Hacking / Exploiting / cheating in Online Games

Shahin Ramezany
www.abysssec.com
shahin@abysssec.com
Twitter : @abysssec
Who am I?
CTO AT Abysssec
Doing some:

- Next Generation hacking
- Exploit-development
- Reversing
- Web-Audit
What is this talk about?

I. This talk is about hacking / exploiting cheating in online games.

II. Real world Cheating is mainly focused in this talk, because it's fun and legal. During this talk we will have a tour into all of ways to manipulating an online game and we will end up with bypassing latest anti-cheating technologies and manipulating the game to our heart's desire.

III. After this talk you can go and play a bit and test your learned lesson.
Part I : introduction
Part II : Hacking Online Game Servers
Part III : Exploiting Online Games
Part IV : Cheating in Online Games
Part V : Creating your own cheats
Part VI : Bypassing anti-cheat engines
Part I : Introduction
Why Exploit/Hack Online Games?

- Why not?
- Millions of players around
- Cheating make you a pornstar in games
- Impress your friends
- Get some unique insults as well!!
State of Online games!

- Counter-Strike
- Current Server: 94,964
- Player Minute in month: 51,576
- Current Unique Players: 2,834,131 / per month!
State of Online games!

- Word Of Warcraft (WoW)
- Current Player: 12 million
State of Online games!

- Call of duty (COD)
- Current Player: 14 million
Hacking VS. Exploiting VS. Cheating

1- Hacking

For Hacking a game server / client you can use normal penetration testing ways.

Server:
- Normal network based attacks
- Our lovely web based attacks

Client:
- Social engineering family
- SET / Metasploit
- Exploits / Bots / Key loggers / Custom Malware, Trojans,...
Hacking VS. Exploiting VS. Cheating

2- Exploiting
For Exploiting game server / client you can use normal ways to audit both server / client.

Server:
- Fuzzing
- Reverse Engineering
- Code audit

Client:
- Same as server but in servers you should try to fuzz protocol but for client you should focus on game imports from client
Hacking VS. Exploiting VS. Cheating

3- Cheating

Cheating is a bit different in some case even if you PWN the server or client you can not use silent and cool cheats on servers

Server:
- Sending crafted packets for changing some functionality in game (depends on server)

Client:
- Changing game models and add custom ones
- Using Bots for automating gaming tasks (play with AI)
- Using game features against it!
Part II: Hacking Online Games
Hacking online games

As I already said for hacking a game server we can use available methods for penetration testing projects.

Most simplest example is finding a gaming portal (used for players statics, game server states, etc.) PWN the portal then PWN the game. Due to lots of game portals have permission to game database hence you may see lots of them use root or SA for their game servers.
Hacking online games

Here is just some example of vulnerable Gaming CMSs

- Mafia Game Script SQL injection Vulnerability
- mygamingladder MGL Combo System <= 7.5 game.php SQL injection Exploit
- Chipmunk Pwngame Multiple SQL Injection Vulnerabilities
- Joomla Component Gamesbox com_gamesbox 1.0.2 (id) SQL Injection Vulnerability
- Eyeland Studio Inc. (game.php) SQL Injection Vulnerability
- PHP Gamepage SQL Injection Vulnerability
- Games Script (Galore) Backup Dump Vulnerability
- GameScript v3.0 SQL Injection Vulnerability

Even the CMS itself does not have access to Game server the CMS admin mostly have.
Hacking online games

We found a simple one in IGaming CMS during MOAUB and didn’t report it.
Hacking online games

Really Old-School Blind SQL Injection in iGamingCMS in gamedetails.php

gamedetails.php file line 32:

$result = $db->Execute("SELECT * FROM `sp_games` WHERE `id` = "'$_REQUEST[id]' LIMIT 1");

PoC:
http://lamesite.com:/iGamingCMS/gamedetails.php?id=[Inject the code here]

Sounds like still 0day but I call it garbage 0day.
Hacking online games

I found a similar bug in online charging game portal that lead me to completely PWN the server. You can search and do similar things like me.

But there is an important note: most of these server hacking styles are possible in PUBLIC / PRIVATE game servers, not the main game developer server. For hacking main developer servers you have to do much more.

E.G: blizzard server is that easy to PWN.
Part III: Exploiting Online Games
Exploiting online games

1337 stuff. Exploiting online games as I have already said, is like finding vulnerability and exploiting normal applications, so normal attacks works for game engines too. But the most important thing you should know about vulnerability discovery and exploiting online games, is where games receive INPUTS. Basically, if you are not a player it's not clear for you. But at least all of games have some parsers for input files and packets.
Exploiting online games

A normal game (with capability of multiplying) at least will have following inputs:

- Network packets (for all playing stuff)
- Save games / stats / scripts
- Models and items
- Levels and maps
- Maybe movies and sounds
- And so on …
Exploiting online games

Unlike a normal program when you are auditing a game maybe you have to pass some simple or advanced encryptions.

- Packed / protected binaries
- Encrypted network packets
- Encrypted models, levels saves, items
- Encrypted sounds, movies
- Maybe movies and sounds
- And other encrypted stuff.
Exploiting online games

Here is some games those use encryption for their packets:

- Half-life
- Halo
- GS4
- Call of duty
- World of Warcraft
- ...
void hlenc(unsigned char *buff, unsigned int pcksz) {
    #define HL_NTOHL(x) \
        ((((x) & 0xff000000) >> 24) | \
         ((((x) & 0x00ff0000) >>  8) | \
         ((((x) & 0x0000ff00) <<  8) | \
         ((((x) & 0x000000ff) << 24))

    const static unsigned char hlhash[] =
        "x05""x61""x7A""xED""x1B""xCA""x0D""x9B""x4A""xF1""x64""xC7""xB5""x8E""xDF""xA0";
    unsigned char *ptrebpc;
    unsigned int *lbuff = (unsigned int *)buff,
        pcknum, 
        invnum, 
        ebpc;
    int 
        cont,
        i;

    if(pcksz < 9) return;
    pcknum = *buff;
    invnum = ~pcknum;
    pcksz = (pcksz - 8) >> 2;
    lbuff += 8;
    cont = 0;

    while(pcksz--) {
        ebpc = *lbuff ^ invnum;
        ebpc = HL_NTOHL(ebpc);

        ptrebpc = (unsigned char *)&ebpc;
        for(i = 0; i < 4; i++) {
            *ptrebpc ^= ((hlhash[(cont + i) & 0xf] | (i << i)) | i) | 0xA5);
            ptrebpc++;
        }

        *lbuff = ebpc ^ pcknum;
        lbuff++;
        cont++;
    }
}
void hldec(unsigned char *buff, unsigned int pcksz) {
  #define HL_NTOHL(x) 
          (((x) & 0xff000000) >> 24) | 
          (((x) & 0x00ff0000) >>  8) | 
          (((x) & 0x0000ff00) <<  8) | 
          (((x) & 0x000000ff) << 24))

const static unsigned char hlhash[] =
  
  "\x05\x61\x7A\xED\x1B\xCA\x0D\x9B\x4A\xF1\x64\xC7\xB5\x8E\xDF\xA0";

unsigned char  *ptrebpc;
unsigned int   *lbuff = (unsigned int *)buff,
               pcknum,
               invnum,
               ebpc;
int            cont,
               i;

if(pcksz < 9) return;
pcknum = *buff;
invnum = ~pcknum;
pcksz = (pcksz - 8) >> 2;
*lbuff += 2;
cont = 0;

while(pcksz--) {
  ebpc = *lbuff ^ pcknum;

  ptrebpc = (unsigned char *)&ebpc;
  for(i = 0; i < 4; i++) {
    *ptrebpc ^= (((hlhash[(cont + i) & 0xf] | (i << i)) | i) | 0xA5);
    ptrebpc++;
  }

  *lbuff = HL_NTOHL(ebpc) ^ invnum;
  *lbuff += 2;
  cont++;
}
}
Exploiting online games

Finding vulnerabilities in games is not totally new stuff, Luigi Auriemma is most active researcher (that I know) in hunting vulnerabilities in game engines. Some examples:

- Invalid memory access in Unreal Tournament 3 2.1
- Failed assertion in old games based on Unreal engine
- Two vulnerabilities in Ghost Recon Advanced Warfighter 1 and 2
- Clients unicode buffer-overflow in Unreal engine 2.5
- Negative memcpy in id Tech 4 engine
- Buffer-overflow in the Electronic Arts games that use Gamespy
- Files uploading vulnerabilities in the Source engine (build 3933 and 3950)
- Format string in Crysis 1.21 and Crysis Wars/Warhead 1.5
- Half-Life broadcast client's buffer-overflow (versions 1.1.1.0)
- Half-Life servers: buffer-overflow and freeze (versions 1.1.1.0, 4.1.1.1c1 and 3.1.1.1c1)
Exploiting online games

You can even see some really old-school vulnerabilities in game engines. A bug found by Luigi in 2004 in unreal engine secure packet.

send a similar UDP packet to the query port of the game server: 
\secure\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...aaaa

Both code execution and spoof where possible by using this vulnerability.
Exploiting online games

Unfortunately, there is no exploit mitigation available is most of games. Due to lack of OS exploit mitigation, Exploiting games even in most modern systems is not hard.

No DEP+ASLR in half-life 😊 ☹️
Exploiting online games

Finally, if you understand game algorithms for sending / receiving packets and pass encryptions correctly, you still can find great vulnerabilities using fuzzing and static analysis.
Part IV: Cheating in Online Games
Cheating in online games

Why we should do cheat instead of playing like a good person?
Cheating in online games

Because it’s better than cheating friend.
Cheating in online games

And because we can't cheat death.
Cheating in online games

Because you can get $$$ from cheating. People will pay for working cheats in multiplayer online games.
Cheating in online games

How to create cheat for games? When you are not dealing with online games, cheating is not that hard. All you should do is find values in memory and then freeze or change them.

- Health
- Money
- Ammo
- Time
- And even functionalities like:
  - Jump
  - Speed
  - Fly
  - Swim
  - And so on...
Cheating in online games

For finding values you can use differential-reversing or simple and great available tools. The best tool I know is Cheat-Engine which is free and open source.
Cheat Engine is an open source tool designed to help you with modifying single player games running under Windows so you can make them harder or easier depending on your preference (E.G: Find that 100hp is too easy, try playing a game with a max of 1 HP), but also contains other useful tools to help debugging games and even normal applications. It comes with a memory scanner to quickly scan for variables used within a game and allow you to change them, but it also comes with a debugger, disassembler, assembler, speedhack, trainer maker, direct 3D manipulation tools, system inspection tools and more.
Cheating in online games

Will this cheat-engine and other kind of cheating programs work on online games?

In most case answer is clear:

NO !!!
Cheating in online games
Also for packet editing there is a really simple program called WPE-Pro.
Cheating in online games

Basically, WPE-Pro in just a real-time sniffer and packet editor and due to being easy to use is popular. It is used widely for game hacking even in Online games. But when hack will be done using the packets server will fix it too. Let's see a real example in World of Warcraft.
Cheating in online games

First you need find a pattern and then change the value some values are just BOOLS and need to change 1–0 or 0–1.
Cheating in online games

Here is an example for buying any item without money.
Cheating in online games

Will this kind of hacks still works? Maybe YES in some private servers but in main servers the answer is AGAIN:

NO !!!
Cheating in online games

Why these doesn’t work or worked for a while ?

- Anti-cheats (like exploit mitigations)
- Some checks are server side only
- CRC-checks and anti-modifications
- Patching patterns
- Because even cheats are not free!!
Cheating in online games

So what is the solution?
Creating your OWN cheat!!!
Part V: Creating your own cheats
Creating your own cheat

Before going forward we should know what kind of hacks we can do in multiplayer online games? Most of silent and popular hacks we can do are:

- Wallhack
- AutoAim
- Sky/Water/flash/smoke removal
- Speed hacks
- ESP
- Fly Hack
- Model modification
Creating your own cheat

In this section we want to talk about creating custom cheats for Counter Strike game. We will go steps from scratch to making reliable cheat for game. For making cheating specially for online games there is some important factors:

1. what game server know about cheat?
2. how many checks are server side ?
3. What we can do on client ?
4. If it detect the cheats how will do it?
Creating your own cheat

Warning: Counter strike is tactical first person shooter game and is amazingly addictive!
Creating your own cheat

Wall hack:

Wallhacking allows a player to see through solid or opaque objects and/or manipulate or remove textures, to know in advance when an opponent is about to come into targeting range from an occluded area. This can be done by making wall textures transparent, or modifying the game maps to insert polygonal holes into otherwise solid walls.

As with the aimbot, wallhacking relies on the fact that an FPS server usually sends raw positional information for all players in the game, and leaves it up to the client's 3D renderer to hide opponents behind walls, in plant foliage, or in dark shadows. If the game map rendering could be turned off completely, all players could be seen moving around in what appears to be empty space. Complete map hiding offers no advantage to a cheater as they would be unable to navigate the invisible map pathways and obstacles. However if only certain surfaces are made transparent or removed, this leaves just enough of an outline of the world to allow the cheater still to navigate it easily.
Creating your own cheat

Wall hack example:

Can you see guy beyond wall?
Creating your own cheat

Aimbot (autoAim):

An aimbot (sometimes called "auto-aim") is a type of computer game bot used in multiplayer first-person shooter games to provide varying levels of target acquisition assistance to the player. While most common in first person shooter games, they exist in other game types and are often used in combination with a TriggerBot, which shoots automatically when an opponent appears within the field-of-view or aiming reticule of the player.

Aimbotting relies on the fact that each client computer must be typically sent information about all players, whether seen or unseen. Targeting is simply a matter of finding the position difference of where the player is located and where any opponent is located, and pointing the player's weapon at the target. This targeting works regardless of whether the opponent is behind walls or too far away to be seen directly.
Creating your own cheat

By default when you run the game and create a server, a few things will be checked (for example when you playing from LAN) and you can do a lot of modifications from your client. But for modification, you should know what you want modify. So first step in every game hacking is playing the game.
Creating your own cheat

For example after a bit playing you will understand there a cross-hair for every guns except AWP and SCOUT
Creating your own cheat

So one of the modifications would be adding a NICE cross-hair to these guns. But how?

First step is find the gun model. So we can search the valve folder for AWP. After a while, you’ll end up with v_awp.mdl. Now how we can do modification on it? Just search for a model editor. I found jed’s half-life model editor by a bit of searching.
Creating your own cheat
Jed’s half-life model editor and AWP.
Creating your own cheat AWP with Crosshairs! You can decompile models using mdlsec.exe
Creating your own cheat

After recompiling model you can restart game and see what you did in game.

As you can see the added crosshair.
Creating your own cheat

The most important thing here is local server didn’t detect our modification. So we can do a lot of more by using modification; but there is a question. Won’t it detect other model modifications? What about if we remove flash-bang and smoke models completely?
Creating your own cheat

So I removed all models those have flashbang in their name and restart the game and I got a fatal error about it.

So the game didn’t detect modification but detects removing objects. What about patching checks in game?
Creating your own cheat

Ok, we know we can do modification, but how we can implement for example Wallhack by ourselves and inject it to game?

The answer is simple. We should write our Wallhack as a DLL and inject it into the game. Next question is how we can write Wallhack? The answer is again simple; you should know a bit about game developing really, really a BIT! Or patch the process for it.
Creating your own cheat

Here is code for patching HL.exe for a sample wallhack.

```c
void WallHackRipped()
{
    BYTE Patch[] = {0x68, 0x71, 0x0B, 0x00, 0x00, 0xFF, 0x15, 0x5C, 0x89, 0x7E, 0x02};
    BYTE Original[] = {0x68, 0x04, 0x04, 0x00, 0x00, 0xFF, 0x15, 0x50, 0x88, 0x7E, 0x02};
    DWORD HLB = (DWORD)GetModuleHandle(NULL);
    DWORD Addr1 = HLB + 0x99098C;
    DWORD Addr2 = HLB + 0x94663E;
    switch (isWallHackActivated)
    {
    case TRUE:
        memcpy((LPVOID)Addr1, Patch, 6); 
        memcpy((LPVOID)Addr2, Patch, 6);
        break;

    case FALSE:
        memcpy((LPVOID)Addr1, Original, 6);
        memcpy((LPVOID)Addr2, Original, 6);
        break;
    }
}
```
Creating your own cheat

And here is the code for wallhack without patching hl.exe using API hooking.

```c
void WINAPI MyglBegin(DWORD dwMode)
{
    typedef float GLfloat;
    GLfloat col[4];
    BOOL isSmoke = false;

    if ((dwMode == GL_TRIANGLE_STRIP) || (dwMode == GL_TRIANGLE_FAN))
    {
        switch (isWallHackActivated)
        {
        case TRUE:
            // disables wallhack
            glDisable(GL_DEPTH_TEST);
            break;
```
Creating your own cheat

case FALSE:
    // enables wallhack
    glEnable(GL_DEPTH_TEST);
    break;

else
{
    glEnable(GL_DEPTH_TEST);
}

pglBegin(dwMode);
}
Creating your own cheat

How to inject it to game? The simplest way is using CreateRemoteThread API.

/*******************************************************************/

BOOL InjectDll(DWORD pid, LPTSTR dllname)
{
    LPVOID     hRemoteMem;
    HANDLE     hProcess, hRemoteThread;
    HMODULE    hModule;

    //open remote process
    if((hProcess = OpenProcess(PROCESS_CREATE_THREAD|PROCESS_VM_OPERATION|PROCESS_VM_WRITE|PROCESS_VM_READ, FALSE, pid)) == NULL) {
        printf("Injection: OpenProcess failed\n");
        return FALSE;
    }

    Cont'd in next slide.
Creating your own cheat

allocate memory in remote process

```c
if((hRemoteMem = VirtualAllocEx(hProcess, NULL, strlen(dllname), MEM_RESERVE|MEM_COMMIT, PAGE_READWRITE)) == NULL)
{
    printf("Injection: VirtualAllocEx failed\n");
    return FALSE;
}
```

//copy the dll name to memory allocated in the remote processes' address space
```c
if(!WriteProcessMemory(hProcess, hRemoteMem, (LPVOID)dllname, strlen(dllname), NULL))
    printf("Injection: WriteProcessMemory failed\n");
    VirtualFreeEx(hProcess, hRemoteMem, strlen(dllname), MEM_RELEASE|MEM_DECOMMIT);
    return FALSE;
```

//need kernel32's handle for call to CreateRemoteThread()
```c
hModule = GetModuleHandle("KERNEL32.DLL");
```

//create thread in remote process, passing the address of LoadLibraryA for the thread's entry point
```c
//and the address of the DLL's pathname as an argument to the thread
hRemoteThread = CreateRemoteThread(hProcess, NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(hModule, "LoadLibraryA"), hRemoteMem, 0, NULL);
```

```c
if(hRemoteThread == NULL) {
    printf("Injection: CreateRemoteThread failed\n");
    VirtualFreeEx(hProcess, hRemoteMem, strlen(dllname), MEM_RELEASE|MEM_DECOMMIT);
    return FALSE;
}
```

Cont'd in next slide
Creating your own cheat

//cleanup
WaitForSingleObject(hRemoteThread, WAIT_TIMEOUT);
VirtualFreeEx(hProcess, hRemoteMem, strlen(dllname),
MEM_RELEASE|MEM_DECOMMIT);
CloseHandle(hProcess);

return TRUE;
}

Here was simplest injector using CreateRemoteThread if you search a bit you can find tones of working codes.
Creating your own cheat

//cleanup
WaitForSingleObject(hRemoteThread, WAIT_TIMEOUT);
VirtualFreeEx(hProcess, hRemoteMem, strlen(dllname),
MEM_RELEASE|MEM_DECOMMIT);
CloseHandle(hProcess);

return TRUE;

}
Creating your own cheat

Object removal: for removing functionality of an object you need first detect that object; for example flash / smoke or ... also you should be aware of conflicts during removing an object. Here is simplest flash / smoke hack for counter strike.
Creating your own cheat

To do this, again we need to hook OpenGL functions. This time let's use disassembler I used Beaengine.

```c
BOOL WINAPI HookFunctionDis(LPCSTR lpModule, LPCSTR lpFuncName, LPVOID lpNewFunction)
{
    // Getting the address of AP
    DWORD OriginalFunction = (DWORD)GetProcAddress(GetModuleHandle(lpModule), lpFuncName);
    HOOK_DATA *hinfo = GetHookInfoFromFunction(OriginalFunction);

    if (hinfo)
    {
        OutputDebugString("Already Hooked!");
        return FALSE;
    }
    return TRUE;
}
```

Cont'd in next slide
Creating your own cheat

DWORD BridgeAddr = CreateBridge(OriginalFunction, 6);

HookData[NumberOfHooks].Function = OriginalFunction;
HookData[NumberOfHooks].Hook = (DWORD) lpNewFunction;
HookData[NumberOfHooks].Bridge = BridgeAddr;

// Replaces the start of API with PUSH xxxx, RET
BYTE JUMP[6] = { 0x68,
    0x00, 0x00, 0x00, 0x00,
    0xc3
};

// Address of our new API (MyCreateProcessAW)
DWORD dwCalc = (DWORD) lpNewFunction;

// Building PUSH MyAPI, RET
memcpy(&JUMP[1], &dwCalc, 4);

// Writing PUSH MyAPI
if (WriteProcessMemory(GetCurrentProcess(), (LPVOID) OriginalFunction, JUMP, 6, 0))
{
    NumberOfHooks++;
    return TRUE;
}
else
{
    MessageBox(NULL, "Unable to hook", "Error...", MB_ICONSTOP);
    return FALSE;
}
Creating your own cheat

```c
void WINAPI MyglVertex3fv(const GLfloat *v)
{
    if (isSmoke==false)
    {
        typedef void (WINAPI *LPFNglVertex3fv)(const GLfloat *
        MyglVertex3fv);;
        LPFNglVertex3fv pglVertex3fv = (LPFNglVertex3fv) GetOriginalFunction((ULONG_PTR)
        pglVertex3fv(v);
    }
    else
    
        OutputDebugString("Smoke detected");
}

if (dwMode == GL_QUADS)
{
    GLfloat col;
    glGetFloatv(GL_CURRENT_COLOR, col);
    switch(isSmokeHackActivated)
    {
        case TRUE:
                isSmoke = true;
            break;
        case FALSE:
            isSmoke = false;
            break;
    }
}
Creating your own cheat

As we said it's possible to hack and patch game functionalities. Here is our patch for free nightvision forever.

```c
isNightVisionActivated = TRUE;

__asm
{
    push eax
    mov eax, 0x1957ea9
    mov byte ptr [eax], 0x75
    pop eax
}

break;

case TRUE:
isNightVisionActivated = FALSE;
__asm
{
    push eax
    mov eax, 0x1957ea9
    mov byte ptr [eax], 0x74
    pop eax
}

break;
```

Creating your own cheat

DEMO Cheating in Lan !!!
Part VI: Bypassing anti-cheat engines
Bypassing anti-cheat engines

Nice! But will these kind of hacks work in online servers? The answer is again: NO !!!
Bypassing anti-cheat engines

Why? The game will detect our modifications. Because of Anti-Cheats !!!
Bypassing anti-cheat engines

The most popular anti-cheats are:
Valve anti cheat
SXE injected
Aequitas
BlackEye
Custodia
SSClient
GameGuard
...

[Image of a creature]
Bypassing anti-cheat engines

The most popular are SXE and VAC and we will focus on them.
Bypassing anti-cheat engines

Bypassing the SXE-Injected!
Bypassing anti-cheat engines

Main feature of these programs are:

- Anti-Wallhack
- 16bpp detection
- Screenshot
- Local ban
- Speed hack detection
- Model modification detection
- Behavior detection
Bypassing anti-cheat engines

How will they do it?

By HOOKING. For example, if you remember correctly we talked about wallhack and how we implemented it. An anti cheat will hook necessary functions for wall hack and if you want to hook them again it will detect you. Also for more security SXE will use Ringo SSDT hooks for not allowing you to unhook those functions.
Bypassing anti-cheat engines

We can use kernel detective to detect hooks.
Bypassing anti-cheat engines

As you can see ddsxei.sys is responsible for Ring0 hooks and it will hooks some functions like NtprotectVirtualMemory and NtReadVirtualMemory by hooking these functions it will not be possible to unhook Ring3 hooks due to we need WriteProcessMemory and ReadProcessMemory. So what we should do?! There is a bypass. SXE won’t load its driver on X64 systems! So on X64 systems you need only understand Ring3 hooks and unhook them.
Bypassing anti-cheat engines

But it’s only for x64 systems what about 32-bit systems? Well it’s still possible to unhook functions? You need to write a windows Driver to unhook them. We used modified version of Antimida driver by Daniel Pistelli.
Bypassing anti-cheat engines

We changed driver code, removes some sections and add more functions like following one:

```c
case CODE_VIRTUAL_PROTECT:
{
    Input_ZwProtectVirtualMemory Input;
    RtlCopyMemory(&Input, pInput, sizeof (Input_ZwProtectVirtualMemory));

    try
    {
        RtlCopyMemory(&Input, pInput, sizeof (Input_ZwProtectVirtualMemory));
    }
    except (EXCEPTION_EXECUTE_HANDLER)
    {
        DbgPrint("Exception occurred: 0x%08X\n");
        return STATUS_UNSUCCESSFUL;
    }

    DbgPrint("ZwProtectVirtualMemory is called");

    return pZwProtectVirtualMemory(Input.ProcessHandle, Input.BaseAddress, 
    Input.NumberOfBytesToProtect, Input.OldAccessProtection, 
    Input.NewAccessProtection);
}
```
Bypassing anti-cheat engines

Even simple DLL injection won't work here. It means we can't use CreateRemoteThread method because of hooking WriteProcessMemory by sXe Injected; but there is at least two ways to bypass it.

1. Using system-wide hooks
2. Restoring WriteProcessMemory hook to make CreateRemoteThread method available
Bypassing anti-cheat engines

Here is example of using system-wide hooks

```c
BOOL InjectDll(char *dllName, DWORD dwTid)
{
    HMODULE hDll = LoadLibraryA(dllName);
    unsigned long pCBTProc = (DWORD) GetProcAddress(hDll, "CBTProc");
    unsigned long pGetMsgProc = (DWORD) GetProcAddress(hDll, "GetMsgProc");

    SetWindowsHookEx(WH_CBT, (HOOKPROC)pCBTProc, hDll, dwTid);
    SetWindowsHookEx(WH_GETMESSAGE, (HOOKPROC)pGetMsgProc, hDll, dwTid);
    SetWindowsHookEx(WH_KEYBOARD_LL, (HOOKPROC)pLowLevelKeyboardProc, hDll, dwTid);

    return TRUE;
}
```
Bypassing anti-cheat engines

For restoring CreateRemoteThread method you need unhook these functions in Ringo.

NtQuerySystemInformation
NtopenProcess
NtProtectVirtualMemory
NtWriteVirtualMemory
NtCreateThread

Sxe-Injected hooks NtProtectVirtualMemory to prevent changes in memory permissions of following file names: hl.exe, cstrike.exe, czero.exe, day of defeat.exe, and rev-hl.exe
Bypassing anti-cheat engines

Sxe-Injected calls NtDeviceIoControlFile to create a unique HID that will be used for local ban.

Sxe-injected.exe itself and sxe.dll are protected by latest version of Themida (wl) SXE.dll is main protector / hooker which detects and kicks / bans you out of game.
Bypassing anti-cheat engines

OEP of SXE.dll (semi-unpacking for doing patches)

```assembly
; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason,
public DllEntryPoint
DllEntryPoint proc near ; DATA XREF: sub_103470B2:1
hinstDLL = dword ptr 8
fdwReason = dword ptr 0Ch
lpvReserved = dword ptr 10h
push ebp
mov ebp, esp
push ebx
mov ebx, [ebp+hinstDLL]
push esi
mov esi, [ebp+fdwReason]
push edi
mov edi, [ebp+lpvReserved]
test esi, esi
jnz short loc_10066915
cmp ds:dword_100FD74C, 0
jmp short loc_1006693B
loc_10066915: ; CODE XREF: DllEntryPoint+
cmp esi, 1
jz short loc_1006691F
cmp esi, 2
jnz short loc_10066941
loc_1006691F: ; CODE XREF: DllEntryPoint+
mov eax, ds:dword_100FD754
```
Bypassing anti-cheat engines

Now let’s talk about SXE-Injected user mode hooks for breaking all your debugging programs. It hooks:

```
DbgBreakpoint
DbgUiRemoteBreakin
```

So before unhooking them you can’t even attach a debugger to HL process that been protected by SXE.dll.
Bypassing anti-cheat engines

SXE-Injected also hooks VirtualProtect in user mode to prevent memory permission changes. This technique easily disables most of public cheats which need to patch some memory addresses. It's also the main part of protection in x64 systems. Unhooking this API is necessary for patching the code of sxe.dll.
Bypassing anti-cheat engines

SXE—Injected user mode hooks for breaking all your removal smoke / flash / is all about it hooks to OpenGL functions.

- hl.exe!OPENGL32.dll->glBegin
- hl.exe!OPENGL32.dll->glEnd
- hl.exe!OPENGL32.dll->glVertex2f
- hl.exe!OPENGL32.dll->glVertex2fv
- hl.exe!OPENGL32.dll->glVertex3f
- hl.exe!OPENGL32.dll->glVertex3fv
- hl.exe!OPENGL32.dll->glDisable
- hl.exe!OPENGL32.dll->glEnable
- hl.exe!OPENGL32.dll->glPopMatrix
- hl.exe!OPENGL32.dll->glPushMatrix
- hl.exe!OPENGL32.dll->glPolygonOffset
- hl.exe!OPENGL32.dll->glClear
- hl.exe!OPENGL32.dll->glCullFace
- hl.exe!OPENGL32.dll->glFrontFace
- hl.exe!OPENGL32.dll->glPolygonMode
- hl.exe!OPENGL32.dll->glShadeModel
- hl.exe!OPENGL32.dll+0x2FAE
- hl.exe!OPENGL32.dll->glDepthMask
- hl.exe!OPENGL32.dll->glDepthFunc
- hl.exe!OPENGL32.dll->glDepthRange
Bypassing anti-cheat engines

You can find these functions by available programs like hookshark.
Bypassing anti-cheat engines

After you found them it’s time to inject a DLL to game and unhook which SXE hooks there is again at least two ways to doing it:

1. Patching SXE detours
2. Unhook API completely
Bypassing anti-cheat engines

The detour of hooks set by `sxe.dll` checks for blacklisted arguments of functions. Here is the `VirtualProtect` detour:

```
- SXE proxy to check...
  ... arguments of `VirtualProtect`
- bypassing `VirtualProtect`
  \-> original bytes of `VirtualProtect`
- return to `VirtualProtect`
```
Bypassing anti-cheat engines

As the detour code is too simple, its bypass is too simple as well. By nopping all instructions before original instructions of hooked function, it can be bypassed easily. Just we need to read 5 bytes at the start of function, calculates the address of JMP to find the address of detour, and patch first 0x22 bytes to NOP.
Bypassing anti-cheat engines

Here is the code snippet for patching SXE detour:

```c
BOOL PatchSxeDetour(HANDLE hProc, char* DllName, char* APIName, DWORD HlAddr)
{
    DWORD AddrHook = 0;

    if (HlAddr == NULL)
    {
        AddrHook = (DWORD)GetProcAddress(GetModuleHandle(DllName), APIName);
    }
    else if (DllName == NULL && APIName == NULL)
    {
        AddrHook = HlAddr;
    }

    DWORD JMPtoSxeAddr = 0, SxeHookCode = 0, nBytesRead = 0;
    BYTE Nop = 0x90;
    int i = 0;
    char DebugMessage[100] = {" "};

    if (!AddrHook)
    {
        wsprintf(DebugMessage,"Fail to get address of \%s", APIName);
        OutputDebugString(DebugMessage);
        return FALSE;
    }
}
**Bypassing anti-cheat engines**

```c
ReadProcessMemory(hProc, (LPVOID)(AddrHook + 1), &JMPtoSxeAddr, sizeof(DWORD), &nBytesRead);

SxeHookCode = 5 + AddrHook + JMPtoSxeAddr;

SxeUnhooked = SxeHookCode;

for (i=0; i<0x22; i++)
{
    WriteProcessMemory(hProc, (LPVOID)(SxeHookCode + i), &Nop, 1, &nBytesRead);
}

test TRUE;
```

These function will patch SXE.dll Detour you can use it like:

```c
PatchSxeDetour(hHL, "kernel32.dll", "VirtualProtect", NULL);
PatchSxeDetour(hHL, "opengl32.dll", "glBegin", NULL);
```
Bypassing anti-cheat engines

It's also possible (and better) to completely unhook the API. Here is the code:

```c
BOOL UnHookAPI(HANDLE hProc, char* DllName, char* APIName, int RestoreLength)
{
    DWORD DllImgBase = (DWORD)GetModuleHandle(DllName);
    DWORD AddrAPI = (DWORD)GetProcAddress((HMODULE)DllImgBase, APIName);
    DWORD nBytesRead = 0;
    DWORD OldProtection = 0;

    char DebugMessage[100] = "";
    wsprintf(DebugMessage,"Unhooking %s", APIName);
    OutputDebugString(DebugMessage);

    if (!AddrAPI)
    {
        return FALSE;
    }

    BYTE OriginalBytes[10] = "";

    GetFunctionOriginalBytes(DllName, APIName, OriginalBytes, RestoreLength);

    WriteProcessMemory(hProc, (LPVOID)AddrAPI, OriginalBytes, RestoreLength, &nBytesRead);

    wsprintf(DebugMessage,"%s was unhooked.", APIName);
    OutputDebugString(DebugMessage);

    return TRUE;
}
```
Bypassing anti-cheat engines

GetFunctionOriginalBytes function

BOOL GetFunctionOriginalBytes(char* DllName, char* FunctionName, BYTE* OriginalBytes, int Length) {
    char Buffer[MAX_PATH];

    GetSystemDirectory(Buffer, MAX_PATH);

    strcat(Buffer, "\"\");  
    strcat(Buffer, DllName);

    HANDLE hFile = CreateFile(Buffer, GENERIC_READ, FILE_SHARE_READ, NULL, OPEN_EXISTING, 0, NULL);

    if (hFile == INVALID_HANDLE_VALUE)
        return FALSE;

    DWORD FileSize = GetFileSize(hFile, NULL);

    BYTE *ptrDll = (BYTE *) VirtualAlloc(NULL, FileSize, MEM_COMMIT, PAGE_READWRITE);

    if (ptrDll == NULL)
    {
        CloseHandle(hFile);
        return FALSE;
    }
}
Bypassing anti-cheat engines

CloseHandle(hFile);

IMAGE_DOS_HEADER *ImgDosHdr = (IMAGE_DOS_HEADER *) ptrDll;

IMAGE_NT_HEADERS *ImgNtHdrs = (IMAGE_NT_HEADERS *)
  &ptrDll[ImgDosHdr->e_lfanew];

ULONG_PTR EP_Rva = 0;

if (!GetExport(ptrDll, &EP_Rva, NULL, FunctionName))
{
    VirtualFree(ptrDll, 0, MEM_RELEASE);
    return FALSE;
}

BYTE *ptr = (BYTE *) (EP_Rva + (ULONG_PTR) ptrDll);

memcpy(OriginalBytes, ptr, Length);

VirtualFree(ptrDll, 0, MEM_RELEASE);

return TRUE;
Bypassing anti-cheat engines

Also, there is a note if you join a server with our modified model (AWP) SXE will detect it and will kick you from the game. SXE will detect it by getting MD5 Checksum of each model.

```
mov [ebp-9c0h], eax
mov ecx, ds:word_1007b834
mov [ebp-9c6h], cx
xor edx, edx
mov [ebp-9c4h], edx
mov [ebp-9c6h], edx
mov ecx, 8
mov esi, offset a81c1eced1532c
leal edi, [ebp-9d0h]  ; "81c1eced1532cfft9d9fca9161dc8eb"
rep movsd
movsd
xor eax, eax
mov [ebp-9f0h], eax
mov [ebp-9f4h], ax
mov [ebp-9f4h], al
mov ecx, ds:dword_1007b824
mov [ebp-9e0h], ecx
mov edx, ds:dword_1007b828
mov [ebp-9d8h], edx
mov eax, ds:dword_1007b82c
mov [ebp-998h], eax
mov ecx, ds:dword_1007b830
mov [ebp-994h], ecx
mov dx, ds:dword_1007b834
mov [ebp-990h], dx
xor eax, eax
```
Bypassing anti-cheat engines

Again there is at least two ways to bypass it:

1. Alter the checksum with your model MD5
2. Patch the modification Check routine.
**Bypassing anti-cheat engines**

For altering the main object with your object, just calculate the new model MD5 and replace it in SXE.dll using Injection.

![HashCalc](image)

<table>
<thead>
<tr>
<th>Data Format</th>
<th>Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>M:\AppData\Local\Temp\Rar\DR4.4.72\v_awp.md5</td>
</tr>
</tbody>
</table>

- **HMAC**
- **MD5**: 81c1eeced1532c7fbb459bfa91e1dc9eb
- **MD4**: 
- **SHA1**: 836662666ea14932e295835337eb291480e1e5a5
- **SHA256**: 
- **SHA384**: 
- **SHA512**: 
- **RIPEMD160**: 31f7d390c92f3c3701da93e466a8e7dc05664e47
- **PANAMA**: 
- **TIGER**: 
- **MD2**: 
- **ADLER32**: 
- **CRC32**: 44a3c463

- **eDonkey/ eMule**: 

---

110101001010110011101010111001010110000110011000000000000000000
Bypassing anti-cheat engines

To completely patch the routine, find the beginning of the routine contains the MD5 of object and patch first byte to RET. You can use following code to do that:

```c
// Disable model modification check:
DWORD AddrModelCheck = 0x4010; // offset of patch
    // ImageBase is hSxeDll
BYTE PatchModelCheck [1] = {0xC3};
WriteProcessMemory( hHL, (LPVOID)((DWORD)hSxeDll + AddrModelCheck),
    PatchModelCheck, 1, &nBytesWritten);
```

The hSxeDll is Handle of SXE.dll we can get by using GetModuleHandle which is the ImageBase of sxe.dll technically.
Bypassing anti-cheat engines

OK, SXE bypassing looks simple, but wait! It detects memory modifications and hook removal. We can patch this routine too.

```
; FUNCTION CHUNK AT .data:1032ECEC SIZE 0000008A BYTES

sub_10012880 proc near
    push    ebp
    mov     ebp, esp
    sub     esp, 48h
    push    ebx
    push    esi
    push    edi
    jmp     loc_1032ECEC

sub_10012880 endp
```
Bypassing anti-cheat engines

And here is the code of patcher.

```c
// Disable sXe memory patch check
DWORD AddrSxeHooker = 0x128B0;
BYTE PatchSxeHooker[1] = {0xC3};
DWORD AddrSxeHookerMagicJump = 0x7AAB6;
WriteProcessMemory(hHL, (LPVOID)((DWORD)hSxeDll + AddrSxeHooker), PatchSxeHooker, 1, &nBytesWritten);
```
Bypassing anti-cheat engines

Well there is some other kind of checks in SXE, but with these unhooking / patching, we have mutilated SXE to not be useful as it should be, so we can use almost all available cheats on SXE to proof it. We used some old hack and did some modification to work on modern OS and also latest SXE and finally finished our job. We can now do cheat and always be BEST player in the map !!!
Bypassing anti-cheat engines

DEMO
Cheating in Protected Server by both SXE and VAC !!!
Conclusion

So cheating in online games is not that easy and needs strong reversing skills as well as programming. Also anti-cheat can make your life harder by protecting manipulation on client side. But at all it's still possible to cheat even with latest protections and online games !!!
Questions ?!

if you have any question
mail it to: shahin@abysssec.com
follow @abysssec in twitter