

Progression of Something Awesome Project:

Game Hacking AssaultCube

CSGO

- No prior knowledge of game hacking, but highly interested.
- Started by creating CSGO hacks using Python.
- Created a bhop script and triggerbot for CSGO project.
- No recorded footage of the hacks in action.
- Used pymem documentation and followed this tutorial: [YouTube tutorial link](#).

BHOP:

```
python > bhop.py > bhop
 1 import pymem, keyboard, time
 2 import pymem.process
 3 import win32api
 4
 5 LOCAL_PLAYER = 14596508
 6 FORCE_JUMP = 86756744
 7 HEALTH = 256
 8 FLAGS = 260
 9
10 def bhop() -> None:
11     print("Bhop has launched.")
12     pm = pymem.Pymem('csgo.exe')
13
14     # module address
15     client = pymem.process.module_from_name(pm.process_handle, "client.dll").lpBaseOfDll
16
17     # hack loop
18     while True:
19         time.sleep(0.01)
20
21         # check space bar is not pressed
22         if not win32api.GetAsyncKeyState(0x20):
23             continue
24
25         local_player: int = pm.read_uint(client + LOCAL_PLAYER)
26
27         # check valid player
28         if not local_player:
29             continue
30
31         # is alive
32         if not pm.read_int(local_player + HEALTH):
33             continue
34
35         if pm.read_uint(local_player + FLAGS) & 1 << 0:
36             pm.write_uint(client + FORCE_JUMP, 6)
37             time.sleep(0.01)
38             pm.write_uint(client + FORCE_JUMP, 4)
39
40     if __name__ == '__main__':
41         bhop()
```

TRIGGERBOT:

```
python > trigger.py > ...
 1 import keyboard
 2 import pymem
 3 import pymem.process
 4 import time
 5
 6 dwEntityList = (81793068)
 7 dwForceAttack = (52620952)
 8 dwLocalPlayer = (14596508)
 9 m_iCrosshairID = (71736)
10 m_iTeamNum = (244)
11
12 def main():
13     print("Trigger has launched.")
14     pm = pymem.Pymem('csgo.exe')
15     client = pymem.process.module_from_name(pm.process_handle, "client.dll").lpBaseOfDll
16
17     while True:
18         localPlayer = pm.read_int(client + dwLocalPlayer)
19         crosshairID = pm.read_int(localPlayer + m_iCrosshairID)
20         getTeam = pm.read_int(client + dwEntityList + (crosshairID - 1) * 0x10)
21         localTeam = pm.read_int(localPlayer + m_iTeamNum)
22         crosshairTeam = pm.read_int(getTeam + m_iTeamNum)
23
24         if crosshairID > 0 and crosshairID < 32 and localTeam != crosshairTeam:
25             pm.write_int(client + dwForceAttack, 6)
26
27         if keyboard.is_pressed('x'):
28             time.sleep(2)
29
30         time.sleep(0.1)
31
32     if __name__ == '__main__':
33         main()
```

Learning about offsets:

- First encounter with "offsets," known to hold values for game features.
- Located offsets at: [hazedumper GitHub link](#).
- Learned offsets change with game updates or new releases.
- Created a script to fetch current offsets from the GitHub page.

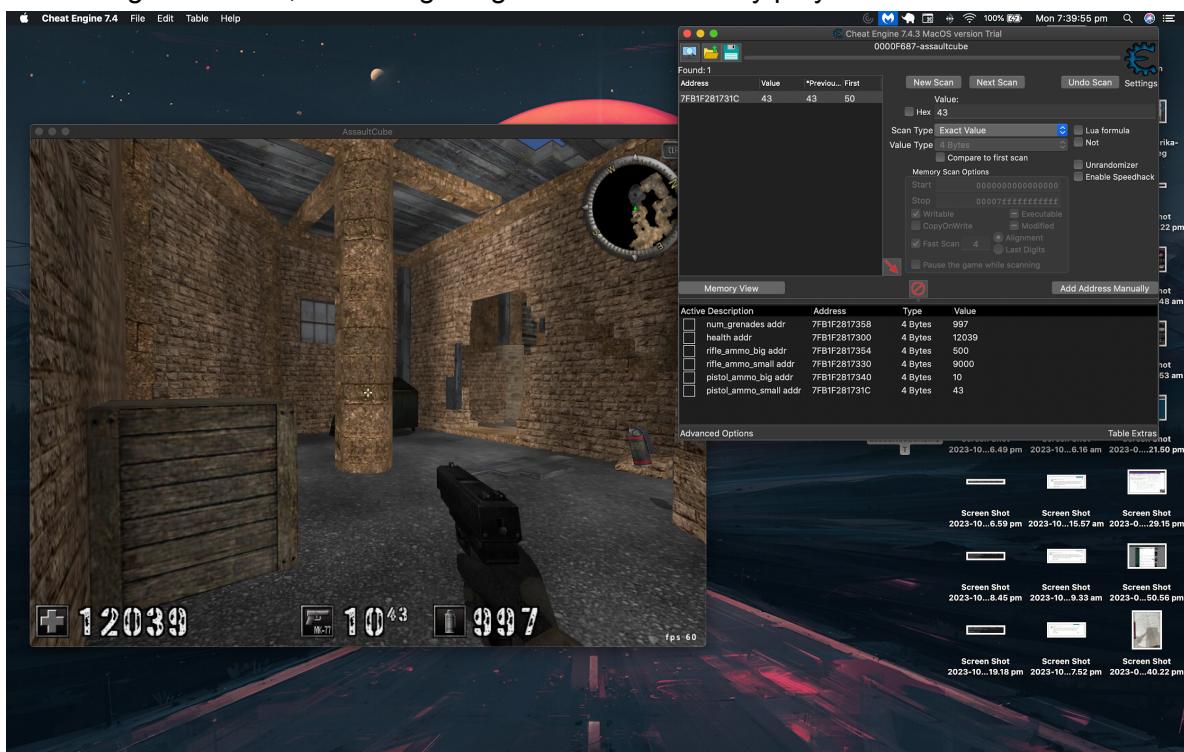
```
python > update_offsets.py > ...
1  import requests
2  import json
3
4  def write_items(dictionary, file):
5      for k, v in dictionary.items():
6          sigs = f'{k}: {v}'
7          file.write(sigs)
8          file.write('\n')
9
10 def update():
11     r = requests.get("https://github.com/frk1/hazedumper/blob/master/csgo.json").json()
12     rawLines = r["payload"]["blob"]["rawLines"]
13     data_dict = json.loads("".join(rawLines))
14     signatures = data_dict['signatures']
15     netvars = data_dict['netvars']
16     with open("offsets.txt", "w") as f:
17         write_items(signatures, f)
18         write_items(netvars, f)
19         f.close()
20
21 if __name__ == "__main__":
22     update()
```

```
python > offsets.txt
1 anim_overlays: 10640
2 clientstate_choked_commands: 19760
3 clientstate_delta_ticks: 372
4 clientstate_last_outgoing_command: 19756
5 clientstate_net_channel: 156
6 convar_name_hash_table: 197024
7 dwClientState: 5894556
8 dwClientState_GetLocalPlayer: 384
9 dwClientState_IsHLTV: 19784
10 dwClientState_Map: 652
11 dwClientState_MapDirectory: 392
12 dwClientState_MaxPlayer: 904
13 dwClientState_PlayerInfo: 21184
14 dwClientState_State: 264
15 dwClientState_ViewAngles: 19856
16 dwEntityList: 81793068
17 dwForceAttack: 52620952
18 dwForceAttack2: 52620964
19 dwForceBackward: 52621024
20 dwForceForward: 52621012
21 dwForceJump: 86756744
22 dwForceLeft: 52620888
23 dwForceRight: 52620892
24 dwGameDir: 6532688
25 dwGameRulesProxy: 87229844
26 dwGetAllClasses: 14733220
27 dwGlobalVars: 5893728
28 dwGlowObjectManager: 87407312
29 dwInput: 86369792
30 dwInterfaceLinkList: 10080132
31 dwLocalPlayer: 14596508
32 dwMouseEnable: 86221408
33 dwMouseEnablePtr: 86221368
34 dwPlayerResource: 52613584
35 dwRadarBase: 86211332
```

Starting AssaultCube

- CS2 replaced CSGO, not suitable for the project.
- Recommended to switch to AssaultCube by Nick.
- Research showed AssaultCube is good for hacking beginners.
- Installed Cheat Engine and completed the tutorial.
- Overcame difficult sections with help from the video: [Cheat Engine for Idiots](#).

After doing the tutorial, I started getting basic values for my player character in AssaultCube.



Explanation for how to get “dynamic addresses”

- Selected player health (starting at 100) as the value to manipulate.
- Conducted initial scan for addresses with 100 health points, resulting in many addresses.
- Threw a grenade at player character to reduce health and narrow down addresses with subsequent scans.
- Repeated health reduction and scanning to isolate 2-3 possible addresses.
- Added final addresses to a list and tested each by toggling "Active" and adjusting health values.
- Identified the correct address as the one that maintained a static health value when "Active" was enabled.

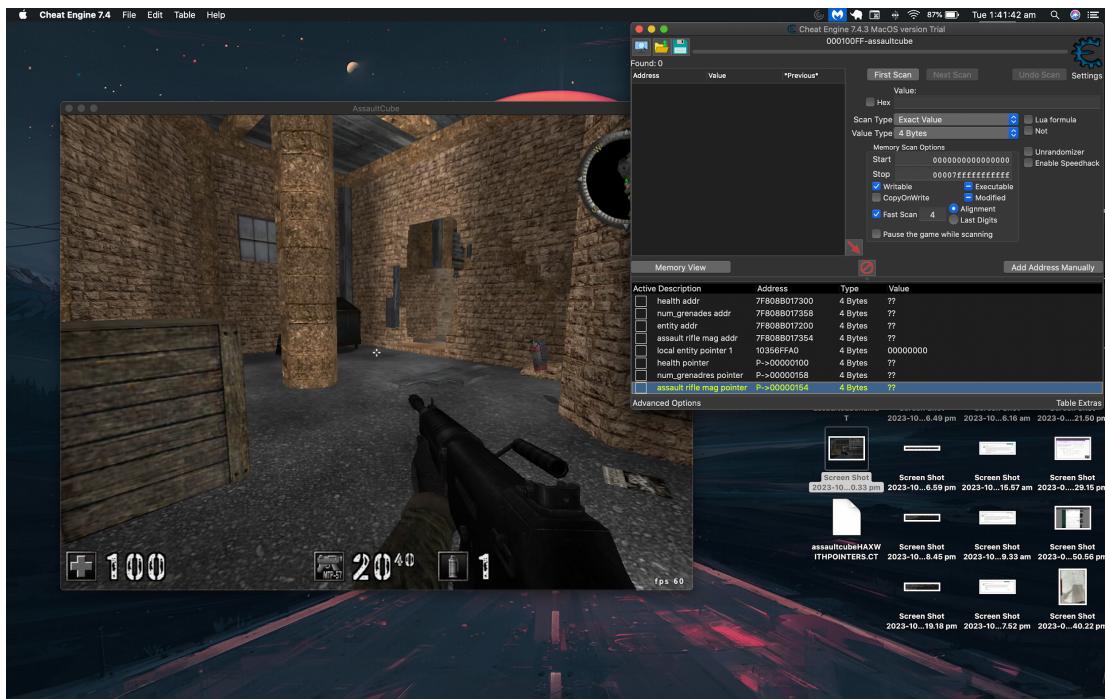
To summarise these steps:

1. Decide on the value I want the address of.
2. Make a new scan of the initial value.
3. Change value within the game.
4. Enter the new value and click “Next scan.”
5. Repeat 3 and 4 until the list of addresses found is around 2-3.
6. Add addresses to the address list.
7. Test each address.
 - a. Tick “Active” for the address.
 - b. Try to change the value within the game.
 - c. If the value does not change, use the address.
8. Delete other addresses.

These steps were repeated to find all values.

Learning how to get “static base addresses”

Unfortunately, I found that everytime I restarted the game, I would have to redo the whole process of finding the addresses of the values as the addresses change every time the game restarts.



- Values are deallocated and reallocated. I termed these as "dynamic addresses."
- Found forum post: [Cheat Engine Forum Link](#).
- Learned about "static addresses" from YouTube tutorials on Cheat Engine for AssaultCube.
- Static addresses needed for consistent base address pointers across game resets.
- Used a tutorial for finding static addresses: [YouTube Tutorial](#).

Using pointer scan to get static addresses

To summarise these steps:

1. Find the “dynamic address” like before.
2. Perform a pointer scan on the address (initial scans will give anywhere from 1000s-100,000s static pointers).
3. Close the game and keep the pointer scanner window open.
4. Reopen the game and attach the game to Cheat Engine.
5. Repeat step 1.
6. Get the address of this dynamic address.
7. In the pointer scanner window, click on ‘Pointer scanner’ then ‘Rescan memory...’
8. Enter the address from step 5.
9. Repeat steps 1-8 until satisfied.
10. Select the static pointers at random to be added to the address list.
11. Repeat steps 3 and 4 (Some of these static pointers will be lost when the game restarts).

12. Keep the static pointers that still work.

13. Choose any and remove the excess static pointers.

I did this for a couple of values such as health, armour and ammo and grenades. Though, I don't know what all the extra files are for as I only performed 2-4 rescans on each value.

<input type="checkbox"/> 1.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 3:46 PM	5 File	119 KB	<input type="checkbox"/> nadesp1.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	PTR File	3 KB
<input type="checkbox"/> 1.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 3:46 PM	6 File	99 KB	<input checked="" type="checkbox"/> nadesp1.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	zip	90 KB
<input type="checkbox"/> 1.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 3:46 PM	7 File	116 KB	<input type="checkbox"/> nadesp1.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	1 File	55 KB
<input type="checkbox"/> 1.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 3:46 PM	8 File	123 KB	<input type="checkbox"/> nadesp1.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	2 File	97 KB
<input type="checkbox"/> 2.PTR	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	PTR File	3 KB	<input type="checkbox"/> nadesp1.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	3 File	88 KB
<input checked="" type="checkbox"/> 2.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	zip	2 KB	<input type="checkbox"/> nadesp1.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	4 File	60 KB
<input type="checkbox"/> 2.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	1 File	2 KB	<input type="checkbox"/> nadesp1.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	5 File	76 KB
<input type="checkbox"/> 2.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	2 File	2 KB	<input type="checkbox"/> nadesp1.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	6 File	81 KB
<input type="checkbox"/> 2.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	3 File	2 KB	<input type="checkbox"/> nadesp1.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	7 File	82 KB
<input type="checkbox"/> 2.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	4 File	1 KB	<input type="checkbox"/> nadesp1.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	8 File	78 KB
<input type="checkbox"/> 2.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	5 File	1 KB	<input type="checkbox"/> nadesp1.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	PTR File	3 KB
<input type="checkbox"/> 2.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	6 File	2 KB	<input checked="" type="checkbox"/> nadesp1.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	zip	3 KB
<input type="checkbox"/> 2.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	7 File	2 KB	<input type="checkbox"/> nadesp1.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	1 File	1 KB
<input type="checkbox"/> 2.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 3:50 PM	8 File	5 KB	<input type="checkbox"/> nadesp1.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	2 File	4 KB
<input type="checkbox"/> armourptr.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	PTR File	3 KB	<input type="checkbox"/> nadesp1.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	3 File	2 KB
<input checked="" type="checkbox"/> armourptr.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	zip	129 KB	<input type="checkbox"/> nadesp1.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	4 File	3 KB
<input type="checkbox"/> armourptr.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	1 File	98 KB	<input type="checkbox"/> nadesp1.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	5 File	4 KB
<input type="checkbox"/> armourptr.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	2 File	93 KB	<input type="checkbox"/> nadesp1.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	6 File	2 KB
<input type="checkbox"/> armourptr.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	3 File	120 KB	<input type="checkbox"/> nadesp1.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	7 File	2 KB
<input type="checkbox"/> armourptr.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	4 File	148 KB	<input type="checkbox"/> nadesp1.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 4:09 PM	8 File	3 KB
<input type="checkbox"/> armourptr.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	5 File	96 KB	<input checked="" type="checkbox"/> pistolptr.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	PTR File	3 KB
<input type="checkbox"/> armourptr.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	6 File	107 KB	<input checked="" type="checkbox"/> pistolptr.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	zip	398 KB
<input type="checkbox"/> armourptr.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	7 File	80 KB	<input checked="" type="checkbox"/> pistolptr.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	1 File	292 KB
<input type="checkbox"/> armourptr.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 4:23 PM	8 File	72 KB	<input type="checkbox"/> pistolptr.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	2 File	186 KB
<input type="checkbox"/> armourptr2.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	PTR File	3 KB	<input type="checkbox"/> pistolptr.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	3 File	239 KB
<input checked="" type="checkbox"/> armourptr2.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	zip	2 KB	<input type="checkbox"/> pistolptr.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	4 File	245 KB
<input type="checkbox"/> armourptr2.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	1 File	3 KB	<input type="checkbox"/> pistolptr.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	5 File	161 KB
<input type="checkbox"/> armourptr2.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	2 File	5 KB	<input type="checkbox"/> pistolptr.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	6 File	258 KB
<input type="checkbox"/> armourptr2.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	3 File	3 KB	<input type="checkbox"/> pistolptr.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	7 File	336 KB
<input type="checkbox"/> armourptr2.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	4 File	3 KB	<input type="checkbox"/> pistolptr.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 4:15 PM	8 File	323 KB
<input type="checkbox"/> armourptr2.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	5 File	3 KB	<input type="checkbox"/> pistolptr.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	PTR File	3 KB
<input type="checkbox"/> armourptr2.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	6 File	2 KB	<input checked="" type="checkbox"/> pistolptr2.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	zip	0 KB
<input type="checkbox"/> armourptr2.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	7 File	4 KB	<input type="checkbox"/> pistolptr2.PTR.results.SA3344	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	SA3344 File	1 KB
<input type="checkbox"/> armourptr2.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 4:25 PM	8 File	2 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F0E4	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F0E4 File	1 KB
<input type="checkbox"/> grav1.PTR	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	PTR File	3 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F1F0	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F1F0 File	1 KB
<input checked="" type="checkbox"/> grav1.PTR.results.0	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	zip	5,060 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F5AB	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F5AB File	1 KB
<input type="checkbox"/> grav1.PTR.results.1	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	1 File	5,483 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F5B0	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F5B0 File	1 KB
<input type="checkbox"/> grav1.PTR.results.2	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	2 File	3,839 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F5DC	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F5DC File	1 KB
<input type="checkbox"/> grav1.PTR.results.3	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	3 File	3,040 KB	<input type="checkbox"/> pistolptr2.PTR.results.19F6C8	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F6C8 File	1 KB
<input type="checkbox"/> grav1.PTR.results.4	<input checked="" type="checkbox"/>	17/10/2023 6:08 PM	4 File	2,717 KB	<input checked="" type="checkbox"/> nictnltr2.PTR.results.19F988	<input checked="" type="checkbox"/>	10/10/2023 4:17 PM	19F988 File	1 KB
<input type="checkbox"/> healthptr.PTR	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	PTR File	3 KB					
<input checked="" type="checkbox"/> healthptr.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	zip	83 KB					
<input type="checkbox"/> healthptr.PTR.results.1	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	1 File	89 KB					
<input type="checkbox"/> healthptr.PTR.results.2	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	2 File	124 KB					
<input type="checkbox"/> healthptr.PTR.results.3	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	3 File	81 KB					
<input type="checkbox"/> healthptr.PTR.results.4	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	4 File	93 KB					
<input type="checkbox"/> healthptr.PTR.results.5	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	5 File	98 KB					
<input type="checkbox"/> healthptr.PTR.results.6	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	6 File	80 KB					
<input type="checkbox"/> healthptr.PTR.results.7	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	7 File	83 KB					
<input type="checkbox"/> healthptr.PTR.results.8	<input checked="" type="checkbox"/>	10/10/2023 3:58 PM	8 File	91 KB					
<input type="checkbox"/> healthptr2.PTR	<input checked="" type="checkbox"/>	10/10/2023 4:01 PM	PTR File	3 KB					
<input checked="" type="checkbox"/> healthptr2.PTR.results.0	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	SA3344 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F1F0	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F1F0 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F200	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F200 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F2F0	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F2F0 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F300	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F300 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F3E8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F3E8 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F480	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F480 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F5A8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F5A8 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F5B8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F5B8 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F5CC	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F5CC File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F5D0	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F5D0 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F5D8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F5D8 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F684	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F684 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F6E8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F6E8 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F6E6C	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F6E6C File	2 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C2	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C2 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C3	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C3 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C7	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C7 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C4	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C4 File	1 KB					
<input type="checkbox"/> healthptr2.PTR.results.19F7C8	<input checked="" type="checkbox"/>	10/10/2023 4:07 PM	19F7C8 File	1 KB					

Creating a player entity with Cheat Engine's structure feature

- Found manual value collection for player position and perspective tedious.
- Decided to develop aimbot and ESP wallhacks.
- Used tutorial: [YouTube tutorial link](#).
- Tutorial helped in creating an entity object for the player character.

- Entity struct estimated all associated values for the player character.

To summarise the steps:

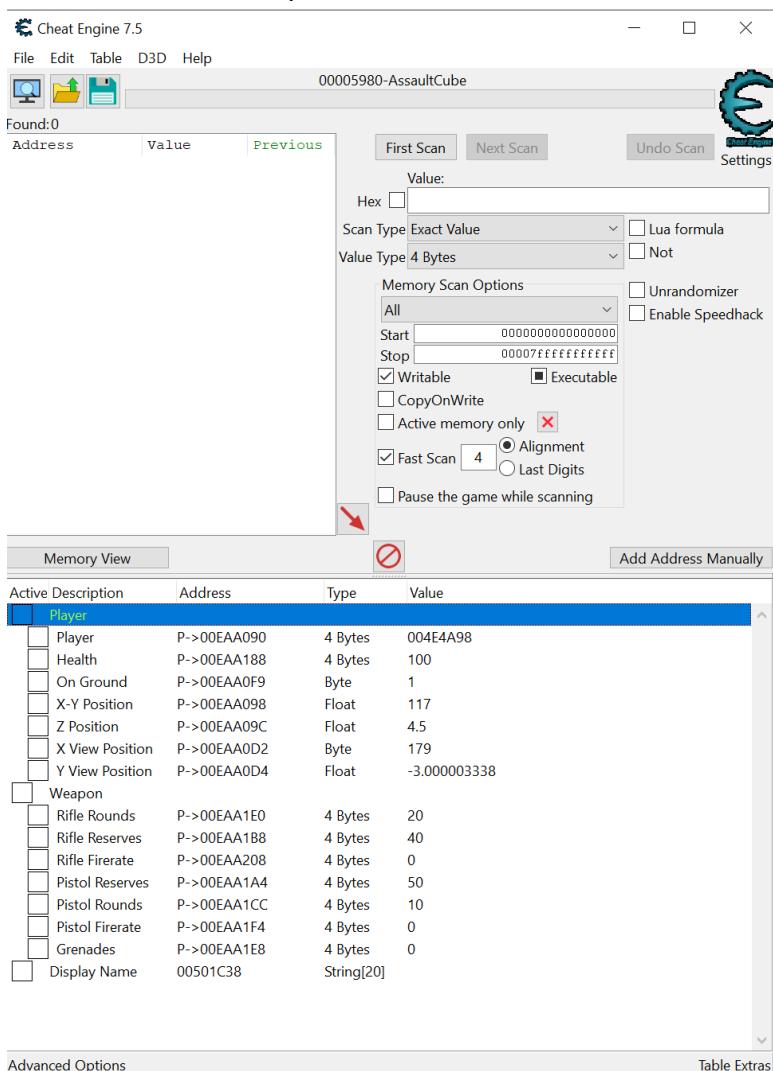
1. Find the dynamic address of a value.
2. Right-click that address and click on “Check what accesses this address” which should open a debugger window.
3. Perform the action that changes the value of the address. E.g. if looking for health, reduce/increase health value.
4. A new entry should be made to the debugger window.
5. Click on this entry and click on “More information.”
6. Save the address mentioned as the likely address.
7. Return to Cheat Engine and click on “New scan.”
8. Tick “Hex” and enter the saved address.
9. Choose any of the green static addresses in the list of addresses to add to the address list (these are the base addresses/pointers).
10. Double-click on the address and tick the “pointer” box.
11. Enter the ‘address’ that appears in the address list into the pointer box and save.
12. Click on the address and press “ctrl+b” to open the memory viewer then “ctrl+d” to open the structure dissect window.
13. In the structure dissect window, click on “structures” then “define new structure.”
14. Name the new structure.

This was the result:

Offset-description	Address: Value
0040 - Byte	EAA0D0 : 163
0041 - Byte	EAA0D1 : 139
0042 - Byte	EAA0D2 : 179
0043 - Byte	EAA0D3 : 67
0044 - Float	EAA0D4 : -3.000000
0048 - 4 Bytes	EAA0D8 : 0
004C - 4 Bytes	EAA0DC : 00000000
0050 - Float	EAA0E0 : 16
0054 - Double	EAA0E4 : 0.014062
005C - Float	EAA0EC : 4.5
0060 - Float	EAA0F0 : 4.5
0064 - String	EAA0F4 : 333?
0069 - Byte	EAA0F9 : 1
006A - Byte	EAA0FA : 0
006B - Byte	EAA0FB : 0
006C - Pointer	EAA0FC : P->01000
0070 - 4 Bytes	EAA100 : 1
0074 - 4 Bytes	EAA104 : 0
0078 - 4 Bytes	EAA108 : 0
007C - 4 Bytes	EAA10C : 0
0080 - 4 Bytes	EAA110 : 0
0084 - 4 Bytes	EAA114 : 0
0088 - 4 Bytes	EAA118 : 0
008C - 4 Bytes	EAA11C : 0
0090 - 4 Bytes	EAA120 : 0
0094 - 4 Bytes	EAA124 : 0
0098 - 4 Bytes	EAA128 : 0
009C - 4 Bytes	EAA12C : 0
00A0 - Float	EAA130 : 100
00A4 - 4 Bytes	EAA134 : 0
00A8 - 4 Bytes	EAA138 : 0
00AC - 4 Bytes	EAA13C : 0
00B0 - 4 Bytes	EAA140 : 0
00B4 - Float	EAA144 : 100
00B8 - 4 Bytes	EAA148 : 0
00BC - 4 Bytes	EAA14C : 0
00C0 - 4 Bytes	EAA150 : 0
00C4 - 4 Bytes	EAA154 : 0
00C8 - Float	EAA158 : 100
00CC - 4 Bytes	EAA15C : 0
00D0 - 4 Bytes	EAA160 : 0
00D4 - 4 Bytes	EAA164 : 0
00D8 - 4 Bytes	EAA168 : 0
00DC - Float	EAA16C : 100
00E0 - 4 Bytes	EAA170 : 42949672
00E4 - 4 Bytes	EAA174 : 42949672
00E8 - 4 Bytes	EAA178 : 0
00EC - 4 Bytes	EAA17C : 0
00F0 - 4 Bytes	EAA180 : 0
00F4 - Pointer	EAA184 : P->004E4
00F8 - 4 Bytes	EAA188 : 100

- Identified F8 as the health offset. I confirmed with the in-game health value of 100.
- Collected numerous useful values during gameplay.
- Added values to the address list as pointers with base address “0x509b74” and corresponding offsets.

This is what I ended up with:

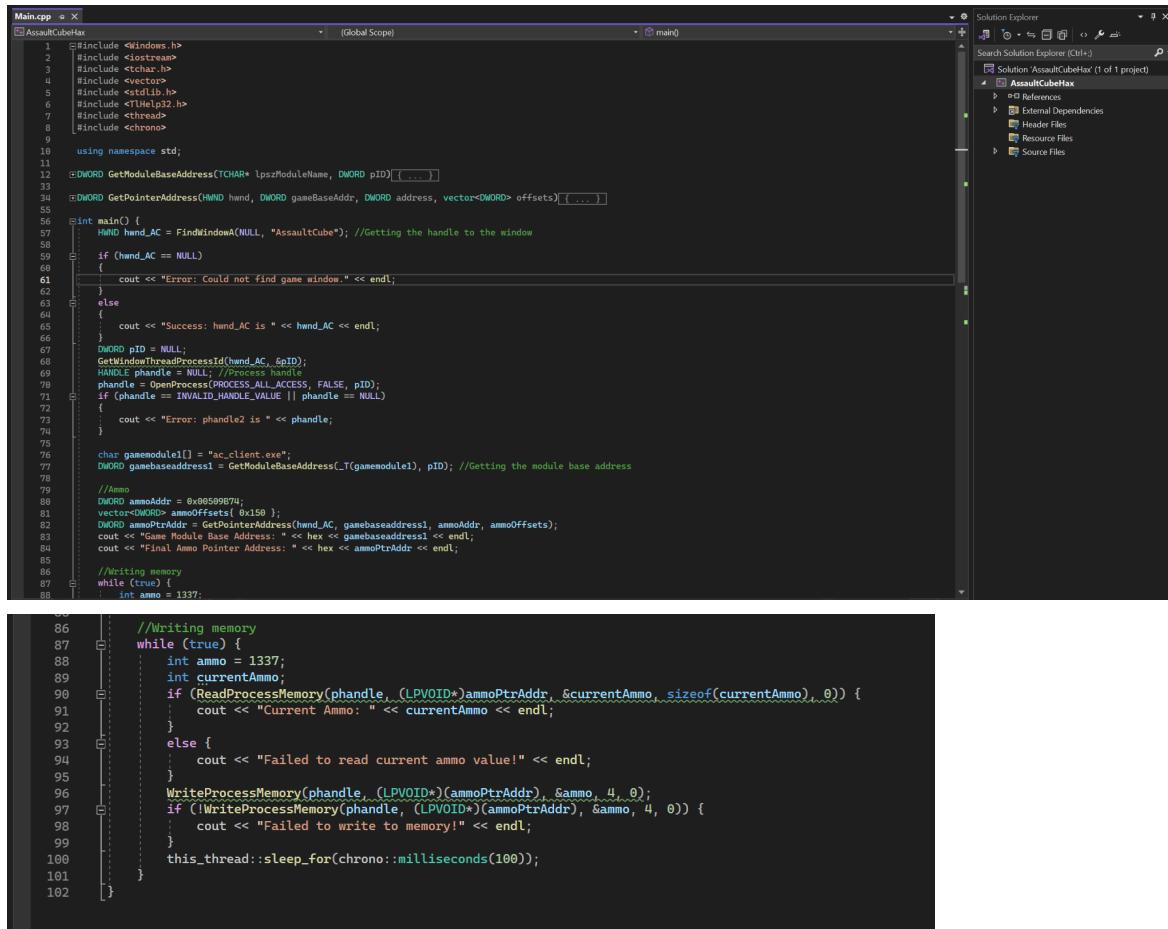


Writing the scripts

- Decided to write scripts for basic hacks.
- Chose C++ on Visual Studio following tutorials and recommendations over Visual Studio Code.
- Researched DLL injections and C++ documentation.
- Downloaded Extreme Injector v3.7 for DLL injection tasks.

C++ Scripts

- Created a c++ project in Visual Studio.
- Struggled to start so followed this tutorial: [Youtube tutorial link](#)
- Gained understanding of C++ modules: windows.h and tlhelp32.h.
- Applied this knowledge to Python programs.
- Offsets in the tutorial differed due to game version variations.
- Tutorial used multi-level pointers. My personal code had one base address "0x509b74" and a single offset.
- Adapted and debugged code without success.
- Attempted code injection with Extreme Injector. I faced an issue due to C++ file not being a dll.
- Unaware that dll files could be written in C++ at the time.
- Progress was made despite challenges.



The screenshot shows the Visual Studio IDE interface. The main window displays the `Main.cpp` file for the `AssaultCubeHax` project. The code is written in C++ and performs the following tasks:

- Includes necessary headers: windows.h, iostream, cchar, tchar, windows.h, stdlib.h, tlib.h, tlib32.h, thread, chrono.
- Defines a function `GetModuleBaseAddress` that takes a module name and a PID, and returns the base address.
- Defines a function `GetPointerAddress` that takes a handle, base address, and a vector of offsets to find a specific pointer.
- In the `main` function:
 - Gets the handle to the game window using `FindWindowA`.
 - If the handle is null, it prints an error message.
 - Creates a process handle using `OpenProcess` with `PROCESS_ALL_ACCESS`.
 - If the handle is invalid or null, it prints an error message.
 - Gets the module base address of "ac_client.exe" using `GetModuleBaseAddress`.
 - Creates a memory block at `0x00509b74` and writes the value `1337` to it.
 - Reads the current ammo value from memory at the same address.
 - If reading fails, it prints an error message.
 - Writes the value `1337` back to the memory address.
 - If writing fails, it prints an error message.
 - Pauses the thread for 100 milliseconds.

Python Scripts

- Decided on a solution in Python for game hacking.
- Revisited previous CSGO Python scripts for reference.
- Successfully created first Python script for infinite health and ammo hacks.

INFINITE HEALTH AND AMMO

```
Python > ACstatshack.py > {} time

1  import pymem
2  import time
3
4  LOCAL_PLAYER = 0x00509b74
5  HEALTH = 0xf8
6  AMMO = 0x150
7
8  def infinitehealthammo():
9      print("HAX has launched.")
10     pm = pymem.Pymem('ac_client.exe')
11
12
13     while True:
14         time.sleep(0.01)
15
16         local_player: int = pm.read_uint(LOCAL_PLAYER)
17
18         health_address = local_player + HEALTH
19         pm.write_int(health_address, 420)
20
21         ammo_address = local_player + AMMO
22         pm.write_int(ammo_address, 420)
23
24     if __name__ == "__main__":
25         infinitehealthammo()
```

Video for Infinite health and ammo demo

- Modified the BHOP script from CSGO for use with AssaultCube.
- Adjusted script by updating offsets.
- Removed unnecessary code to obtain the base client address, already known as 0x4.

BHOP

```
python > ACbhop.py > ...
1  import pymem, keyboard, time
2  import win32api
3
4  LOCAL_PLAYER = 0x509b74
5  FORCE_JUMP = 0x6B
6  HEALTH = 0xF8
7  ON_GROUND = 0x69
8
9  def bhop() -> None:
10    print('Bhop has launched.')
11    pm = pymem.Pymem('ac_client.exe')
12
13    # hack loop
14    while True:
15        time.sleep(0.01)
16
17        # space bar
18        if not win32api.GetAsyncKeyState(0x20):
19            continue
20
21        local_player: int = pm.read_uint(LOCAL_PLAYER)
22
23        if not local_player:
24            continue
25
26        # is alive
27        if not pm.read_int(local_player + HEALTH):
28            continue
29
30        # on ground
31        if (pm.read_uint(local_player + ON_GROUND) == 1):
32            pm.write_uint(local_player + FORCE_JUMP, 1)
33
34
35    if __name__ == '__main__':
36        bhop()
```

Video for Bhop demo

- Also found FORCE_JUMP offset from this forum: [Forum link](#)
- Created a rapid fire hack that increased the rate of fire of weapons.

RAPID FIRE

```
python > ACRapidfire.py > rapidfire

1 import pymem, keyboard, time
2 import win32api
3
4 LOCAL_PLAYER = 0x509b74
5 HEALTH = 0xF8
6 RIFLE_FIRERATE = 0x178
7 PISTOLE_FIRERATE = 0x164
8
9 def rapidfire() -> None:
10     print("Rapid fire has been launched.")
11     pm = pymem.Pymem('ac_client.exe')
12
13     while True:
14         time.sleep(0.01)
15         # left-click
16         if not win32api.GetAsyncKeyState(0x01):
17             continue
18
19         local_player: int = pm.read_uint(LOCAL_PLAYER)
20
21         if not local_player:
22             continue
23
24         # is alive
25         if not pm.read_int(local_player + HEALTH):
26             continue
27
28         if pm.read_uint(local_player + RIFLE_FIRERATE) == 120:
29             pm.write_uint(local_player + RIFLE_FIRERATE, 60)
30
31         # Pistol kinda useless though
32         if pm.read_uint(local_player + PISTOLE_FIRERATE) == 160:
33             pm.write_uint(local_player + RIFLE_FIRERATE, 60)
34
35 if __name__ == '__main__':
36     rapidfire()
```

[Video for rapid fire demo](#)

- Attempted to create a no-recoil hack for centred bullet spread.
- Difficulty in detecting bullet spread using Cheat Engine.
- Conducted scans for unknown values, adjusting as shooting started and stopped.
- Initial addresses tested did not affect recoil.
- Observed that only the pitch-axis (y-axis) changed during shooting, not the yaw-axis (x-axis).
- Thought of maintaining a constant y-axis position upon shooting.

NO RECOIL #1

```
python > ACnorecoil.py > norecoil

1 import pymem
2 import time
3 import win32api
4
5 LOCAL_PLAYER = 0x509b74
6 HEALTH = 0xF8
7 RIFLE_FIRERATE = 0x178
8 PISTOLE_FIRERATE = 0x164
9 Y_VIEW = 0x44
10
11 def norecoil() -> None:
12     print("No recoil has been launched.")
13     pm = pymem.Pymem('ac_client.exe')
14
15     while True:
16         time.sleep(0.01)
17         if not win32api.GetAsyncKeyState(0x01):
18             continue
19
20         local_player: int = pm.read_uint(LOCAL_PLAYER)
21
22         if not local_player:
23             continue
24
25         if not pm.read_int(local_player + HEALTH):
26             continue
27
28         if pm.read_uint(local_player + RIFLE_FIRERATE) > 0:
29             pm.write_int(local_player + Y_VIEW, 0)
30
31 if __name__ == '__main__':
32     norecoil()
```

[Video for no recoil demo #1](#)

- Identified issue with no-recoil hack: inability to aim vertically while shooting.
- Thought of a compensation strategy to counteract recoil by adjusting the y-view downwards.
- Aimed for a low-recoil effect, allowing mouse movement during firing.

NO RECOIL #2

```
python > AChnorecoil.py > norecoil
 1  import pymem
 2  import time
 3  import win32api
 4
 5  LOCAL_PLAYER = 0x509b74
 6  HEALTH = 0xF8
 7  RIFLE_FIRERATE = 0x178
 8  PISTOLE_FIRERATE = 0x164
 9  Y_VIEW = 0x44
10
11 def norecoil() -> None:
12     print("No recoil has been launched.")
13     pm = pymem.Pymem('ac_client.exe')
14
15     while True:
16         time.sleep(0.01)
17         if not win32api.GetAsyncKeyState(0x01):
18             continue
19
20         local_player: int = pm.read_uint(LOCAL_PLAYER)
21
22         if not local_player:
23             continue
24
25         if not pm.read_int(local_player + HEALTH):
26             continue
27
28         if pm.read_uint(local_player + RIFLE_FIRERATE) > 0:
29             current_y = pm.read_float(local_player + Y_VIEW)
30             recoil_amount = 0.28
31             new_y = current_y - recoil_amount
32             pm.write_float(local_player + Y_VIEW, new_y)
33
34     if __name__ == '__main__':
35         norecoil()
```

Video for no recoil demo #2

- Attempted to develop a rifle jump hack for higher jumps when shooting at the ground simultaneously.
- Put in minimal effort. Hence, the hack was not completed.

RIFLE JUMP:

```
python > ACRiflejump.py > {} pymem
 1  import pymem, keyboard, time
 2  import win32api
 3
 4  LOCAL_PLAYER = 0x509b74
 5  FORCE_JUMP = 0x6B
 6  HEALTH = 0xF8
 7  ON_GROUND = 0x69
 8  FORCE_ATTACK = 0x224
 9  Y_VIEW = 0x44
10
11 def bhop() -> None:
12     print('Bhop has launched.')
13     pm = pymem.Pymem('ac_client.exe')
14
15     # hack loop
16     while True:
17         time.sleep(0.01)
18
19         # space bar
20         if not win32api.GetAsyncKeyState(0x20):
21             continue
22
23         local_player: int = pm.read_uint(LOCAL_PLAYER)
24
25         if not local_player:
26             continue
27
28         # is alive
29         if not pm.read_int(local_player + HEALTH):
30             continue
31
32         # on ground
33         if win32api.GetAsyncKeyState(0x20) and win32api.GetAsyncKeyState(0x01):
34             pm.write_uint(local_player + FORCE_JUMP, 6)
35             pm.write_uint(local_player + Y_VIEW, -90)
36             # pm.write_uint(local_player + FORCE_ATTACK, 1092831209)
37
38
39 if __name__ == '__main__':
40     bhop()
```

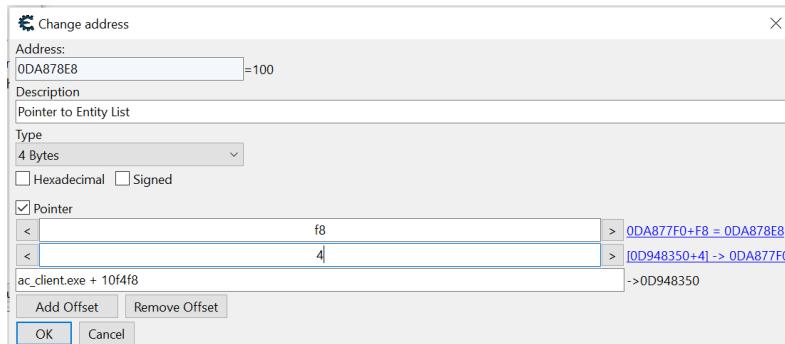
Working on TriggerBot - Big challenge

Getting Entity List

- Aimed to create an AssaultCube triggerbot, drawing from CSGO triggerbot experience.
- Found the task challenging, with difficulty locating the entityList.
- Used a helpful tutorial for help: [YouTube Tutorial on AssaultCube Triggerbot](#).

To summarise these steps:

1. Get the address of one of your player's attributes i.e. name, health etc.
2. Check what accesses the code and look at the disassembler.
3. Search for something that looks like a 'eax, [edi + esi * 4]', which means it's a pointer to an array of entity pointers and pointers in a 32-bit app are 4 bytes.
4. Slightly above it, search for something like 'edi, [ac_client.exe + 10F4F8]' which is likely to be the pointer to the entityList.
5. Add address to address list.
6. Test as done so in the tutorial by adding offsets to test values for different characters.



- Clearly works as the first entity is at offset 0x4 and their health offset is 0xf8.
- Began the script by first working on finding the entity list and how to access the values. I got this:

PRINTING ENTITY LIST

```
1 import pymem
2
3 ENTITY_LIST = 0x10f4f8
4 LOCAL_PLAYER = 0x509b74
5 HEALTH_OFFSET = 0xF8
6 CLIENT = 0x400000
7 ENTITY_SIZE = 0x4
8
9 def main():
10     print("Trigger has been launched")
11     pm = pymem.Pymem("ac_client.exe")
12
13     # entity list base addr
14     entity_list_base = pm.read_int(CLIENT + ENTITY_LIST)
15     print(f"Entity list base address: {hex(entity_list_base)}")
16
17     # 16 for testing rn. need to find way to get entity count
18     # skip first range since always empty
19     for i in range(1, 16):
20         # calculate the address of the entity
21         entity_address = pm.read_int(entity_list_base + i * ENTITY_SIZE)
22         health = pm.read_int(entity_address + HEALTH_OFFSET)
23         print("entity address: ", entity_address)
24         # skip invalid addresses
25         if health == 0 or health == None:
26             continue
27
28         # read the health value of the entity
29         health = pm.read_int(entity_address + HEALTH_OFFSET)
30         print(f"Entity {i} health: {health}")
31
32 if __name__ == "__main__":
33     main()
34
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS
[Running] python -u C:\Users\Oryan\Local\CompuScience\python\AC\triggerbot.py
Trigger has been launched
Entity list base address: 0x2b42280
entity address: 227319832
Entity 1 health: 100
entity address: 45587744
Entity 2 health: 100
entity address: 45568880
Entity 3 health: 100
entity address: 227052216
Entity 4 health: 100
entity address: 45413568

- Initially struggled to understand accessing the entity list.
- Recognised the entity list as a collection of pointers, not direct values.
- Managed to iterate and print the entity list. I encountered various numbers, including 0, -1, 1, and double digits.
- Needed to filter valid from invalid entities.
- Initially attempted to filter entities with addresses above 1000.
- Later used entity attributes (e.g. health) to validate entities.
- Misunderstood player count, expected 8 entities due to 4v4 match but realised only 7 were valid as player was the 8th outside of entity list.
- Final task was refining the triggerbot script.

TRIGGERBOT #1

```
python > ATriggerbot.py > [⑥] FORCE_ATTACK
 5 ENTITY_LIST = 0x10f4f8
 6 LOCAL_PLAYER = 0x509b74
 7 HEALTH_OFFSET = 0xF8
 8 CLIENT = 0x400000
 9 ENTITY_SIZE = 0x4
10 FORCE_ATTACK = 0x224
11 TEAM_OFFSET = 0x32C
12 NAME_OFFSET = 0x225
13 DISPLAY_NAME = 0x101C38
14
15
16 def get_character_entities(pm):
17     # entity list base addr
18     entity_list_base = pm.read_int(CLIENT + ENTITY_LIST)
19
20     entities_data = {}
21     # 16 for testing rn. need to find way to get entity count
22     # skip first range since always empty
23     for i in range(1, 16):
24         try:
25             entity_address = pm.read_int(entity_list_base + i * ENTITY_SIZE)
26             health = pm.read_int(entity_address + HEALTH_OFFSET)
27             team = pm.read_int(entity_address + TEAM_OFFSET)
28             name = pm.read_string(entity_address + NAME_OFFSET)
29             # skip invalid addresses or if health is zero
30             if health == 0 or health == None:
31                 continue
32             entity_data = {'name': name, 'health': health, 'team': team}
33             entities_data[i] = entity_data
34         except pymem.exception.MemoryReadError as e:
35             continue
36     return entities_data
37
38 def get_entity_by_name(entities_info):
39     return {info['name']: entity_num for entity_num, info in entities_info.items() if info['name']}
40
41 def main():
42     print("Trigger has been launched")
43     pm = pymem.Pymem("ac_client.exe")
44     entities = get_character_entities(pm)
45     print(entities)
46     # lookup dict for names to entity numbers
47     entity_lookup_by_name = get_entity_by_name(entities)
48     while True:
49         time.sleep(0.01)
50         local_player: int = pm.read_uint(LOCAL_PLAYER)
51         display_name = CLIENT + DISPLAY_NAME
52         current_name = pm.read_string(display_name)
53
54         # check if current name matches any known entity's name
55         if current_name in entity_lookup_by_name:
56             entity_num = entity_lookup_by_name[current_name]
57             entity_details = entities[entity_num]
58             team = entity_details['team']
59             if team == 0:
60                 pm.write_uint(local_player + FORCE_ATTACK, 1)
61                 # print(pm.write_bytes(LOCAL_PLAYER + FORCE_ATTACK, (1).to_bytes(1, byteorder='little')), 1))
62                 # print(pm.write_uint(LOCAL_PLAYER + FORCE_ATTACK, 1853441793))
63             elif team == 1:
64                 pm.write_uint(local_player + FORCE_ATTACK, 0)
65                 continue
66
67     if __name__ == "__main__":
68         main()
```

[Video of triggerbot demo #1](#)

- Compiled entity stats (health, team, name) into a dictionary with entities as keys.
- Created a 'get_character_entities' function to organise code.
- Utilised display name offset for identifying character names in-game.
- Developed 'get_entity_by_name' function to verify existence of names in the entity dictionary.
- Identified own team as '1' and enemies as '0' through Cheat Engine.
- Implemented logic to shoot at enemy team entities (team '0').
- Encountered issues with character shooting continuously. My first fix attempts commented out.
- Debugged and added code to read 'LOCAL_PLAYER' and check FORCE_ATTACK status.

- Realised shooting continued due to the DISPLAY_NAME address retaining the last name targeted.
- Introduced a prev_name variable to compare with current_name to prevent continuous shooting.
- Adjusted the logic sequence to compare names before shooting.
- Finalised the code with these corrections.

TRIGGERBOT #2

```

prev_name = None
can_shoot = False
while True:
    time.sleep(0.01)
    local_player: int = pm.read_uint(LOCAL_PLAYER)
    player_team = pm.read_int(LOCAL_PLAYER + TEAM_OFFSET)
    current_name = pm.read_string(CLIENT + DISPLAY_NAME)

    print(f"prev_name1: ", prev_name)
    print(f"current_name1: ", current_name)

    # check if current name matches any known entity's name
    if prev_name != current_name:
        prev_name = current_name
        if current_name in entity_lookup_by_name:
            entity_num = entity_lookup_by_name[current_name]
            entity_details = entities[entity_num]
            team = entity_details['team']
            if team != player_team:
                can_shoot = True
            else:
                can_shoot = False
        else:
            can_shoot = False
    else:
        can_shoot = False

    if can_shoot:
        pm.write_uint(local_player + FORCE_ATTACK, 1)
    else:
        pm.write_uint(local_player + FORCE_ATTACK, 0)

    print(f"prev_name2: ", prev_name)
    print(f"current_name2: ", current_name)

```

[Video of triggerbot demo #2](#)

Fixed the display name issue

- I FIXED IT!!!!!!
- Solved issue with display name not updating correctly.
- Wrote null terminators to the DISPLAY_NAME address after each new name detection.
- Allowed DISPLAY_NAME to briefly update for accurate current_name refresh.
- Set up flags to control the shooting mechanism based on name update.

TRIGGERBOT #3

```
python > ATriggerbot.py > main
43     print("Trigger has been launched")
44     pm = pymem.Pymem("ac_client.exe")
45     entities = get_character_entities(pm)
46     print(entities)
47     # lookup dict for names to entity numbers
48     entity_lookup_by_name = get_entity_by_name(entities)
49     can_shoot = False
50     while True:
51         time.sleep(0.01)
52         local_player: int = pm.read_uint(LOCAL_PLAYER)
53         player_team = pm.read_int(LOCAL_PLAYER + TEAM_OFFSET)
54         current_name = pm.read_string(CLIENT + DISPLAY_NAME)
55         pm.write_string(CLIENT + DISPLAY_NAME, "\0" * 20)
56
57         # check if current name matches any known entity's name
58         if current_name in entity_lookup_by_name:
59             entity_num = entity_lookup_by_name[current_name]
60             entity_details = entities[entity_num]
61             team = entity_details['team']
62             if team == player_team:
63                 can_shoot = True
64             else:
65                 can_shoot = False
66             else:
67                 can_shoot = False
68
69
70             if can_shoot:
71                 pm.write_uint(local_player + FORCE_ATTACK, 1)
72             else:
73                 pm.write_uint(local_player + FORCE_ATTACK, 0)
```

[Video of triggerbot demo #3](#)

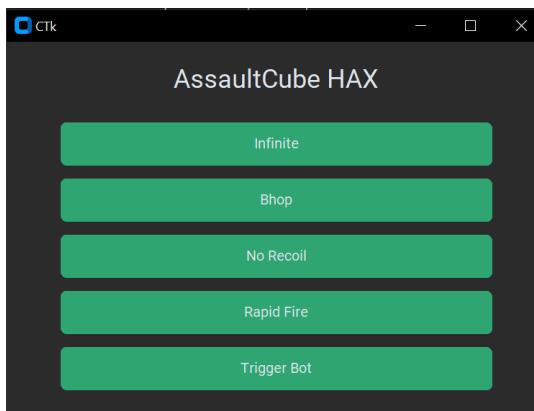
Working on GUI Mod Menu for Cheats

- Decided to create a mod menu GUI for the hacks.
- Found and chose CustomTKinter as a Python GUI tool: [Github CustomTKinter](#)
- Used CustomTKinter documentation and a YouTube tutorial for guidance: [Youtube Tutorial](#)
- Successfully created a mod menu with CustomTKinter.

GUI #1

```
python > ACmain.py > bhop
1 import customtkinter
2 import tkinter
3 from ACinfinitehealthammo import infinitehealthammo
4 from ACbhop import bhop
5 from ACnorecoil import norecoil
6 from ACrapidfire import rapidfire
7 from ACtriggerbot import triggerbot
8
9 customtkinter.set_appearance_mode("dark")
10 customtkinter.set_default_color_theme("green")
11
12 menu = customtkinter.CTk()
13 menu.geometry("500x350")
14
15 frame = customtkinter.CTkFrame(master=menu)
16 frame.pack(fill="both", expand=True)
17
18 label = customtkinter.CTkLabel(master=frame, text="AssaultCube HAX", font=("Roboto", 24))
19 label.pack(pady=20, padx=60)
20
21 infinitebutton = customtkinter.CTkButton(master=frame, text="Infinite", command=infinitehealthammo, width=400, height=40)
22 infinitebutton.pack(pady=6, padx=8)
23
24 bhopbutton = customtkinter.CTkButton(master=frame, text="Bhop", command=bhop, width=400, height=40)
25 bhopbutton.pack(pady=6, padx=8)
26
27 norecoilbutton = customtkinter.CTkButton(master=frame, text="No Recoil", command=norecoil, width=400, height=40)
28 norecoilbutton.pack(pady=6, padx=8)
29
30 rapidfirebutton = customtkinter.CTkButton(master=frame, text="Rapid Fire", command=rapidfire, width=400, height=40)
31 rapidfirebutton.pack(pady=6, padx=8)
32
33 triggerbotbutton = customtkinter.CTkButton(master=frame, text="Trigger Bot", command=triggerbot, width=400, height=40)
34 triggerbotbutton.pack(pady=6, padx=8)
35
36 menu.mainloop()
```

and the GUI menu looks like this:



Now when I run a hack with the menu, it does work like so:

[Video for GUI demo #1](#)

Issue with menu crashing after running one hack due to threading

- Experienced crashes in the mod menu after running a hack.
- Menu requires restarting or game closure to function again post-crash.
- Online search yielded no relevant solutions.
- Sought advice from ChatGPT on potential causes for the issue.

BR

I have created a mod menu GUI with some cheats in python for a game. When I run the cheats for the game through the mod menu, they work initially but then the mod menu stops responding. Why is that?

QA

There could be several reasons why your mod menu GUI stops responding after running cheats for a game in Python. Here are some common issues and potential solutions:

1. Resource Consumption:
 - * The cheats or the mod menu itself might consume a significant amount of system resources, such as CPU or memory. This can lead to the GUI becoming unresponsive or sluggish. Make sure your code is optimized and not causing resource spikes.
2. Infinite Loops or Blocking Code:
 - * Check your code for any infinite loops or blocking operations that could prevent the GUI from responding. Long-running operations should be executed in separate threads or processes to keep the GUI responsive.
3. Exception Handling:
 - * Unhandled exceptions can cause your program to crash or become unresponsive. Ensure that you have proper exception handling in place to catch and handle any errors that may occur during the execution of your cheats.
4. Memory Leaks:
 - * Memory leaks can accumulate over time and cause your program to slow down or become unresponsive. Use tools like memory profilers to identify and fix any memory leaks in your code.
5. Game Updates:
 - * Sometimes, game updates can break compatibility with mods or cheats. Ensure that your cheats are up to date and compatible with the latest version of the game.

6. External Dependencies:

- * If your mod menu relies on external libraries or APIs, ensure that they are functioning correctly and are compatible with your code. Any issues with external dependencies can affect the stability of your mod menu.

7. Thread Synchronization:

- * If you are using multiple threads in your program, make sure you have proper thread synchronization mechanisms in place to avoid race conditions and other threading-related issues.

8. Debugging:

- * Use debugging tools and techniques to identify the specific point in your code where the mod menu becomes unresponsive. This will help you pinpoint the problem and find a solution more effectively.

9. Profiling:

- * Profile your code to identify performance bottlenecks and areas that may be causing the slowdown. Profiling tools can help you optimize your code for better performance.

10. Game Anti-Cheat Measures:

- * Some games have anti-cheat measures that detect and block cheats or mods. If your cheats are being detected by the game's anti-cheat system, it could lead to instability or unresponsiveness. Be cautious when using cheats in games with active anti-cheat systems.

Without specific details about your code and the game you are modifying, it's challenging to provide a more precise solution. However, by investigating these common issues and debugging your code thoroughly, you should be able to identify and address the root cause of the mod menu's unresponsiveness.

Explanation for threading issue

- Identified threading as the probable cause of mod menu crashes.
- Noted that GUI and hacks running on the "main thread" caused update issues.
- Learned that hacks using infinite loops blocked the GUI's main thread.
- Realised the need for separate threads for each hack for concurrent execution.
- Looked through [Real Python's threading tutorial](#) for guidance.
- Implemented threading in the hack scripts to resolve mod menu crashing.

GUI #2

```
python > ACmain.py > start_rapidfire
  8 import threading
  9
10 # threading for responsiveness
11 def start_infinitehealthammo():
12     | threading.Thread(target=infinitehealthammo).start()
13
14 def start_bhop():
15     | threading.Thread(target=bhop).start()
16
17 def start_norecoil():
18     | threading.Thread(target=norecoil).start()
19
20 def start_rapidfire():
21     | threading.Thread(target=rapidfire).start()
22
23 def start_triggerbot():
24     | threading.Thread(target=triggerbot).start()
25
26 customtkinter.set_appearance_mode("dark")
27 customtkinter.set_default_color_theme("green")
28
29 menu = customtkinter.CTk()
30 menu.geometry("500x350")
31
32 frame = customtkinter.CTkFrame(master=menu)
33 frame.pack(fill="both", expand=True)
34
35 label = customtkinter.CTkLabel(master=frame, text="AssaultCube HAX", font=("Roboto", 24))
36 label.pack(pady=20, padx=60)
37
38 infinitebutton = customtkinter.CTkButton(master=frame, text="Infinite", command=start_infinitehealthammo, width=400, height=40)
39 infinitebutton.pack(pady=6, padx=8)
40
41 bhopbutton = customtkinter.CTkButton(master=frame, text="Bhop", command=start_bhop, width=400, height=40)
42 bhopbutton.pack(pady=6, padx=8)
43
44 norecoilbutton = customtkinter.CTkButton(master=frame, text="No Recoil", command=start_norecoil, width=400, height=40)
45 norecoilbutton.pack(pady=6, padx=8)
```

And this is how it worked after implementing threading:

[Video for GUI demo #2](#)

GUI works but triggerbot has a bug

- Threading implementation successful, as multiple hacks are now usable simultaneously.
- Triggerbot bug as it activates on teammates instead of enemies.
- Noted that triggerbot was programmed to target non-teammates.
- Discovered the team offset always showed '1' for the player, regardless of actual team.
- Bots' team values were consistent: '1' for red, '0' for blue.
- Found previous oversight: local player offset 0x509b74 wasn't the sole address identified.
- Located two additional possible addresses: 0x50f4f4 and 0x51e20c. Though, both showed '0' for team value.
- Decided on a new solution using the entity list player count discrepancy.
- In a team match, one team would appear to have one fewer player due to our player not being in the entity list.
- Created a "get_player_team" function to determine the player's team by counting the number of entities per team.
- Assigned the player to the team with fewer entities to correct the player_team.

TRIGGERBOT #4

```
def get_player_team(entities):
    if len(entities) == 0:
        return None
    team_counts = {}
    # dict for num on each team
    for entity_num, entity_details in entities.items():
        team = entity_details['team']
        if team not in team_counts:
            team_counts[team] = 0
        team_counts[team] += 1

    # find team with less members
    min_team = min(team_counts, key=team_counts.get)
    player_team = min_team

    return player_team
```

- This produced more consistent results across various games.

[Video for triggerbot and mod menu demo](#)

Clearly, it works much better and I'm happy with it!