

Crackmes 0xD HackMe

Language: C/C++

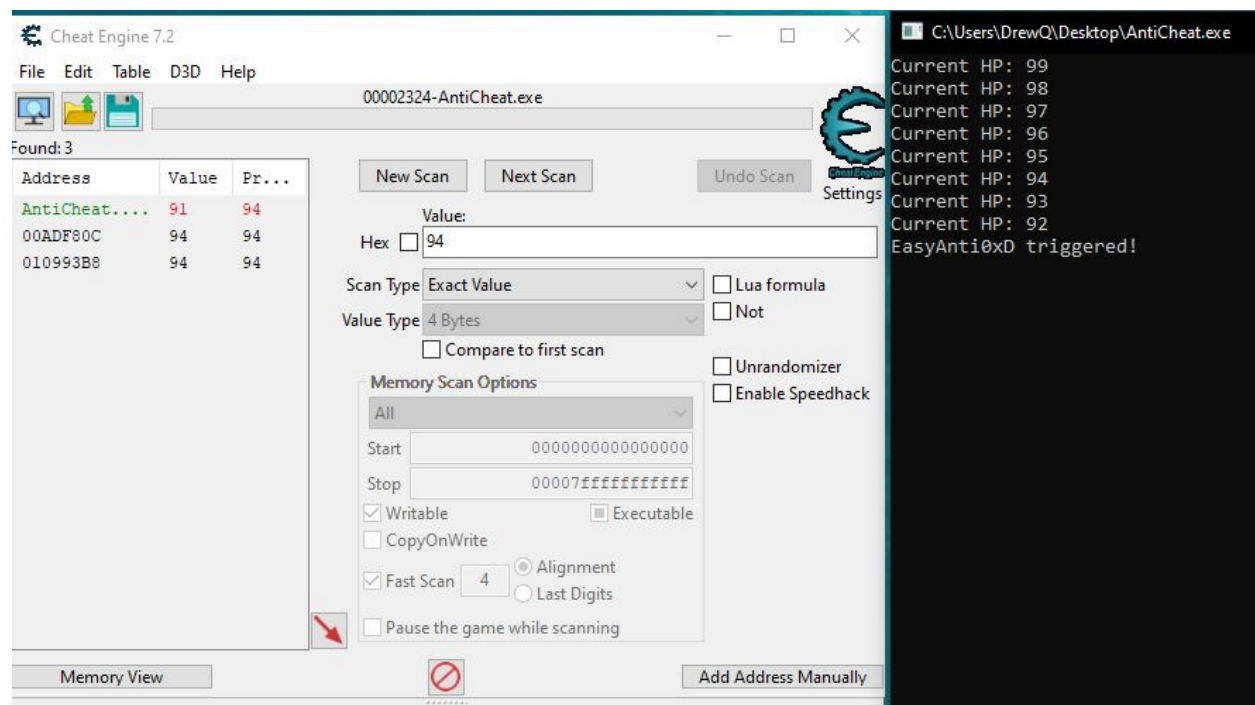
Level: Medium (3)

Description: Attempt to change the HP while it is running, without triggering the AntiCheat.

If you cheat that is your problem otherwise, congratz!

*I hope this is the right place. src: <https://github.com/alxalx14/hackMe/tree/master>

Link: <https://crackmes.one/crackme/5f38acbf33c5d42a7c667d18>



Trying to change the value directly using [Cheat Engine](#) causes the AntiCheat to be triggered.

Analyzing The Binary With Ghidra

Searching for the text "Current HP: " led me to FUN_0041a5d0

```

Decompile: FUN_0041a5d0 - (AntiCheat.exe)
33  puStack12 = &LAB_0042081d;
34  local_10 = *in_FS_OFFSET;
35  iVar2 = 0x3d;
36  puVar3 = (undefined4 *) &stack0xfffffffffc;
37  while (iVar2 != 0) {
38      iVar2 = iVar2 + -1;
39      *puVar3 = 0xffffffff;
40      puVar3 = puVar3 + 1;
41  }
42  uVar1 = DAT_00427010 ^ (uint) &stack0xfffffffffc;
43  *in_FS_OFFSET = (int *) &local_10;
44  local_14 = uVar1;
45  @__CheckForDebuggerJustMyCode@4(&DAT_0042a033);
46  while ((DAT_00427000 < 0x65 && (DAT_00427000 != 0))) {
47      uVar4 = thunk_FUN_00419690();
48      if ((uVar4 & 0xff) == 0) {
49          uVar8 = 0xf0;
50          puVar6 = &DAT_00423db0;
51          __acrt_iob_func(1, &DAT_00423db0, "EasyAnti0xD triggered!", uVar1);
52          uVar5 = __RTC_CheckEsp(extraout_ECX, extraout_EDX);
53          thunk_FUN_0041a550((int) uVar5, puVar6, uVar8);
54          thunk_FUN_00416000();
55          local_8 = 0;
56          thunk_FUN_00413c60(cin_exref, local_34);
57          local_100 = 1;
58          local_8 = 0xffffffff;
59          uVar5 = thunk_FUN_00416bc0();
60          uVar8 = (undefined) ((ulonglong) uVar5 >> 0x20);
61          goto LAB_0041a73f;
62      }
63      thunk_FUN_0041a160(&DAT_00427000);
64      DAT_00427000 = DAT_00427000 + -1;
65      DAT_00427300 = DAT_00427300 + 1;
66      pcVar7 = "Current HP: %d\n";
67      iVar2 = DAT_00427000;
68      __acrt_iob_func(1, "Current HP: %d\n", DAT_00427000, uVar1);
69      uVar8 = (undefined) iVar2;
70      uVar5 = __RTC_CheckEsp(extraout_ECX_00, extraout_EDX_00);
71      thunk_FUN_0041a550((int) uVar5, pcVar7, uVar8);

```

Line 46 looks like the main HP loop. It seems to check to see if DAT_00427000 is less than 100 but not equal to 0. 0x65 is 100 when converted to decimal . We can rewrite this line as:

while(HP < 100 && HP != 0)

Line 47 calls the function FUN_00419690 and stores the result into uVar4. If it returns 0 it prints "EasyAnti0xD triggered!". FUN_00419690 must be the anti-cheat function.

We can patch the Exe so the anti-cheat function never gets called.

0041a639	0f b6 c0	MOVZX	antiCheatResult, antiCheatResult
0041a63c	85 c0	TEST	antiCheatResult, antiCheatResult
0041a63e	74 2b	JZ	LAB_0041a66b

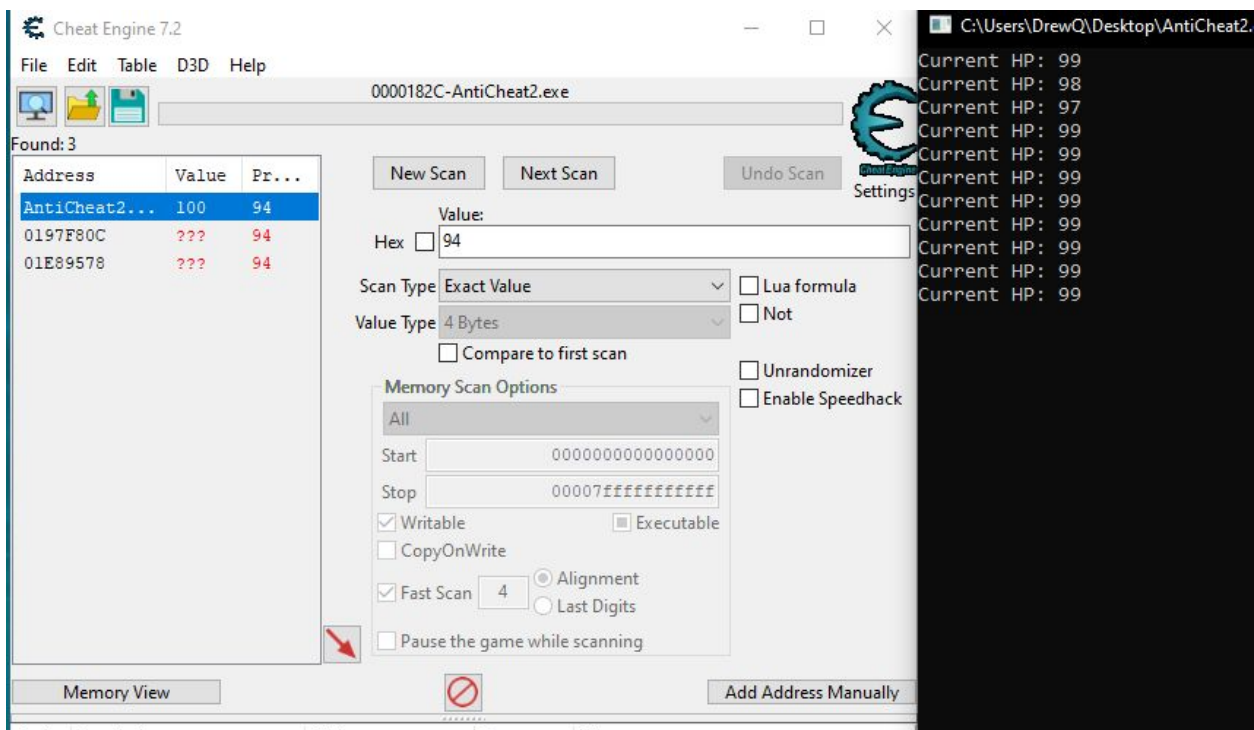
This is the assembly instructions for line 48.

Line 48: if ((antiCheatResult & 0xff) == 0)

If we change the jump instruction to JS instead of JZ everything under the if statement will never be called.

JS means jump if only the previous value is negative. Since the output will never be negative it will never trigger!

```
0041a639 0f b6 c0    MOVZX    antiCheatResult,antiCheatResult
0041a63c 85 c0       TEST     antiCheatResult,antiCheatResult
0041a63e 78 2b       JS       LAB_0041a66b
```



The anti cheat has been avoided!