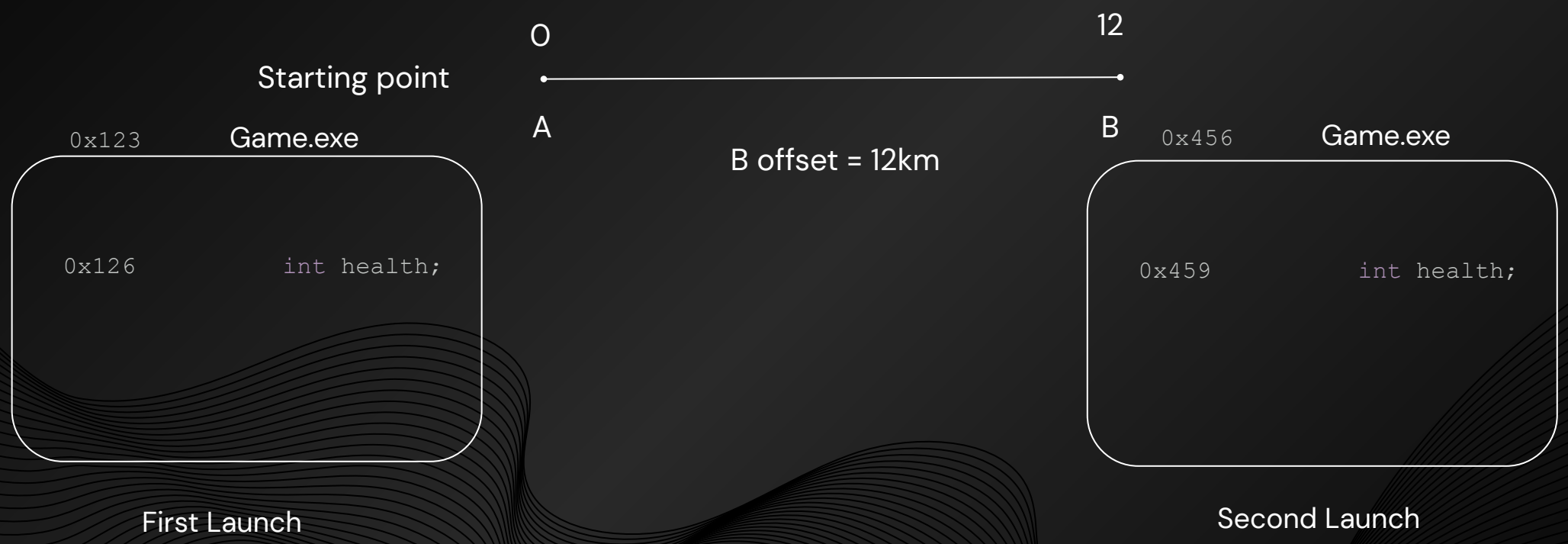
Debugging(Cheat Engine)

Disassembling(IDA pro)





What are Offsets?



Health Offset from game.exe is same in both cases i.e. 0x3

What are Offsets?

Game.exe

0x0 0x10

```
struct enemy {   //0x0
      int health;      //0x0
      int ammo;        //0x4
      float x;         //0x8
      float y;         //0xC
      float z;         //0x10
};

struct player { //0x10
      int health;      //0x0
      int ammo;        //0x4
      float x;         //0x8
      float y;         //0xC
      float z;         //0x10
};
```

ammo = 40

↓

*(ammoAdd) = 40

↓

*(playerAdd + 0x4) = *(ammoAdd) = 40

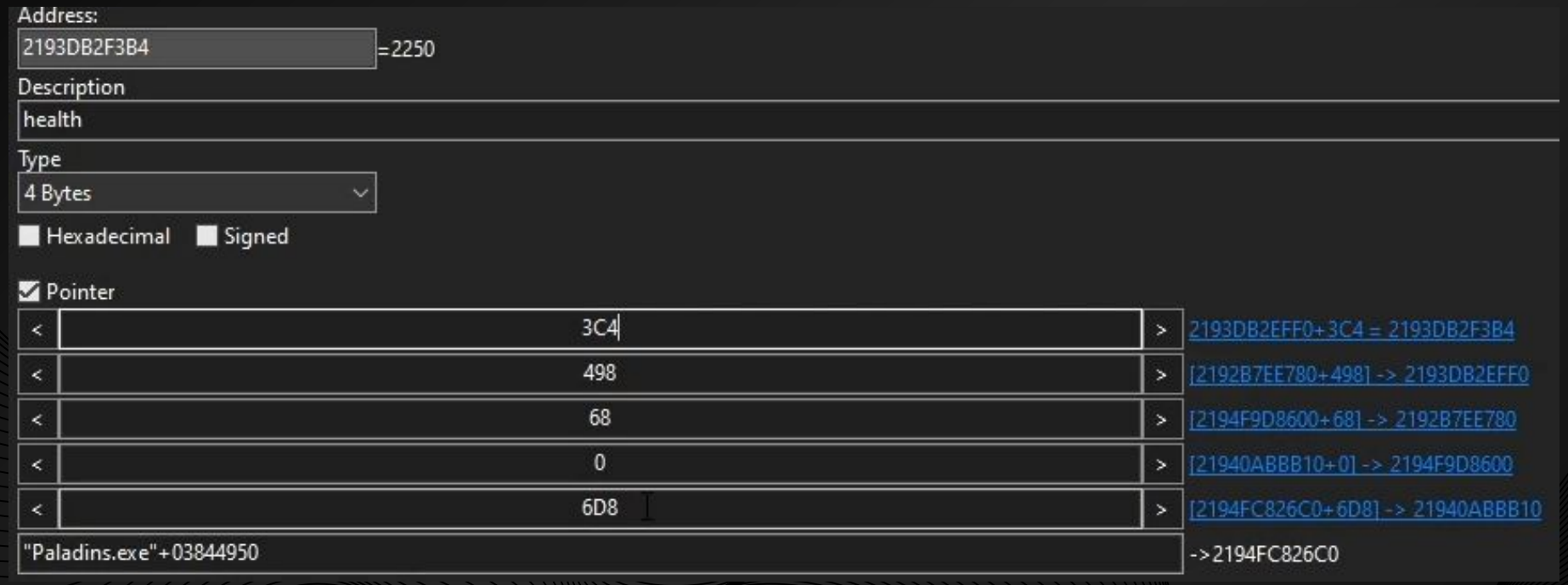
↓

((GameAdd + 0x10) + 0x4)) = *(ammoAdd) = 40

Ammo Offset = Game.exe + 0x10 -> 0x4

Quick demo for Offsets

Paladins health Offset



Paladins.exe + 0x3844950 -> 0x6D8 -> 0x0 -> 0x68 -> 0x498 -> 0x3C4

```cpp
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <Windows.h>
#include "proc.h"

int main()
{
    //Get ProcId of Paladins.exe
    DWORD procId = GetProcId(L"Paladins.exe"); //wrapper around Process32First

    //Get address of Paladins.exe
    uintptr_t moduleBase = GetModuleBaseAddress(procId, L"Paladins.exe"); //wrapper around Module32First

    //Get Handle to Paladins
    HANDLE hProcess = 0;
    hProcess = OpenProcess(PROCESS_ALL_ACCESS, NULL, procId);

    //pointer to baseClass Structure
    uintptr_t pbaseClass = moduleBase + 0x3844950;

    //Address of health pointer
    std::vector<unsigned int> healthOffset = { 0x6D8 , 0x0, 0x68, 0x498, 0x3C4 };
    uintptr_t healthAddr = addFromOffset(hProcess, pbaseClass, healthOffset); //wrapper

    //Read Ammo value
    int healthValue = 0;
    ReadProcessMemory(hProcess, (BYTE*)healthAddr, &healthValue, sizeof(healthValue), nullptr);
    std::cout << "Curent health = " << std::dec << healthValue << std::endl;

    //Write to it
    int newHealth = 1337;
    WriteProcessMemory(hProcess, (BYTE*)healthAddr, &newHealth, sizeof(newHealth), nullptr);

    std::cout << "New health = " << std::dec << healthValue << std::endl;

    getchar();

```

2. Creating Kernel Driver

Requirements

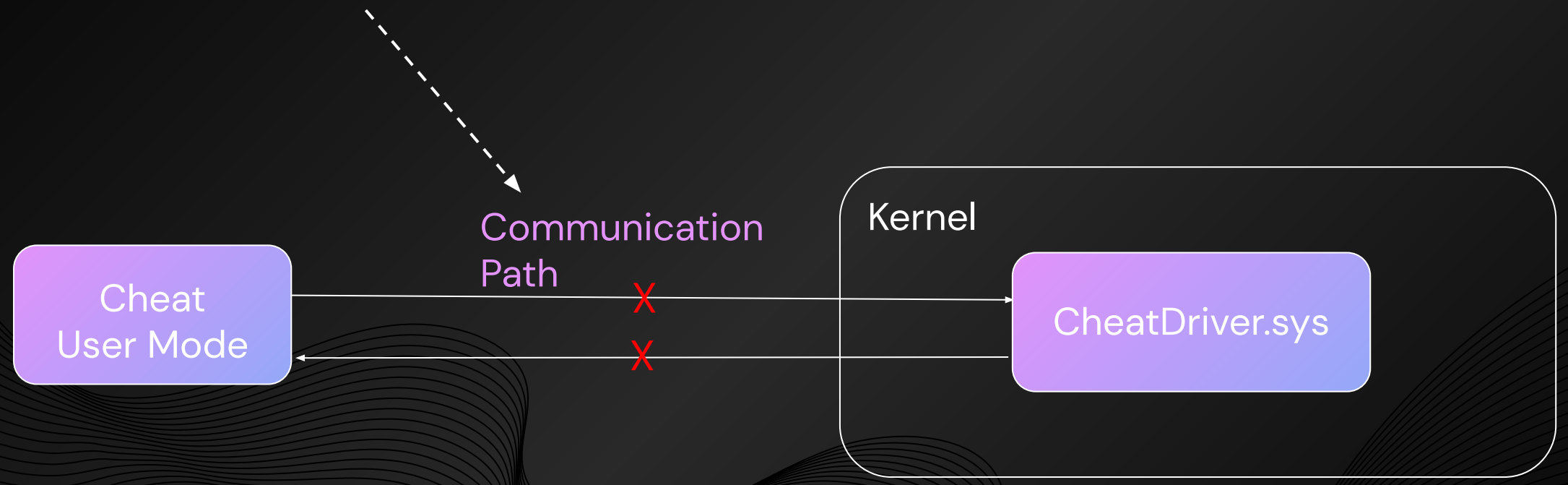
Visual Studio

Windows Driver Kit

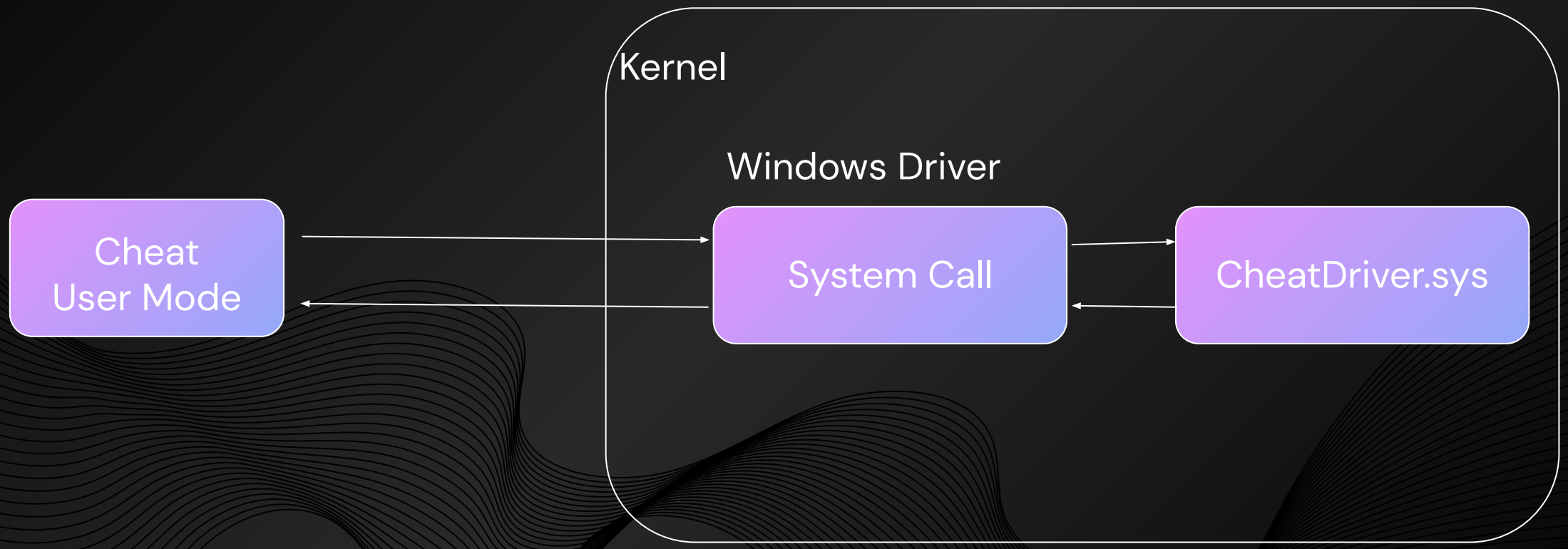
WinDBG / DBGview

Hello World Driver

3. Hooking


Communication
Path
Kernel
Cheat
User Mode
CheatDriver.sys

Hooking


Kernel
Windows Driver
Cheat
User Mode
System Call
CheatDriver.sys

User Mode

```
systemCall(instruction)
```

Kernel

WindowsDriver.sys

```
systemCall

mov rax, ourFunctionAddress
jmp rax
```

Our shell code

OurDriver.sys

```
ourFunction(instruction){

if (instruction==READ)
    readMem()
else if (instruction==WRITE)
    writeMem()
};
```

Till now...

A basic hello world driver

Communication = Hooking

What else our driver needs to do?

What makes a Kernel Cheat Driver?

1. System Call Addr.

We need the address of system module & function in order to place our hook

2. Hooking

Placing our shellcode into hook function to jump to our driver in kernel

3. Hook Handler

Handler that handles the instructions from User mode, executes and response back.

4. Clearing Traces

Clearing our loaded driver traces from PiDDBCacheTable & MmUnloadedDrivers.

Detection vectors while using hook

1. Hook Function

Almost all public system calls are sigged by AntiCheats ex: NtOpenCompositionSurfaceSectionInfo. Find your own system call that you can hook to.

2. Shell Code

`mov rax,xxx & jmp rax` is a classic hook shell code which is well known and sigged by AntiCheats. Create your own Shell Code that prefers mid function hooking.

```
PVOID* function = reinterpret_cast<PVOID*>(get_system_module_export("\\SystemRoot\\System32\\drivers\\dxgkrnl.sys",
    "NtOpenCompositionSurfaceSectionInfo")); // NtOpenCompositionSurfaceSectionInfo

if (!function)
    return false;

BYTE orig[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };   //0x00, 0x00, 0x00, 0x00

BYTE shell_code[] = { 0x48, 0xB8 }; // mov rax, xxx // 0x48 - 0xB8
BYTE shell_code_end[] = { 0xFF, 0xE0 }; //jmp rax   // 0xFF - 0xE0
```

4. Loading Driver

Test Mode	Pay to load	Exploit
Only good for testing our driver	Get your driver signed by Microsoft by paying.	Exploit official signed driver to load our driver.
Doesn't work with Modern Anti-Cheats	Can be revoked easily if gets reported	Easy pzzz

KDMapper

https://github.com/TheCruZ/kdmapper

Exploits iqvw64e.sys Intel driver to manually map non-signed drivers in kernel memory.

Automatically clears major Traces such as MmUnloadedDrivers, PiDDBCacheTable & g_KernelHashBucketList but not all like PoolBigPageTable.

Loading driver is easy: `kdmapper.exe driver.sys`

5. Creating User Mode

1. Call Hooked Function

Call the hooked function or system call with our shell code

2. Prepare Instructions

Construct the instruction according to your needs for sending to the hooked function

3. Handle the cheat logic

Aimbot, ESP, etc.

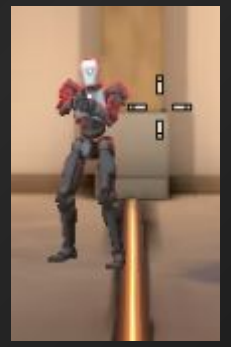
04 Demonstration

UEFI & Arduino Hacks

```
-45ED-9C20-0FBA129511D6,0x13A800,0x8000)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x1F,0x2)/Sata(0x0,0xFFFF,0x0)/HD(4,GPT,FD03F504-E2EB
-4F05-B1F4-436B4FAB51DE,0x142800,0x12ABC800)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
Shell> FS1:
FS1:\> ls
Directory of: FS1:\
04/28/2020  18:01 <DIR>          512  EFI
04/29/2020  15:30             49,634  memory.efi
          1 File(s)      49,634 bytes
          1 Dir(s)
FS1:\> load memory.efi

  ___ / _(_)___ _ __  ___ _ __  ___ _ _ _  _
 / -_)  _| |___| '  \/ -_) '  \/ _ \ '_| || |
 \___|_| |_|   |_|_|_\___|_|_|_\___/_|  \_, |
                                         |__/
Made by: Samuel Tulach
Thanks to: @Mattiwatti (EfiGuard), Roderick W. Smith (rodsbooks.com)

Driver has been loaded successfully. You can now boot to the OS.
Image 'FS1:\memory.efi' loaded at 7FB9C000 - Success
FS1:\> _
```



Kernel Cheat Driver Release

https://github.com/nahoragg

References

https://guidedhacking.com/

https://www.unknowncheats.me/

https://www.youtube.com/c/NullTerminator

Thank you!

Contact me:

nahoragg@gmail.com
Twitter: nahoragg

#plsdontbanmeEAC