I'VE INJECTED A DLL - YOU WON'T BELIEVE WHAT HAPPENED NEXT!

BY @CAPTNBANANA

WHO R U MAN

- I do red teaming / pentesting
- Interested in reversing & exploit development
- And: Game hacking
- https://bananamafia.dev/

MOTIVATION

MOTIVATION: MONEY

- Win at Tournament: $$$
- Cheat: Easier $$$?
- Cheat subscription
  - 7 Days: 7€
  - 30 Days: 30€
  - 90 Days: 28€

PAID CHEATS

- "Humanized Bot"
- "We are undetected, we swear!"
- "If you attach a debugger we will ban you from the cheat service"
- "We don't log, trust us!"

CHEAT TYPES

- Wallhack
- Aimbot
- Game Specific:
  - No Flash
  - Anti Grip
  - See invisible players
  - Crosshair hack

HACK TYPES

- Internal
- External
- (Instrumented)

TOOLING

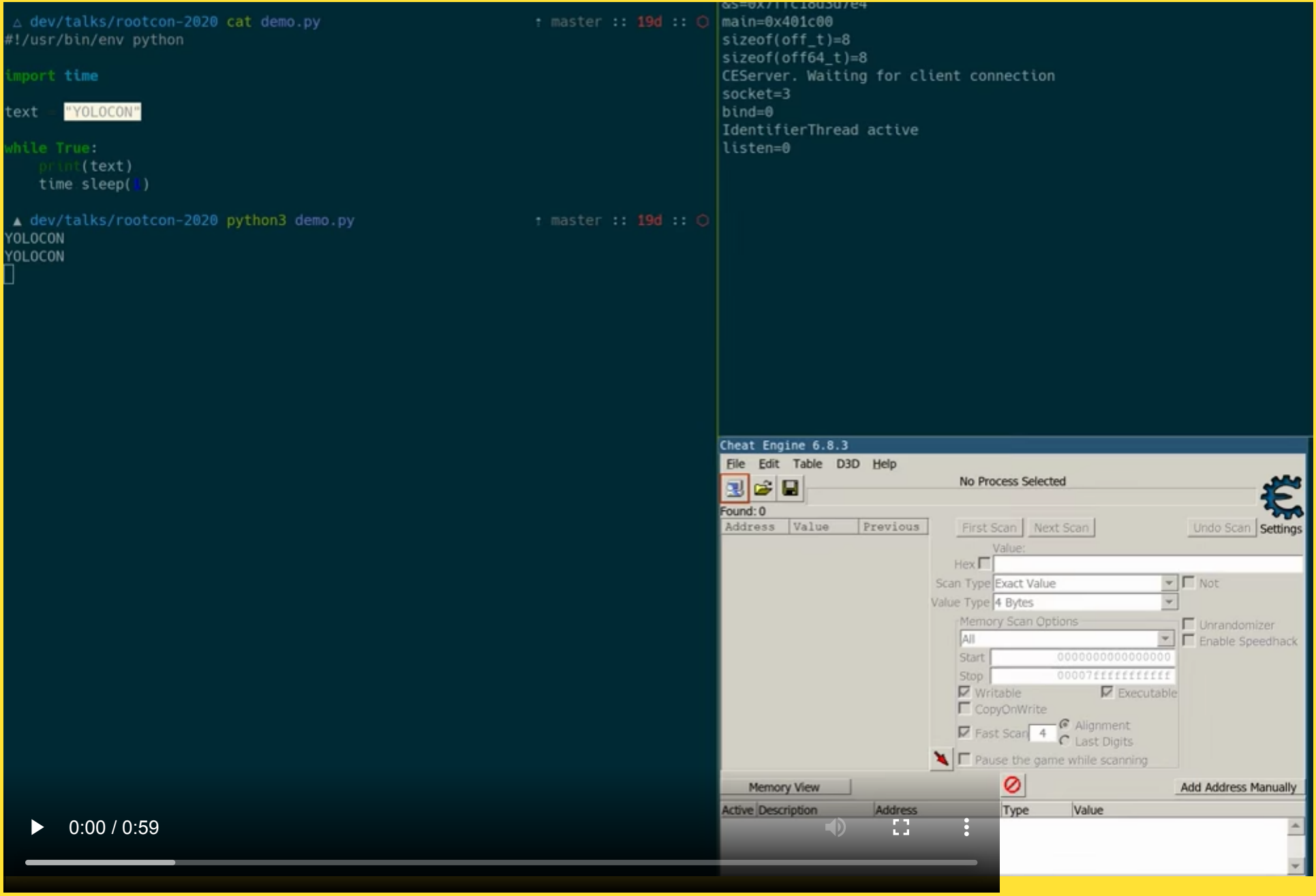
- Visual Studio: C++
- Debugger, e.g. x64dbg
- IDA/Ghidra/Radare2/Cutter/...
- **Cheat Engine**

ABOUT CHEAT ENGINE

- It's great
- Inspect and analyze process memory
- Disassembler
- Scripting engine
- Windows / Linux
  - `ceserver` + GUI (wine)

HANDY CHEAT ENGINE FEATURES

- Scan for known values
- "What writes/reads" this address
- Freeze values


△ dev/talks/rootcon-2020 cat demo.py
↑ master :: 19d ::
#!/usr/bin/env python
import time
text = "YOLOCON"
while True:
print(text)
time sleep(1)
▲ dev/talks/rootcon-2020 python3 demo.py
↑ master :: 19d ::
YOLOCON
YOLOCON
main=0x401c00
sizeof(off_t)=8
sizeof(off64_t)=8
CEServer. Waiting for client connection
socket=3
bind=0
IdentifierThread active
listen=0
Cheat Engine 6.8.3
File Edit Table D3D Help
No Process Selected
Found: 0
Address Value Previous
First Scan
Next Scan
Undo Scan
Settings
Value:
Hex
Scan Type Exact Value
Not
Value Type 4 Bytes
Memory Scan Options
Unrandomizer
Enable Speedhack
All
Start 0000000000000000
Stop 00007fffffffffff
Writable
Executable
CopyOnWrite
Fast Scan 4
Alignment
Last Digits
Pause the game while scanning
Memory View
Add Address Manually
Active Description Address
Type Value
0:00 / 0:59

INTERNAL HACK ON WINDOWS

- For Jedi Academy and Counter Strike: GO
- Plan:
  - Build DLL loader
  - Build actual DLL
  - Inject DLL
  - Profit

LOADER CODE

```
HANDLE procHandle = OpenProcess(
            PROCESS_ALL_ACCESS,
            FALSE,
            PID);

LPVOID loadFunctionAddress = (LPVOID)GetProcAddress(
            GetModuleHandle("kernel32.dll"),
            "LoadLibraryA");

LPVOID allocatedMem = LPVOID(VirtualAllocEx(
            procHandle,
            nullptr,
            MAX_PATH,
            MEM_RESERVE | MEM_COMMIT,
```

THE INJECTED DLL

```
BOOL APIENTRY DllMain (HMODULE hModule, DWORD ul_reason_for_ca
    switch (ul_reason_for_call) {
        case DLL_PROCESS_ATTACH:
            MessageBox(0, "Cool, Works!", "1337 DLL", 0);
            break;
    }
    return TRUE;
}
```

GAME ENGINES

- From now: Game and engine specific
- What to hook and what to manipulate depends!
- Jedi Academy: idTech3 (Quake3) Engine
- CS:GO: Source engine

JEDI ACADEMY

WALLHACK, AIMBOT, ANTI GRIP

WALLHACK

cl_cgame.c (Quake3)

```
int CL_CgameSystemCalls(int *args) {
    switch(args[0]) {
        [...]
        case CG_R_ADDREFENTITYTOSCENE:
            re.AddRefEntityToScene(VMA(1));
            return 0;
        [...]
    }
}
```

`CL_CgameSystemCalls()`

- Implemented in `cgamex86.dll`
- Called by main executable (`jamp.exe`)
- Needs to be hooked

CALL INTO DLL: PROGRAM FLOW

- `jamp.exe` loads `cgamex86.dll`
- `jamp.exe` calls `GetProcAddress()` for desired function
- GetProcAddress() returns address of function inside of DLL
- `jamp.exe` executes `function@Address`

PLAN (1)

- Hook `GetProcAddress()` and manipulate returned function address
- -> Execute own code instead
- -> Call original function in the end

CGAMEX86.DLL EXPORTS: DLLENTRY()

- Receives function pointer as parameter
- **Hooked to manipulate existing code (e.g. for Wallhack)**
- Events: Entity added, entity moves, game data received from server

```
Q_EXPORT void dllEntry(intptr_t (QDECL *syscallptr)( intptr_t
    Q_syscall = syscallptr;
    TranslateSyscalls();
}
```

PLAN (2)

- Hook `GetProcAddress()`
- -> hook `dllEntry()`
- Intercept calls with command `CG_R_ADDREFENTITYTOSCENE`
- Manipulate entity parameter
- Done!

HOOK SETUP: GetProcAddress()

Redirect into own `dllEntry()`

```
Mhook_SetHook(
    (PVOID*)&originalGetProcAddress,
    hookGetProcAddress
);

[...]

FARPROC WINAPI hookGetProcAddress(HMODULE hModule, LPCSTR lpPr
{
    [...]
    if (isSubstr(lpProcName, "dllEntry")) {
        return (PROC)hookDLLEntry;
    }
    return (FARPROC)originalGetProcAddress(hModule, lpProcName
```

THE WALLHACK

```
int syscall_hook(int cmd, ...) {
    [...]
    case CG_R_ADDREFENTITYTOSCENE: {
        // get the passed parameter (an entity)
        refEntity_t *ref = (refEntity_t *)arg[0];

        // HAX!!1!
        ref->renderfx |= RF_DEPTHHACK;

        break;
    }

    [...]
    // call the original
```

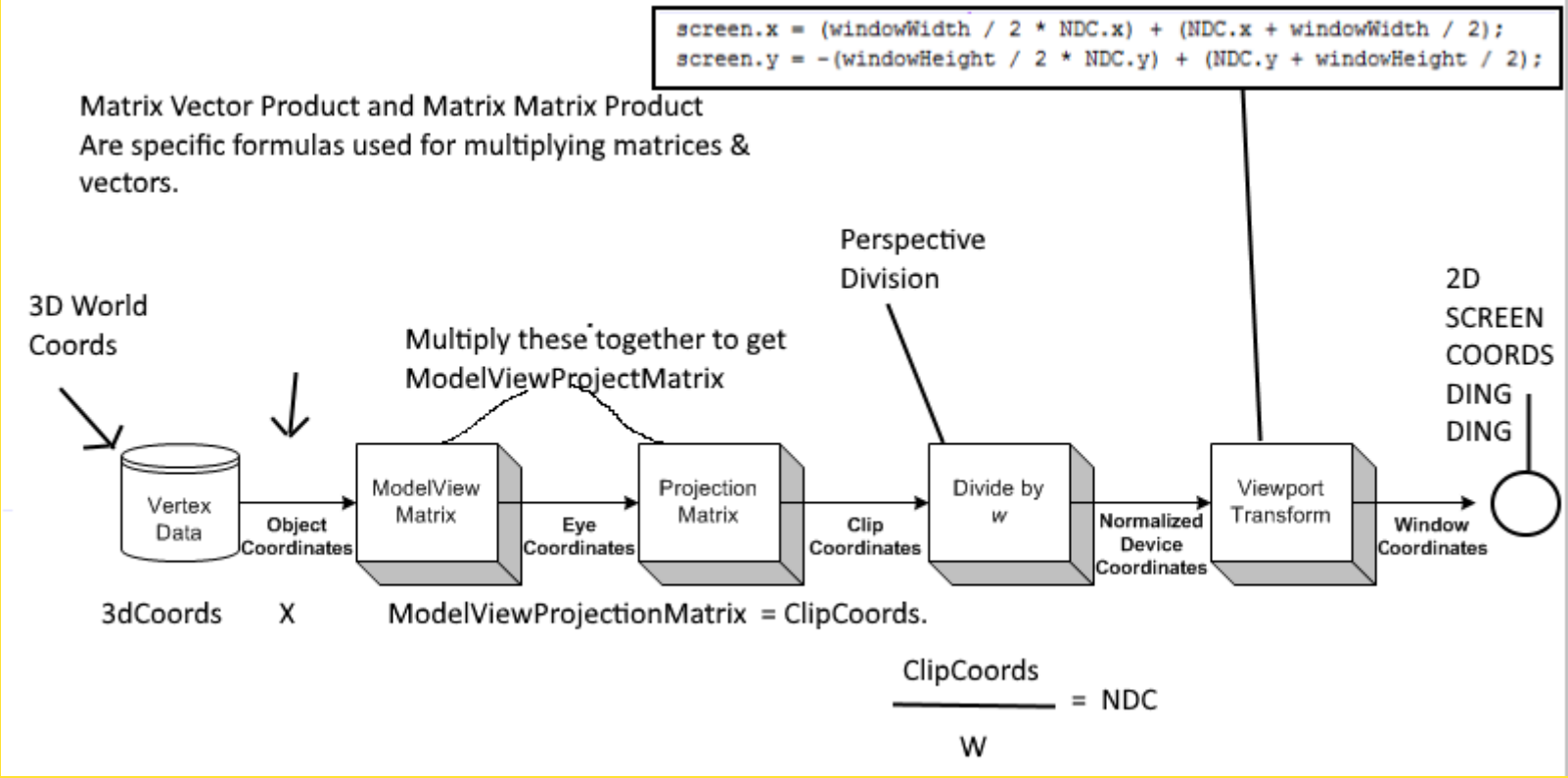
AIMBOT

- Lock view at enemy
- Designated Aim key
- Requirements:
  - Engine Structures: Done via `cgame` hooks
  - Get enemy entity via crosshair
  - Calculate correct angle: World to Screen
  - Set angle programmatically

REQUIRED STRUCTURES: ENEMY

```
int crosshairClientNum = client_game->crosshairClientNum;
auto ent = entFromClientNum(crosshairClientNum);

centity_t* entFromClientNum(int clientNum) {
    [...]
    for (int i = 0; i < MAX_GENTITIES; i++) {
        centity_t* cur = pEntities[i];
        if (!cur) { continue; }
        if (cur->playerState->clientNum == clientNum) {
            res = cur;
            break;
        }
    }
}
```

WORLD TO SCREEN



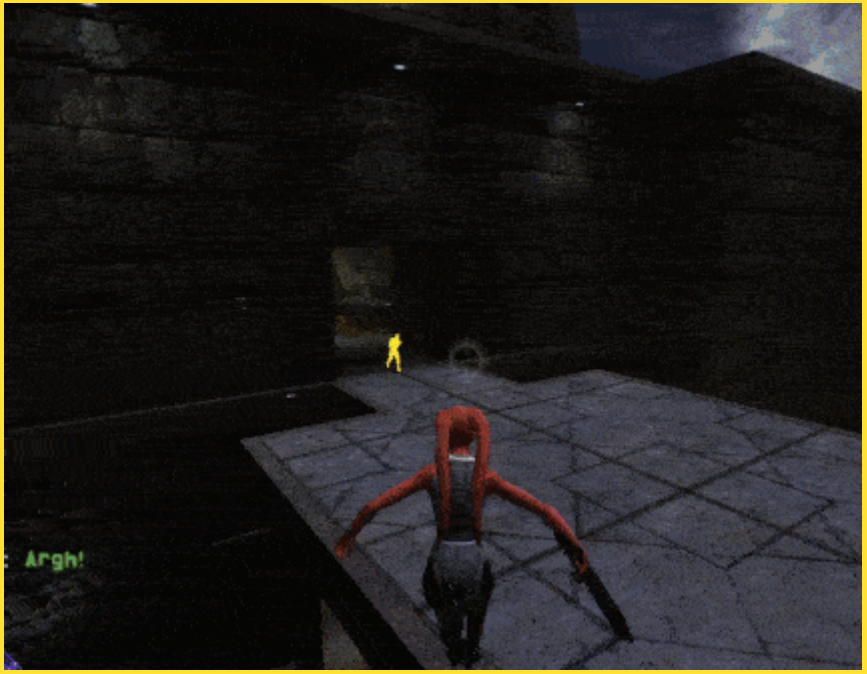
Picture and Code Source: GuidedHacking

WORLD TO SCREEN

```
bool w2s(float fovx, float fovy, float windowWidth, float wind
{
    v3_t transform;
    float xc = 0, yc = 0;
    float px = 0, py = 0;
    float z = 0;

    px = tan(fovx * M_PI / 360.0);
    py = tan(fovy * M_PI / 360.0);

    transform = this->sub(origin); //this = destination

    xc = windowWidth / 2.0;
    yc = windowHeight / 2.0;
```

SETTING THE CAMERA ANGLE

```
void moveMouse(v3_t SCREEN)
{
    [...]

    INPUT Input = {0};
    Input.type = INPUT_MOUSE;
    Input.mi.dwFlags = MOUSEEVENTF_MOVE;
    Input.mi.dx = SCREEN.x - client_game->refdef.width / 2;
    Input.mi.dy = SCREEN.y - client_game->refdef.height / 2;

    SendInput(1, &Input, sizeof(INPUT));
}
```


Argh!

NO GRIP

THE PLAN

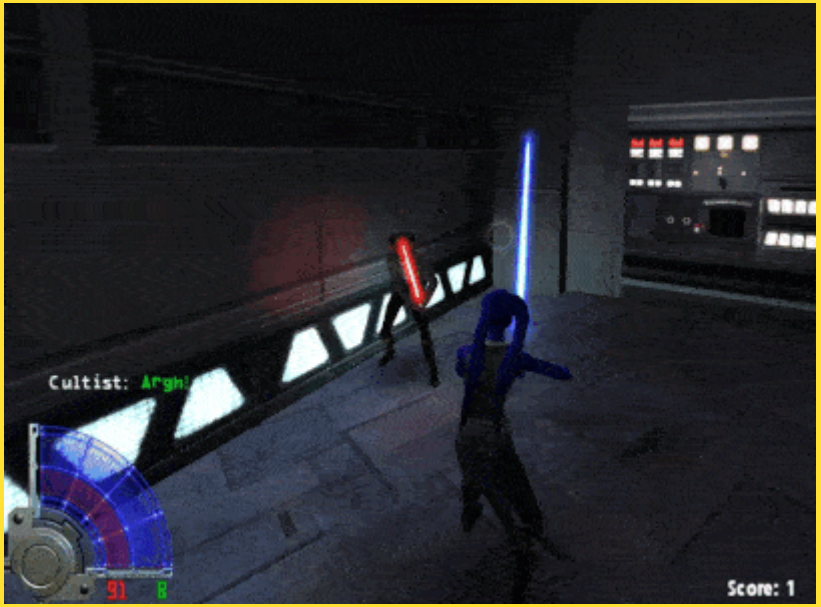
- Get enemy entity
- Automatically aim at enemy (Use Aimbot)
- Execute force throw -> Cancel grip

NO GRIP

```
// If we are currently being gripped
if (ps && (ps->fd.forceGripBeingGripped || ps->fd.forceGripCri
    auto ent = entFromClientNum(ps->persistant[PERS_ATTACKER])
    // focus the current target -> use aimbot
    focusEnt(ent);
    syscall_hook(CG_SENDCONSOLECOMMAND, "force_throw;");
}
```


Score: 0

ANTI TRICK



HOW IT WORKS

```
// modify local player state
ps.fd.forcePowersActive |= (1 << FP_SEE);
```

- Works even though force is disabled on server

ALLOWING BLOCKED SETTINGS

```
syscall_hook(CG_CVAR_SET, "sv_cheats", "1");
```

- Makes the game think that cheat settings are OK
- Enable e.g. `r_fullbright`
- -> No shadows on the map

CS:GO

CROSSHAIR HACK

DIRECT3D EndScene()

- Queues scene for output (~ a frame)
- Executed after scene creation is completed
- Ideal for hooking

HOOKING EndScene()

- Create DLL with `endSceneHook()` function
- Important: Accept same parameters as original
- Inject DLL that hooks the function
- Use same loader as before

HOOK FUNCTION: PROTOTYPE

```
void APIENTRY endSceneHook(LPDIRECT3DDEVICE9 p_pDevice);
```

- It's parameterless
- But: Implicit `this` parameter

REQUIREMENTS

- Get address of EndScene( )
- Function that performs the actual hooking

DIRECT3D PSEUDO DEVICE TECHNIQUE

- Game restart, memory mapping, library versions, etc. :
  - No idea where `endScene()` actually is
  - Need reliable way to find it
  - Possible but not universal
- Here comes the Pseudo Device Technique
- Credits: GuidedHacking

DIRECT3D PSEUDO DEVICE TECHNIQUE

- Create own D3D device
- It contains a vTable with an entry to `EndScene( )`
- Copy this function address
- Throw the device away
- Hook the function at the retrieved address

DIRECT3D SOURCE CODE: d3d9.h

```
typedef struct IDirect3DDevice9ExVtbl
{
    [... 41 Elements ...]
    HRESULT (WINAPI *EndScene)(IDirect3DDevice9Ex *This);
    [...]
}
```

LOL HOW

```
// see https://guidedhacking.com/threads/get-direct3d9-and-dir
bool d3dHelper::getD3D9Device() {
    IDirect3D9* d3dSys = Direct3DCreate9(D3D_SDK_VERSION);
    IDirect3DDevice9* dummyDev = NULL;

    // Options to create dummy device
    D3DPRESENT_PARAMETERS d3dpp = {};
    d3dpp.Windowed = false;
    d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
    d3dpp.hDeviceWindow = hwnd;

    HRESULT dummyDeviceCreated = d3dSys->CreateDevice(D3DADAPT

    // Copy memory to our own data structure
```

THAT WAS EASY

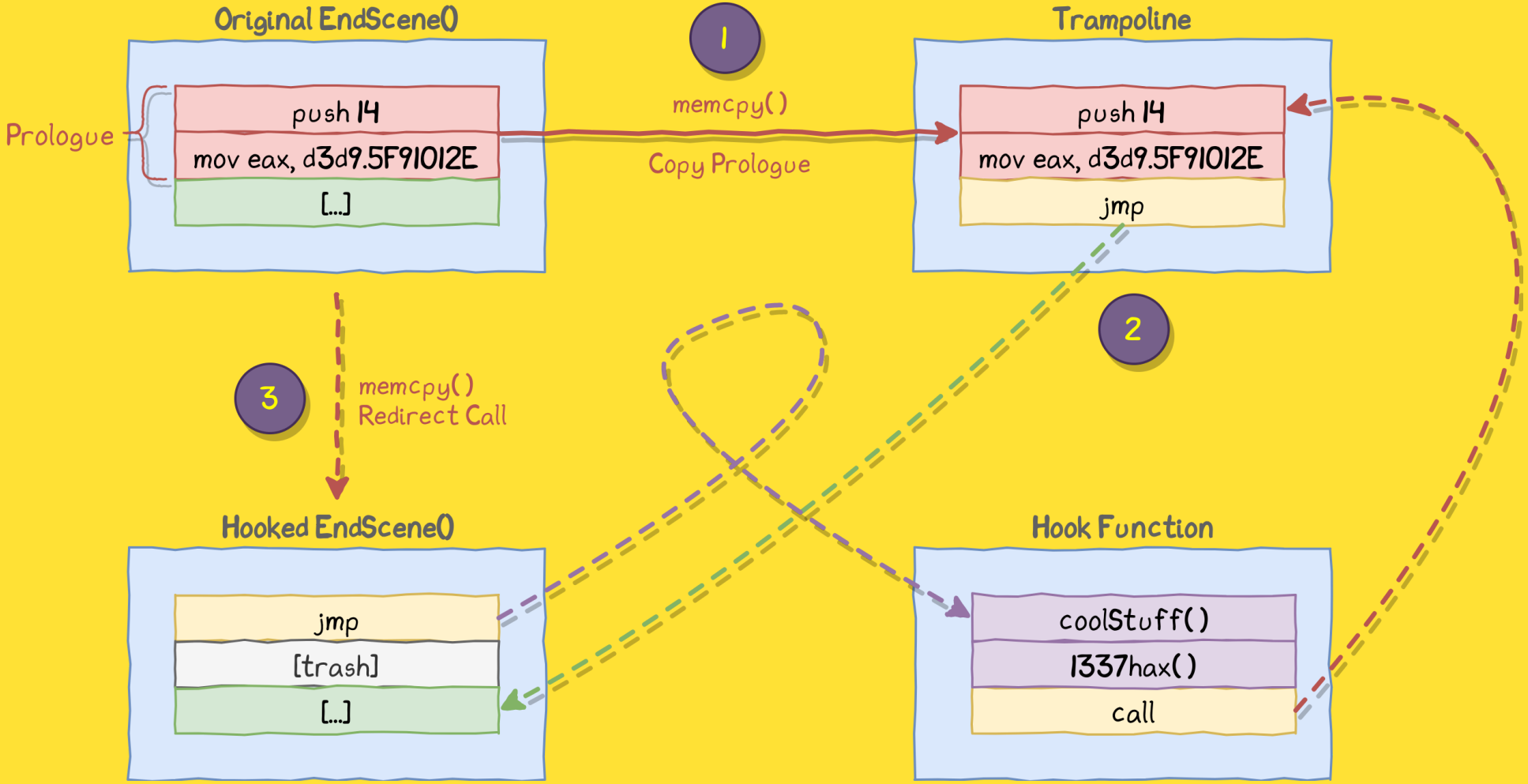
```
char* ogEndSceneAddress = d3dHelper.d3d9DeviceTable[42];
```

WHAT'S MISSING?

- We now know what to hook
- We know how `endSceneHook()` has to be implemented
- How to actually hook?

TRAMPOLINE HOOKS

TRAMPOLINE HOOKS: OVERVIEW



TRAMPOLINE HOOKS: CODE

```
// adapted from https://guidedhacking.com/threads/simple-x86-c
const char* REL_JMP = "\xE9";
const unsigned int SIZE_OF_REL_JMP = 5;

void* WINAPI hookFn(char* hookedFn, char* hookFn, int copyByte

    // Backup original
    ReadProcessMemory(GetCurrentProcess(), hookedFn, backupByt

    // Trampoline setup
    char* trampoline = (char*)VirtualAlloc(0, copyBytesSize +
    PAGE_EXECUTE_READWRITE);
    memcpy(trampoline, hookedFn, copyBytesSize);
```

TRAMPOLINE HOOKS: ASM VIEW

```
; Original EndScene()
0x5F8F46A0 6A 14 push 14 ; Prologue
0x5F8F46A2 B8 2E01915F mov eax,d3d9.5F91012E ; Prologue
0x5F8F46A7 E8 3E8B0100 call d3d9.__EH_prolog3_catch ; Actual c
;[more code]

; Hooked EndScene():
0x5F8F46A0 E9 XXXXXXXX jmp dll.EndSceneHook ; Jump to Own Code
0x5F8F46A5 91 ??? ; Trash, never executed
0x5F8F46A6 5F ??? ; Trash, never executed
0x5F8F46A7 E8 3E8B0100 call d3d9.__EH_prolog3_catch ; Actual c
;[more code]

; The Trampoline()
```

EndSceneHook()
IMPLEMENTATION

```
// the returned trampoline
extern endSceneFunc trampEndScene;
void APIENTRY d3dHelper::endSceneHook(LPDIRECT3DDEVICE9 p_pDev
    [...]
    // Do own stuff
    drawRectangle(25, 25, 100, 100, D3DCOLOR_ARGB(255, 255, 25

    // Call original function using the trampoline
    trampEndScene(d3dDevice);
}
```

COMPILING

- Add DirectX SDK to linker libraries
- Compile DLL for **x86**

DEBUGGING WITH SYMBOLS

Base	Module	Party	P
79640000	acgenral.dll	System	C
77020000	advapi32.dll	System	C
70CE0000	amsi.dll	System	C
70B20000	antimalware_provider.dll	User	C
68BB0000	apphelp.dll	System	C
62DF0000	audioses.dll	System	C
62740000	avifil32.dll	System	C
62FC0000	avrt.dll	System	C
770F0000	bcrypt.dll	System	C
77850000	bcryptprimitives.dll	System	C
78770000	cairo.dll	User	C
777A0000	cfgmgr32.dll	System	C
76AC0000	clbcatq.dll	System	C
34E00000	client_panorama.dll	User	C
75DF0000	combase.dll	System	C
6FDA0000	comctl32.dll	System	C
66EE0000	coremessaging.dll	System	C
66C80000	coreuicomponents.dll	System	C
67000000	crashhandler.dll	User	C
76CC0000	crypt32.dll	System	C
75560000	cryptbase.dll	System	C
62A20000	cryptnet.dll	System	C
77290000	cryptsp.dll	System	C
00F80000	csgo.exe	User	C
5F890000	d3d9.dll	System	C

Address	Type	Ordina	Symbol
5F8CB544	Symbol		??_C@_0BB@KBBLCIKD@EndScene?5f
5F8CB4B8	Symbol		??_C@_0BD@OIBIHEEB@BeginScene?
5F8CF5B8	Symbol		??_C@_0CF@DJAIAKBD@Driver?5f
5F8CB4CC	Symbol		??_C@_0CJ@BHAMGENP@EndScene?0?
5F8CC2C0	Symbol		??_C@_0CK@OAECIGMH@Need?5to?
5F8CB434	Symbol		??_C@_0DB@OPBNNBBE@BeginScene?
5F8F45A0	Symbol		?BeginScene@CD3DBase@@UAGJXZ
5F8F46A0	Symbol		?EndScene@CD3DBase@@UAGJXZ
5F99E0A3	Symbol		?OnBeginScene@?$ShaderDebugge
5F99E10F	Symbol		?OnEndScene@?$ShaderDebugger@
5F9A4220	Symbol		?RefRastSceneCapture@@YGKPAU_
5F9D8CD0	Symbol		?SceneCapture@CD3DDDIDX6@@UAE
5F8F93C0	Symbol		?SceneCapture@CD3DDDIDX7@@UAE
5F99E330	Symbol		?SceneCapture@RefDev@@QAEXK@Z
5F982EA5	Symbol		?extractNotIScene@CRMHeap@@QA
5F8F45A0	Symbol		CD3DBase::BeginScene
5F8F46A0	Symbol		CD3DBase::EndScene


تبغ
TABAC


تبغ

CS:GO

EXTERNAL CHEAT

HOW IT WORKS

- Read and analyze game memory
- Manipulate memory accordingly
- Works without code injection

WHAT CAN BE IMPLEMENTED?

- Aimbot
- No Flash
- Auto-Bunnyhop
- Probably more

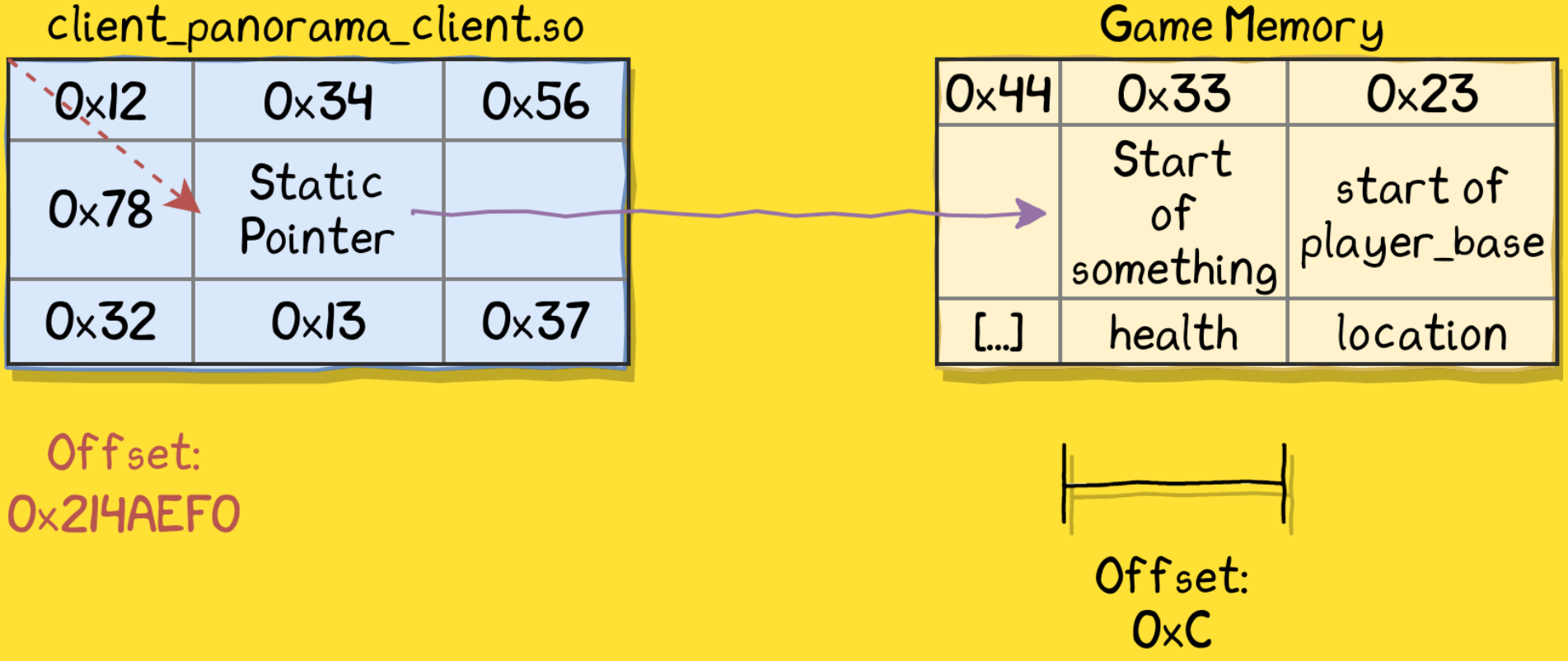
MEMORY ANALYSIS

- Attach CheatEngine
- Scan for known values
- Reverse structures

MEMORY ANALYSIS

- Use leaked game source code
- Refer to existing cheat source code

STATIC POINTERS



HAZE DUMPER

hazedumper

🚀 Up to date offset and dumper-config for Counter-Strike: Global Offensive. For more informations visit the release page on UnKnoWnCheaTs.

Local Player

Since in the past the signature for the LocalPlayer was broken a few times and/or pointed to something wrong, I want to offer you an alternative. All required offsets are already in the repo, why 99% of people don't use them is questionable - or they are just too incompetent and complain that nothing works, but they can't find a single signature themselves. However, pretty much every hack uses the entity list (`dwEntityList`) and also ClientState (`dwClientState`). All you need is a third offset which is located in ClientState, called `dwClientState_GetLocalPlayer` .

```
const auto client_state = read_memory<std::uint32_t>( engine_image->base + hazedumper::signatures::dw
if( client_state ) {
    const auto local_player = get_client_entity(
        read_memory<std::int32_t>( client_state + hazedumper::signatures::dwClientState_GetLocalPlaye
    );

    if( local_player ) {
        printf(
            "[+] Found local player: 0x%X, health: %d\n",
            local_player,
```

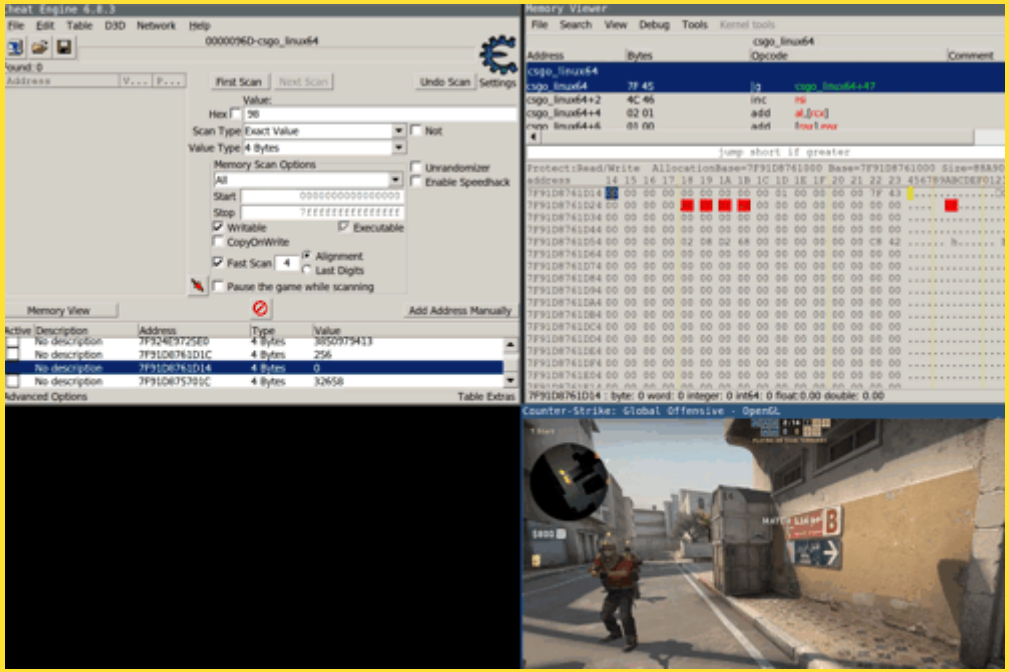
```
            read_memory<std::int32_t>( local_player + hazedumper::netvars::m_iHealth )
        );
    }
}
```

UPDATES

- Game update -> Cheat update
- Different offsets, addresses, struct members

IMPLEMENTING CHEATS: NO FLASH

- Write `0` to member of `LocalPlayer`
- It's that easy


Cheat Engine 6.8.3
File Edit Table D3D Network Help
0000096D-csgo_linux64
First Scan
Next Scan
Undo Scan
Settings
Value:
Hex
50
Scan Type
Exact Value
Not
Value Type
4 Bytes
Memory Scan Options
All
Start
Stop
Writable
Executable
CopyOnWrite
Fast Scan
4
Alignment
Last Digits
Pause the game while scanning
Unrandomizer
Enable Speedhack
Memory View
Add Address Manually
Active
Description
Address
Type
Value
No description
7F924E972SE0
4 Bytes
3050979413
No description
7F91D8761D1C
4 Bytes
256
No description
7F91D8761D14
4 Bytes
0
No description
7F91D875701C
4 Bytes
32658
Advanced Options
Table Extras
Memory Viewer
File Search View Debug Tools Kernel tools
csgo_linux64
Address
Bytes
Opcode
Comment
csgo_linux64
csgo_linux64
7F 45
jg
csgo_linux64+47
csgo_linux64+2
4C 46
inc
r9
csgo_linux64+4
02 01
add
al,[rcx]
csgo_linux64+6
00 00
add
[rax],al
jump short if greater
Protect:Read/Write AllocationBase=7F91D8761000 Base=7F91D8761000 Size=88A90
address 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 456789ABCDEF0123
7F91D8761D14 00 00 00 00 00 00 00 00 31 00 00 00 00 00 7F 43
7F91D8761D24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D34 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D54 00 00 00 00 02 08 D2 68 00 00 00 00 00 00 C8 42
7F91D8761D64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D84 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D94 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DA4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DB4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DC4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DD4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DE4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761DF4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761E04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761E14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7F91D8761D14 : byte: 0 word: 0 integer: 0 int64: 0 float:0.00 double: 0.00
Counter-Strike: Global Offensive - OpenGL


CT Start
1:42
$300
Desert Eagle
100
100
7 / 35

I NEED MORE INFORMATION!

Check out my talk from BSides Munich 2020!

GAME HACKING WITH FRIDA

HOW IT WORKS

- Inject JS code into black box process
- Hook, trace

github.com/taviso/sharapi

Simpsons: Hit & Run API

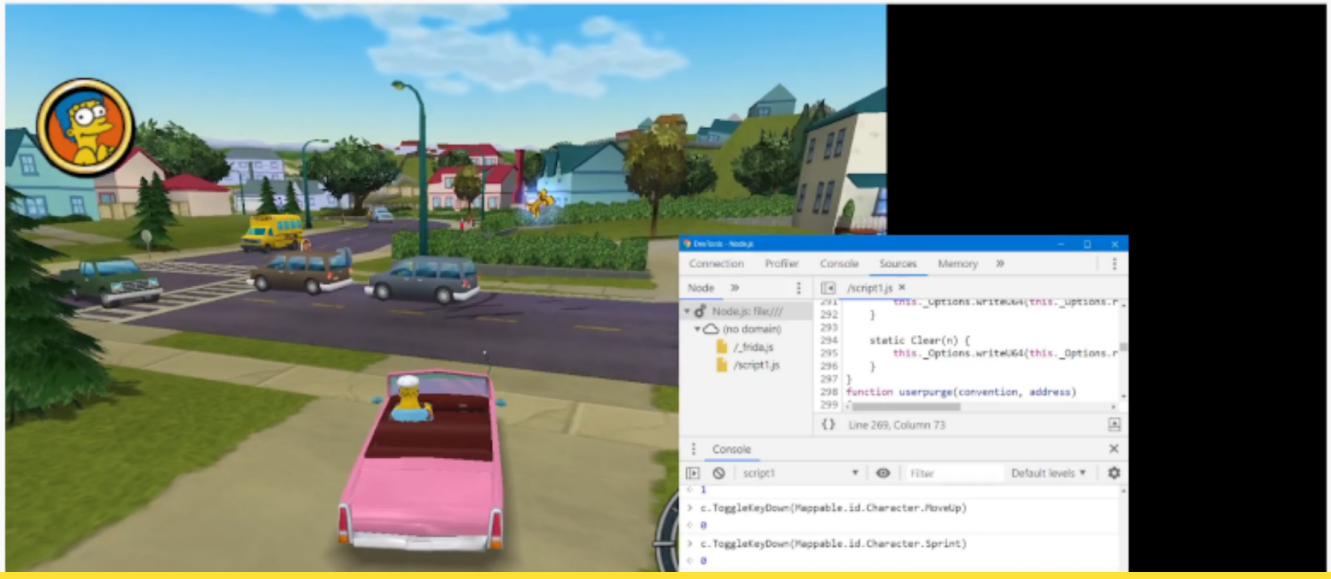
> ***This code is in a pre-pre-pre-alpha experimental state.***

This is a library to automate the abandonware game Simpsons: Hit & Run with JavaScript. It uses frida to access internal state, and exposes JavaScript classes that can be used to query and control the game.

The intention is to allow easy scriptable access to state, in a similar way to how pysc2 enabled deepmind to learn how to play Starcraft II. Eventually I'd like to be able to automate finding glitches, crashes, strategies, routes and so on for speedrunning.

> *This is a solo hobby project, I'm a long way off from that point.*

Want to see it in action? Here is an early demo video using the debugging console.



```
function kick(count = 1) {
    while (count-- > 0) {
        input.SimulateKeyPress(Mappable.id.Character.Attack);
        Thread.sleep(0.5);
    }
}

[...]

// Move in current direction.
step();

// Did we find the object?
if (myPos.distanceTo(object) <= 1) {
```

CHEAT DETECTION

VAC DETECTION

- "VAC is a Joke"
- Uses signatures (among other things)
- Detects specific kinds of hooks
- Solution: Hook mid function
- Don't use public code
- Manual Mapping, Polymorphism, Unhooking
- **But: Kernel mode anti cheats exist**

MANUAL MAPPING?

- A fancier DLL injector
- Bypass:
  - `LoadLibrary() Hooks`
  - `CreateToolhelp32Snapshot()`
  - `EnumModules()`
  - `NtQueryVirtualMemory()`
- DLL doesn't show up in loaded modules
- Also not in Process Environment Block (PEB)

MANUAL MAPPING

- Re-Implement `LoadLibrary()`
- -> Kernel doesn't know a DLL is loaded :)

MANUAL MAPPING: HOW IT WORKS

- Load raw DLL into own memory
- Map DLL sections in game process
- Inject and run loader shellcode in game process
  - Relocate
  - Fix imports
  - TLS callbacks
  - Call `DllMain()`
- Cleanup: Free memory

MANUAL MAPPING: RELOCATIONS

- Required if allocated space != `ImageBase` of DLL
- DLL includes relocation information
- For global variables, addresses for CALL instructions
- Relocate: Adjust addresses based on new base address

MANUAL MAPPING: IMPORTS

- Injected DLL may require additional functions of other DLLs
- Fixing imports:
  - Loading these DLLs
  - Setting pointers to imported functions in DLL header

MANUAL MAPPING: TLS CALLBACKS

- TLS = Thread Local Storage
- Executed before DLL entry point
- TLS Table in DLL header
- Executing TLS Callbacks -> Initialize per-thread data

@CaptnBanana

REFERENCES

- My Blog Posts
- Source Code: CS:GO EndScene Hook
- d3d9.h Source Code
- Simpsons Hit and Run Frida API
- GuidedHacking: World to Screen Functions
- GuidedHacking: Manual Mapping
- Rohitab: Manual Mapping
- TLS Section
- Slides for my Talk @ BSides Munich