

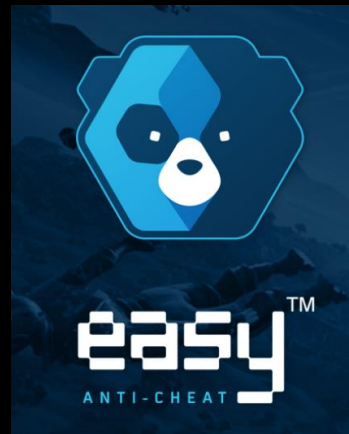
The background of the slide is a grayscale image of a circuit board. It features various traces, pads, and circular components. A solid black horizontal band runs across the middle of the image, serving as a backdrop for the title text.

Memory Editing. Hacking in Video Games

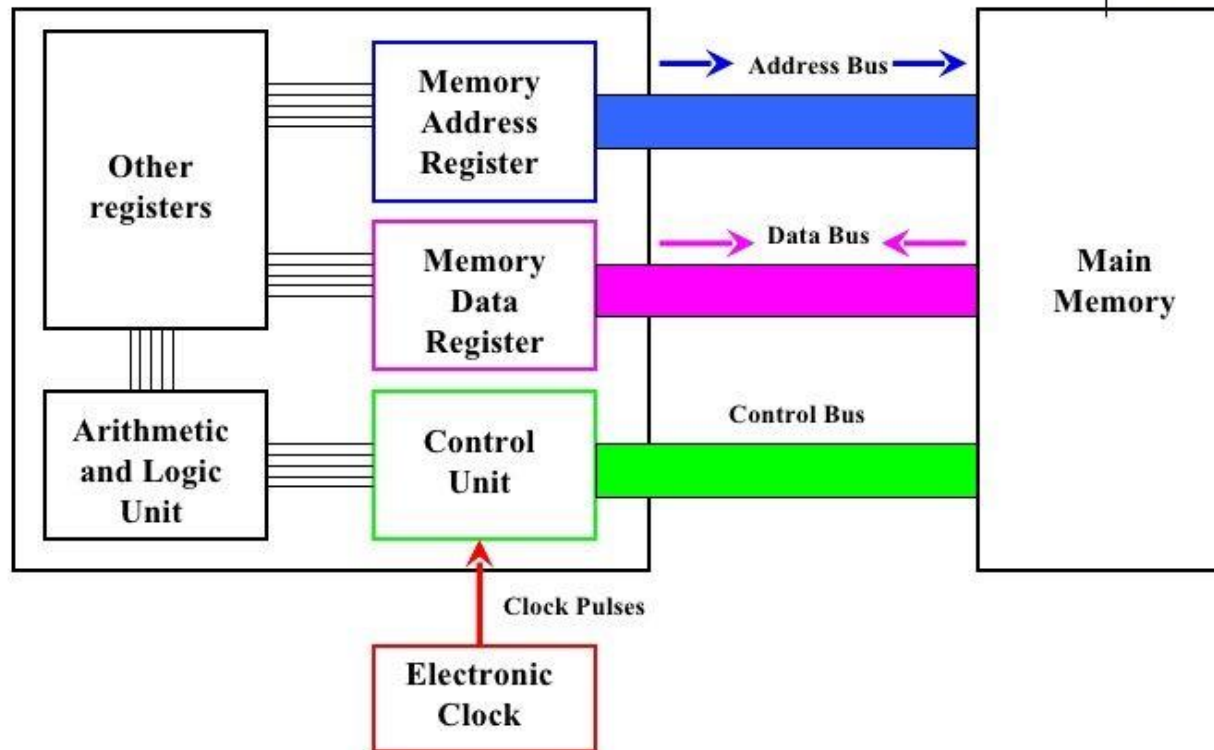
By Benjamin Haedt
10 March 2023

Introduction

- Video Game hacking has been around for as long as video games have been played. From Game Genie for the Nintendo Entertainment System, to Counter Strike: GO aimbots, people have always wanted to find a way to gain an unfair advantage to win at their favorite games.
- Some people cheat in video games because they want to win, others try to find exploits just see if they can, and others do it to make money.
- Individuals and companies make millions developing cheats for video games and then selling their products.
- Like in other areas of hacking, there are always people trying to develop ways to stop the hacking from happening. One example of anti-cheat software is Easy ANTI-CHEAT, from EPIC Games. This developer in particular makes money by licensing their product to other game developers.



The Processor Structure



The (very) Basics of Computer Architecture

- The brains of any computer is the CPU, or the Central Processing Unit.
- The CPU performs calculations and executes code.
- While performing calculations on data, registers are used to store temporary data.
- Then once the instructions are processed, it stores the outcome in storage, or memory. With memory editing, we are looking at the memory that will be sent to the processor to be executed.

What is Memory Editing?

- Every piece of code that runs on our computer is loaded into memory when we run a program. As different parts of the program are called, the instructions are sent to the processor.
- Memory holds instructions for programs that can be accessed at anytime. When accessed, the instructions at that location will be sent to the processor to be executed.
- Memory Editing is the process of accessing the current memory of a process and either changing values directly, changing the assembly instructions that tell the processor what to do with the code, or injecting our own code to change what happens when we execute a piece of code.
- We view this data as Memory Addresses. These addresses hold the instructions on what to do with its data. The instructions can be seen in the form of assembly code, hexadecimal values, and ASCII.

FileViewDebugPluginsOptionsWindowHelp

004861BB: 55 PUSH EBP

004861BC: 8BEC MOV EBP,ESP

004861BE: 6A FF PUSH -1

004861C0: 68 30C94A00 PUSH Df2.004AC930

004861C5: 68 546D4800 PUSH Df2.00486054

004861CA: 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]

004861D0: 50 PUSH EAX

004861D1: 64:8925 000000 MOV DWORD PTR FS:[0],ESP

004861D8: 83EC 58 SUB ESP,58

004861DB: 53 PUSH EBX

004861DC: 56 PUSH ESI

004861DD: 57 PUSH EDI

004861DE: 8965 E8 MOV DWORD PTR SS:[EBP-10],ESP

004861E1: FF15 DCC04A00 CALL DWORD PTR DS:[<&KERNEL32.GetVersion

004861E7: 33D2 XOR EDX,EDX

004861E9: 8AD4 MOV DL,AH

004861EB: 8915 50DB5000 MOV DWORD PTR DS:[50DB50],EDX

004861F1: 8BC8 MOV ECX,EAX

004861F3: 81E1 FF000000 AND ECX,0FF

004861F9: 890D 4CDB5000 MOV DWORD PTR DS:[50DB4C],ECX

004861FF: C1E1 08 SHL ECX,8

00486202: 03CA ADD ECX,EDX

00486204: 890D 48DB5000 MOV DWORD PTR DS:[50DB48],ECX

0048620A: C1E8 10 SHR EAX,10

0048620D: A3 44DB5000 MOV DWORD PTR DS:[50DB44],EAX

00486212: 33F6 XOR ESI,ESI

00486214: 56 PUSH ESI

00486215: E8 24290000 CALL Df2.00488B3E

0048621A: 59 POP ECX

0048621B: 85C0 TEST EAX,EAX

0048621D: 75 08 JNZ SHORT Df2.00486227

0048621F: 6A 1C PUSH 1C

00486221: E8 B0000000 CALL Df2.004862D6

00486226: 59 POP ECX

00486227: 8975 FC MOV DWORD PTR SS:[EBP-4],ESI

0048622A: E8 23150000 CALL Df2.00487752

0048622F: FF15 A0C04A00 CALL DWORD PTR DS:[<&KERNEL32.GetComm

00486235: A3 B419E100 MOV DWORD PTR DS:[E119B4],EAX

0048623A: E8 2A520000 CALL Df2.00488B49

0048623F: A3 2CDB5000 MOV DWORD PTR DS:[50DB2C],EAX

SE handler installation

KERNEL32.GetVersion

CGetCommandLineA

Registers (FPU)

EAX 0019FFCC

ECX 004861BB Df2.<ModuleEntryPoint>

EDX 004861BB Df2.<ModuleEntryPoint>

EBX 00200000

ESP 0019FF78

EBP 0019FF84

ESI 004861BB Df2.<ModuleEntryPoint>

EDI 004861BB Df2.<ModuleEntryPoint>

EIP 004861BB Df2.<ModuleEntryPoint>

C 0 ES 002B 32bit 0(FFFFFFFF)

P 1 CS 0023 32bit 0(FFFFFFFF)

A 0 SS 002B 32bit 0(FFFFFFFF)

Z 1 DS 002B 32bit 0(FFFFFFFF)

S 0 FS 0053 32bit 210000(FFF)

T 0 GS 002B 32bit 0(FFFFFFFF)

D 0

O 0 LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,6E,LE)

ST0 empty 0.0

ST1 empty 0.0

ST2 empty 0.0

ST3 empty 0.0

ST4 empty 0.0

ST5 empty 0.0

ST6 empty 0.0

ST7 empty 0.0

FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)

FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

Df2.<ModuleEntryPoint>+47

AddressHexDumpASCII

004AF14073 6E 64 32 00 00 00 00 32 64 6E 73 B0 28 40 00snd2....2dns@(.
004AF15073 6E 64 33 00 00 00 00 33 64 6E 73 10 29 40 00snd3....3dns@).
004AF16073 6C 67 74 00 00 00 00 74 67 6C 73 F0 CC 42 00slgt....tgls@fB.
004AF17074 72 65 65 00 00 00 00 65 65 72 74 80 29 40 00tree....eertC@).
004AF18076 65 68 30 00 00 00 00 30 68 65 76 C0 37 42 00veh0....0hev^7B.
004AF19068 65 6C 6F 00 00 00 00 6F 6C 65 68 00 C2 42 00helo....oleh.^B.
004AF1A073 70 61 72 00 00 00 00 72 61 70 73 30 D0 42 00spar....raps0^B.
004AF1B073 67 75 6C 00 00 00 00 6C 75 67 73 E0 D2 42 00sgul....lugs@B.
004AF1C062 74 72 65 00 00 00 00 65 72 74 62 70 D1 42 00btre....ertbp^B.
004AF1D070 72 6E 68 00 00 00 00 68 6E 72 70 20 D8 42 00prnh....hnrp.B.
004AF1E076 75 6C 74 00 00 00 00 74 6C 75 76 E0 DF 42 00vult....tluv@B.
004AF1F06D 69 6E 65 00 00 00 00 65 6E 69 6D B0 E3 42 00mine....enlm@B.
004AF20067 6E 72 63 00 00 00 00 63 72 6E 67 80 E5 42 00gnrc....crngC@B.
004AF21067 6E 72 6C 00 00 00 00 6C 72 6E 67 80 13 40 00gnrl....lrngC@B.
004AF22067 6E 6C 32 00 00 00 00 32 72 6E 67 20 14 40 00gnl2....2rng @B.
004AF23063 61 63 74 00 00 00 00 74 63 61 63 20 E6 42 00caot....tcaoc ^B.
004AF2406C 61 6D 70 00 00 00 00 70 6D 61 6C 20 E7 42 00lamp....pmal ^B.
004AF25070 61 6C 6D 00 00 00 00 6D 6C 61 70 40 ED 42 00palm....mlap@B.
004AF2606F 61 68 74 00 00 00 00 74 68 61 6F B0 E8 42 00oakt....tkao@B.
004AF27077 64 73 6D 00 00 00 00 6D 73 64 77 60 E9 42 00wdsn....msdw^B.
004AF28077 64 6D 64 00 00 00 00 64 6D 64 77 E0 E9 42 00wdnd....dwdw@B.
004AF29077 64 6C 67 00 00 00 00 67 6C 64 77 60 EA 42 00wdlg....gl dw^B.
004AF2A062 6C 64 67 00 00 00 00 67 64 6C 62 B0 EF 42 00bldg....gd lb@B.
004AF2B070 65 6E 67 00 00 00 00 67 6E 65 70 30 F0 42 00peng....gnep@B.
004AF2C076 74 72 67 00 00 00 00 72 74 76 40 F7 42 00vtrg....grtv@B.
004AF2D066 6C 61 67 00 00 00 00 67 61 6C 66 B0 12 40 00flag....galf@B.
004AF2E02B 00 00 00 43 44 46 53 00 00 00 00 44 45 4C 54+...CDF\$....DELT
004AF2F041 46 4F 52 43 45 32 00 25 73 5C 00 72 62 00 00AFORCE2.%s\..rb..
004AF30025 73 5C 55 73 5C 25 73 00 00 00 00 44 46 32 53%\$%\$%\$....DF2\$
004AF31045 54 55 40 00 00 00 00 44 46 32 49 4E 54 52 4FETUP....DF2INTR0
004AF32025 42 00 00 00 4C 5F 48 50 00 00 00 53 54 62 4D.BIK....TeamOrde
004AF33072 42 00 00 00 4C 5F 48 50 00 00 00 53 54 62 4Drs....L_HP....STRM
004AF34049 53 43 33 37 00 00 00 4C 5F 48 50 43 4D 00 00ISCS37....L_HPCH..
004AF35053 54 52 4D 49 53 43 33 36 00 00 00 4C 5F 48 50STRMISC6....L_HP
004AF36048 46 00 00 53 54 53 4D 49 53 43 33 3E 00 00 00MC....STRMISC6

0019FF7000000000

0019FF7400000000

0019FF7876707D69RETURN to KERNEL32.76707D69

0019FF7C0020D000

0019FF8076707D50KERNEL32.BaseThreadInitThunk

0019FF840019FFDC

0019FF8877B5B74BRETURN to ntdll.77B5B74B

0019FF8C0020D000

0019FF90B0D06F8A

0019FF9400000000

0019FF9800000000

0019FF9C0020D000

0019FFA000000000

0019FFA400000000

0019FFA800000000

0019FFAC00000000

0019FFB000000000

0019FFB400000000

0019FFB800000000

0019FFBC00000000

0019FFC000000000

0019FFC40019FF90

0019FFC800000000

0019FFCC0019FFEAPointer to next SEH record

0019FFD077B6E8B0SE handler

0019FFD4C709A596

0019FFD800000000

0019FFDC0019FFEC

0019FFE077B5B6CFRETURN to ntdll.77B5B6CF from ntdll.77B5B720

0019FFE4FFFFFFFFEnd of SEH chain

0019FFE877B88684SE handler

0019FFEC00000000

0019FFF000000000

0019FFF4004861BBDf2.<ModuleEntryPoint>

0019FFF80020D000

0019FFFC00000000

How does it relate to hacking in video games?

- Game hacking looks at the running process of a video game and manipulates the data using Memory Editing to perform actions otherwise not allowed in the game. This is also known as Memory Editing.
 - Some examples of exploits in games include unlimited ammunition, invincibility, teleportation, and auto-aim.
 - Extreme examples include exploits to crash servers or ban players without admin privileges.
 - Found exploits are known as “pokes” or instructions.

```
000067CC00 78 68 61 77 6B 78 78 00 00 00 00 00 00 00 00 xhawkxx.....
000067CC10 00 00 00 00 00 00 00 00 44 30 32 4D 30 31 2E 42 .....D02M01.B
000067CC20 4D 53 00 00 00 00 00 00 00 00 00 00 00 00 00 MS.....
000067CC30 00 00 00 00 00 00 00 00 00 00 00 00 53 69 78 20 .....Six
000067CC40 6F 66 20 4F 6E 65 00 00 00 00 00 00 00 00 00 of One.....
000067CC50 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000067CC60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

```
0044CD0E . 8B4A 14 MOV ECX,DWORD PTR DS:[EDX+14]
0044CD11 . 8B348D 04A379 MOV ESI,DWORD PTR DS:[ECX*4+79A3D4]
0044CD18 . 8D048D 04A379 LEA EAX,DWORD PTR DS:[ECX*4+79A3D4]
0044CD1F . 85F6 TEST ESI,ESI
0044CD21 . 5E POP ESI
0044CD22 . 74 3E JE SHORT Df2.0044CD62
0044CD24 . C7048D 04A279 MOV DWORD PTR DS:[ECX*4+79A2D4],1
0044CD2F . 8B0D E8D36700 MOV ECX,DWORD PTR DS:[67D3E8]
0044CD35 . F6C5 01 TEST CH,1
0044CD38 . 75 02 JNZ SHORT Df2.0044CD3C
0044CD3A . FF08 DEC DWORD PTR DS:[EAX]
0044CD3C > 8338 00 CMP DWORD PTR DS:[EAX],0
0044CD3F . 75 05 JNZ SHORT Df2.0044CD46
0044CD41 . E9 9A230000 JMP Df2.0044F0E0
```

Tools used for Memory Editing

Tsearch

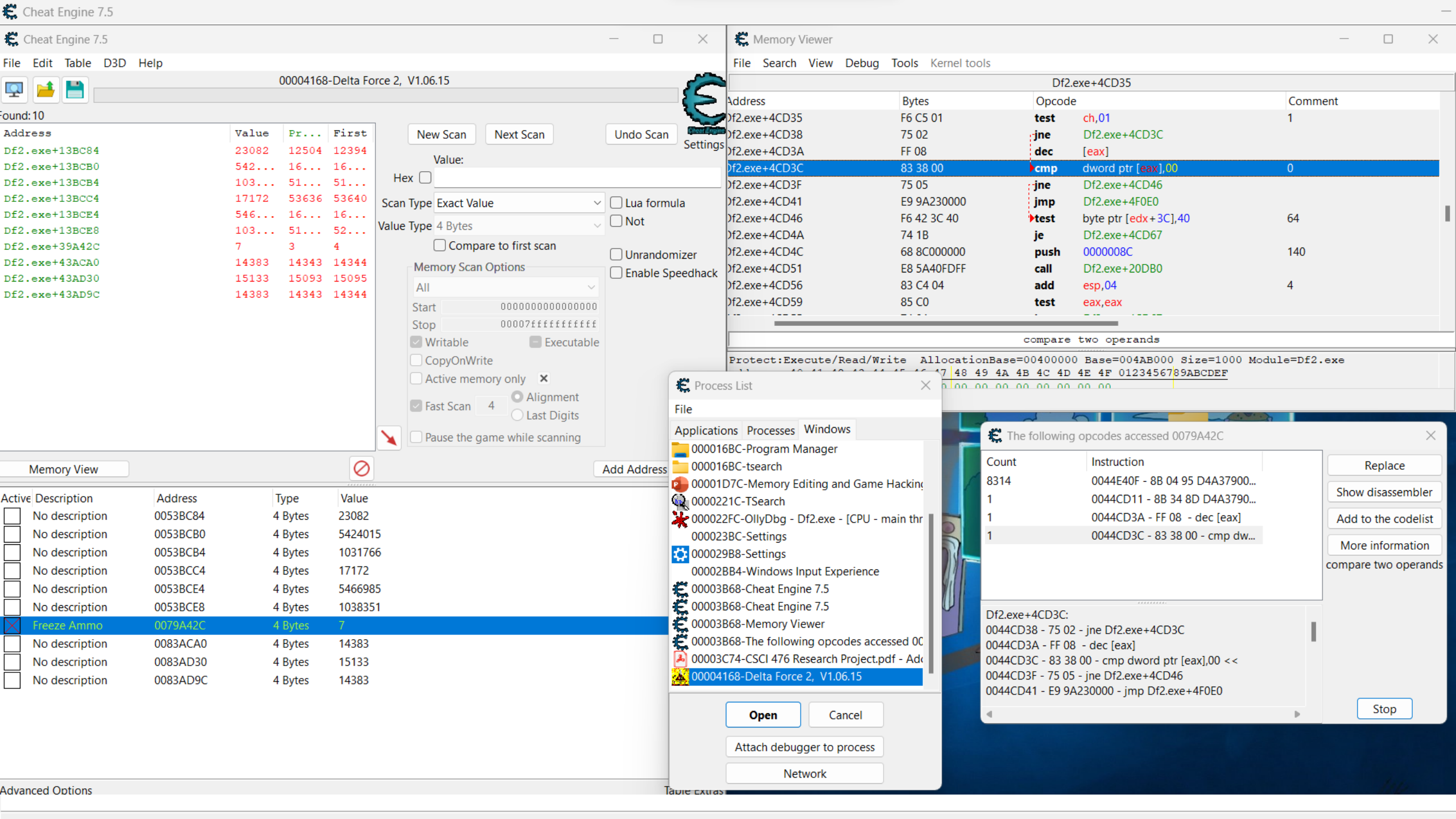
- Good for beginners
- Easy to use and can automate simple tasks
- Has an “Auto-hack” feature.
- Save found pokes in tables.
- Can freeze values
- Limited on how advanced your hacks can be.

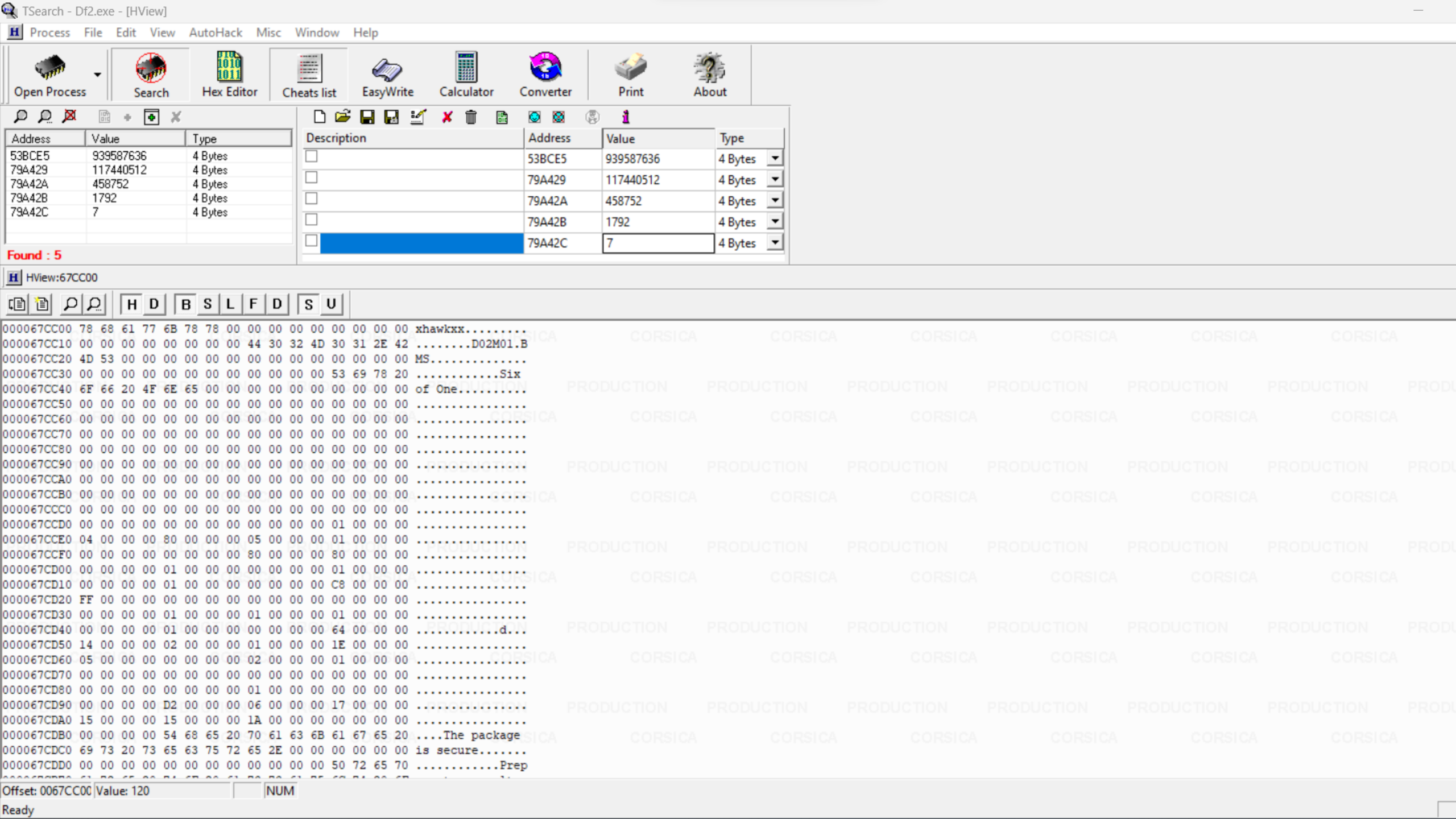
Cheat Engine

- The standard for most game hackers.
- Can find easy to manipulate data, similar to Tsearch
- Allows code injection.
- Change assembly instructions
- Find pointers easily
- Save your finds as a “Cheat table”
- See what reads or writes to a specific address.

OllyDBG

- For advanced users
- Can show
 - assembly instructions
 - hex and ASCII values
 - Show the registers, pointers, the flow of instructions, and everything associated with the process.
- Set break points and write custom assembly code.
- Follow the flow of the instruction set on the stack.





Prevention

- Handle more data on the server side.
 - This means that the more the server handles, such as keeping track of how much ammunition a player has, the harder it is for the player to change the amount of ammunition they have on the client side.
- Using pointers that change data locations
 - When pointers are used to change the location of data, it can be harder to find a poke in memory and once it changes the values will have to be found again.
- Client-side memory monitors
 - This includes special programs that run with the game that can detect unauthorized changes in memory or detect the use of memory editing software
- Use floating-point and decimal variables
 - This makes it harder to easily find values such as health that can be found using simple searching through the games memory


The Community

- The people I knew in this community, back in the 2000's, didn't hack games to make money or to gain an unfair advantage, they did it because it was fun to see what we could figure out. There used to be blogs where we would share our finds and share different techniques. We were always trying to out-do and challenge each other. It wasn't all about malicious intent.



In this screenshot from DF2, I was able to change the players skin textures to load a pink color. This was used as part of an aim-bot.

Sources

- bamdad. (2018, December 3). *Image result for CPU structure diagram*. Pinterest. Retrieved March 17, 2023, from <https://www.pinterest.com/pin/792915078121864419/>
- *Cheat engine*. Cheat Engine. (n.d.). Retrieved March 17, 2023, from <https://cheatengine.org/>
- *Don't bear with the Cheaters*. Easy Anti-Cheat. (2023). Retrieved March 17, 2023, from <https://www.easy.ac/en-us/>
- *Privatecheatz* : Next Gen Undetected Private Hacks & Cheats 2022. PrivateCheatz. (2023, March 13). Retrieved March 17, 2023, from <https://www.privatecheatz.com/>
- *What is register?: Types of registers: Importance: Advantages*. EDUCBA. (2022, December 13). Retrieved March 17, 2023, from <https://www.educba.com/what-is-register/>
- Wikimedia Foundation. (2023, February 27). *Game genie*. Wikipedia. Retrieved March 17, 2023, from https://en.wikipedia.org/wiki/Game_Genie
- Wikimedia Foundation. (2023, January 9). *Call stack*. Wikipedia. Retrieved March 17, 2023, from https://en.wikipedia.org/wiki/Call_stack
- Woodford. , C. (2022, August 22). *How does computer memory work? Explain that Stuff*. Retrieved March 17, 2023, from <https://www.explainthatstuff.com/how-computer-memory-works.html>
- Yuschuk, O. (n.d.). *OLLYDBG v1.10*. OllyDbg v1.10. Retrieved March 17, 2023, from <https://www.ollydbg.de/>