Bypassing Anti-Cheats & Hacking Competitive Games
Hello I’m...

Rohan Aggarwal @nahoragg
Founder at DefCore Security
Bug Bounties
Live Hacking Event(H1 & Intigriti)
Love playing games(Valorant, Elden Ring, etc)
Outline

01 - Cheats vs Anti-Cheats
02 - Game Hacking Basics
03 - Kernel Cheats
04 - External Hardware Cheats
05 - Demonstration
What kind of Game Hacking?
History of Game Hacking
1980 - Modding
1995 - First Aimbot
Early 2000 - Multiplayer Cheats
Early 2000 - AntiCheats

2000

2002
Popular Kernel AntiCheats
Anti-Cheats

Detect Cheats - Signatures

Discover Cheats - Reports, Manual

Prevent Cheats - Obfuscation, Sandbox
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
User Mode

Restricted access to system resources

Private virtual space for each process

Kernel Mode

direct and unrestricted access to system resources

Single virtual space for whole kernel
Game Hacking Basics
## Types of Cheats

<table>
<thead>
<tr>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injected into the target process itself.</td>
<td>Have their own process that manipulates target process</td>
</tr>
<tr>
<td>Complex to make (depends on engine)</td>
<td>Easy to make (Any lang.)</td>
</tr>
<tr>
<td>Much Flexible and great performance</td>
<td>Enough Flexibility and performance</td>
</tr>
<tr>
<td>Prefered for Games with low-level or No AC</td>
<td>Prefered for Games with Strong AC</td>
</tr>
</tbody>
</table>
Cheat.exe
Cheat User Mode

Game.exe
Game Memory

read/write

Kernel

AntiCheat.sys
Kernel

AntiCheat.sys

Game.exe

Game Memory

Cheat.exe

Cheat User Mode

read/write
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?
Kernel Cheat Development
Cheat Development

1. Reversing
Reversing Game to find the offsets for required cheat.

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

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

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

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?

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

- https://dumps.host/
- https://www.unknowncheats.me/
- https://github.com/
Hello World Driver

Requirements:

1. Visual Studio
2. Windows Driver Kit
3. WinDBG / DBGview
2. Hooking

Cheat User Mode

Communication Path

Kernel

CheatDriver.sys
2. Hooking

shell code: Mov rax, CheatDriver_address
            Jmp rax
User Mode

```
systemCall(instruction)
```

Kernel

```
WindowsDriver(dxgkrnl.sys)
```

```
ourFunction (instruction)
mov rax, ourFunctionAddress
jmp rax
```

```
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.
```

```c
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

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

<table>
<thead>
<tr>
<th>Test Mode</th>
<th>Pay to load</th>
<th>Exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only good for testing our driver</td>
<td>Get your driver signed by Microsoft by paying.</td>
<td>Exploit official signed driver to load our driver.</td>
</tr>
<tr>
<td>Doesn’t work with Modern Anti-Cheats</td>
<td>Can be revoked easily if gets reported</td>
<td>Easy pzzz</td>
</tr>
</tbody>
</table>
KDMapper

https://github.com/TheCruZ/kdmapper

Exploits `iqvw64e.sys` Intel driver CVE-2015-229 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`
More Vuln Drivers?

https://github.com/hfiref0x/KDU

https://github.com/eclypsium/Screwed-Driver

https://guidedhacking.com/threads/vulnerable-kernel-drivers-for-exploitation.15979/

https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=driver

Exploit drivers on your own
5. Creating User Mode

1. Call Hooked Function
   Call the hooked function or system call with our shell code

2. Prepare Instruction
   Construct the instruction according to your needs for sending to the hooked function

3. Handle the cheat logic
   Aimbot, ESP, etc.
Kernel WallHack Showcase
Moving onto Ext. Hardware Cheat
Pixelbots

Color based aimbots
How do these work?

PC

- pixelbot
- valorant

Monitor
How do these work?

pixelbot

- Capture Pixels
- Find enemy
- Move Mouse
Benefits?

Very hard to detect

Easy & Works on almost all games

Good code = Good aimbot
Tools required

Arduino Leonardo

USB host shield soldered
Pixelbots Gen-1 Mouse Mov.

PC

pixelbot

valorant

pyautogui
pydirectinput
NtUserSendInput
SendInput
PostMessage
SendMessage
MouseEvent
CreateCursor
SetCursorPos
SetCursor
Pixelbots Gen-2 Mouse Mov.

PC

- pixelbot
- valorant

Interception Driver
Mouse Drivers - Razer, Logitech, etc
Chinese drivers - https://github.com/ddxoft/master
Pixelbots Gen-3 Mouse Mov.
Pixelbots Gen-4 Mouse Mov.

PC

pixelbot
valorant

Arduino + Hyper-V

Virtual Second Mouse
Pixelbots Gen-5 Mouse Mov.

PC

- pixelbot
- valorant

Spoofed Mouse

Arduino +
USB host shield

Real Mouse
Pixelbot Code

Capture Pixels → Color Filtering → Calculate Coordinates → Send Coordinates
Challenge

Actually aim on enemy and not other things with similar color
Finding Color Range using OpenCV
Finding Color Range using OpenCV

```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

image = cv2.imread('media/astra.jpg')

# Converting the image to hsv
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# define range of purple color in HSV
lower_purple = np.array([160, 50, 50])
upper_purple = np.array([180, 255, 255])

# Threshold the HSV image using inRange function to get only purple colors
mask = cv2.inRange(hsv, lower_purple, upper_purple)

plt.figure(figsize=[13,13])
plt.subplot(121); plt.imshow(image[:, :, ::-1]); plt.title("Original Image", fontdict={'fontsize': 25}); plt.axis('off');
plt.subplot(122); plt.imshow(mask, cmap='gray'); plt.title("Mask of purple Color", fontdict={'fontsize': 25}); plt.axis('off');
```

https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/
Arduino code

Take Coordinates → Simulate Mouse Mov → Simulate Real Mouse
Different arduino communication methods

Serial Comm
Web server
Wireless transmitter
Serial getting monitored by vgc

Still u can hide the coordinates inside garbage
Spoofing arduino

C:\Program Files
(x86)\Arduino\hardware\arduino\avr\boards.txt


```
leonardo.name=Arduino Leonardo
leonardo.pid.0=0x2341
leonardo.pid.1=0x0036
leonardo.pid.2=0x2341
leonardo.pid.3=0x8036
leonardo.pid.4=0x2343
leonardo.pid.5=0x0036
leonardo.pid.6=0x2343
leonardo.pid.7=0x8036
leonardo.upload_tool=avrdude
leonardo.upload_protocol=avr109
leonardo.upload.maximum_size=28672
leonardo.upload.maximum_address_size=2560
leonardo.upload.speed=57600
leonardo.upload.disably_flushing=true
leonardo.upload.use_1200bps=True
leonardo.upload.wait_for_upload_port=true
leonardo.bootloader_tool=avrdude
leonardo.bootloader_low_fuses=0xff
leonardo.bootloader_high_fuses=0x88
leonardo.bootloader_extended_fuses=0x00
leonardo.bootloader_file=caterina/Caterina-Lenardo.hex
leonardo.bootloader_unlock_bits=0x3f
leonardo.bootloader_lock_bits=0x2f
leonardo.build.mcu=atmega32u4
leonardo.build.f_cpu=160000000L
leonardo.build.vid=0x2341
leonardo.build.pid=0x8036
leonardo.build.vid=0x2341
leonardo.build.vid=0x8036
leonardo.build.usb_product="Arduino Leonardo"
leonardo.build.board=Uno_LEONARDO
leonardo.build.core=arduino
leonardo.build.variant=leonardo
leonardo.build.extra_flags=[build.usb_flags]
```
External Hardware Aimbot Showcase
What valorant can do?

Remove outlines?

GAMEPLAY SYSTEMS UPDATES

- Added the ability to hide Agent outlines and fresnel (the color outline on Agents)
  - Go to Settings >> General >> Under “Other”, there is an option to toggle “Hide Outlines and Fresnel”.
YOLO AI

Release

https://github.com/nahoragg/KernelDriver

https://github.com/nahoragg/Arduino
References

https://guidedhacking.com/

https://www.unknowncheats.me/

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

JK

ANTI-CHEAT
Thank you!
nahoragg@gmail.com
Twitter: nahoragg