Unveiling the underground world of ANTI-CHEATS

Joel Noguera
Security Consultant at Immunity Inc
@niemand_sec - niemand.com.ar
What are we going to talk about?
FIRST RULE OF THE GAMING CLUB, YOU DON'T CHEAT (or get caught doing it)
GIF DEL PIBE QUE LO ENCUENTRA CON WORD.exe EN MEDIO DEL TORNEO
Anti Cheats
Anti-Cheats
Let’s see some numbers...

336,500,000

Monthly Active Users

EAC  275,000,000

XC3  500,000

BE   30,000,000

VAC  31,000,000
Вы используете: Аура Силы.
Черный Заряд Духа (+4) будет использоваться автоматически.
Ваше сражение наполнено силой.
Цена низкая - глубоко врага.
Уменьшение слабых лавин в игре.

Награды: Спаси брата. Хир ДА и пой.
Благодарен: Кунду Рик и благоразумие.
Зачем? Могу нанести УДАЩУ.
Удалось попасть на зал.
Уникальный ками - купил Рест и лайт.
Заклинание мощной бомбы урон фанатичке +112 с 300.
Поверните назад.
Предложение: ВЗ.
Пожалуйста, дайте мне клан."Волшебный Камень".
Варлок в шкуре.
Послушайте, пожалуйста.
Кому в клане Арка.
Сцена в копье.
Сцена в копье.

Пожалуйста, дайте мне клан.
Волшебный Камень +122 СЛ.
Пожалуйста, дайте мне клан.
Сцена в копье.
Сцена в копье.

Anti-Cheat Components
Kernel Driver

- Handle stripping/Access Control
- Register kernel callbacks
- Rejection of Kernel/User mode debugging
- Analysis of privileged process (lsass and csrss)
- Block blacklisted/unsigned drivers
- Monitoring of kernel function calls
DLL inside Games

- Control of access flags to different sections
- Identification of hooks
- Thread Hijacking
- DLL Injection
- Function signatures
- VEH/SEH modification
- Game resources modification
- Detection of virtual environment
External Ring 3 Process

- Process/File Controls
- Blacklisted programs detection
- Manage logic from Driver
- Control of game client and DLL hashes
- Multi-client detection
- Program integrity controls
Cheats
LOL

WRONG CHEAT CODE
## Internal (DLL) vs External (Process)

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
</table>
| **External** |  - Quick for small patches  
- Easy to master  
- Can be closed in certain cases |  - Slow  
- Easy to detect  
- Limited potential  
- Requires a Handle |
| **Internal**  |  - Great performance  
- Direct access to memory  
- Hard to detect if you are good enough |  - Hard to master  
- Easier to detect if you mess it up |
Aimbots
Wallhack/ESP
Pro players getting caught? Why not
Automatization

Utility

coll-bot
gold-farmer
exploits
scripting
multi-client
Auto-fooling
infinite-resources
automatic-grinding
stats
invincible
invincible
unlimited-health

leveling

flying

teleport
unlimited-health
unlimited-mana
player-detection
unlimited-ammunition

Automatization
Utility
Motivation!
Let me tell you a story...

We decided to reverse a cheat for Lineage 2

Characteristics: Made in Russia, good bypasses for AC, Lineage 2

Extra Gold Coins for:
- Emiliano Del Peon (@Dolphin01684386)
- Lautaro Fain (@LautaroFain)
Let me tell you a story
Let me tell you a story

Old version is detected by ACs

The new version moved to a stealthier approach: **FileMapping**
Parallel Market
Parallel Market

Cheat Prices:
U$S 1 to U$S 25
Some up to U$S 500

Ex: 2500 paid members
U$S 10 * 2500 = U$S 25000
(150000 memberships)

U$S 1,25 M
PER YEAR
(Wait... what?)
Are they fighting back?

Apex claims:

- More than 770k players banned
- Over 300K account creations blocked
- Over than 4k cheat sellers accounts (spammers) banned in 20 days

Temporary EAC Bypass

Navigate to game folder (Origin Games\Apex)

1. Rename EasyAntiCheat_launcher.exe to EasyAntiCheat_launcher.back
2. Rename r5apex.exe to EasyAntiCheat_launcher.exe

Have some luls while it lasts.
Analyzing Anti-Cheats
Methodology

Goal:

- Read/Write/Alloc Memory (Internal & External)
- Run Code inside Game’s Process
- Be as **stealthy** as possible
Hijacking Techniques

AC usually control/block/reject new HANDLEs to the game process:

- Driver that protects game and AC processes

Some process need to be whitelisted: **lsass, csrss, AC**

Hijacking techniques come to our rescue:

- Handle Hijacking
- Stealth Handle Hijacking
- Hooking
Hijacking Techniques

LSASS (exe)

Attacker DLL

Handle 0x4A

GAME (exe)

CHEATS (exe)
Hijacking Techniques - NamedPipe

```
\Device\NamedPiped\270F59B0075AA3D3
```
Hijacking Techniques - NamedPipe

Disadvantages

- Suspicious new HANDLEs
- Hooks to user-mode WIN API
- Thread with suspicious context
- Downgrade of HANDLE privileges
Imagine a world where our shared memory does not leave an open HANDLE and we can cover better our tracks.
File mapping object does not close until all references to it are released

HANDLE CreateFileMappingA(
    HANDLE hFile,
    LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
    DWORD flProtect,
    DWORD dwMaximumSizeHigh,
    DWORD dwMaximumSizeLow,
    LPCSTR lpName
);

BOOL UnmapViewOfFile(
    LPCVOID lpBaseAddress
);

We can call CloseHandle without calling to UnmapViewOfFile.
Hijacking Techniques - FileMapping

Manual spinlocks to avoid mutex/semaphores HANDLEs

We can make it even better by delaying the execution
Hijacking Techniques - FileMapping

**Disadvantages**

- Suspicious new HANDLEs
- Hooks to user-mode WIN API
- Thread with suspicious context
- Downgrade of HANDLE privileges
Hijacking Techniques - Bypass Hooks

EAC also hook functions on `lsass.exe`:

- Validate/Control/Track each action done against the game
Hijacking Techniques - Bypass Hooks

```asm
.ZwWriteVM proc
    mov r10, rcx
    mov eax, 3Ah
    syscall
    ret
.ZwWriteVM endp

.ZwReadVM proc
    mov r10, rcx
    mov eax, 3Fh
    syscall
    ret
.ZwReadVM endp
```

Hijacking Techniques - Bypass Hooks

Disadvantages

- Suspicious new HANDLEs
- Hooks to user-mode WIN API
- Thread with suspicious context
- Downgrade of HANDLE privileges
Hooking
Cheats usually hook functions from Graphic Engines:

- IAT hooking, JMPs on Prolog functions, etc

But AC usually control this.

Inside their own game is easy, but what about trusted external libraries?

- Steam Overlay
- Open Broadcaster Software (OBS)
Steam Overlay

Redirects execution to gameoverlayrenderer64.dll:$8A480

Open Broadcaster Software

Redirects to graphics-hook64.7FFEB97AE4D0
JMPs to unmapped regions still works.
Hooking - Code Caves and NamedPipes?
Refresher - Bypass Hooks

Disadvantages

- Suspicious new HANDLEs
- Hooks to user-mode WIN API
- Thread with suspicious context
- Downgrade of HANDLE privileges
Moving to kernel...Drivers
Drivers

Cheat developers also develop their own to fight inside the kernel.

Loading a Driver:

- Test Mode
- Sign your own Driver ($$$$$$)
- Abuse of another driver
EAC downgrading the HANDLE

<table>
<thead>
<tr>
<th>Type</th>
<th>Handle</th>
<th>Name</th>
<th>Access</th>
<th>Decoded Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>0x9A8</td>
<td>ServiceHub.DataWarehouseHost.exe</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x146C</td>
<td>sedsvg.exe (7312)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x14B8</td>
<td>SearchUI.exe (10180)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0xE10</td>
<td>SearchIndexer.exe (7103)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x1B5C</td>
<td>Scripted Sandbox64.exe (15372)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x1840</td>
<td>SCM.exe (6204)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x186C</td>
<td>RuntimeBroker.exe (7604)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x15A0</td>
<td>RuntimeBroker.exe (12244)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x14FC</td>
<td>RuntimeBroker.exe (10640)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x1854</td>
<td>rdpapp.exe (6048)</td>
<td>0x00001440 DUP_HANDLE</td>
<td>QUERY_INFORMATION</td>
</tr>
<tr>
<td>Process</td>
<td>0x1910</td>
<td>QHSafe Tray.exe (14228)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x06C</td>
<td>QHActiveDefense.exe (3495)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x17C4</td>
<td>proexp64.exe (4923)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0xAD0</td>
<td>PerfWatson2.exe (3880)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
<tr>
<td>Process</td>
<td>0x1804</td>
<td>PerfWatson2.exe (12088)</td>
<td>0x00001478 VM_OPERATION</td>
<td>VM_READ</td>
</tr>
</tbody>
</table>

We need to find a different approach.
Driver - Synapse (CVE-2017-9769)

- IOCTL gives us access to ZwOpenProcess

- If AC control the access at kernel level it won't work :(  

- We need a better approach
Driver - GIGABYTE Drivers

- CVE-2018-19320 (ring0 memcpy with VA)
- CVE-2018-19321 (read/write arbitrary physical memory)

- Non-privileged user processes are able to get a HANDLE and issue IOCTL codes

- How could we use this?
Driver - DKOM

1) Load the vulnerable Driver and get a HANDLE (open DACL)
2) Search for EPROCESS Struct in kernel
   
   ```
   typedef struct
   {
       CHAR    ImageFileName[15];
       DWORD   PriorityClass;
   };
   ```
3) Obtain the ObjectTable (HANDLE_TABLE)
4) Use ExpLookupHandleTableEntry(HandleTable, Handle)
5) Retrieve HANDLE
6) Modify GrantedAccess
7) Overwrite kernel memory
8) Profit
```c
unsigned __int64 v2; // rdx@1
__int64 v3; // r8@2
signed __int64 v4; // rax@2
ULONG v5; // rax@3
unsigned __int64 result; // rax@4

v2 = handle & 0xFFFF000000000000u64;
if (v2 >>= *(DWORD *)a1)
    result = 0x164;
else
    v3 = *(__int64 *)(a1 + 8);
    v4 = *(__int64 *)(a1 + 8) & 0x164;
    if (*((DWORD)v4 == 1)
```

Refresher- Bypass Hooks

Disadvantages

- Suspicious new HANDLEs
- Hooks to user-mode WIN API
- Thread with suspicious context
- Downgrade of HANDLE privileges
Conclusions

- Fight at kernel level
- It could be trivial
- Blacklisting all drivers is impossible
- Compatibility with Windows and 3rd applications is a problem
Conclusions

AntiCheat-Testing-Framework

- CheatHelper & DriverHelper
- DriverDisabler
- HandleHijackingDLL and HandleHijackingMaster
- StealthHijackingDLL and StealthHijackingMaster
- WinApi Hooking Bypass (Direct call to syscalls)
- Lua Hooking (with pattern scanning)
- Synapse Driver exploit (Razer)
- Handle Elevation (Gigabyte Driver)

Github: niemand-sec/AntiCheat-Testing-Framework
THANK YOU!

More information at niemand.com.ar