

Game Security

Tencent
Kelvin

About Me

Name: Kelvin Chan

Work: Tencent Game Security and Windows Kernel Researcher

Facebook: Kelvin Chan

Email : KelvinChan@tencent.com

Introduction

- Understanding why a games security exists.
- Sharing some specific work and operation of a Commercial Security Research team
- Explaining how a cheat develop.
- Understanding why low-level knowledge strongly related to security research.
- Understanding the overview of how a game security production.

Outline

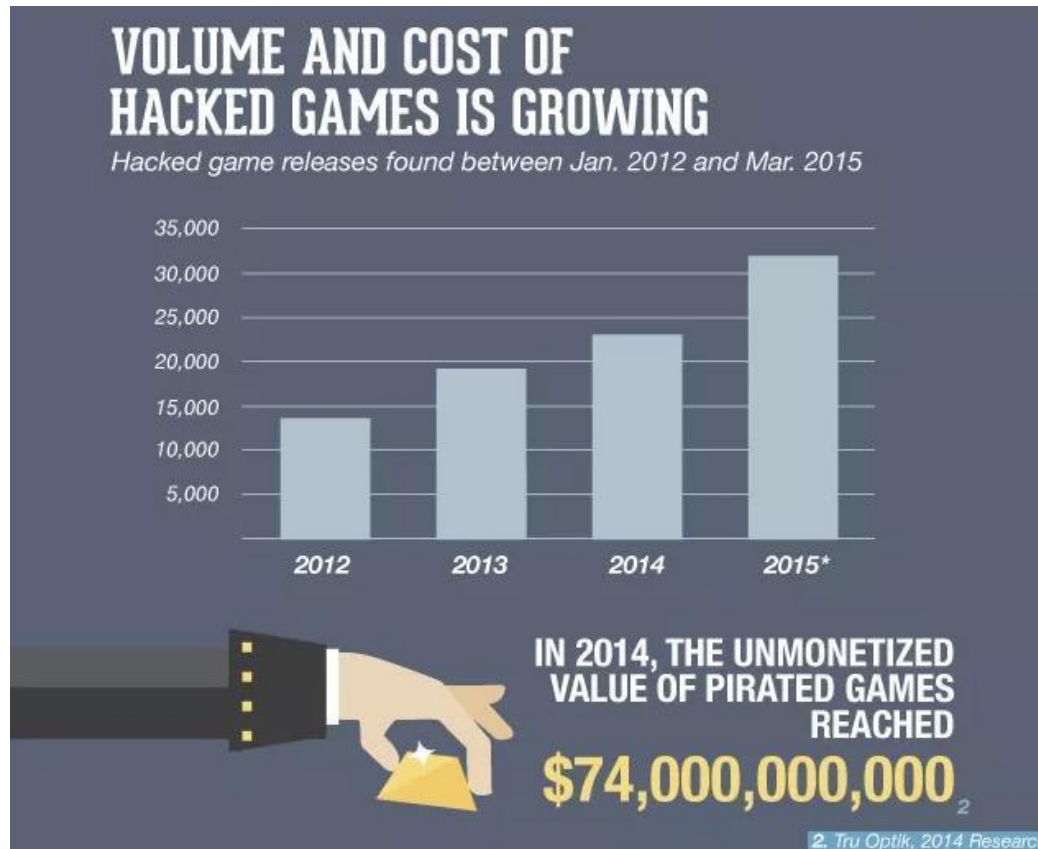
- Type of dark market
- Value of Game Security
- How cheat is developed
- Regular Works of Game Security
- How to defense a hackers?
- OS / CPU based research
- How research being used and example
- Monitoring
- Production and Stability

Type of black market

- DDOS
- Trojan
- Spam
- Malware
 - E.g. Ransomware
- Game Cheat Studio

Values of Game Security

- It is a big “business” in dark market, really profitable
- In 2014, whole games industry is lost 74 billion in the US, because of game hacked.



Revenue of Tencent Game

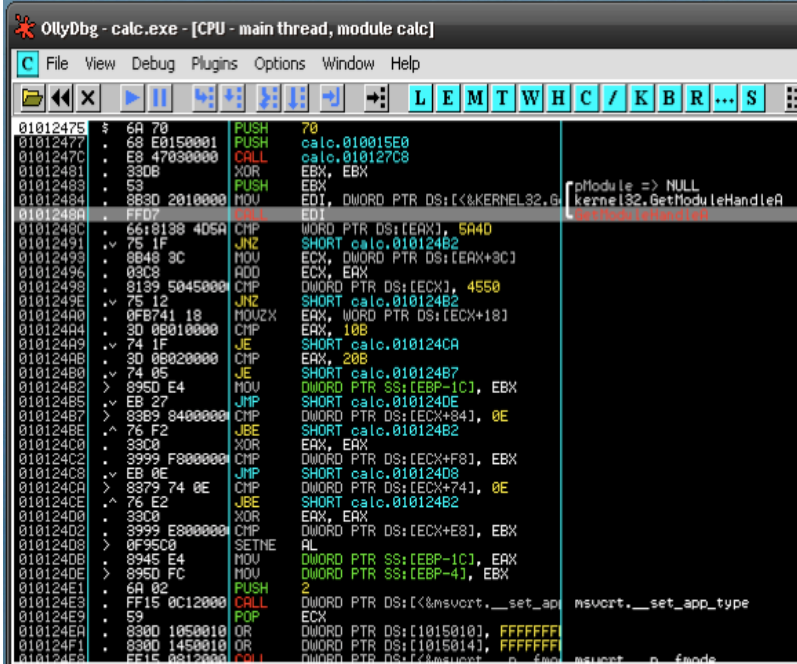


Type of Cheat

- Automation attack, run, do a mission, emulate any process specific with game logic, etc.
- I/O Device emulation, keyboard, mouse, joystick, etc.
- Game Logic modification, invincible character, unlimited HP(Health Point)
- Speed Acceleration
- Network Packet Emulation, fully emulate a client with reverse engineering whole network protocol

How cheat is developed?

- Online-Game, actually is a process that running on a specific OS (e.g. Windows)
- Normally, Hackers will analysis a game first, by **Reverse Engineering(Debugging)**
- Hackers will get the information what they needs. Such as, an address of Function(e.g. attack function), a character object base address, monster object base address. any character attribute loaded in the memory.
- Typical skills, stack-trace, hardware breakpoint, Software breakpoint, and step-into, so on.



OllyDbg - calc.exe - [CPU - main thread, module calc]

File View Debug Plugins Options Window Help

Assembly window showing instructions:

```
01012475 $ 6A 70 PUSH 70
01012477 68 E0150001 PUSH calc.010015E0
0101247C E8 47030000 CALL calc.010127C8
01012481 330B XOR EBX, EBX
01012483 53 PUSH EBX
01012484 8B3D 20100000 MOV EDI, DWORD PTR DS:[<<KERNEL32.6
0101248A FFD5 DD
0101248C 66:8138 405A CMP WORD PTR DS:[EAX], 6A40
01012491 75 1F JNZ SHORT calc.010124B2
01012493 8B48 3C MOV ECX, DWORD PTR DS:[EAX+3C]
01012496 03C8 ADD ECX, EAX
01012498 8139 50450000 CMP DWORD PTR DS:[ECX], 4550
0101249E 75 12 JNZ SHORT calc.010124B2
010124A0 0FB741 18 MOVBX, WORD PTR DS:[ECX+18]
010124A4 3D 0B010000 CMP EAX, 10B
010124A9 74 1F JE SHORT calc.010124CA
010124AB 3D 0B020000 CMP EAX, 20B
010124B0 74 05 JE SHORT calc.010124B7
010124B2 895D E4 MOV DWORD PTR SS:[EBP-1C], EBX
010124B5 EB 27 JMP SHORT calc.010124DE
010124B7 8B89 84000000 CMP DWORD PTR DS:[ECX+84], 0E
010124BE 76 F2 JBE SHORT calc.010124B2
010124C0 33C0 XOR EAX, EAX
010124C2 3999 F8000000 CMP DWORD PTR DS:[ECX+F8], EBX
010124C5 EB 0E JMP SHORT calc.010124D8
010124C8 3379 74 0E CMP DWORD PTR DS:[ECX+74], 0E
010124CE 76 E2 JBE SHORT calc.010124B2
010124D0 33C0 XOR EAX, EAX
010124D2 3999 E8000000 CMP DWORD PTR DS:[ECX+E8], EBX
010124D8 0F95C0 SETNE AL
010124DB 8945 E4 MOV DWORD PTR SS:[EBP-1C], EAX
010124DE 895D FC MOV DWORD PTR SS:[EBP-4], EBX
010124E1 6A 02 PUSH 2
010124E3 FF15 0C120000 CALL ECX
010124E9 53 POP ECX
010124EA 3380 10500100 OR EDI, DWORD PTR DS:[1015010], FFFFFFFF
010124F1 3380 14500100 OR EDI, DWORD PTR DS:[1015014], FFFFFFFF
010124F3 FF15 08120000 CALL ECX
```

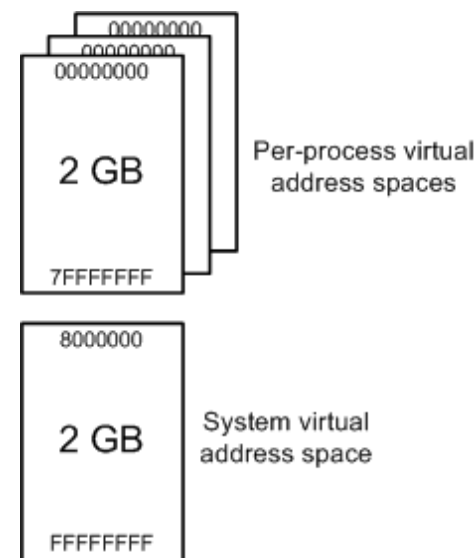
Hex dump window showing memory values:

Address	Hex dump	ASCII
01014000	03 00 00 00 01 00 00 00	...0...
01014008	20 00 00 00 0A 00 00 00	...
01014010	0A 00 00 00 40 00 00 00	...
01014018	53 00 63 00 69 00 43 00	S.O.L.C.
01014020	61 00 6C 00 63 00 00 00	a.l.c...
01014028	00 00 00 00 2E 00 00 00	...
01014030	00 00 00 00 00 00 00 00	...
01014038	2C 00 00 00 00 00 00 00	...
01014040	00 00 00 00 30 00 00 00	...
01014048	01 00 00 00 00 00 57 00	...M...
01014050	58 00 56 01 5C 02 5D 02	X.U0010
01014058	07 03 53 03 5E 03 59 03	..V*7*

Analysing calc: 158 heuristical procedures, 274 calls to known, 167 calls to guessed functions

How cheat is developed?

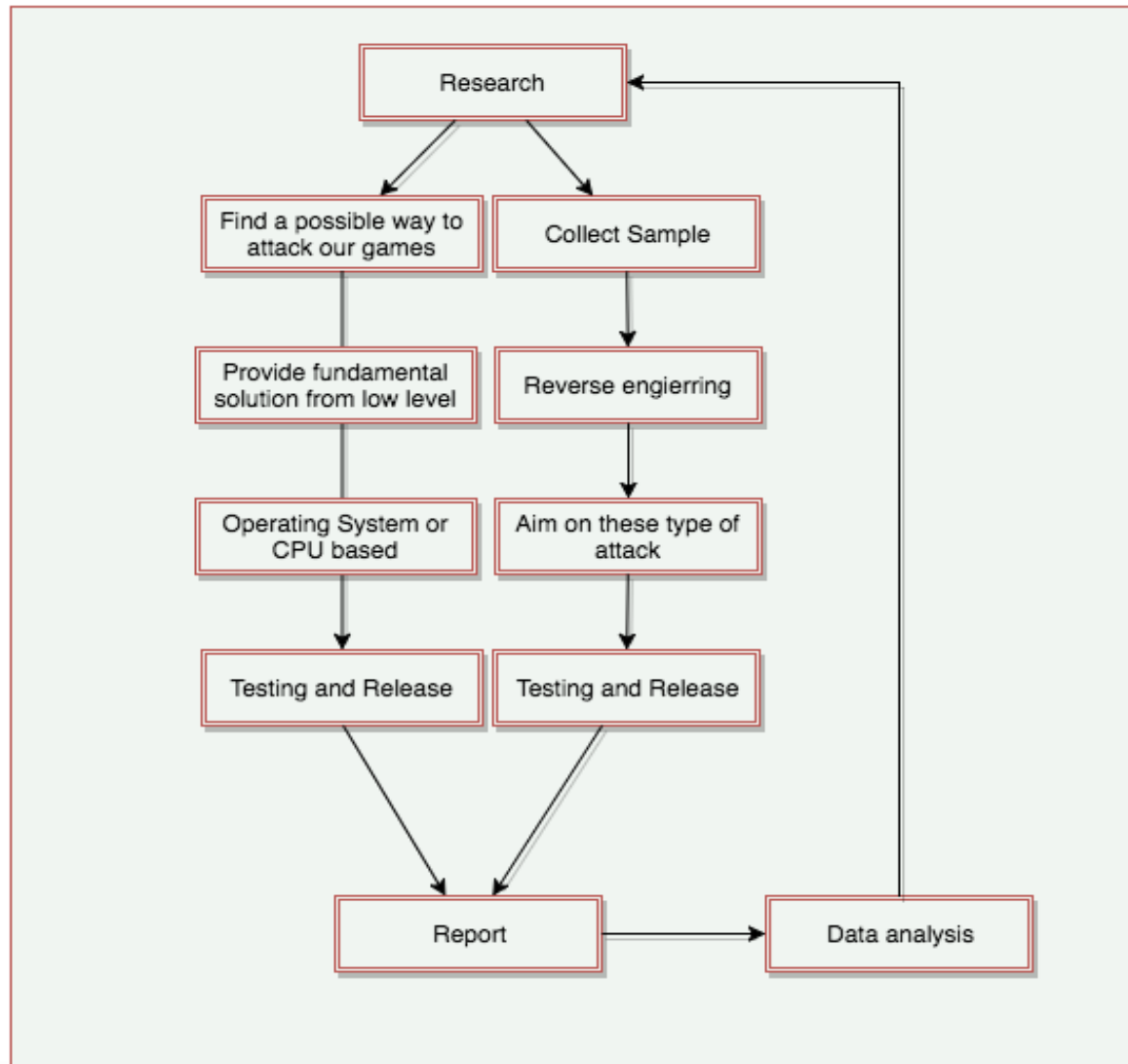
- Hacker will then build a cheat program / module.
- And since the memory address space is isolated when we started protected-mode and paging from CPU start-up period.
- Some common method they may used:
 - Directly use windows API, **WriteProcessMemory**
 - Inject a module into game process, and directly read, write by normal memory access
 - Driver direct read/write (relatively rare)
 - ShellCode injection
 - so on.... (unlimited method for hacking...)



How cheat is developed?

- Generic Attack Method:
 - Attack on windows kernel protection mechanism, such as, attack patch guard for hooking some critical API
- Game Logic modification and Automation :
 - Memory hiding
 - attack on windows kernel protection mechanism
- Speedy Acceleration :
 - Modify a clock, and fake a system the next second is actually second * 2 (depend on how many we what)
- Keyboard emulation :
 - RGB capture
 - Emulated by SendMessage to Game Window
 - Direct write to the I/O port which is corresponding to the keyboard and mouse.
 - Driver emulation, send a keyboard / mouse message packet from kernel to user mode.
 - Game Enginee hacked (e.g. Direct-X)

How to defense a hackers?



Why OS / CPU based research

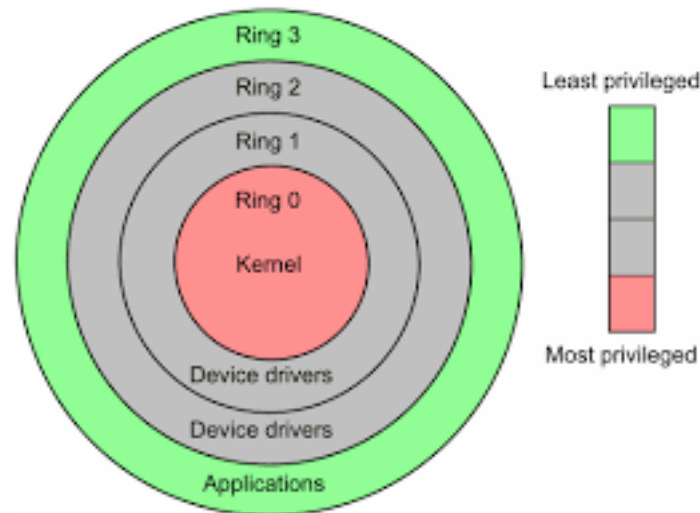
- Any cheat, running in a specific OS, it is just a process, or module. So, from the OS aspect to protect our game. It is a good choice.

- As low level as possible. Because all of the API a cheat used will finally be called into the Ring 0(kernel mode).

- More Accurately, some instruction will be running on Ring 0 level, so the granularity of detection point can be a instruction execution (if we can, VT-x may be a good choice.)

- p.s. Windows is only used Ring 0 and Ring 3 as kernel mode and user mode respectively.

- Ring 1 and Ring 2 is reserved, for software-based-virtualization, they maybe used for isolation between VMs

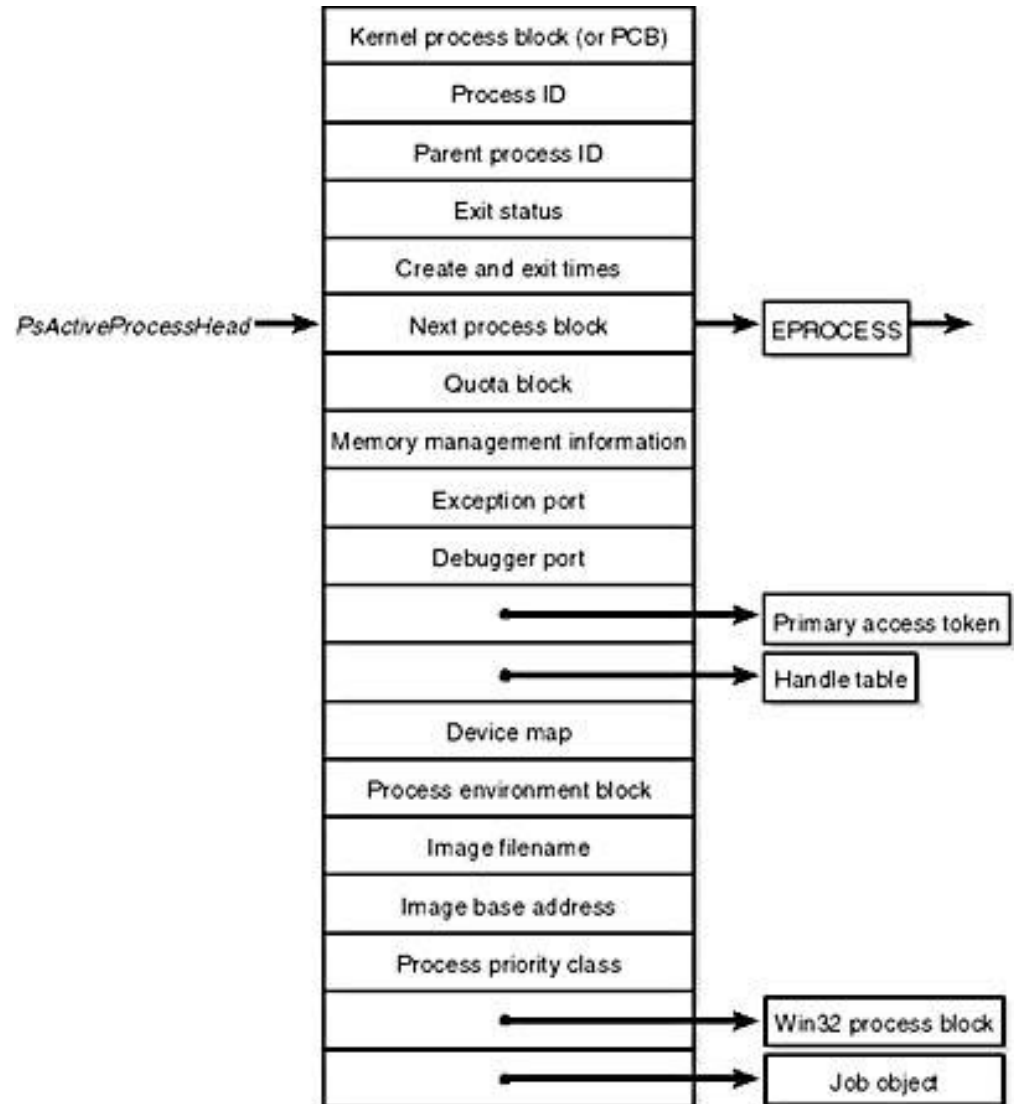


How research being used ?

- Since hackers need to ensure they are able to debug(reversing) our games, then getting the information they wants.
- So, anti-debugging will be the very first wall for defending their attack.
- As we said, a process is running on Windows, so, we can reused this concept for anti-debugging.
- So the question is : How debugger / disassembler works?
- We need to proceed a research of debugging and analysis, for understand how debugging works in specific windows.

Example

- Since we need to anti-debugging
- After studying how a debugging is supported by Windows kernel.
- We can know that, debugger and debuggee is connected by a kernel-maintained structure,
- This structure is stored in process structure as a member, called **DebugPort**, which is an pointer of debugport structure.
- If we detect or clear this member, we can know if we are being debugged, or stop the debugging instantly.



Monitoring

- Exception exploitation

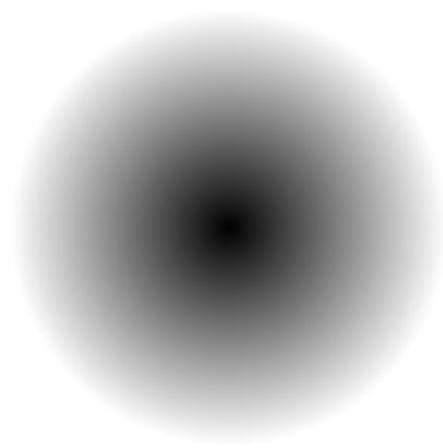
- For example:
 - We directly set a specific memory address's page is invalid, and any process access this page will issue a Page-Fault.

- Virtualization-based method

- Intel VT-x provided Extended page table(EPT) for virtualizing a physical memory.
- This ability is good for monitoring a memory accessing.
- Because Exception exploitation is not a perfect method for monitoring, it is dangerous in case we modify the OS logic.

Production and Stability

- Full System test – we need to make a full system test on Windows before we release a protect solution.
- Gray Scale - we need to ensure the stability of over 100 million end-user environment is stable.



Conclusion

- Explained why game security needs R&D
- Given a overview of the war between hackers and research team.
- That's why low level is required for playing hacking or security research.
- In real commercial security production, we need to 100% ensure, our solution is not affecting the games(product) functionality.

<https://github.com/Kelvinhack/kHypervisor>

End