This file contains the walkthrough for the challenge "Misprotected" part 1 and 2.

> [!WARNING] The binaries referenced in this walkthrough are not the same as those that will be distributed to participants. As a result, the specific addresses and offsets noted here are for reference only and will not align exactly with the binaries provided during the competition.

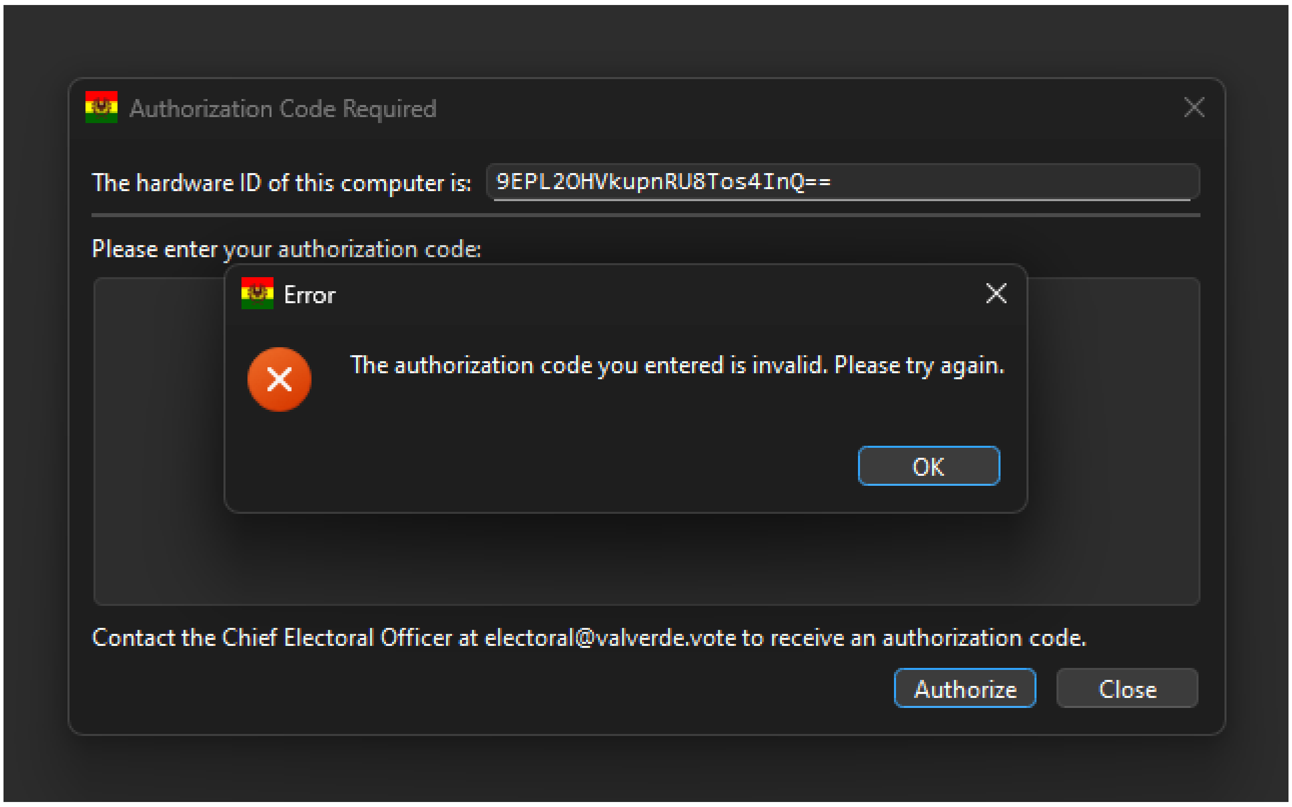
Part 1

This part is deliberately made easy. The code relevant to this part can be analyzed statically since it is not virtualized or mutated.

The "authorization code" mechanism is essentially a serial number verified with VMProtect SDK's `VMProtectGetSerialNumberState` function. VMProtect's obfuscations are most likely too sophisticated to break within the amount of time available, so participants will need to find another way.

Pretty much all of the functions in the `MainWindow` class are protected with VMProtect's `Ultra` configuration, which is virtualization + mutation. They will be too difficult to analyze. However, the serial number validation logic lives unprotected in the authentication dialog's code, allowing easy manipulation of the dialog's return status.

When you enter the wrong authorization code in the dialog, you will see this error message:


Authorization Code Required

The hardware ID of this computer is: 9EPL2OHVkupnRU8Tos4InQ==

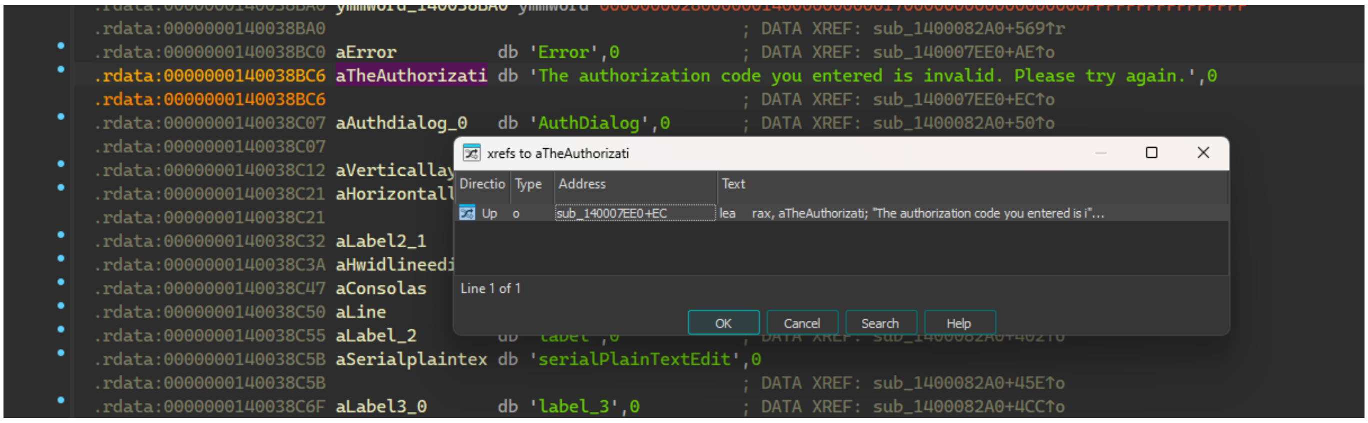
Please enter your authorization code:

Error

The authorization code you entered is invalid. Please try again.

Contact the Chief Electoral Officer at electoral@valverde.vote to receive an authorization code.

Since both the string and the function are unprotected, you should be able to find this string and see what accesses this string in static analysis:



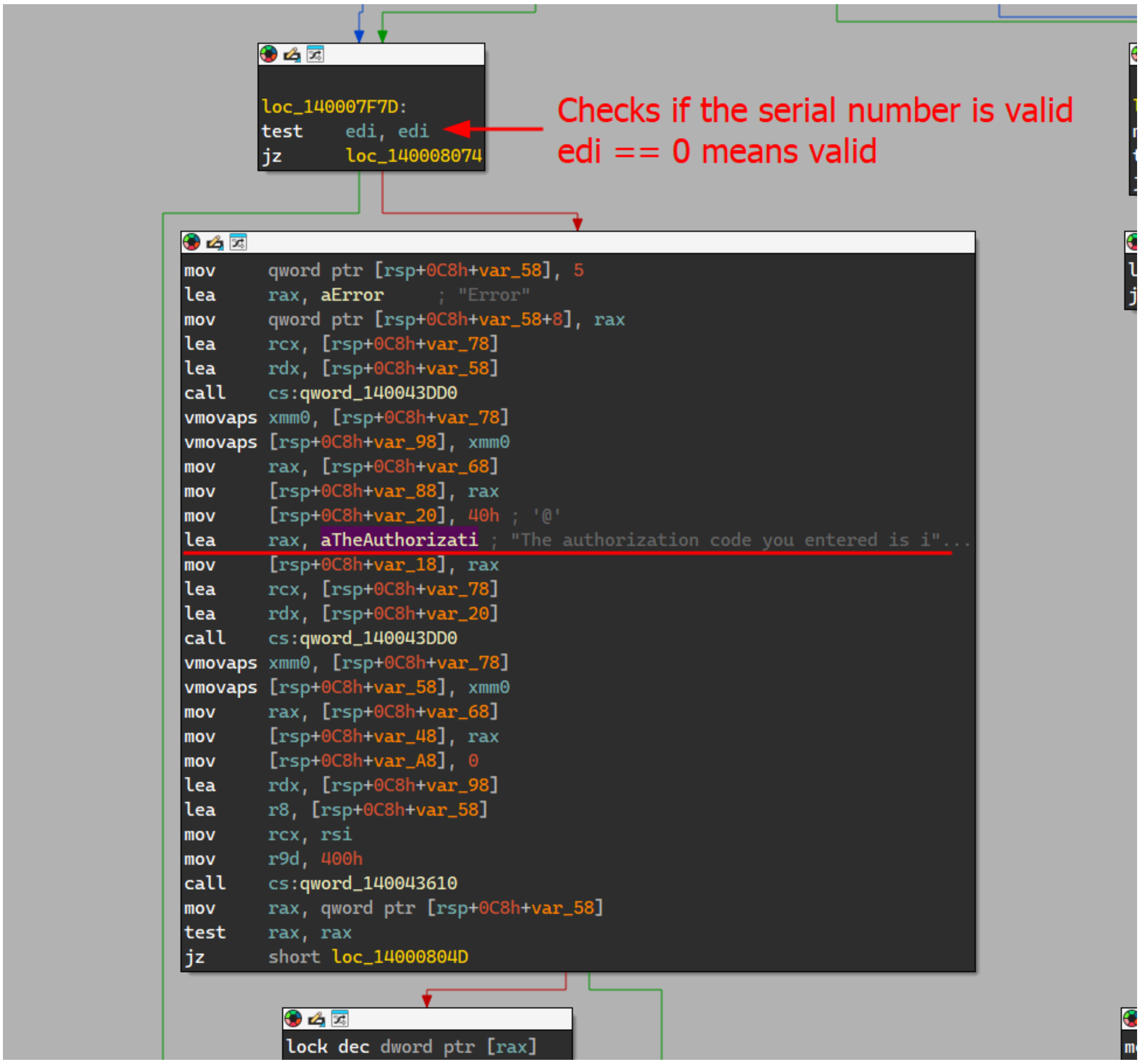
```
.rdata:0000000140038BA0                                          ; DATA XREF: sub_1400082A0+569↑r
.rdata:0000000140038BC0 aError          db 'Error',0             ; DATA XREF: sub_140007EE0+AE↑o
.rdata:0000000140038BC6 aTheAuthorizati db 'The authorization code you entered is invalid. Please try again.',0
.rdata:0000000140038BC6                                          ; DATA XREF: sub_140007EE0+EC↑o
.rdata:0000000140038C07 aAuthdialog_0   db 'AuthDialog',0        ; DATA XREF: sub_1400082A0+50↑o
.rdata:0000000140038C07
.rdata:0000000140038C12 aVerticallay
.rdata:0000000140038C21 aHorizontall
.rdata:0000000140038C21
.rdata:0000000140038C32 aLabel2_1
.rdata:0000000140038C3A aHwidlineedi
.rdata:0000000140038C47 aConsolas
.rdata:0000000140038C50 aLine
.rdata:0000000140038C55 aLabel_2
.rdata:0000000140038C5B aSerialplaintex db 'serialPlainTextEdit',0
.rdata:0000000140038C5B                                          ; DATA XREF: sub_1400082A0+45E↑o
.rdata:0000000140038C6F aLabel3_0       db 'label_3',0           ; DATA XREF: sub_1400082A0+4CC↑o
```

xrefs to aTheAuthorizati

Directio	Type	Address	Text
Up	o	sub_140007EE0+EC	lea rax, aTheAuthorizati; "The authorization code you entered is i"...

Line 1 of 1

We can see that the code checks if `edi` is `0` here. It's pretty easy to tell that if `edi` isn't `0`, it will call some functions and pass that error message as an argument, so we can assume we want `edi` to be `0` here, which means that the serial number is valid.

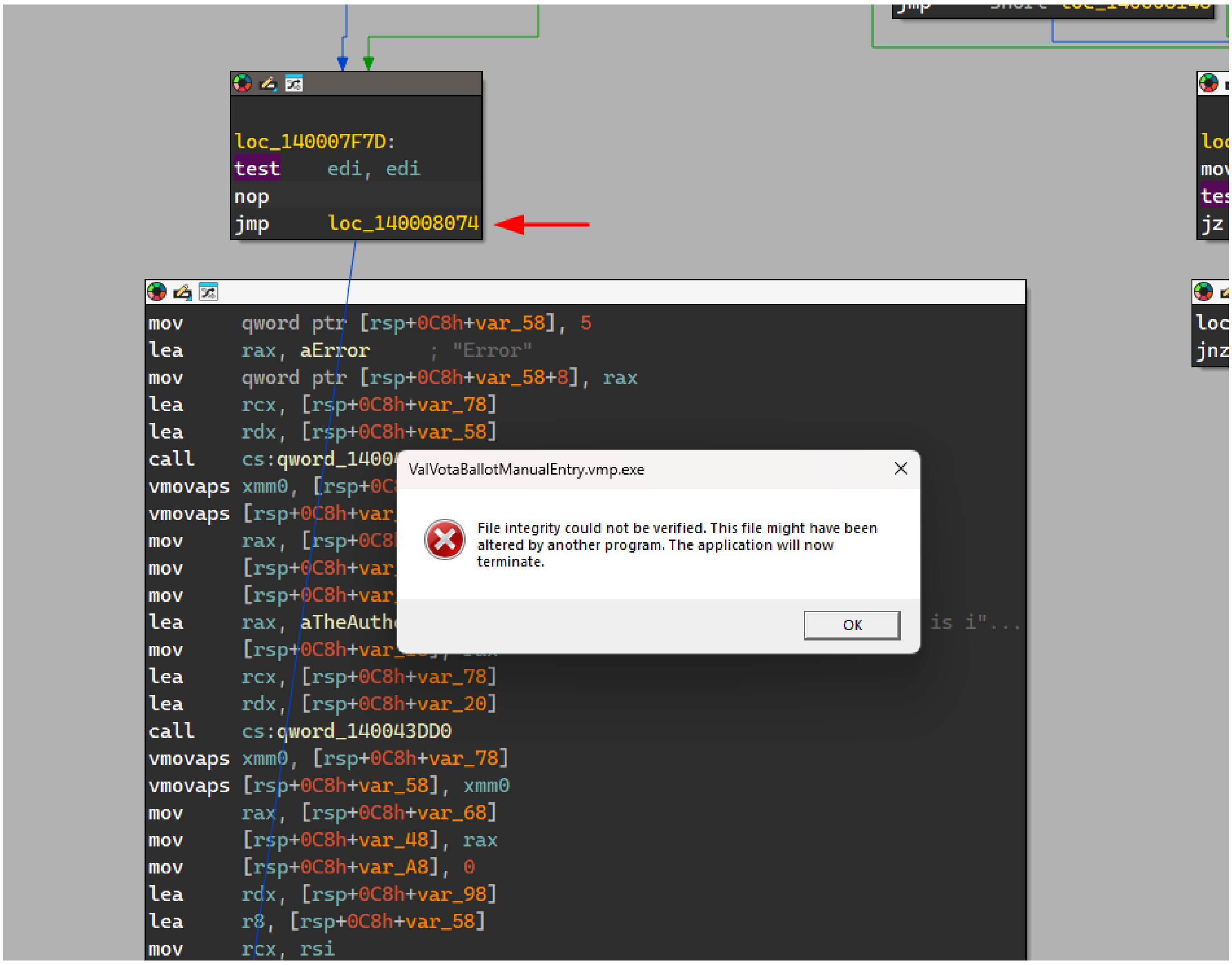


```
loc_140007F7D:
test    edi, edi
jz      loc_140008074
```

```
mov     qword ptr [rsp+0C8h+var_58], 5
lea     rax, aError     ; "Error"
mov     qword ptr [rsp+0C8h+var_58+8], rax
lea     rcx, [rsp+0C8h+var_78]
lea     rdx, [rsp+0C8h+var_58]
call    cs:qword_140043DD0
vmovaps xmm0, [rsp+0C8h+var_78]
vmovaps [rsp+0C8h+var_98], xmm0
mov     rax, [rsp+0C8h+var_68]
mov     [rsp+0C8h+var_88], rax
mov     [rsp+0C8h+var_20], 40h ; '@'
lea     rax, aTheAuthorizati ; "The authorization code you entered is i"...
mov     [rsp+0C8h+var_18], rax
lea     rcx, [rsp+0C8h+var_78]
lea     rdx, [rsp+0C8h+var_20]
call    cs:qword_140043DD0
vmovaps xmm0, [rsp+0C8h+var_78]
vmovaps [rsp+0C8h+var_58], xmm0
mov     rax, [rsp+0C8h+var_68]
mov     [rsp+0C8h+var_48], rax
mov     [rsp+0C8h+var_A8], 0
lea     rdx, [rsp+0C8h+var_98]
lea     r8, [rsp+0C8h+var_58]
mov     rcx, rsi
mov     r9d, 400h
call    cs:qword_140043610
mov     rax, qword ptr [rsp+0C8h+var_58]
test    rax, rax
jz      short loc_14000804D
```

```
lock dec dword ptr [rax]
```

If you try to patch this jump in the binary on the disk, the program will refuse to launch because VMProtect's `Mmeory Protection` feature checks the integrity of the file before passing the control to the original program:



```
loc_140007F7D:
test    edi, edi
nop
jmp     loc_140008074
```

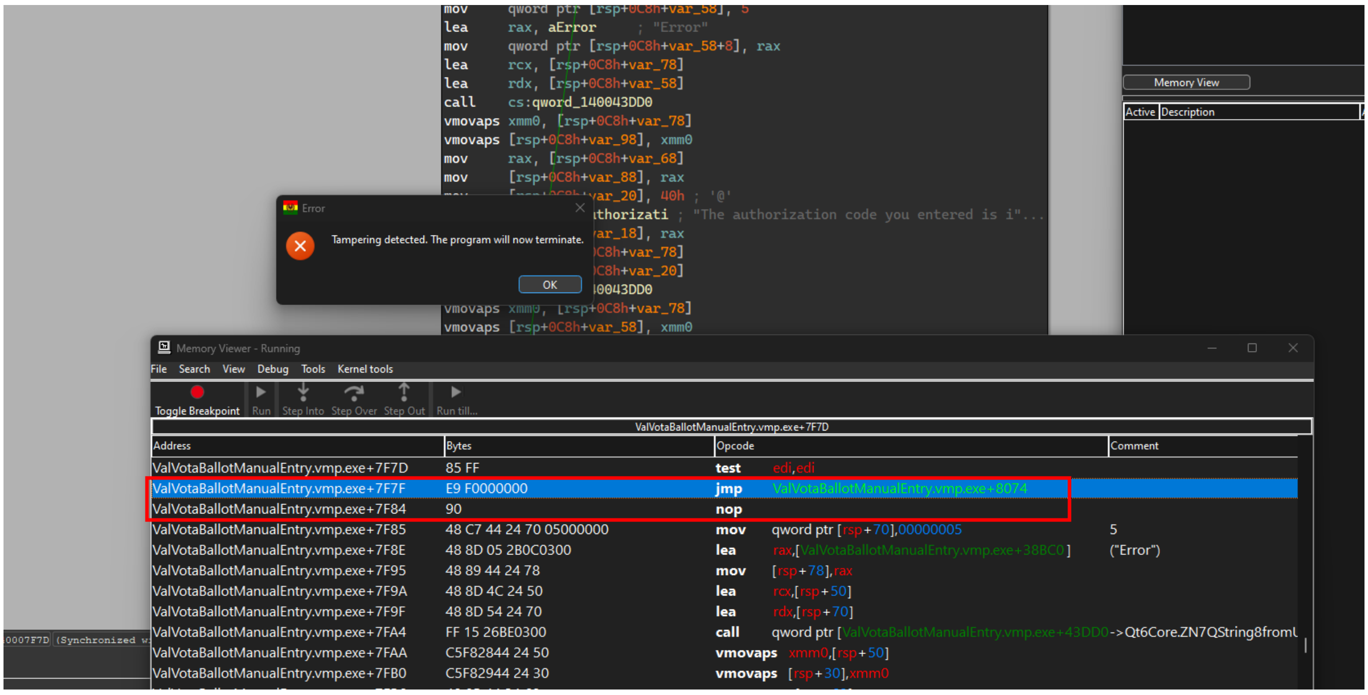
```
mov     qword ptr [rsp+0C8h+var_58], 5
lea     rax, aError      ; "Error"
mov     qword ptr [rsp+0C8h+var_58+8], rax
lea     rcx, [rsp+0C8h+var_78]
lea     rdx, [rsp+0C8h+var_58]
call    cs:qword_1400
vmovaps xmm0, [rsp+0C
vmovaps [rsp+0C8h+var
mov     rax, [rsp+0C8
mov     [rsp+0C8h+var
mov     [rsp+0C8h+var
lea     rax, aTheAuth                                  is i"...
mov     [rsp+0C8h+var
lea     rcx, [rsp+0C8h+var_78]
lea     rdx, [rsp+0C8h+var_20]
call    cs:qword_140043DD0
vmovaps xmm0, [rsp+0C8h+var_78]
vmovaps [rsp+0C8h+var_58], xmm0
mov     rax, [rsp+0C8h+var_68]
mov     [rsp+0C8h+var_48], rax
mov     [rsp+0C8h+var_A8], 0
lea     rdx, [rsp+0C8h+var_98]
lea     r8, [rsp+0C8h+var_58]
mov     rcx, rsi
```

ValVotaBallotManualEntry.vmp.exe

File integrity could not be verified. This file might have been altered by another program. The application will now terminate.

OK

Patching the program in-memory when it's already running also triggers an error since the program calls VMProtect SDK's `VMProtectIsValidImageCRC` function to check for the file image's integrity when it verifies the validity of the serial number:



```
mov     qword ptr [rsp+0C8h+var_58], 5
lea     rax, aError      ; "Error"
mov     qword ptr [rsp+0C8h+var_58+8], rax
lea     rcx, [rsp+0C8h+var_78]
lea     rdx, [rsp+0C8h+var_58]
call    cs:qword_140043DD0
vmovaps xmm0, [rsp+0C8h+var_78]
vmovaps [rsp+0C8h+var_98], xmm0
mov     rax, [rsp+0C8h+var_68]
mov     [rsp+0C8h+var_88], rax
        var_20], 40h ; '@'
thorizati ; "The authorization code you entered is i"...
        ar_18], rax
        C8h+var_78]
        C8h+var_20]
        0043DD0
vmovaps xmm0, [rsp+0C8h+var_78]
vmovaps [rsp+0C8h+var_58], xmm0
```

Error

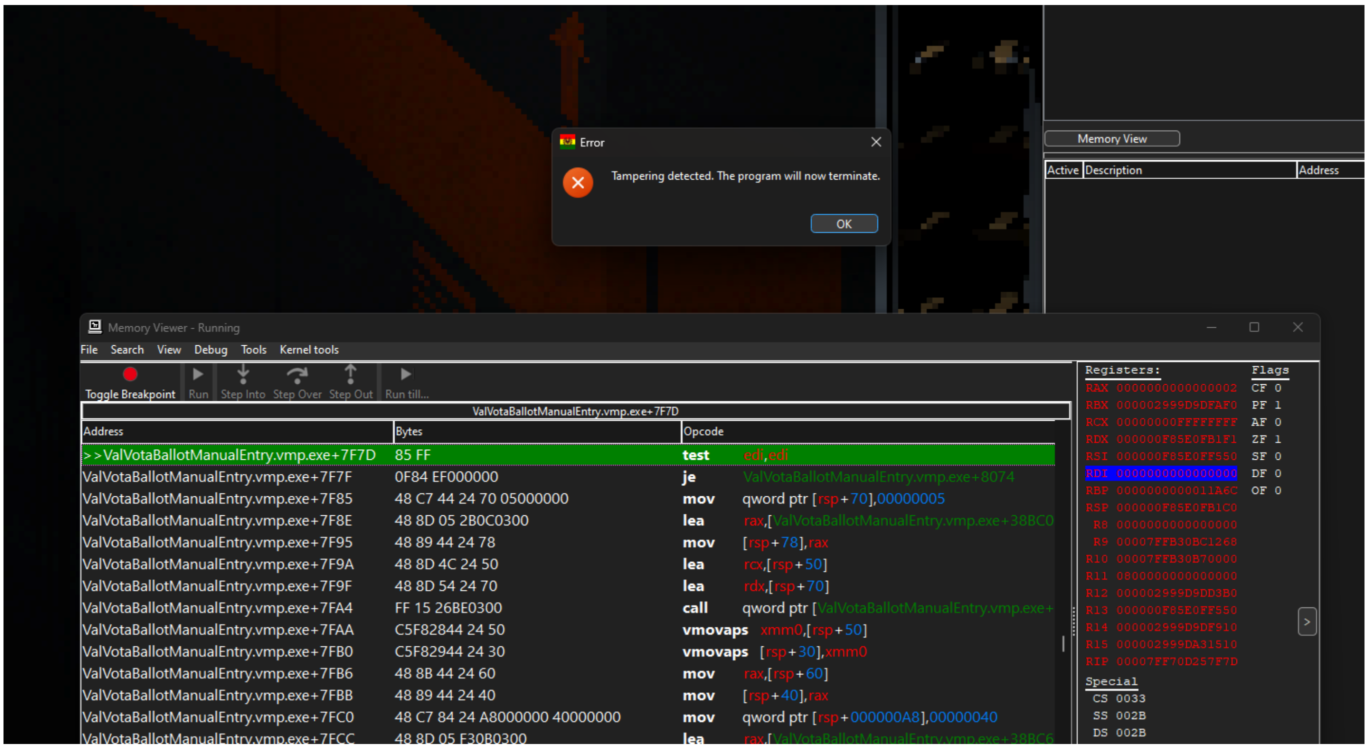
Tampering detected. The program will now terminate.

OK

ValVotaBallotManualEntry.vmp.exe+7F7D

Address	Bytes	Opcode	Comment
ValVotaBallotManualEntry.vmp.exe+7F7D	85 FF	test edi,edi	
ValVotaBallotManualEntry.vmp.exe+7F7F	E9 F0000000	jmp ValVotaBallotManualEntry.vmp.exe+8074	
ValVotaBallotManualEntry.vmp.exe+7F84	90	nop	
ValVotaBallotManualEntry.vmp.exe+7F85	48 C7 44 24 70 05000000	mov qword ptr [rsp+70],00000005	5
ValVotaBallotManualEntry.vmp.exe+7F8E	48 8D 05 2B0C0300	lea rax,[ValVotaBallotManualEntry.vmp.exe+38BC0]	("Error")
ValVotaBallotManualEntry.vmp.exe+7F95	48 89 44 24 78	mov [rsp+78],rax	
ValVotaBallotManualEntry.vmp.exe+7F9A	48 8D 4C 24 50	lea rcx,[rsp+50]	
ValVotaBallotManualEntry.vmp.exe+7F9F	48 8D 54 24 70	lea rdx,[rsp+70]	
ValVotaBallotManualEntry.vmp.exe+7FA4	FF 15 26BE0300	call qword ptr [ValVotaBallotManualEntry.vmp.exe+43DD0->Qt6Core.ZN7QString8fromU	
ValVotaBallotManualEntry.vmp.exe+7FAA	C5F82844 24 50	vmovaps xmm0,[rsp+50]	
ValVotaBallotManualEntry.vmp.exe+7FB0	C5F82944 24 30	vmovaps [rsp+30],xmm0	

Instead, we can attach a debugger to the program and manipulate the value of the `edi` register. However, if we attach a regular Windows Debugger, the program will also detect it and exit since it uses VMProtect SDK's `VMProtectIsDebuggerPresent` function to detect the presence of debuggers.


Error

Tampering detected. The program will now terminate.

OK

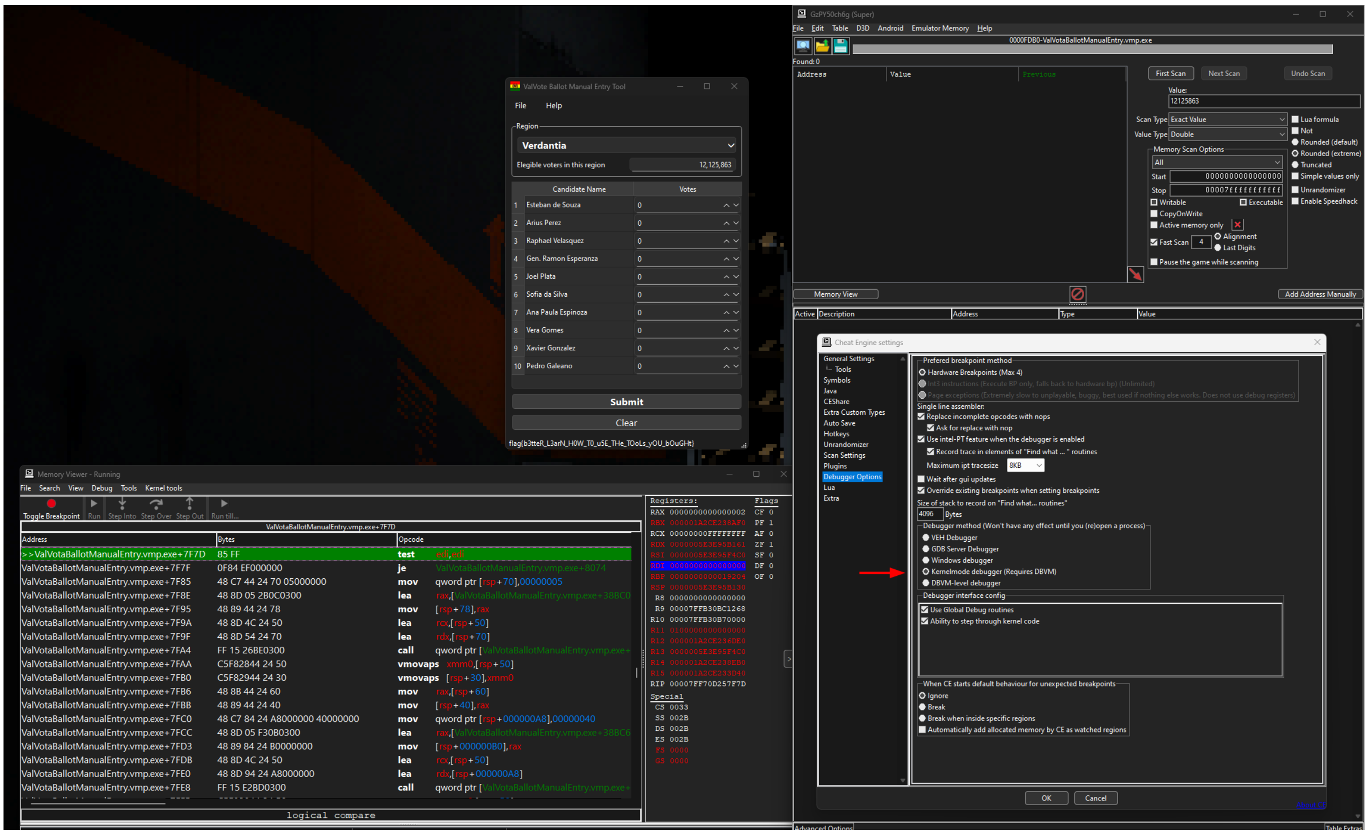
Memory Viewer - Running

ValVotaBallotManualEntry.vmp.exe+7F7D

```
Address                                  Bytes                      Opcode
>>ValVotaBallotManualEntry.vmp.exe+7F7D  85 FF                      test      edi,edi
ValVotaBallotManualEntry.vmp.exe+7F7F    0F84 EF000000              je        ValVotaBallotManualEntry.vmp.exe+8074
ValVotaBallotManualEntry.vmp.exe+7F85    48 C7 44 24 70 05000000    mov       qword ptr [rsp+70],00000005
ValVotaBallotManualEntry.vmp.exe+7F8E    48 8D 05 2B0C0300          lea       rax,[ValVotaBallotManualEntry.vmp.exe+38BC0
ValVotaBallotManualEntry.vmp.exe+7F95    48 89 44 24 78             mov       [rsp+78],rax
ValVotaBallotManualEntry.vmp.exe+7F9A    48 8D 4C 24 50             lea       rcx,[rsp+50]
ValVotaBallotManualEntry.vmp.exe+7F9F    48 8D 54 24 70             lea       rdx,[rsp+70]
ValVotaBallotManualEntry.vmp.exe+7FA4    FF 15 26BE0300             call      qword ptr [ValVotaBallotManualEntry.vmp.exe+
ValVotaBallotManualEntry.vmp.exe+7FAA    C5F82844 24 50             vmovaps   xmm0,[rsp+50]
ValVotaBallotManualEntry.vmp.exe+7FB0    C5F82944 24 30             vmovaps   [rsp+30],xmm0
ValVotaBallotManualEntry.vmp.exe+7FB6    48 8B 44 24 60             mov       rax,[rsp+60]
ValVotaBallotManualEntry.vmp.exe+7FBB    48 89 44 24 40             mov       [rsp+40],rax
ValVotaBallotManualEntry.vmp.exe+7FC0    48 C7 84 24 A8000000 40000000  mov   qword ptr [rsp+000000A8],00000040
ValVotaBallotManualEntry.vmp.exe+7FCC    48 8D 05 F30B0300          lea       rax,[ValVotaBallotManualEntry.vmp.exe+38BC6
```

```
Registers:                 Flags
RAX 0000000000000002       CF 0
RBX 000002999D9DFAF0       PF 1
RCX 00000000FFFFFFFF       AF 0
RDX 000000F85E0FB1F1       ZF 1
RSI 000000F85E0FF550       SF 0
RDI 0000000000000000       DF 0
RBP 00000000000011A6C      OF 0
RSP 000000F85E0FB1C0
 R8 0000000000000000
 R9 00007FFB30BC1268
R10 00007FFB30B70000
R11 0800000000000000
R12 000002999D9DD3B0
R13 000000F85E0FF550
R14 000002999D9DF910
R15 000002999DA31510
RIP 00007FF70D257F7D
Special
 CS 0033
 SS 002B
 DS 002B
```

We can combat this by using a debugger that's more covert. Here I'm using a custom-compiled Cheat Engine DBVM kernel mode debugger. For the stock version of Cheat Engine, the kernel mode debugger will get detected, but not the user-mode VEH debugger or the DBVM-level (ring -1) debugger. Other debuggers such as HyperDbg could work too.


ValVote Ballot Manual Entry Tool

Region: Verdantia

Elegible voters in this region: 12,125,863

	Candidate Name	Votes
1	Esteban de Souza	0
2	Arius Perez	0
3	Raphael Velasquez	0
4	Gen. Ramon Esperanza	0
5	Joel Plata	0
6	Sofia da Silva	0
7	Ana Paula Espinoza	0
8	Vera Gomes	0
9	Xavier Gonzalez	0
10	Pedro Galeano	0

Submit

Clear

flag{b3tteR_L3arN_H0W_T0_u5E_THe_TOoLs_yOU_bOuGHt}

Cheat Engine settings — Debugger Options

Prefered breakpoint method
- [✓ Hardware Breakpoints (Max 4)]
- Int3 instructions (Execute BP only, falls back to hardware bp) (Unlimited)
- Page exceptions (Extremely slow to unplayable, buggy, best used if nothing else works. Does not use debug registers)

Single line assembler:
- ✓ Replace incomplete opcodes with nops
- ✓ Ask for replace with nop
- ✓ Use intel-PT feature when the debugger is enabled
- ✓ Record trace in elements of "Find what ... " routines
- Maximum ipt tracesize: 8KB
- Wait after gui updates
- ✓ Override existing breakpoints when setting breakpoints

Size of stack to record on "Find what... routines": 4096 Bytes

Debugger method (Won't have any effect until you (re)open a process)
- VEH Debugger
- GDB Server Debugger
- Windows debugger
- [✓ Kernelmode debugger (Requires DBVM)]
- DBVM-level debugger

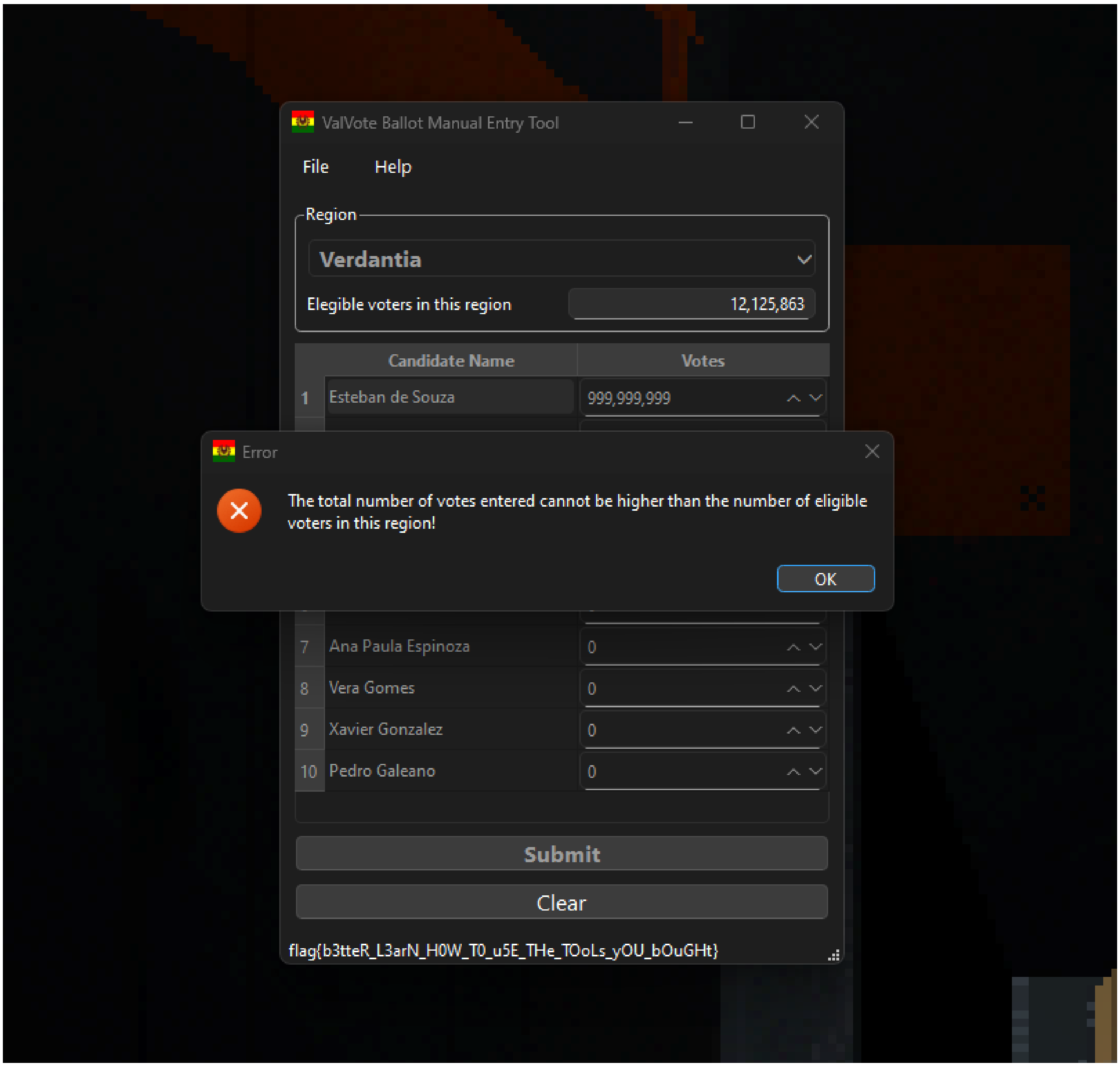
Debugger interface config
- ✓ Use Global Debug routines
- ✓ Ability to step through kernel code

When CE starts default behaviour for unexpected breakpoints
- [✓ Ignore]
- Break
- Break when inside specific regions
- Automatically add allocated memory by CE as watched regions

The flag is shown on the status bar once we enter the "homepage" of the application.

Part 2

When we enter votes that sums up to be higher than the number of eligible voters in the selected region, this error message pops up:



The total number of votes entered cannot be higher than the number of eligible voters in this region!

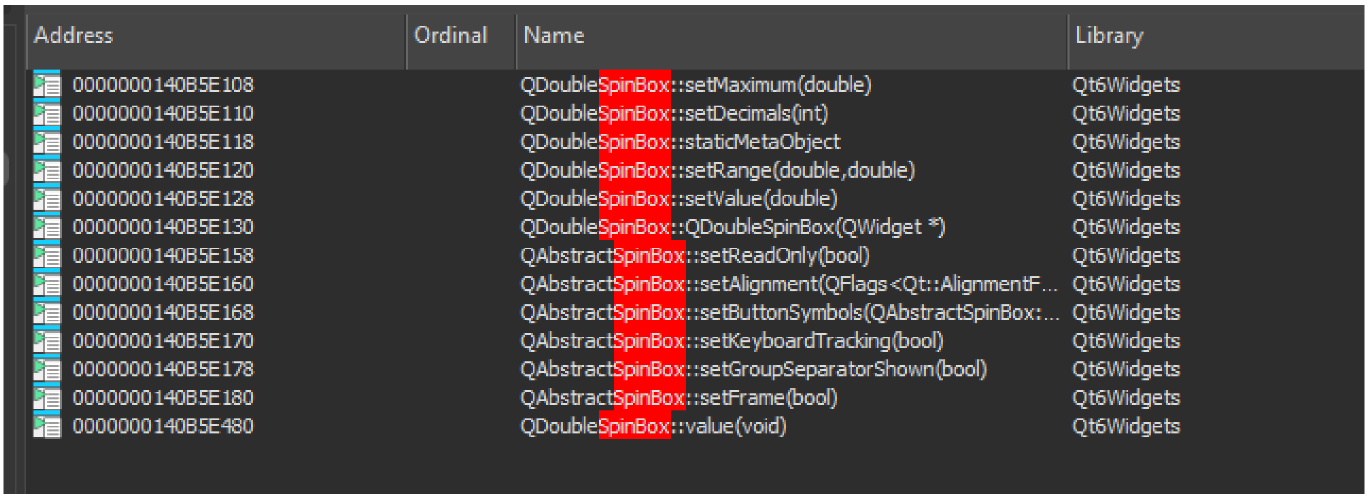
flag{b3tteR_L3arN_H0W_T0_u5E_THe_TOoLs_yOU_bOuGHt}

This string cannot be found with static analysis in the program like in the previous part because it has been encrypted by VMProtect. All the code associated with this part of the challenge are also protected with VMProtect's `Ultra` configuration (virtualization + mutation), so it is not practical to statically analyze what's going on behind this error message.

There are many more ways to solve this challenge such as reverse engineering the gRPC ProtoBuf specs and manipulating the data or forging your own requests. I will walk you through the simplest solution, which is to find and patch the number of eligible voters in-memory.

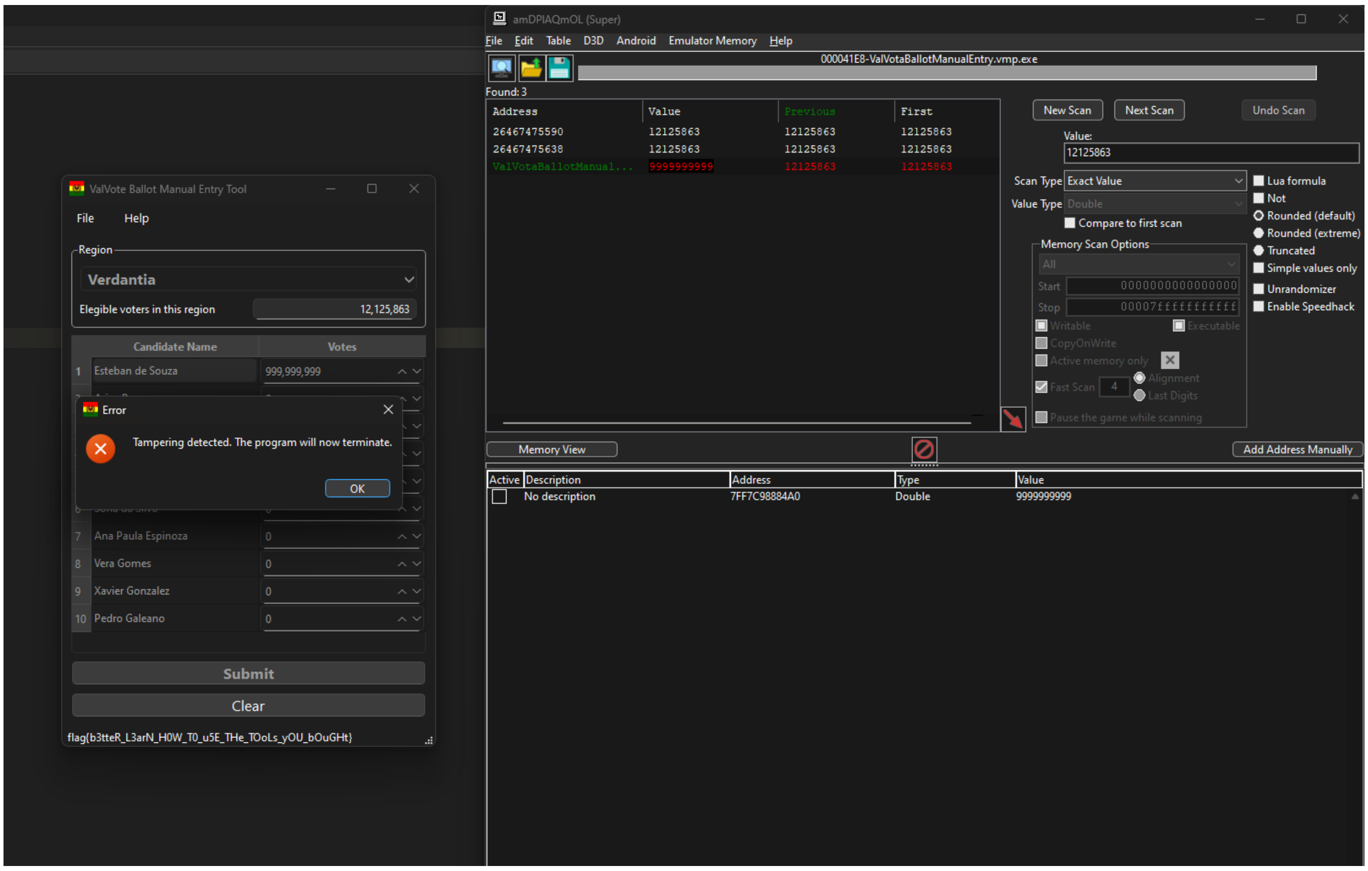
We need to search the memory to find out where the number of eligible voters for each region are stored. Before we can perform the search, we will first need to determine the numbers' data type, and this part may trick some people.

You may assume that these numbers are integers because that is the natural thing to do, but they are actually `double`s. If you examine the import table of the program, you'll see that it only uses `QDoubleSpinBox` instead of `QSpinBox`. This is a clue that all of the numbers you see are `double`s, so you should search for `12125863` with the `double` data type. (This is also the reason why I turned off VMProtect's `Import Protection` feature.)



Address	Ordinal	Name	Library
0000000140B5E108		QDoubleSpinBox::setMaximum(double)	Qt6Widgets
0000000140B5E110		QDoubleSpinBox::setDecimals(int)	Qt6Widgets
0000000140B5E118		QDoubleSpinBox::staticMetaObject	Qt6Widgets
0000000140B5E120		QDoubleSpinBox::setRange(double,double)	Qt6Widgets
0000000140B5E128		QDoubleSpinBox::setValue(double)	Qt6Widgets
0000000140B5E130		QDoubleSpinBox::QDoubleSpinBox(QWidget *)	Qt6Widgets
0000000140B5E158		QAbstractSpinBox::setReadOnly(bool)	Qt6Widgets
0000000140B5E160		QAbstractSpinBox::setAlignment(QFlags<Qt::AlignmentF...	Qt6Widgets
0000000140B5E168		QAbstractSpinBox::setButtonSymbols(QAbstractSpinBox:...	Qt6Widgets
0000000140B5E170		QAbstractSpinBox::setKeyboardTracking(bool)	Qt6Widgets
0000000140B5E178		QAbstractSpinBox::setGroupSeparatorShown(bool)	Qt6Widgets
0000000140B5E180		QAbstractSpinBox::setFrame(bool)	Qt6Widgets
0000000140B5E480		QDoubleSpinBox::value(void)	Qt6Widgets

We can search for the number `12125863` in Cheat Engine. There should only be three results including the number in the read-only section. However, when we try to change the number that is used to perform the checks, we will get an error, and the application closes:



ValVote Ballot Manual Entry Tool

Region: Verdantia

Elegible voters in this region: 12,125,863

	Candidate Name	Votes
1	Esteban de Souza	999,999,999
7	Ana Paula Espinoza	0
8	Vera Gomes	0
9	Xavier Gonzalez	0
10	Pedro Galeano	0

Error: Tampering detected. The program will now terminate.

flag{b3tteR_L3arN_H0W_T0_u5E_THe_TOoLs_yOU_bOuGHt}

amDPIAQmOL (Super) — 000041E8-ValVotaBallotManualEntry.vmp.exe

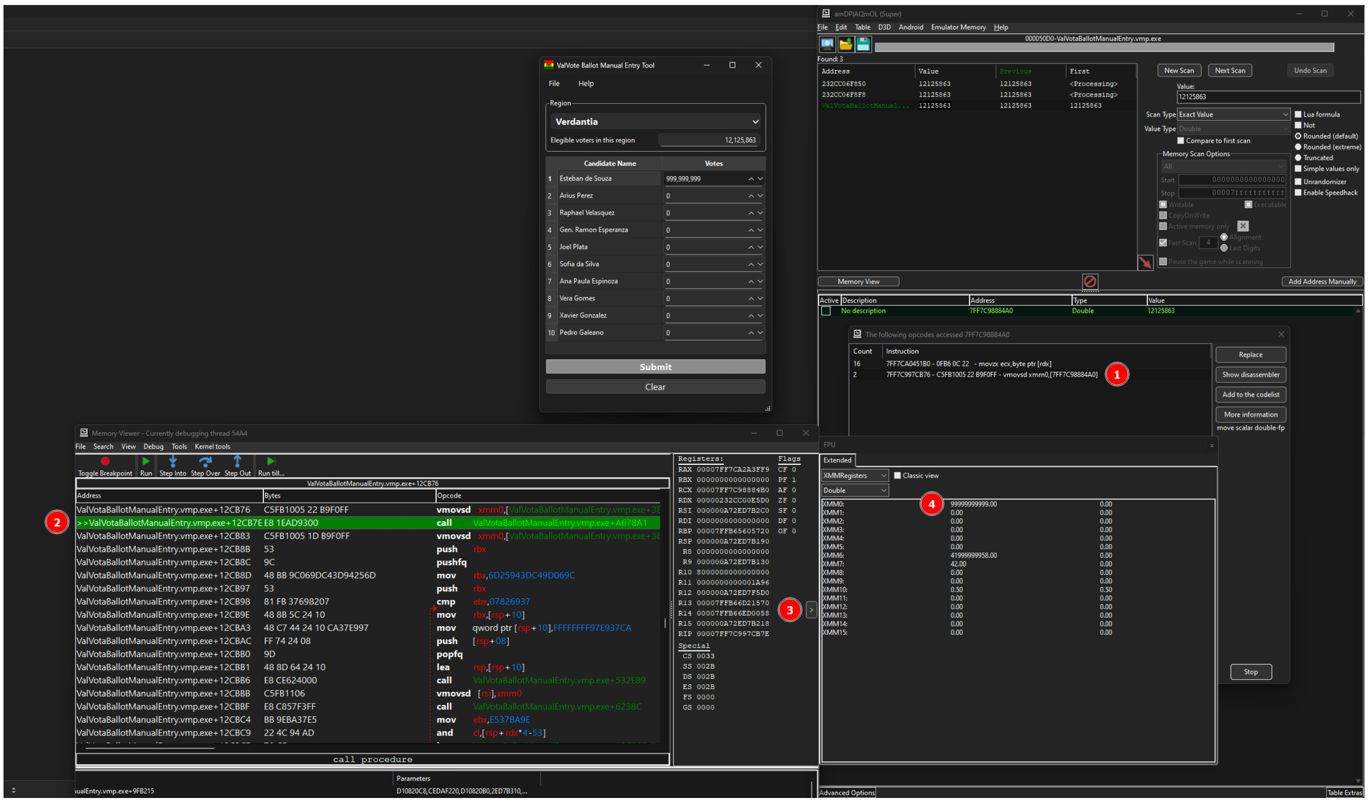
Found: 3

Address	Value	Previous	First
26467475590	12125863	12125863	12125863
26467475638	12125863	12125863	12125863
ValVotaBallotManual...	9999999999	12125863	12125863

Value: 12125863
Scan Type: Exact Value
Value Type: Double
Start: 0000000000000000
Stop: 00007fffffffffff

Active	Description	Address	Type	Value
☐	No description	7FF7C98884A0	Double	9999999999

This is because the application performs an image integrity check as it returns the number of voters for a region. We can use a stealthy debugger to check what is reading this data and alter the readouts in the debugger instead. It should be noted that while the user-mode VEH debugger works for the first part of this challenge, it will not work for this part. You will either need to use a custom DBK driver's kernel-mode debugger, the DBVM-level debugger, or an equivalent debugger that's more stealthy than the VEH debugger.



ValVote Ballot Manual Entry Tool

Region: Verdantia
Elegible voters in this region: 12,125,863

	Candidate Name	Votes
1	Esteban de Souza	999,999,999
2	Arius Perez	0
3	Raphael Velasquez	0
4	Gen. Ramon Esperanza	0
5	Joel Plata	0
6	Sofia da Silva	0
7	Ana Paula Espinoza	0
8	Vera Gomes	0
9	Xavier Gonzalez	0
10	Pedro Galeano	0

Memory Viewer - Currently debugging thread 54A4

```
ValVotaBallotManualEntry.vmp.exe+12CB76
Address                                Bytes                  Opcode
ValVotaBallotManualEntry.vmp.exe+12CB76  C5FB1005 22 B9F0FF    vmovsd  xmm0,[ValVotaBallotManualEntry.vmp.exe+38
>>ValVotaBallotManualEntry.vmp.exe+12CB7E E8 1EAD9300          call    ValVotaBallotManualEntry.vmp.exe+A678A1
ValVotaBallotManualEntry.vmp.exe+12CB83  C5FB1005 1D B9F0FF    vmovsd  xmm0,[ValVotaBallotManualEntry.vmp.exe+38
ValVotaBallotManualEntry.vmp.exe+12CB8B  53                    push    rbx
ValVotaBallotManualEntry.vmp.exe+12CB8C  9C                    pushfq
ValVotaBallotManualEntry.vmp.exe+12CB8D  48 BB 9C069DC43D94256D mov    rbx,6D25943DC49D069C
ValVotaBallotManualEntry.vmp.exe+12CB97  53                    push    rbx
ValVotaBallotManualEntry.vmp.exe+12CB98  81 FB 37698207        cmp     ebx,07826937
ValVotaBallotManualEntry.vmp.exe+12CB9E  48 8B 5C 24 10        mov     rbx,[rsp+10]
ValVotaBallotManualEntry.vmp.exe+12CBA3  48 C7 44 24 10 CA37E997 mov   qword ptr [rsp+10],FFFFFFFF97E937CA
ValVotaBallotManualEntry.vmp.exe+12CBAC  FF 74 24 08           push    [rsp+08]
ValVotaBallotManualEntry.vmp.exe+12CBB0  9D                    popfq
ValVotaBallotManualEntry.vmp.exe+12CBB1  48 8D 64 24 10        lea     rsp,[rsp+10]
ValVotaBallotManualEntry.vmp.exe+12CBB6  E8 CE624000           call    ValVotaBallotManualEntry.vmp.exe+532E89
ValVotaBallotManualEntry.vmp.exe+12CBBB  C5FB1106              vmovsd  [rsi],xmm0
ValVotaBallotManualEntry.vmp.exe+12CBBF  E8 C857F3FF           call    ValVotaBallotManualEntry.vmp.exe+6238C
ValVotaBallotManualEntry.vmp.exe+12CBC4  BB 9EBA37E5           mov     ebx,E537BA9E
ValVotaBallotManualEntry.vmp.exe+12CBC9  22 4C 94 AD           and     cl,[rsp+rdx*4-53]
call procedure
```

```
Registers:               Flags
RAX 00007FF7CA2A3FF9     CF 0
RBX 0000000000000000     PF 1
RCX 00007FF7C98884B0     AF 0
RDX 00000232CC00E5D0     ZF 0
RSI 000000A72ED7B2C0     SF 0
RDI 0000000000000000     DF 0
RBP 00007FFB65605720     OF 0
RSP 000000A72ED7B190
 R8 0000000000000000
 R9 000000A72ED7B130
R10 8000000000000000
R11 0000000000001A96
R12 000000A72ED7F5D0
R13 00007FFB66D21570
R14 00007FFB66ED0058
R15 000000A72ED7B218
RIP 00007FF7C997CB7E
Special
 CS 0033
 SS 002B
 DS 002B
 ES 002B
 FS 0000
 GS 0000
```

Parameters: D10820C8,CEDAF220,D10820B0,2ED7B310,...

amDPIAQmOL (Super) — 000050D0-ValVotaBallotManualEntry.vmp.exe

Found: 3

Address	Value	Previous	First
232CC06F850	12125863	12125863	<Processing>
232CC06F8F8	12125863	12125863	<Processing>
ValVotaBallotManual...	12125863	12125863	12125863

Value: 12125863
Scan Type: Exact Value
Value Type: Double

Active	Description	Address	Type	Value
	No description	7FF7C98884A0	Double	12125863

The following opcodes accessed 7FF7C98884A0

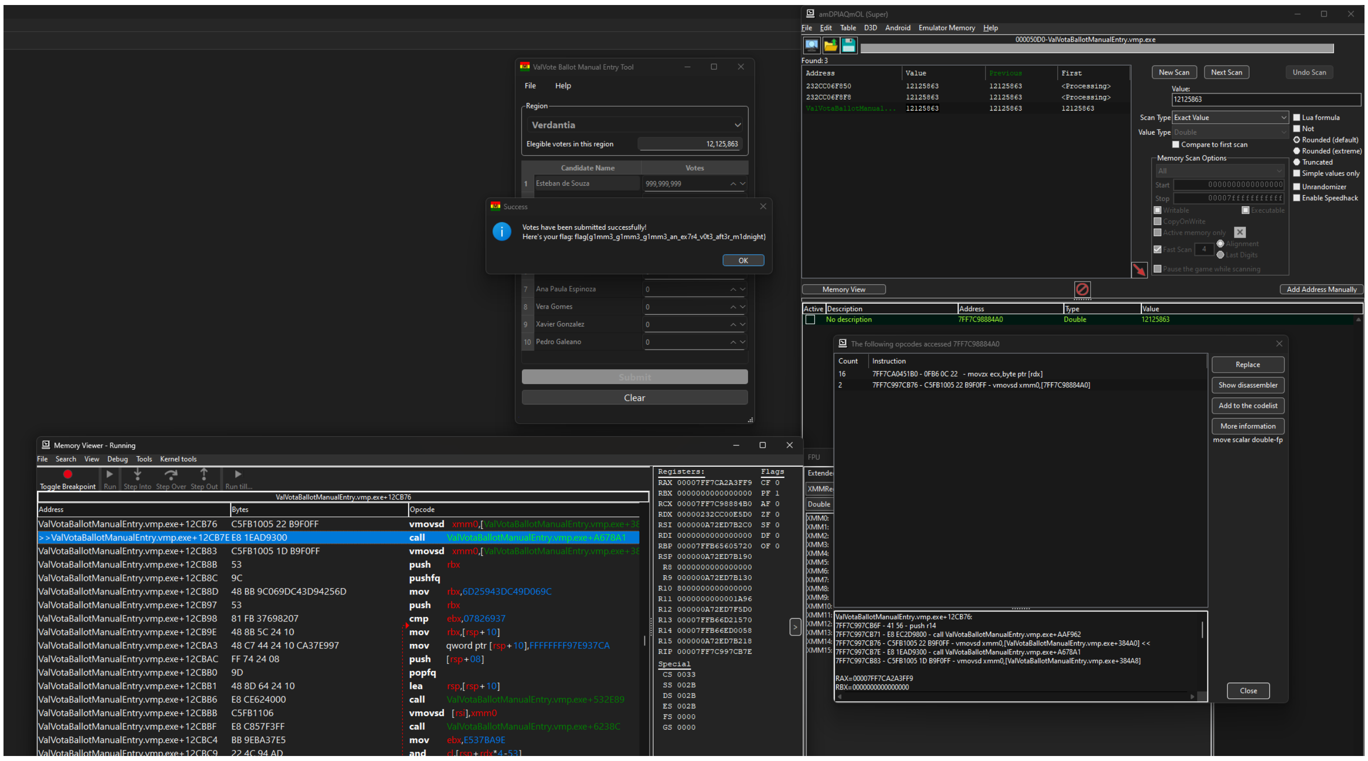
Count	Instruction
16	7FF7CA0451B0 - 0FB6 0C 22 - movzx ecx,byte ptr [rdx]
2	7FF7C997CB76 - C5FB1005 22 B9F0FF - vmovsd xmm0,[7FF7C98884A0]

move scalar double-fp

FPU — Extended — XMMRegisters — Double

```
XMM0:   99999999999.00   0.00
XMM1:   0.00             0.00
XMM2:   0.00             0.00
XMM3:   0.00             0.00
XMM4:   0.00             0.00
XMM5:   0.00             0.00
XMM6:   41999999958.00   0.00
XMM7:   42.00            0.00
XMM8:   0.00             0.00
XMM9:   0.00             0.00
XMM10:  0.50             0.50
XMM11:  0.00             0.00
XMM12:  0.00             0.00
XMM13:  0.00             0.00
XMM14:  0.00             0.00
XMM15:  0.00             0.00
```

Then we can continue the execution and the the number we entered will pass the checks. The server will do the math and return the flag:



Success

```
Votes have been submitted successfully!
Here's your flag: flag{g1mm3_g1mm3_g1mm3_an_ex7r4_v0t3_aft3r_m1dnight}
```

Memory Viewer - Running

```
ValVotaBallotManualEntry.vmp.exe+12CB76:
7FF7C997CB6F - 41 56 - push r14
7FF7C997CB71 - E8 EC2D9800 - call ValVotaBallotManualEntry.vmp.exe+AAF962
7FF7C997CB76 - C5FB1005 22 B9F0FF - vmovsd xmm0,[ValVotaBallotManualEntry.vmp.exe+384A0] <<
7FF7C997CB7E - E8 1EAD9300 - call ValVotaBallotManualEntry.vmp.exe+A678A1
7FF7C997CB83 - C5FB1005 1D B9F0FF - vmovsd xmm0,[ValVotaBallotManualEntry.vmp.exe+384A8]

RAX=00007FF7CA2A3FF9
RBX=0000000000000000
```