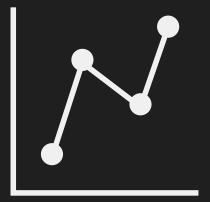
Sounds from the basement: Extreme low-level game cheating

Table of contents

Introduction to game cheating & anti-cheat solutions

Why care about game cheats? - Money



Big business

- Recent study from Birmingham University
- 80 sites analysed, estimated $12.8M and $73.2M of earnings annually

Amount of cheaters is not going down

- Report by Riot Games (Vanguard)
- 3.6M accounts banned in 4 years
- Cheaters don't stop after a ban!

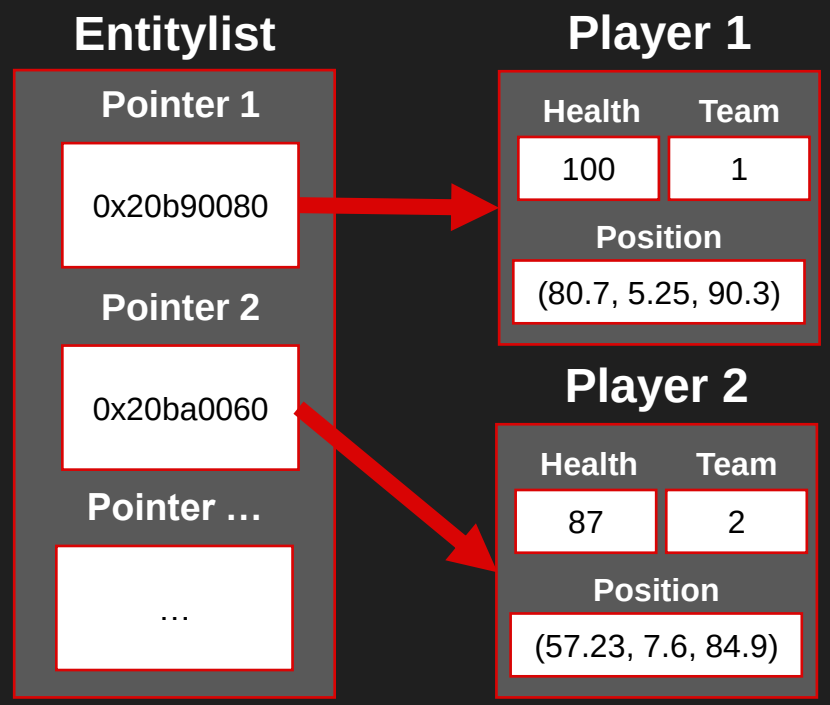
Why care about game cheats? - "Perfect bank heist" in nerdy context

Cat-and-mouse vs. anti-cheats

- Modern anti-cheat systems are very sophisticated
  - Wouldn't it be cool to be undetected from them?
  - Similar to viruses and anti-viruses

Game cheat 101

- Virtually every game has a main entity list containing players
- Each player is a class object with properties and methods (getters & setters for position, angle, health, speed)
  - Cheating requires the cheat to read and/or modify this data!



Game cheating features

Interactive assistance

Interacts with the game for you to improve your performance, examples:

- Aimbot
- Triggerbot
- Scripts

Game cheating features

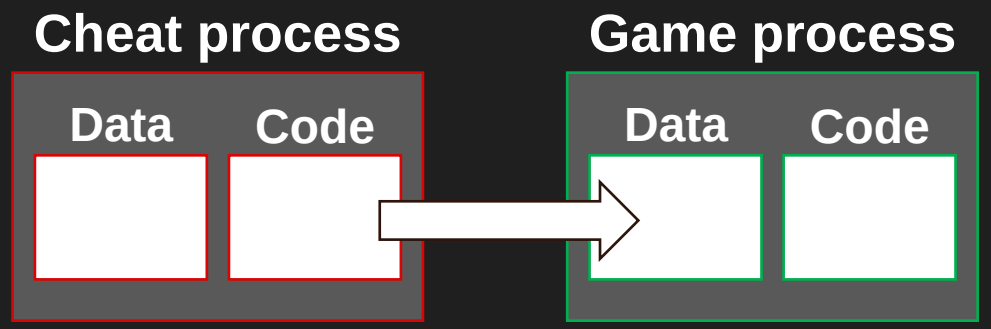
Informational assistance

Gives you undisclosed information for an advantage, examples:

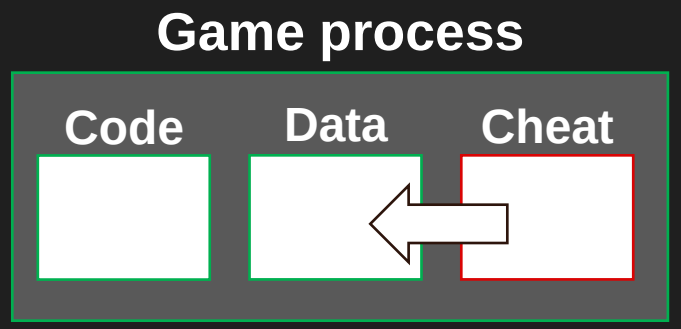
- ESP (visual / audio assistance)
- Radar
- Text on screen, some other medium

Game cheating functionality

External cheat



Internal cheat



Current state of game cheats

«Normal» cheats

- Simple “usermode” application
- Runs in the context of the game or accesses game memory

Driver cheats

- Involves a kernel driver
- Either fully in kernel or acting as middle man to access game memory

DMA cheats

- Involves a physical PCIe hardware device
- Cheat itself runs on a separate device or HW device itself

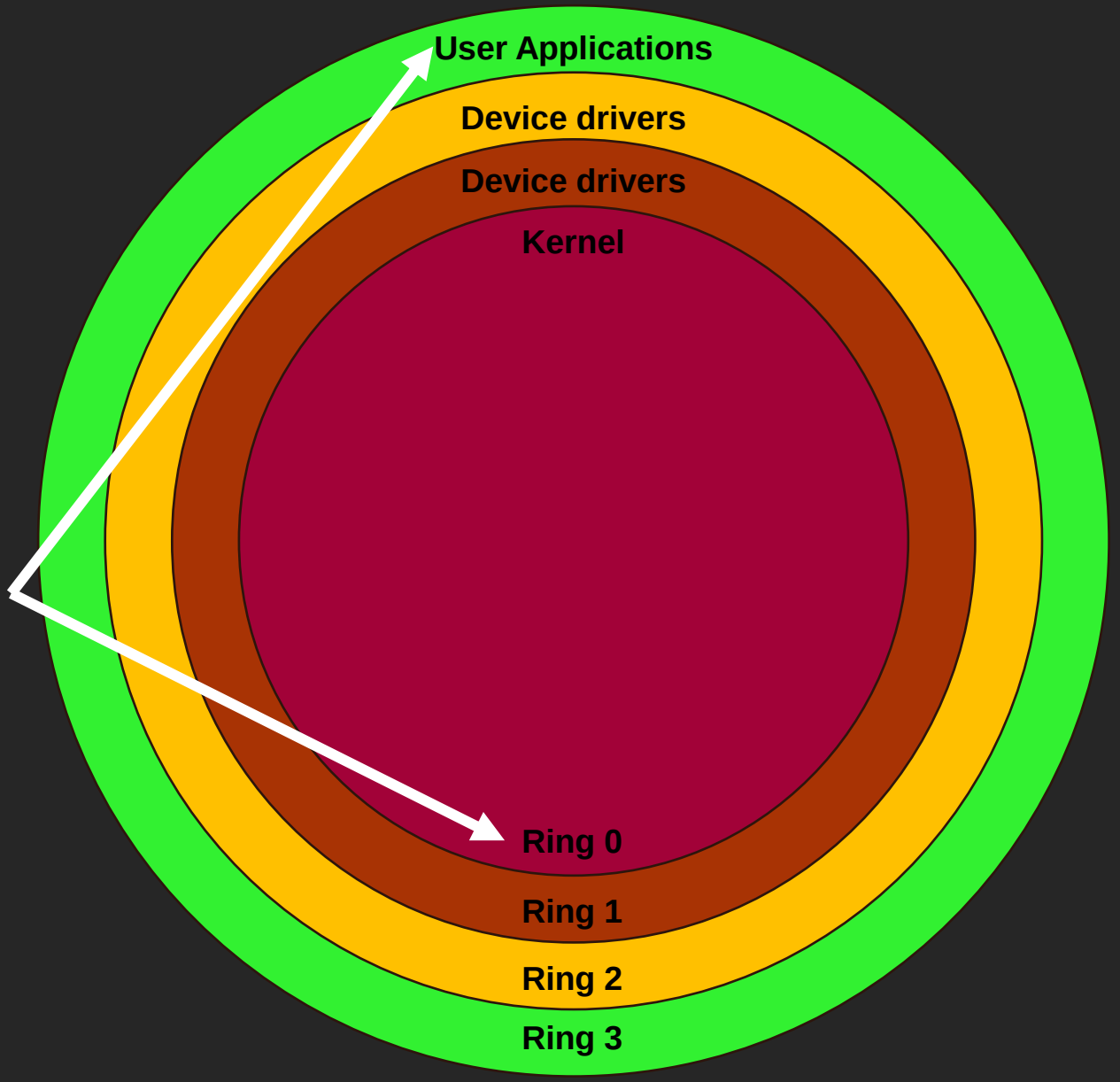
Future state of game cheats?...

SMM cheats

- Running on same CPU but in different operation mode
  - Completely invisible to traditional anti-cheat measures!

Introduction to anti-cheats

Sitting here:



Standard IA protection rings

Challenge when detecting cheats

Clear Evidence

- Anti-cheats require clear evidence to only ban cheaters
- Banning non-cheaters results in loss of trust
- Can correlate some detections to be more confident

Anti-Cheat features

Protection

- System configuration enforcement (HVCI, TPM)

Detection

- Heuristics & Signatures

Prevention

- Game binary & memory obfuscation

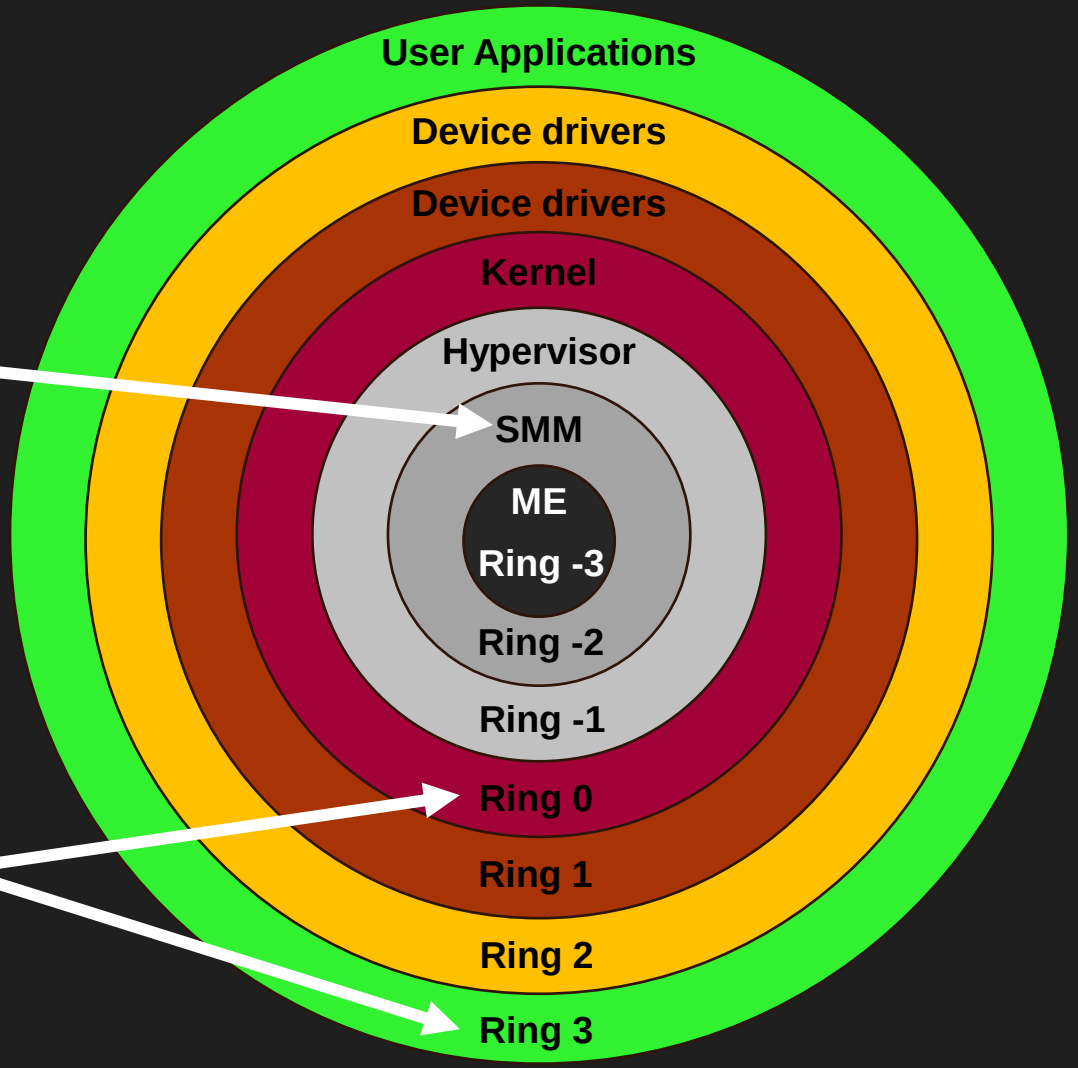
How did we end up choosing SMM?

- Both interested in low-level technologies
    - First looked at Intel Management Engine, but too complex
- SMM already has stealth inbuilt, hard to detect
- Only few projects released in the game cheating scene (**smm** from ekknod, **SmmInfect** from Zepta)
- Using known techniques requires hiding which adds more complexity

Introduction to system management mode (SMM)

SMM sits here:

(recap: Anti-cheats sit here)



Standard & negative IA protection rings

What is system management mode?

- Intended use is by firmware to perform low-level operations while an OS is running
- Separate memory space and fully isolated

Example (expected usage):

- Controlling the fan speed
- Works no matter what operating system is installed

Benefits of SMM

Access to all memory

- Full physical memory access (Hypervisor, Kernel, etc.)

Not many security measures

- Want to overwrite some physical memory? Go ahead!
- No verification of code executed

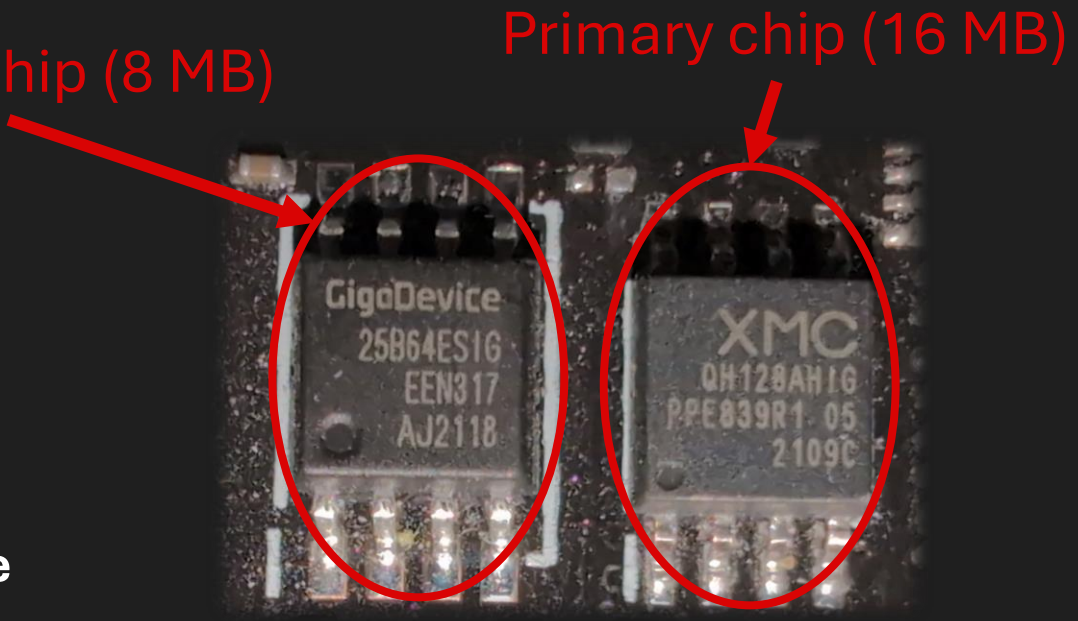
Invisible to runtime

- Operating system can’t inspect SMM
- SMM acts as a blackbox

Where are SMM modules located?

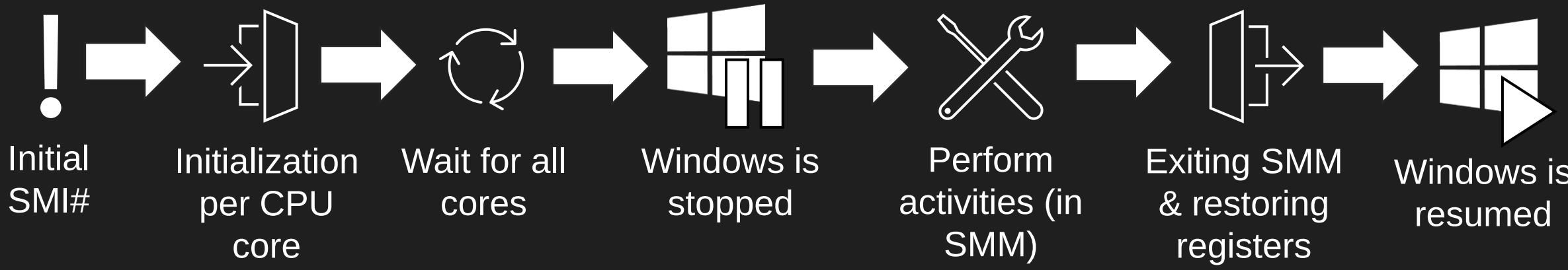
Answer: Inside the UEFI Firmware image!

- Stored on the main firmware chip (SPI NOR Flash)
- Access is possible from inside or outside the system
- Chip itself does not verify the content
- **In this talk, we are focusing on EDK II project as the UEFI specification implementation**



Source: us

How is SMM executed?



EDK II Topology – SMM: Topology of how SMM is set up and executed by Lee Hamel

From a wild idea to a game cheat in SMM

How we first planned the development

1. Setup development environment using EDK II
2. Create Hello World to test SMM abilities
3. Add dynamic debugging options for faster development
4. Create a game cheat

Developed directly on real hardware!

Basic Hello World

Getting the basics right

- Replace an existing SMM module
- Serial for output

```
0$Hello from SMM!
SMM System Table: 0x75f1e018
Memory of SMM System Table:
 0x0: 49 42 49 20 53 59 53 54 50 0 2 0 78 0 0 0
 0x10: 30 17 85 1c 0 0 0 0 18 76 d7 75 0 0 0 0
 0x20: 1b 0 5 0 0 0 0 0 18 f 8e 6d 0 0 0 0
 0x30: 90 bb f9 6f 0 0 0 0 18 f 8e 6d 0 0 0 0
 0x40: 20 bc f9 6f 0 0 0 0 18 f 8e 6d 0 0 0 0
```

Source: us

Fixing memory access

Memory restriction

- Fix memory access (Memory restriction patch)

```
!!!! X64 Exception Type - 0E(#PF - Page-Fault)  CPU Apic ID - 00000000 !!!!
ExceptionData - 0000000000000000  I:0 R:0 U:0 W:0 P:0 PK:0 S:0
RIP  - 000000007F20F26D, CS  - 0000000000000038, RFLAGS - 0000000000010002
RAX  - 000000007ADAA898, RCX - 0000000000001000, RDX - 0000000000001000
RBX  - 0000000000000004, RSP - 000000007F537BD8, RBP - 000000007F537C29
RSI  - 0000000000000001, RDI - 0000000000000001
R8   - 0000000000000000, R9  - 000000007ADAA8A8, R10 - 000000000000030A
R11  - 0000000000000000, R12 - 000000007F7793C8, R13 - 0000000000000000
R14  - 0000000000001000, R15 - 0000000000000010
DS   - 0000000000000020, ES  - 0000000000000020, FS  - 0000000000000020
GS   - 0000000000000020, SS  - 0000000000000020
CR0  - 0000000080010033, CR2 - 000000007ADAA8A0, CR3 - 000000007F50C000
CR4  - 0000000000000668, CR8 - 0000000000000000
DR0  - 0000000000000000, DR1 - 0000000000000000, DR2 - 0000000000000000
DR3  - 0000000000000000, DR6 - 00000000FFFF0FF0, DR7 - 0000000000000400
GDTR - 000000007F50B000 000000000000004F, LDTR - 0000000000000000
IDTR - 000000007F515000 00000000000001FF,   TR - 0000000000000040
FXSAVE_STATE - 000000007F516C60
```

Source: us

Extending to Windows kernel

```
[NT] PML4: 0x1ae000 KernelEntry fffff8010f610c30
[NT] NT kernel: 0xfffff8010f200000
[NT] PISP: 0xfffff8010ff1ea60
[NT] SystemProcess: 0xffffa8023f93b040
[NT] NtVersion: 1000
[NT] NtBuild: 22631
```

Source: us

Extending to Windows kernel

- Port required parts of open source tools (vmread by h33p, pcileech by ufrisk) to SMM
  - Analyze windows kernel and locate necessary structures

Better debugging & faster development

Serial communication

- No interrupts required, small code base
- Connector still exists on modern boards
- «printf» debugging style

Custom debugging application

- Running outside of SMM, for dynamic debug
- Communication via memory buffer
- Example: Hermes from us

From Hello World to Game Cheat in SMM

17.02.2025

What to implement?

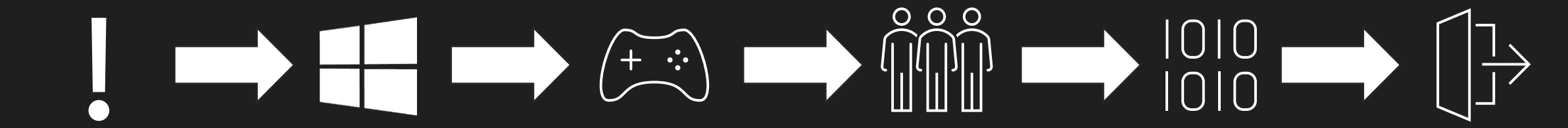
Aimbot

- The classic cheating method
- Assist the aiming of the user
- Requires us to interact with the mouse data

Sound ESP

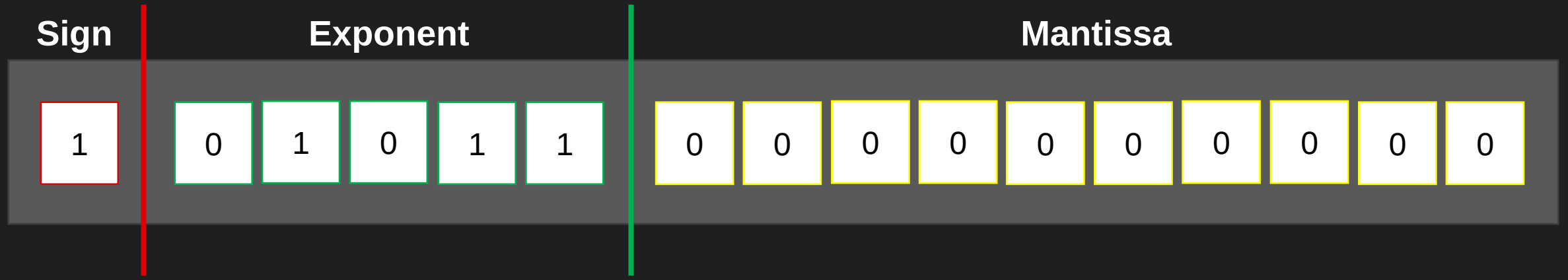
- Visual ESP would introduce a larger detection surface
- Sound ESP provides just in time information to «mentally» prepare yourself

How would it work?



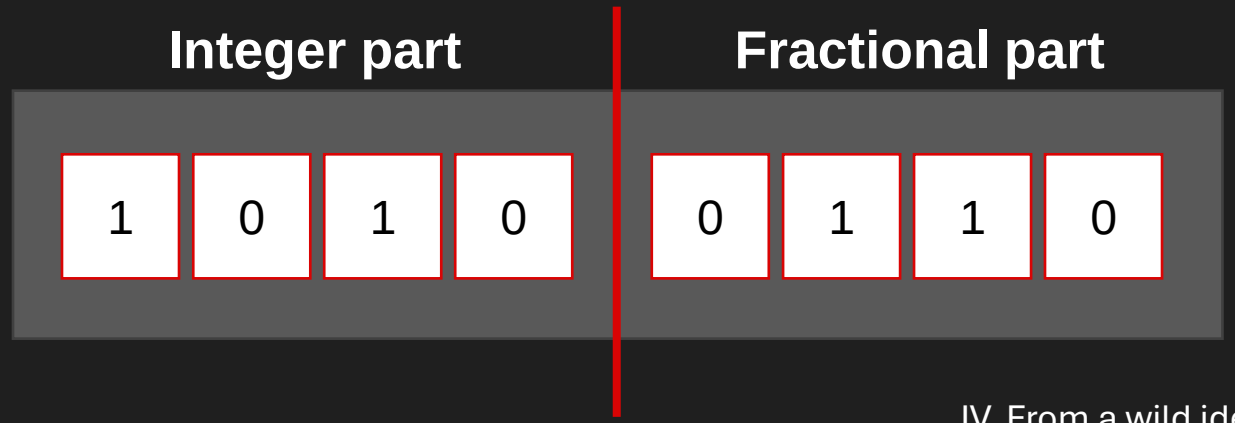
Pain point #1: Floating-point arithmetic

- Games use floating-point to save positions
- Floating-point unit (FPU) not initialized in SMM



Solution for floating-point arithmetic

- Convert floating-point to fixed-point using arithmetic operations
- Recreate the math functions needed for cheat (sqrt, sin, etc.)
- Derived our own library from 32-bit fixed-point lib (libfixmath64)



Pain point #2-3: Execution and user interaction

Require a lot of SMM execution

- To run the cheat in SMM, we need a regular high-frequency amount of executions
- Timers exist, but they are not quick enough (once every 8 seconds)

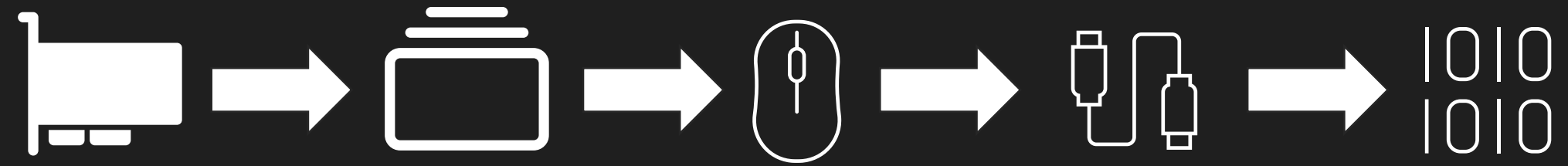
Interaction with system/user required

- Need to inform user/change data
- Should be where the Anti-cheats don’t check

Solution for execution and interaction

- Paper by Joshua Schiffman and David Kaplan (The SMM Rootkit Revisited: Fun with USB, 2014)
- Route interrupts from USB controller to trigger SMI
- Found alternative: USB Controller & Mainboard can trigger SMI on USB events
- Now able to intercept and modify all USB traffic in SMM!

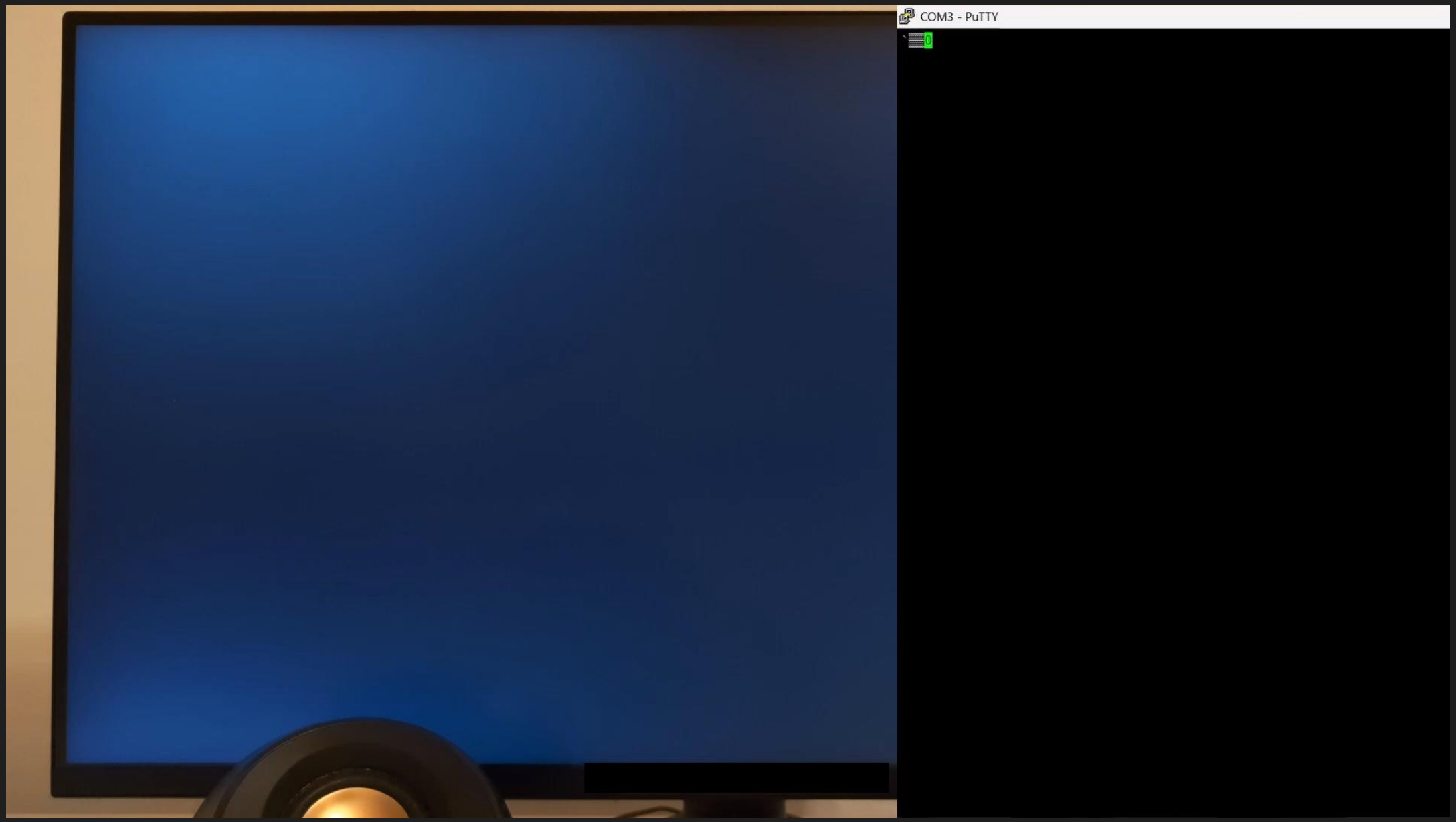
USB Controller parsing



USB Data modification

- Raw data is modified before it reaches the operating system
- Mouse communicates over HID, allows simple manipulation the x, y movements
- Sound sends raw sound data, can just overwrite data to produce “noise”

Demo time!


COM3 - PuTTY

Detection vectors of SMM cheats

Detection using memory honeypots

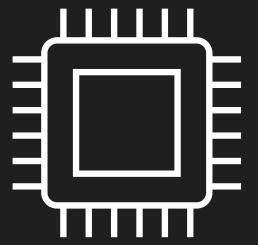
Side channel detection

- Original idea from Everdox (Anti-Cheat Developer @ Vanguard, Riot)
- Uses CPU cache timing to check for recent memory access (E.g. side-channels by h33p)
- Cache can be measured using fake entity

How to prevent lower cache latency

- Disable CPU caching before accessing any memory
  - Huge performance impact

Detection using SPI dump



Dumping the SPI NOR Flash

- Chip content can be read
- Anti-cheat can look for suspicious or out of place modules
- Example: Default compilation tools under windows use 32 byte alignment

How to prevent dumps

- Protected range registers (PRR) to prevent reading the Chip content (🚩)
- Spoofing SPI read operations using SMM, see SpiMitm by Takahiro Haruyama

Other possibilities

What could SMM also be used for?

SmmBackdoorNg
by cr4sh

Hermes
by us

Scotch
by Kevin Leach, Fengwei Zhang, Westley Weimer

Spectre
by Fengwei Zhang, Kevin Leach, Kun Sun, Angelos Stavrou