

Video game hacking:

Cheat Engine Workshop



Lucas PARSY



*"It's
pronounced
Aine jean"*



Resources

- Cheat Engine: cheatengine.org
- My game, Flag Quest:
See QR code
Game, writeup & more
- USB key with everything!



tuxlu.fr/talk_vghacking



- Memory scanner and debugger

Cheat Engine 7.5

File Edit Table D3D Help

00006764-flag_quest_release.exe

Found: 678

Address	Value	Previous	First
E88A9FF35C	55.5	55.5	-0.00003051...
E88B9FF2DC	255.2008057	255.2008057	250.9992065
E88B9FF2F4	255.2008057	255.2008057	250.9992065
E88B9FF30C	55.08114624	55.08114624	-0.00003051...
E88B9FF32C	255.2008057	255.2008057	250.9992065
E88B9FF39C	255.2008057	255.2008057	250.9992065
E88B9FF6D8	0	0	0
E88D9FF2E0	62.46905518	62.46905518	5.99999987E14
E88E9FF260	27.27392578	27.27392578	88.62145996
E88E9FF334	256.563446	256.563446	250.9993896
E88E9FF34C	55.08119202	55.08119202	-0.00003051...
E88E9FF36C	256.563446	256.563446	250.9993896
E88E9FF3DC	256.563446	256.563446	250.9993896
E88F1FF850	45.99002075	45.99002075	5.99999987E14
E88F1FF8BC	262.0027161	262.0027161	250.9993896
E88F1FF8D4	262.0027161	262.0027161	362
E88F1FF90C	257.9138489	257.9138489	362
E88F1FF97C	257.9138489	257.9138489	362
E88F9FF3D0	36.50170898	36.50170898	89.84651611

Buttons: New Scan, Next Scan, Undo Scan, Settings

Scan Type: Increased value | Simple values only

Value Type: Float

☐ Compare to first scan

☐ Unrandomizer

Memory Scan Options

All

Start: 0000000000000000

Stop: 00007FFFFFFF

☒ Writable ☐ Executable

☐ CopyOnWrite

☐ Active memory only

☒ Fast Scan: 4 ☐ Alignment ☐ Last Digits

☐ Pause the game while scanning

Speed: 1 | 0 to 500 | Apply

Memory View

Active	Description	Address	Type	Value
<input type="checkbox"/>	No description	28ADCC91EE0	4 Bytes	5259
<input type="checkbox"/>	No description	28ADC7B33FC	Float	2064
<input type="checkbox"/>	No description	28ADC7EC544	Float	0
<input type="checkbox"/>	No description	28ADCBEB410	Float	2.087934712E-43
<input type="checkbox"/>	No description	28ADCDE9528	Float	4.593582483E-40
<input type="checkbox"/>	No description	28ADCDE96A0	Float	8.407790786E-45
<input type="checkbox"/>	No description	28ADCFC090	Float	-1.892436462E33
<input type="checkbox"/>	No description	28ADC6998C0	Float	Nan
<input type="checkbox"/>	No description	28ADC7B3250	Float	0
<input type="checkbox"/>	No description	28ADD2C7CB8	4 Bytes	100004
<input type="checkbox"/>	No description	28ADD2D17F8	4 Bytes	1000000
<input type="checkbox"/>	No description	28AF6EB4040	4 Bytes	999992

Add Address Manually

CheatEngine

- Memory scanner and debugger
- AutoAssembly and LUA scripting

Hook :

```
retGetGamePlayers_o:
readmem( retGetGamePlayers, 6 )
mov [LocalPlayer],rax
mov rcx, [rax+30]
test rcx,rcx
je short @f
    mov [OakPlayerController],rcx
    mov rcx, [rcx+488]
    test rcx, rcx
    je short @f
    mov rcx, [rax+30]
    mov rcx, [rcx+1988]
    test rcx,rcx
    je short @f
    mov [OakDeveloperPerks],rcx
    test byte ptr [rcx+C8],40
    jne short @f
    or byte ptr [rcx+C8],40

@@:
jmp retGetGamePlayers+6
```



- Memory scanner and debugger
- **AutoAssembly and LUA scripting**

```
function AOBScanAA(script, symbol)
    local success,disableInfo = autoAssemble(script)
    if not success then return nil, disableInfo end -- disab
    local addr = getAddress(symbol)
    autoAssemble(script, disableInfo) -- disable script and
    return addr, 'success'
end

function AOBScanRegion(bytestr, start, stop)
    local script = ([[
[ENABLE]
aobscanregion(luaAOBScanRegionSymbol,%X,%X,%s)
registersymbol(luaAOBScanRegionSymbol)
[DISABLE]
unregistersymbol(luaAOBScanRegionSymbol)
]]):format(getAddress(start), getAddress(stop), bytestr)
    return AOBScanAA(script, 'luaAOBScanRegionSymbol')
end

function AOBScanModule(bytestr, module)
    local script = ([[
[ENABLE]
aobscanmodule(luaAOBScanModuleSymbol,%s,%s)
registersymbol(luaAOBScanModuleSymbol)
[DISABLE]
unregistersymbol(luaAOBScanModuleSymbol)
]]):format(module, bytestr)
    return AOBScanAA(script,'luaAOBScanModuleSymbol')
end
```

CheatEngine

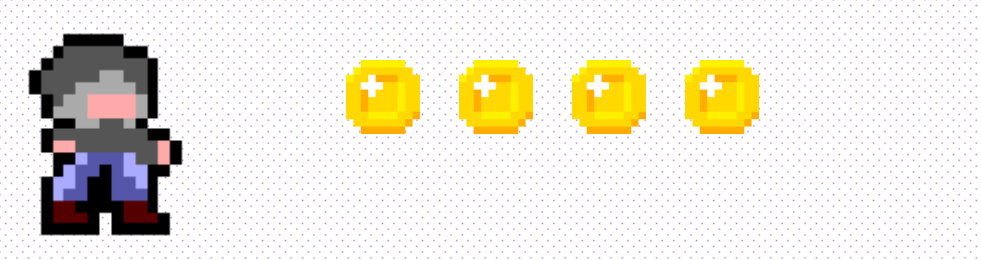
- Memory scanner and debugger
- AutoAssembly and LUA scripting
- GUI 'trainer' generator





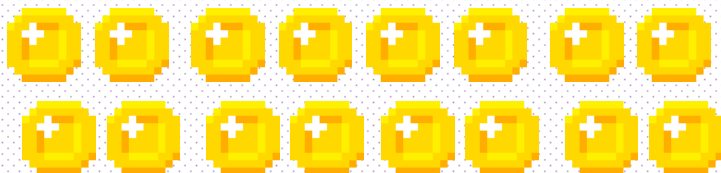
- Memory scanner and debugger
- AutoAssembly and LUA scripting
- GUI 'trainer' generator
- **Not limited to video games**



[illegible]

Finding values

COINS : 4



New ScanNext Scan

Value:

Hex ☐

Scan Type

Found: 113

Address	Value	Previous
1318B...	0	4
1318B...	0	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
13193...	4	4
15C0A...	4	4

Finding values

COINS : 20



New Scan

Next Scan

Value:

☐ Hex 20

Found:4

Address	Value	Previous
15C0D...	20	20
15C0D...	20	20
15C0D...	20	20
15C0E...	20	20

Finding values



COINS : 4000



Active	Description	Address	Type	Value
<input type="checkbox"/>	coins	00000000		
<input type="checkbox"/>	coins	15C0DD859784	Bytes 20	
<input type="checkbox"/>	coins	15C0DD859AC4	Bytes 20	
<input type="checkbox"/>	coins	15C0EC83FD84	Bytes 20	
<input type="checkbox"/>	coins	15C0DD859C84	Bytes 20	

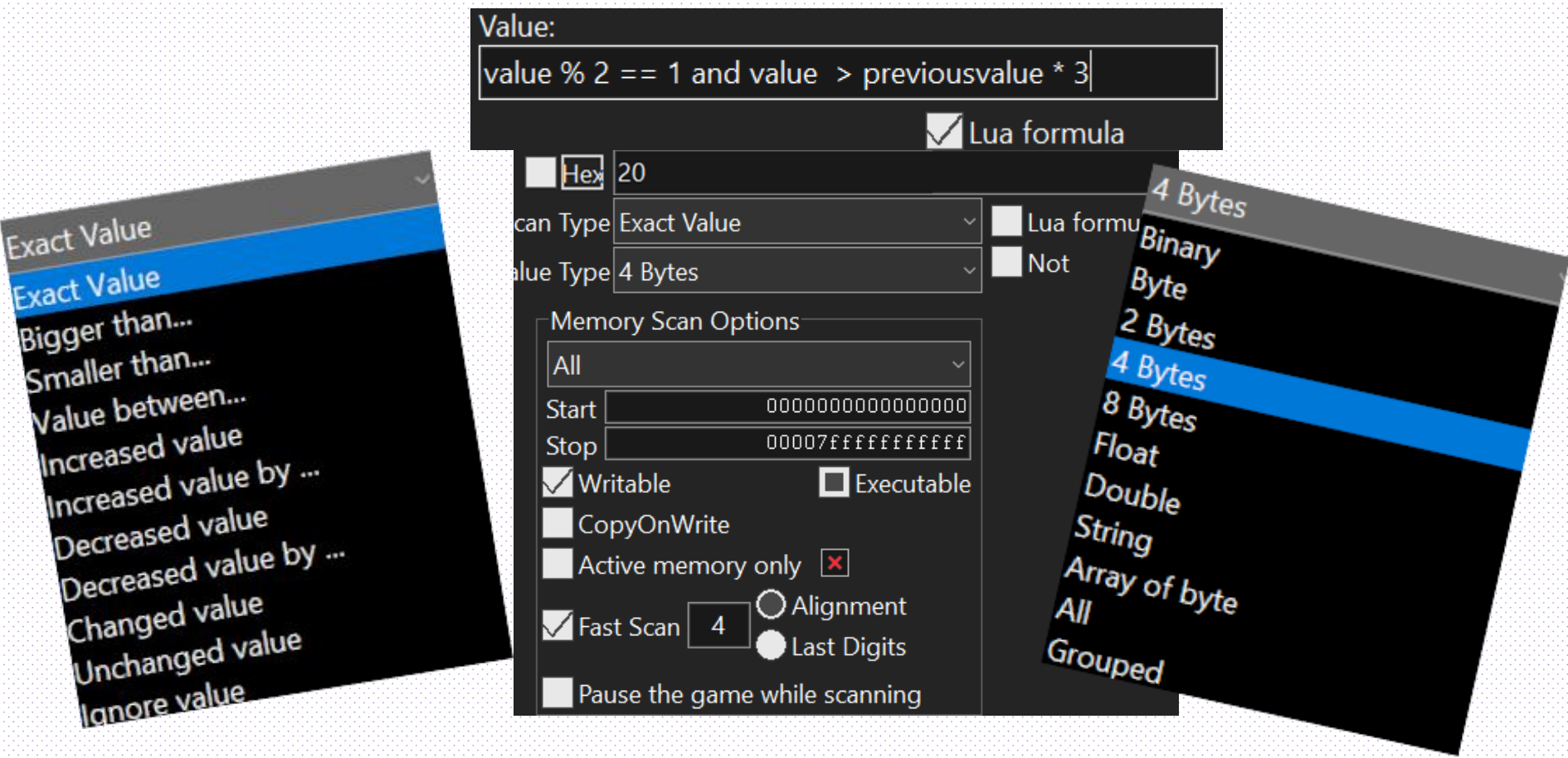
Change Value

what value to change this to?

4000

OK

Finding values: advanced features



Memory viewer

Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

Memory View

Protect:Read/Write AllocationBase=15C0DD40000 Base=

address	80	81	82	83	84	85	86	87	89	ABCDEF	01234567
15C0DD85978	39	05	00	00	64	00	00	00	9	...d...	...\...
15C0DD85988	C8	00	00	00	16	00	00	00	
15C0DD85998	80	51	69	29	5C	01	00	00	Q	i)\...
15C0DD859A8	B0	6A	74	0E	5C	01	00	00	j	t.\...
15C0DD859B8	C6	17	00	00	00	20	00	00		Qm)\...
15C0DD859C8	13	37	00	00	00	01	00	00	.	7.....	B..\...

```
struct Player
{
    int health    = 1337;
    int ???      = ???;
    int ???      = ???;
}
```

Memory viewer


Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

Memory View		Display Type > • 4 Byte decimal	
Protect:Read/Write	AllocationBase=15C0DD40000	Base=	
address	'78	7C	89ABCDEF01234567
15C0DD85978	1337	100	9...d... \...
15C0DD85988	200	22 \...
15C0DD85998	694768000	348	Qi) \.....
15C0DD859A8	242510512	348	jt. \.....
15C0DD859B8	6086	8192 Qm) \...
15C0DD859C8	14099	256	.7..... B.. \...

```
struct Player
{
    int health    = 1337;
    int strength  = 100;
    int defense   = 200;
}
```

Data structures

Description	Address	Type	Value
health	15C0DD85978	4 Bytes	1337

Memory View		Tools  Dissect data/structures	
Offset-description		Address: Value	
Player			
0000	- 4 Bytes	5DA650	: 1337
0004	- 4 Bytes	5DA654	: 100
0008	- 4 Bytes	5DA658	: 200
000C	- 4 Bytes	5DA65C	: 22

```
struct Player
{
    int health    = 1337;
    int strength  = 100;
    int defense   = 200;
}
```


Persisting memory addresses

Activ	Description	Address	Type	Value
<input type="checkbox"/>	coins	00000000		
<input type="checkbox"/>	coins	15C0DD859784	Bytes	??
<input type="checkbox"/>	coins	15C0DD859AC4	Bytes	??
<input type="checkbox"/>	coins	15C0EC83FD84	4 Bytes	??
<input type="checkbox"/>	coins	15C0DD859C84	Bytes	??



Reload game
Lose everything!

Persisting memory addresses

Solution: search *potentially* static addresses
search for all code that points to the address

coins 2216AF30F68 4 Bytes 58



Pointer scan for this address

Recursive
scan



op [(Address - 1) + 01]

op [(Address - 2) + 02]

op [(Address - 3) + 03]

...

7FF7BB2E2724 - 48 03 41 08 - add rax,[rcx+08] RCX=000002216AF30F60

7FF7BB165E68 - 49 89 44 24 08 - mov [r12+08],rax R12=000002216AF30F60

7FF7BB1B9D40 - 49 8B 55 08 - mov rdx,[r13+08] R13=000002216AF30F60



Persisting memory addresses

Problem 1: no results :/

Pointerscanner scanoptions

☒ Scan for address ☐ Scan for addresses with value ☐ Generate pointermap

☐ Use saved pointermap

0152E268

Nr of threads scanning: 12 Normal

Maximum offset value: 128 Max level 6

OK Cancel

4 Bytes Pointer paths 0

Base Address	Offset 0	Poi
--------------	----------	-----



Persisting memory addresses

Problem 2: too many results!

Pointerscanner scanoptions

☒ Scan for address ☐ Scan for addresses with value ☐ Generate pointermap

☐ Use saved pointermap

0152E268

Nr of threads scanning: 12 Normal

Maximum offset value: 128 Max level 6

OK Cancel

4 Bytes Pointer paths 230860

Base Address	Offset 0	Poi
"godot.windows.opt.tools.64.exe"+07212940	10	-
"godot.windows.opt.tools.64.exe"+071DE070	60	-
"godot.windows.opt.tools.64.exe"+07212940	10	-
"godot.windows.opt.tools.64.exe"+071DE070	60	-



Persisting memory addresses

solution: rescan, and compare results

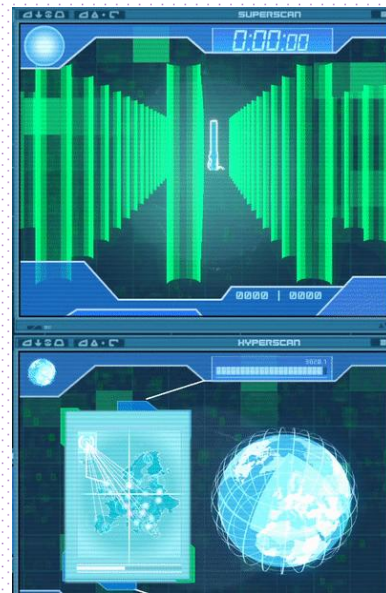
Generate pointermap → Pointer scan for this address

Filename	Address
pointermap_coins5.scandata	248C053F6E8
pointermap_coins3.scandata	1D847EDC898
pointermap_coins1.scandata	247DF1D46D8

4 Bytes Pointer paths 1

Base Address	Offset 0	Points to:
"godot.windows.opt.tools.64.exe"+0717F820	3B8	2216AF30F68 = 206

coins	2216AF30F68	4 Bytes 74
pointerscan result	P->2216AF30F68	4 Bytes 74



What if we don't search a value?

How to search for a condition?



```
def player_move():  
    if collision("coin"):  
        coins += 1  
    if collision("door"):  
        if has_key:  
            open_door()  
    if button("down"):  
        crouch()
```


What if we don't search a value?

How to search for a condition?:
code filter

Memory View Tools  Code Filter



Addresses executed since last filter operation:0

Address List (46093)

Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	No

Load address list

```
def player_move():  
    • if collision("coin"): coins += 1  
    • if collision("door"): open_door()  
    • if has_key: crouch()
```


What if we don't search a value?

How to search for a condition?:

code filter



Addresses executed since last filter operation: 1791

Has been executed

Has not been executed

Start Stop

Address List (44302)

Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	Yes

```
def player_move():  
    if collision("coin"):  
        coins += 1  
    • if collision("door"):  
        • if has_key:  
            open_door()  
    if button("down"):  
        crouch()
```

What if we don't search a value?

How to search for a condition?:

code filter



Addresses executed since last filter operation: 595

☒ Has been executed

☐ Has not been executed

Address List (595)

Address	Executed
Tutorial-i386.exe.text+1BB5	No
Tutorial-i386.exe.text+1BBA	No
Tutorial-i386.exe.text+1BBF	No
Tutorial-i386.exe.text+1BC8	Yes

```
def player_move():  
    if collision("coin"):  
        coins += 1  
    • if collision("door"):  
    •     if has_key:  
            open_door()  
    if button("down"):  
        crouch()
```

What if we don't search a value?

How to search for a condition?:

code filter



Addresses executed since last filter operation: 1

☒ Has been executed

☐ Has not been executed

Address List (1)

Address	Executed
Tutorial-i386.exe.text+1BB5	Yes

```
def player_move():  
    if collision("coin"):   
        coins += 1  
    • if collision("door"):   
    • if has_key:   
        open_door()  
    if button("down"):   
        crouch()
```

Instruction patching

ASM, help!

```
74 02      je      Tutorial-i386.exe.text+26687
EB 49      jmp     Tutorial-i386.exe.text+266D0
A1 B0666500 ▶ mov    eax,[Tutorial-i386.exe+2566B0]
3B 45 E8    cmp     eax,[ebp-18]
74 02      je      Tutorial-i386.exe.text+26693
EB 1F      jmp     Tutorial-i386.exe.text+266B2
C7 45 E8 000... ▶ mov    [ebp-18],00000000
6A 00      push   00
```



Instruction patching

ASM primer

je if ==

jne if !=

jg if >

jg if <

add +=

sub -=

mov x=y

nop do nothing

(padding)

Instruction patching

Replace the *has_key* condition

```
74 02      je      Tutorial-i386.exe.text+26687
EB 49      jmp     Tutorial-i386.exe.text+266D0
A1 B0666500 ▶ mov     eax,[Tutorial-i386.exe+2566B0]
3B 45 E8    cmp     eax,[ebp-18]
74 02      je      Tutorial-i386.exe.text+26693
EB 1F      jmp     Tutorial-i386.exe.text+266B2
C7 45 E8 000... ▶ mov     [ebp-18],00000000
6A 00      push    00
```

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching

Replace the *has_key* condition

```
74 02      je      Tutorial-i386.exe.text+26687
EB 49      jmp     Tutorial-i386.exe.text+266D0
A1 B0666500 ▶ mov     eax,[Tutorial-i386.exe+2566B0]
3B 45 E8    cmp     eax,[ebp-18]
74 02      je      Tutorial-i386.exe.text+26693
EB 1F      jmp     Tutorial-i386.exe.text+266B2
C7 45 E8 000... ▶ mov     [ebp-18],00000000
6A 00      push    00
```

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if has_key:
            open_door()
    if button("down"):
        crouch()
```


Instruction patching

Replace the *has_key* condition

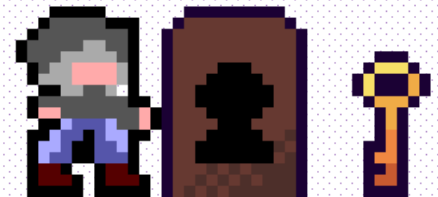
```
74 02      je      Tutorial-i386.exe.text+26687
EB 49      jmp     Tutorial-i386.exe.text+266D0
A1 B0666500 ▶ mov     eax,[Tutorial-i386.exe+2566B0]
3B 45 E8    cmp     eax,[ebp-18]
75 02      jne     Tutorial-i386.exe.text+26693
EB 1F      jmp     Tutorial-i386.exe.text+266B2
C7 45 E8 000... ▶ mov     [ebp-18],00000000
6A 00      push    00
```

```
def player_move():
    if collision("coin"):
        coins += 1
    if collision("door"):
        if not has_key:
            open_door()
    if button("down"):
        crouch()
```

Instruction patching


Replace the *has_key* condition

```
74 02      je      Tutorial-i386.exe.text+26687
EB 49      jmp     Tutorial-i386.exe.text+266D0
A1 B0666500 ▶ mov     eax,[Tutorial-i386.exe+2566B0]
3B 45 E8    cmp     eax,[ebp-18]
75 02      jne     Tutorial-i386.exe.text+26693
EB 1F      jmp     Tutorial-i386.exe.text+266B2
C7 45 E8 000... ▶ mov     [ebp-18],00000000
6A 00      push    00
```



Instruction patching

What if we could do it from known memory addresses?

☐ coins 01723548 4 Bytes 100  Find out what writes to this address

The following opcodes write to 01723548

Count	Instruction
1	004272D7 - 89 02 - mov [edx],eax

Tutorial-i386.exe.text+262D7:
004272CE - 8B 15 B0666500 - mov edx,[Tutorial-i386.exe+2566B0]
004272D4 - 8B 45 F0 - mov eax,[ebp-10]
004272D7 - 89 02 - mov [edx],eax <<

EAX=0000037F
EBX=00000000

Replace

Show disassembler

Add to the codelist

More information

copy memory

Instruction patching: problems

Less obvious instructions

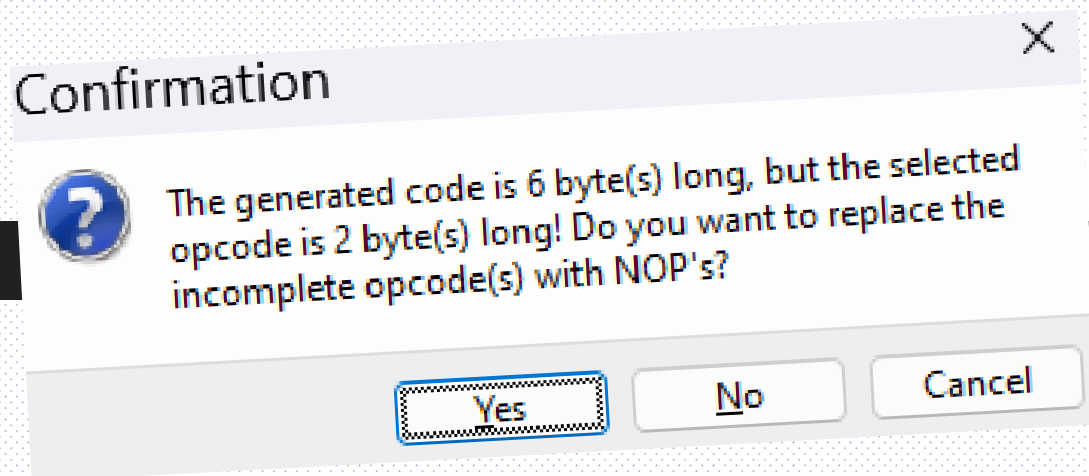
```
mov    [edx],eax
```

where **add** ?



Instruction patching: problems

Instruction size clash



Instruction patching: problems

Instruction size clash

Where to add our code?

```
mov     eax,[ebp-10]
```

```
mov     [edx],00001000
```

```
mov     [edx],000010006.exe+2566B0]
```


Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- Persisting and toggleable

```
[ENABLE]
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx]
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

```
[DISABLE]
dealloc(multiplier)
unregisterSymbol(multiplier)
```

```
dealloc(newmem)
"Tutorial-i386.exe"+272D7:
db 89 02 A1 B0 66 65 00
```




Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory

```
alloc(newmem,2048)
label(return)
```

```
newmem:
// your code here
```

```
jmp return
```

```
"Tutorial-i386.exe"+272D7: //original
jmp newmem                address
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
manages labels, variables...

<input type="checkbox"/>	multiplier	018E0800	4 Bytes	5
--------------------------	------------	----------	---------	---

```
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx]
add eax, [multiplier]
```

```
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods

```
alloc(newmem,2048)
label(return)
alloc(multiplier, 2)
registerSymbol(multiplier)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
"Tutorial-i386.exe"+272D7:
jmp newmem
nop 2
return:
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
newmem:  
mov eax, [edx] //coins += multiplier  
add eax, [multiplier]  
mov [edx],eax  
mov eax, [Tutorial-i386.exe+2566B0]  
jmp return
```

```
"Tutorial-i386.exe"+272D7:  
jmp newmem  
nop 2
```

```
return:  
  
[DISABLE]  
dealloc(multiplier)  
unregisterSymbol(multiplier)
```

```
dealloc(newmem)  
"Tutorial-i386.exe"+272D7:  
db 89 02 A1 B0 66 65 00
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
newmem:  
mov  eax, [edx] //coins += multiplier  
add  eax, [multiplier]  
mov  [edx],eax  
mov  eax, [Tutorial-i386.exe+2566B0]  
jmp  return
```

```
"Tutorial-i386.exe"+272D7:
```

```
jmp  newmem
```

```
nop  2
```

```
return:
```

```
[DISABLE]
```

```
dealloc(multiplier)
```

```
unregisterSymbol(multiplier)
```

```
dealloc(newmem)
```

```
"Tutorial-i386.exe"+272D7:
```

```
db  89 02 A1 B0 66 65 00
```

Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
[ENABLE]
alloc(newmem,2048)
registerSymbol(multiplier)
registerSymbol(INJECT)
aobscanmodule(INJECT,Tutorial-i386.exe,
89 02 A1 B0 66 65 00)

multiplier:
    dd (int)5

newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return

INJECT:
jmp newmem
nop 2
return:
```


Autoassemble

Solution: **auto assembler!**

- Auto-allocate memory
- Create complex methods
- **Persisting and toggleable**

```
[ENABLE]
alloc(newmem,2048)
registerSymbol(multiplier)
registerSymbol(INJECT)
aobscanmodule(INJECT,Tutorial-i386.exe,
89 02 A1 B0 66 65 00)
```

```
multiplier:
    dd (int)5
```

```
newmem:
mov eax, [edx] //coins += multiplier
add eax, [multiplier]
mov [edx],eax
mov eax, [Tutorial-i386.exe+2566B0]
jmp return
```

```
INJECT:
jmp newmem
nop 2
return:
```

```
[DISABLE]
dealloc(multiplier)
unregisterSymbol(multiplier)
```

```
dealloc(newmem)
INJECT:
db 89 02 A1 B0 66 65 00
unregistersymbol(INJECT)
```

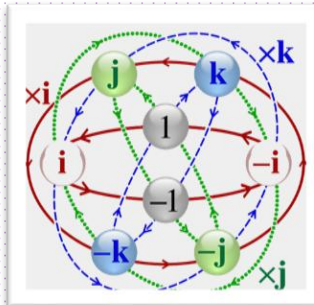
Level 3: thinking 3D

- Where 3D angles?
- Uses quaternions, rotation matrices and radian angles
- See Godot docs for more infos.

Memory View						
2B4CC81B1AC	0.00	0.00	0.00	-6798912...	0.00	0.00
2B4CC81B1C4	0.00	0.00	0.00	-1.00 SX	0.00	0.00
2B4CC81B1DC	0.00	1.00 SY	0.00	0.00	0.00	-1.00 SZ
2B4CC81B1F4	-33.93	14.65	-20.00	-1.00	0.00	0.00
2B4CC81B20C	0.00	1.00	0.00	0.00	0.00	-1.00
2B4CC81B224	-20.20 X	6.64 Y	-27.89 Z	0.00	0.00	0.00
2B4CC81B23C	0.00	1.00	1.00	1.00	0.00	0.00

Properties

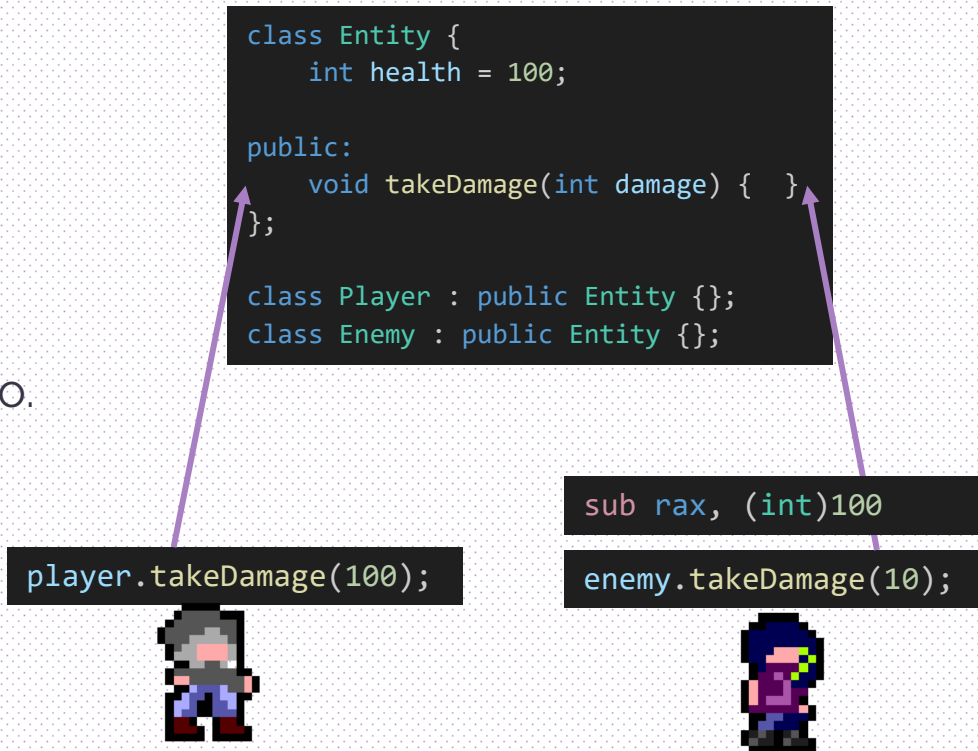
Basis	basis	Basis(1, 0, 0, 0, 1, 0, 0, 0, 1)
Vector3	origin	Vector3(0, 0, 0)



Finding shared instructions

Modify an instruction, suddenly:

- Bugs appear
- Making yourself invincible makes enemies invincible too.
- Game crashes



Finding shared instructions


Solution 1:  Dissect data/structures

- "Find out what addresses this code access"
- Select all: "dissect data".
- Compare value in groups


```
100042D13 - 29 50 60 - sub [rax+60],edx
```

Find out what addresses this code accesses




 Accessed addresses by 100042D13

Code Address	Value	Count
The following 3 addresses have been accessed by the code		
Address	Value	Count
01541CA0	48	20
01541D40	192	5
015429C0	195	5


 Open dissect data with selected addresses



 Structure dissect:unnamed structure

Offset-description	Address: Value	Address: Value	Address: Value
unnamed structure			
> 0000 - Pointer	1541C40 : 0->11541CE0 : 0->1002B4478	1542960 : 0->1002B4478	
0008 - 4 Bytes	1541C48 : 0	1541CE8 : 0	1542968 : 0
000C - 4 Bytes	1541C4C : 0	1541CEC : 0	154296C : 0
0010 - 4 Bytes	1541C50 : 0	1541CF0 : 0	1542970 : 0
0014 - Double	1541C54 : 0.0(1541CF4 : 0.0078125	1542974 : 0.0078125	
001C - Float	1541C5C : 1	1541CFC : 1	154297C : 1
0020 - 4 Bytes	1541C60 : 0	1541D00 : 0	1542980 : 0
0024 - Double	1541C64 : 0.0(1541D04 : -0.0008789066(1542984 : -0.0008789064		
002C - 4 Bytes	1541C6C : 0	1541D0C : 0	154298C : 0

Finding shared instructions

Solution 2:  Find commonalities between addresses

- Auto-find common values between different calls
- Compare structures with only unique distinct values

Accessed addresses by 100042D13

Code Address

The following 3 addresses have been accessed by the code

Address	Value	Count
01541CA0	48	1
01541D40	192	5
015429C0	195	1

Mark selection as group 1

Mark selection as group 2

Scan for commonalities

Commonality scanner

Register	Status
RAX	Doubleclick to launch structure compare
RCX	Doubleclick to launch structure compare
RDX	"Group 2" has common value 0x1

Structure Compare : RAX

Max Level	Structsize	Offset 0	G1:01541C40	G2:01541CE0	G2:01542960
2	4096	28	01541C68 : 10...	01541D08 : 3209481421	01542988 : 32...
		64	01541CA4 : 100	01541D44 : 200	015429C4 : 200
		70	01541CB0 : 0	01541D50 : 1	015429D0 : 1
		78	01541CB8 : 0	01541D58 : 256	015429D8 : 256
		D4	01541D14 : 11...	01541DB4 : 0	01542A34 : 0
		104	01541D44 : 200	01541DE4 : 0	01542A64 : 0
		118	01541D58 : 256	01541DF8 : 0	01542A78 : 0


☒ Only find matching group

New Scan Rescan


Finding shared instructions

Solution 3: manual search

- Some “static” values change only when relaunching the game
- “Find commonalities” does not work across game reboots
- Use some scripts to help scans

 Browse this memory region

```
Protect:Read/Write AllocationBase=014B0000 Base=01541000 Size=6E000
address A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7
01541CA0 30 00 00 00 64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01541CB8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01541CD0 00 00 00 00 00 00 00 00 A1 80 71 00 00 00 00 00 78 44 2B 00 01 00 00 00
01541CE8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 80 3F 00 00 80 3F
01541D00 00 00 00 00 85 EB 51 BE CD CC 4C BF 00 00 00 00 00 00 00 00 E2 B2 2C 4D
01541D18 00 00 00 00 00 00 00 00 00 00 00 00 25 06 01 3E 00 00 00 00 00 00 00 00
01541D30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C8 00 00 00
01541D48 B0 A0 55 01 00 00 00 00 01 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00
01541D60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01541D78 A1 80 7B 00 00 00 00 00 B8 BD 2B 00 01 00 00 00 00 00 00 00 00 00 00 00
01541D90 78 B8 2B 00 01 00 00 00 50 F9 4D 01 00 00 00 00 00 00 00 00 00 00 00 00
01541DA8 00 00 00 00 01 00 22 2C 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01541DC0 01 00 00 00 1C 00 00 00 A0 E5 2B 00 01 00 00 00 E0 74 4D 01 00 00 00 00
01541DD8 02 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

 Copy to clipboard

Ctrl+C

```
cat ./comm_test.txt
80 F4 23 40 F6
65 F4 8A 40 26
68 F4 23 40 F6
python ./detect_commonalities_memory.py
./comm_test.txt 3
-2 = F4
0 = 40
```


LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan, AutoAssembly integration
- GUI development

```
{ $lua }  
if syntaxcheck then return end  
[ENABLE]
```

```
l2sh_counter = 0  
l2sh_timer = createTimer(nil)
```

```
speedhack_setSpeed(500)  
l2sh_timer.Enabled = false  
l2sh_timer.Interval = 100 -- spam keys  
every 0.1s
```

```
function l2SH_loop(v_timer)  
  
    if ((l2sh_counter == -1) or  
        (l2sh_counter == 750)) then  
        l2sh_timer.Enabled = false  
        -- stop after 75s or manual interrupt  
        speedhack_setSpeed(1)  
        return  
    end  
  
    if (getForegroundWindow() ~=  
        findWindow(null, "flag quest")) then  
        return -- do not keypresses when  
        game window is not focused  
    end
```

LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan, AutoAssembly integration
- GUI development

```
function l2SH_loop(v_timer)

    if ((l2sh_counter == -1) or
        (l2sh_counter == 750)) then
        l2sh_timer.Enabled = false
        -- stop after 75s or manual interrupt
        speedhack_setSpeed(1)
        return
    end

    if (getForegroundWindow() ~=
        findWindow(null, "flag quest")) then
        return -- do not keypresses when
        game window is not focused
    end

    if ((l2sh_counter % 10) == 0) then
        doKeyPress(VK_SPACE)
        -- 1/10 space keypress
    else
        doKeyPress(VK_RIGHT)
    end
    l2sh_counter = l2sh_counter + 1
end

-- put this AFTER method definition
l2sh_timer.OnTimer = l2SH_loop
```

LUA autoclicker

Code complex scripts with LUA

- Complete CE API
- Timers, memory scan, AutoAssembly integration
- GUI development

```
speedhack_setSpeed(1)
return
end

if (getForegroundWindow() ~=
findWindow(null, "flag quest")) then
    return -- do not keypresses when
game window is not focused
end

if ((l2sh_counter % 10) == 0) then
    doKeyPress(VK_SPACE)
    -- 1/10 space keypress
else
    doKeyPress(VK_RIGHT)
end
l2sh_counter = l2sh_counter + 1
end

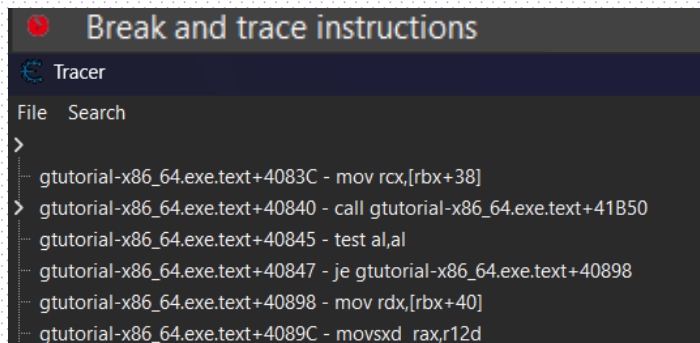
-- put this AFTER method definition
l2sh_timer.OnTimer = l2SH_loop
l2sh_timer.Enabled = true

[DISABLE]
speedhack_setSpeed(1)
l2sh_timer.destroy() --important!
```

Nice CheatEngine features

- Debugger: ultra powerful

stack trace



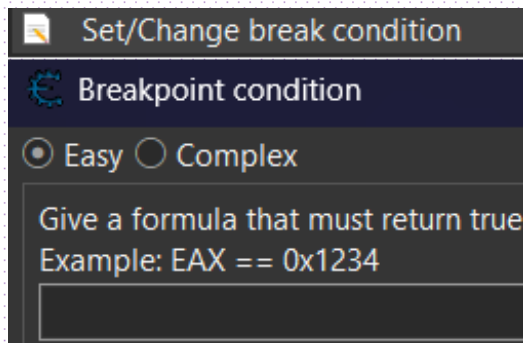
Break and trace instructions

Tracer

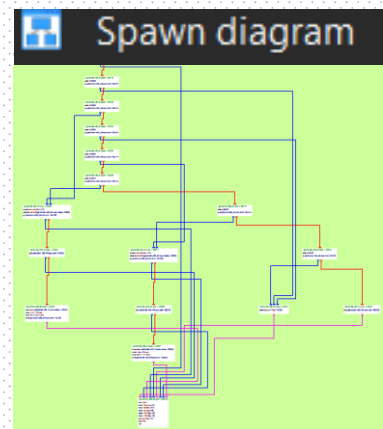
File Search

```
>
| tutorial-x86_64.exe.text+4083C - mov rcx,[rbx+38]
> tutorial-x86_64.exe.text+40840 - call tutorial-x86_64.exe.text+41B50
| tutorial-x86_64.exe.text+40845 - test al,al
| tutorial-x86_64.exe.text+40847 - je tutorial-x86_64.exe.text+40898
| tutorial-x86_64.exe.text+40898 - mov rdx,[rbx+40]
| tutorial-x86_64.exe.text+4089C - movsxd rax,r12d
```

conditional breakpoints

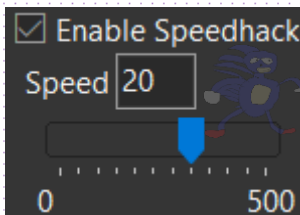


Flow charts



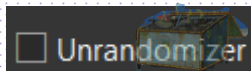
Nice and useless CheatEngine features

- Speedhack: it just works

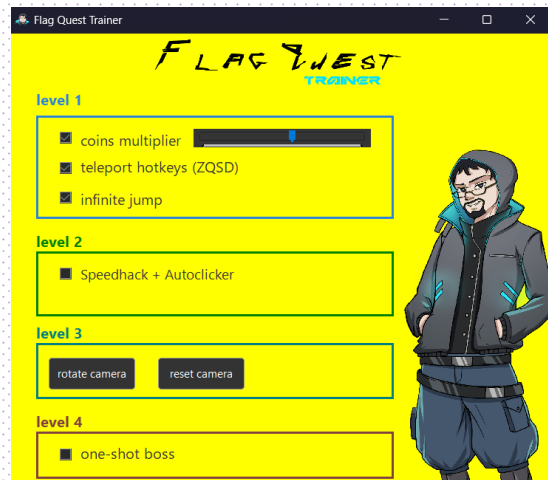


- GUI generator: very powerful
but long topic

- Unrandomizer: never works



- Mono Features: better use Unity Explorer directly



The best CheatEngine feature

- Luacode: put LUA inside your ASM
- Write complex code inside of ASM patch
- Have full LUA API access
- Rename register as variables

```
code:
    cmp r9, rax
    jne end
```

```
{ $LUACODE pos=RBX }
```

```
local npos = pos+0x424 + 8 -- Z pos addr
if npos ~= getAddressSafe("lev3_z") then
    print("Updating all CE 3D position vars")
```

```
    local names = { "z", "y", "x", "sz2",
                    "sz1", "sz0", "sy2", "sy1", "sy0", "sx2",
                    "sx1", "sx0" }
```

```
    for _, name in ipairs(names) do
        registerSymbol("lev3_"..name, npos)
        -- going backward in memory
        npos = npos - 4
    end
end
{ $ASM }
```

```
end:
    movups [rbx+00000420], xmm2 // original
    instruction for setting sz2
    jmp return
```


The best worst CheatEngine feature

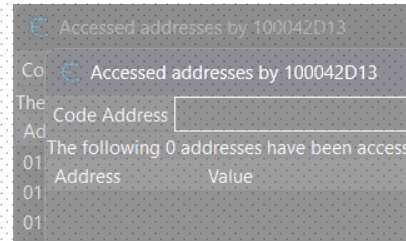
- Luacode: put LUA inside your ASM
- Bugged at the moment
Need to add a space after your comments lines at the start of your cheat
- Not optimized
- May crash your game
- Consider `{ $CCODE }`
write C inside your ASM



Common Cheat Engine problems 1

- I changed a value and my game crashed!
-> *welcome to CE. It will do that.*

- My “find what writes/access” address search has no result
-> *you already have a scan running, but the window is hidden behind another.*



- I added My AutoAssembly cheat to the codelist, but I can't enable it
-> *you clicked on **Execute** and it already patched the game. Relaunch the game.*



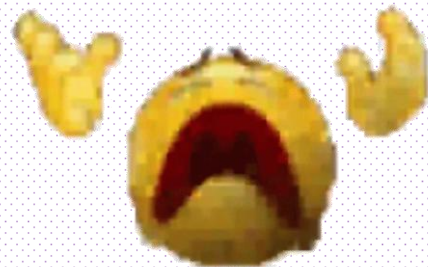
Common Cheat Engine problems 2

- I did a scan, saved my results in a file.
Redid a scan, saved to the same file.
Now the scan doesn't work.
-> *Just don't do that*

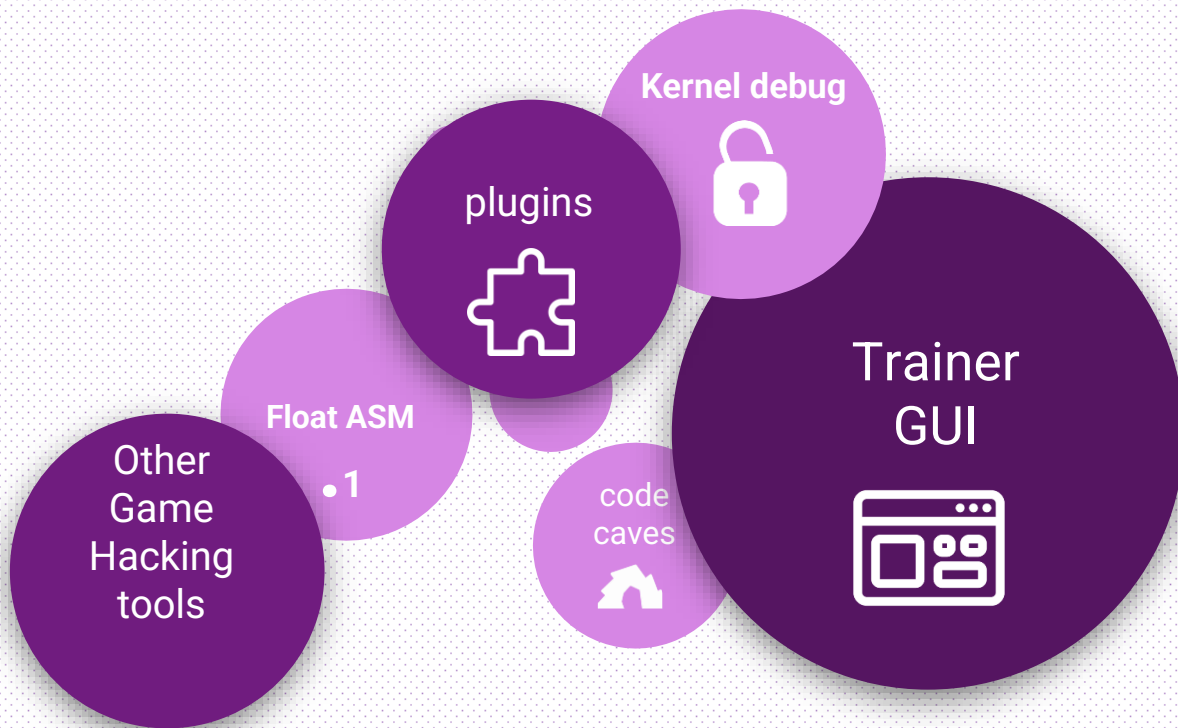
- My LUA script runs twice!
-> *you forgot to add* `if syntaxcheck then return end`

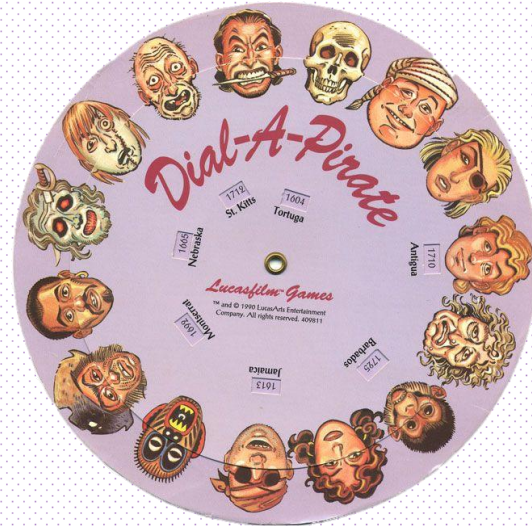
I launched a LUA script that never terminates,
modified it and restarted it. Now I have bugs.

-> *Relaunch Cheat Engine :/*



Next steps



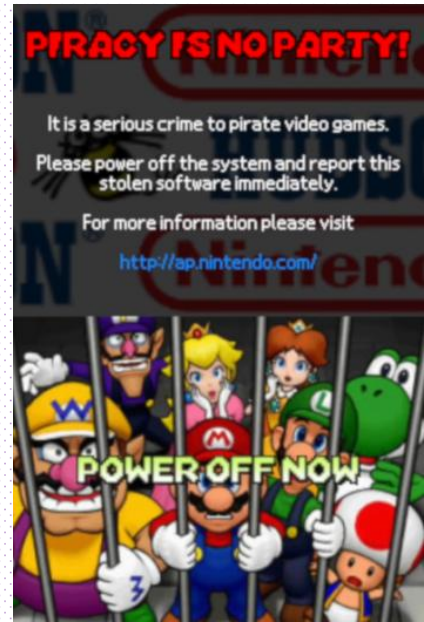


tuxlu.fr/talk_vghacking





tuxlu.fr/talk_vghacking





thank you.

