Next Level Cheating and Leveling Up Mitigations

Nicolas Guigo

Joel St. John
A brief history of cheating in video games
Current state of the arms race (cheating vs anti-cheat)
The future of cheating
Attacking anti-cheat software
Solutions and conclusions
The Rise of eSports

- Online gaming
- LAN tournaments
- Televised gaming
- Worldwide popularity
  - League of Legends 2013 – 32 million viewers
  - Dota 2014 – 20 million viewers
2013 Viewers of Major Sporting Events (in millions)

- Major League Baseball World Series: 14.9
- NCAA Basketball Final Four (Average): 15.7
- NBA Basketball Finals (Game 7): 26.3
- BCS National Football Championship: 26.4
- League of Legends Season 3 World Championship: 32.0

Source: Quartz/qz.com
The Money Aspect

- Multi-billion dollar industry
- Subscription models
- Streaming/Sponsorship
- Virtual grey market
What is Cheating?

• Unfair advantage
  • Abusing game logic
    • Multi-accounts
    • Botting/Scripting
    • Manipulating extraneous client-side data
  • Exploiting client / server code bugs
    • Abusing bugs/glitches
    • Attacking other players or the game server
A History of Cheating

• Early computer games
• Early multiplayer games
• Modern multiplayer games
• Examples!
Common Cheating Vectors

- Speed/Movement hacks
- Botting
Common Cheating Vectors

- Speed/Movement hacks
- Botting
- Scripting/Aim bots
- Player/item finding hacks
Common Cheating Vectors

- Speed/Movement hacks
- Botting
- Scripting/Aim bots
- Player/item finding hacks
- Wall hacks/x-ray mods
The Rise of Anti-Cheat

  - World of Warcraft
  - Starcraft 2
- Valve Anti-Cheat (VAC, 2002)
  - Counter-Strike
  - Team Fortress 2
- BattlEye (2004)
  - Arma 2/3
  - Day-Z

- User-land
- Reactive
- Only a mitigation
The Current State of Cheating in Games

- DLL injection (internal cheating)
  - Loader
  - DLL implementing cheat logic
    - Hook Direct3D calls
    - Read/Write memory
- Network packet manipulation
  - Modify packets in-transit
  - Repeat packets
  - Introduce artificial lag
- External cheating
  - ReadProcessMemory / WriteProcessMemory
  - Transparent window
Current State of Anti-Cheat

- In process
  - Signature checks
  - Game specific checks
  - Hook detection
    - Pointer chain checks
  - Call stacks periodic checks
  - Debug related detections
- Out of process
  - Signature based detection
  - Pattern searching in all processes address space
- Various
  - Scanning for game process handles
  - Scanning files for signatures (offline)
    - Send suspected programs to server for analysis
  - Check DNS history for cheat update servers
  - Obfuscation
  - Etc.
The Future of Cheating

• Architecture
  • Rootkit-like functionality to hide activity
  • Kernel driver
    • Makes the UM portion a protected process (DRM)
    • Maps pages from game memory into the cheat process
    • Install a filter device on the FS stack (TBD)
  • User mode executable
    • Keeps track of game/cheat mappings
    • Implements the cheat logic
status = PsLookupProcessByProcessId((PUINT)ncmmmap->process, (PEPROCESS*)&epb);
if(NT_SUCCESS(status)) {
    ncmd1 = NcAllocateMdl((PUINT)ncmmmap->baseAddress, ncmdmap->len);
    if(ncmd1) {
        KeStackAttachProcess(epb, &kapcstate);
        MmInitializeMdl(&ncmd1->md1, (PUINT)ncmmmap->baseAddress, (SIZE_T)ncmmmap->len);
        _try {
            MmProbeAndLockPages(&ncmd1->md1, UserMode, IOWriteAccess);
        } _except(EXCEPTION_EXECUTE_HANDLER) {
            getout=True;
        }
        KeUnstackDetachProcess(&kapcstate);
        if(!getout) {
            _try {
                userva = (DWORD64)MmMapLockedPagesSpecifyCache(&ncmd1->md1, UserMode, MmCached, NULL, FALSE, NormalPagePriority);
            } _except(EXCEPTION_EXECUTE_HANDLER) {
                userva = 0;
            }
            if(userva) {
                ncmd1->md1.StartVa=(PUINT)userva;
            }
            MmUnlockPages(&ncmd1->md1);
        } // if getout
        else {
            ExFreePoolWithTag(ncmd1, NCDRIVER_TAG);
        } // if ncmd1
    } // if ncmdmap
    ObDereferenceObject((PUINT)epb);
} // if process
Dual-mapping demo
Pros / Cons

• Strengths
  • Generic
  • Virtually undetectable from user-mode
  • Straightforward conversion from publicly available cheat sources
  • Good performance

• Weaknesses
  • Can be challenged by KM anti-cheat
  • Run in debug mode or use signed driver
Attacking Anti-Cheat Software

• Anti-cheat libraries create additional attack surface
  • On client
  • On server
• This attack surface is common to multiple games
• What happens if there is a flaw?
BattlEye

- General architecture
  - On the client
    - DLL in game process
    - System service
  - On the server
    - DLL in game server process
  - Master server
- Hooks game recv() call
BE Packet structure

- Packet structure

<table>
<thead>
<tr>
<th>'B'</th>
<th>'E'</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
<th>LEN</th>
<th>DATA</th>
</tr>
</thead>
</table>

- 2 bytes signature
- Hash
- Len /code
- data
Sign extension
Integer overflow -> heap overwrite
Exploitability

- Denial of Service is trivial
- Remote code execution possible
  - Overwriting heap data
  - Attacker-controlled data
- Very difficult
  - Separate heap limits attack surface
    - Tool: https://github.com/iSECPartners/vtfinder
  - Race condition
    - Code execution must be achieved before thread crashes
    - Must then prevent crash from happening
BattlEye console timing attack

- Length check
- String comparison
BattlEye timing attack demo
Disclosure timeline

• Both vulnerabilities
  • Verified 08/2014
  • Disclosed to vendor 08/2014
• Bugs
  • Memory corruption | fixed (as of 11/2014)
  • Login vulnerability | unpatched as of 11/2-14 (current status unknown)
• New feature
  • Obfuscated kernel driver – iSEC has not investigated
The Future of Anti-Cheat

• Mitigations
  • Move the arms race to the kernel
  • Human factor
• Solutions
  • Full streaming of games
  • Closed platform
Conclusion

- Anti-cheat is a mitigation at best
- Anti-cheat creates additional attack surface
- Current anti-cheat can be completely bypassed
- Fundamental design changes are needed
Follow-up

- “Next-gen” is about to become “Current gen”
- Anti-cheat creates additional attack surface
  - Now running obfuscated code in kernel space!
- Streaming is gaining momentum
  - “Arcadia” Xbox to windows
  - Others to follow (?)
Questions

• Thank you
  • Rachel Engel & Jason Bubolz
  • Rohit Shambhuni (iSEC 2014 Intern, Arizona State)
  • Taylor Trabun (iSEC 2014 Intern, University of Idaho)
  • Too many iSECers to list

Interns are people too!
References

• Vtfinder. https://github.com/iSECPartners/vtfinder
• “eSports, the newest spectator sport” https://www.whitehutchinson.com/blog/2014/04/esports-the-newest-spectator-sport/
UK Offices
Manchester - Head Office
Cheltenham
Edinburgh
Leatherhead
London
Thame

European Offices
Amsterdam - Netherlands
Munich – Germany
Zurich - Switzerland

North American Offices
San Francisco
Atlanta
New York
Seattle

Australian Offices
Sydney