Ingeniería Inversa
Clase 5

Análisis de Malware

martin.uy
Open by default.

Análisis de Malware

Windows Task Manager

File Options View Help

Applications | Processes | Services | Performance | Networking | Users

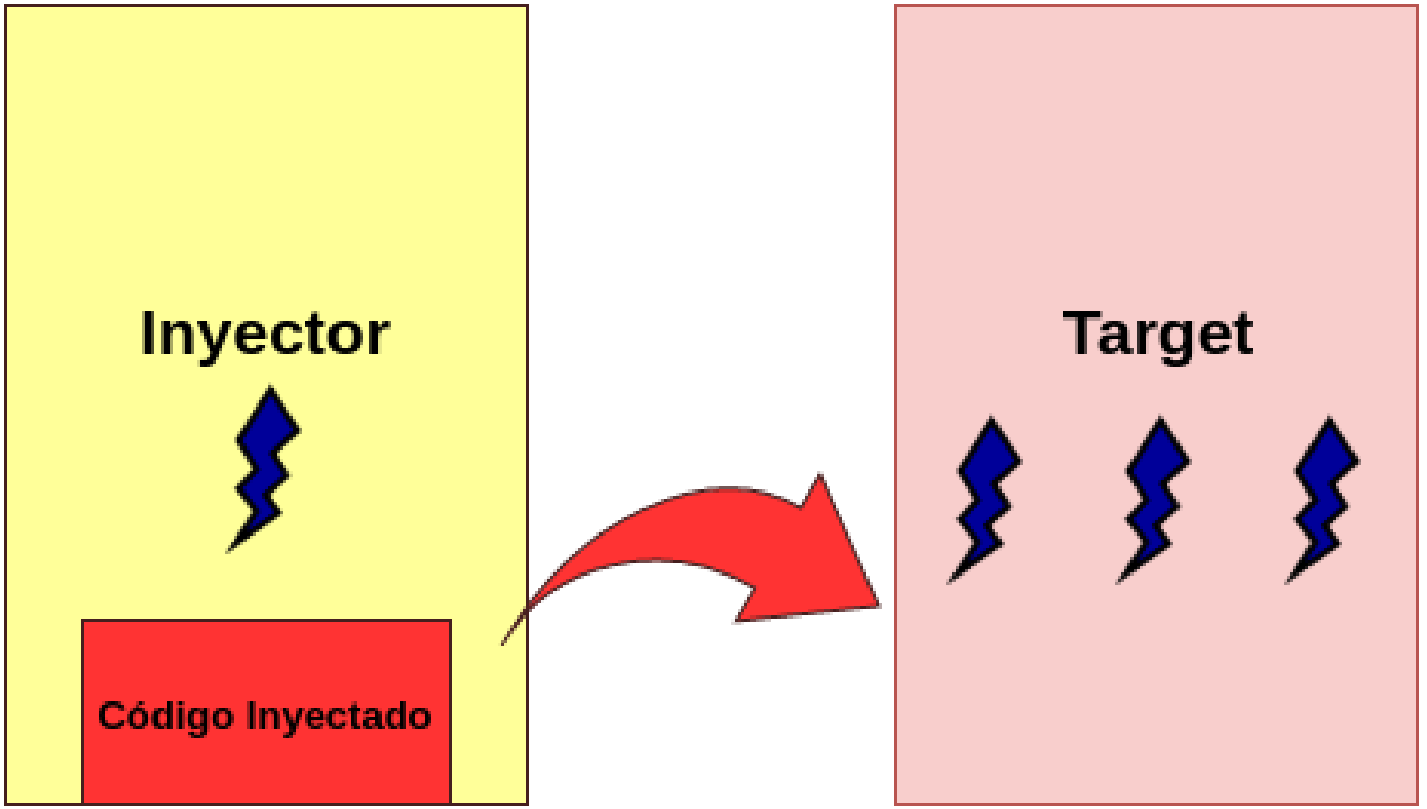
Image Name	User Name	CPU	Memory (Private Working Set)	Command Line
audiodg.exe	LOCAL ...	00	9,940 K	
cmd.exe	martin	00	1,708 K	"C:\Windows\System32\cmd.exe"
conhost.exe	martin	00	1,336 K	\??\C:\Windows\system32\conhost.exe "-706513562104198653 3-943063273662986630153531125 2-112...
conhost.exe	martin	00	1,220 K	\??\C:\Windows\system32\conhost.exe "-491874450166791672 9-770656423151858395414244761 18-17...
csrss.exe	SYSTEM	00	1,248 K	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windo...
csrss.exe	SYSTEM	00	1,304 K	%SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,20480,768 Windo...
dwm.exe	martin	00	98,280 K	"C:\Windows\system32\Dwm.exe"
explorer.exe	martin	00	26,320 K	C:\Windows\Explorer.EXE
idaq.exe *32	martin	00	98,700 K	"C:\Program Files (x86)\IDA 6.95\idaq.exe"
lsass.exe	SYSTEM	00	3,288 K	C:\Windows\system32\lsass.exe
lsm.exe	SYSTEM	00	856 K	C:\Windows\system32\lsm.exe
malware.exe *32	martin	00	408 K	malware.exe
Microsoft.VsHub.Serv...	martin	00	28,376 K	"C:\Program Files (x86)\Common Files\Microsoft Shared\VsHub\1.0.0.0\Microsoft.VsHub.Server.HttpHost....
MpCmdRun.exe	NETWO...	00	2,408 K	"c:\program files\windows defender\MpCmdRun.exe" SpyNetService -RestrictPrivileges -A... 2B453...
msdtc.exe	NETWO...	00	1,096 K	C:\Windows\System32\msdtc.exe
nfsd.exe	SYSTEM	00	8,648 K	C:\Users\martin\Programs\ms-nfs41-client-x64\nfsd.exe

¿Algo extraño?

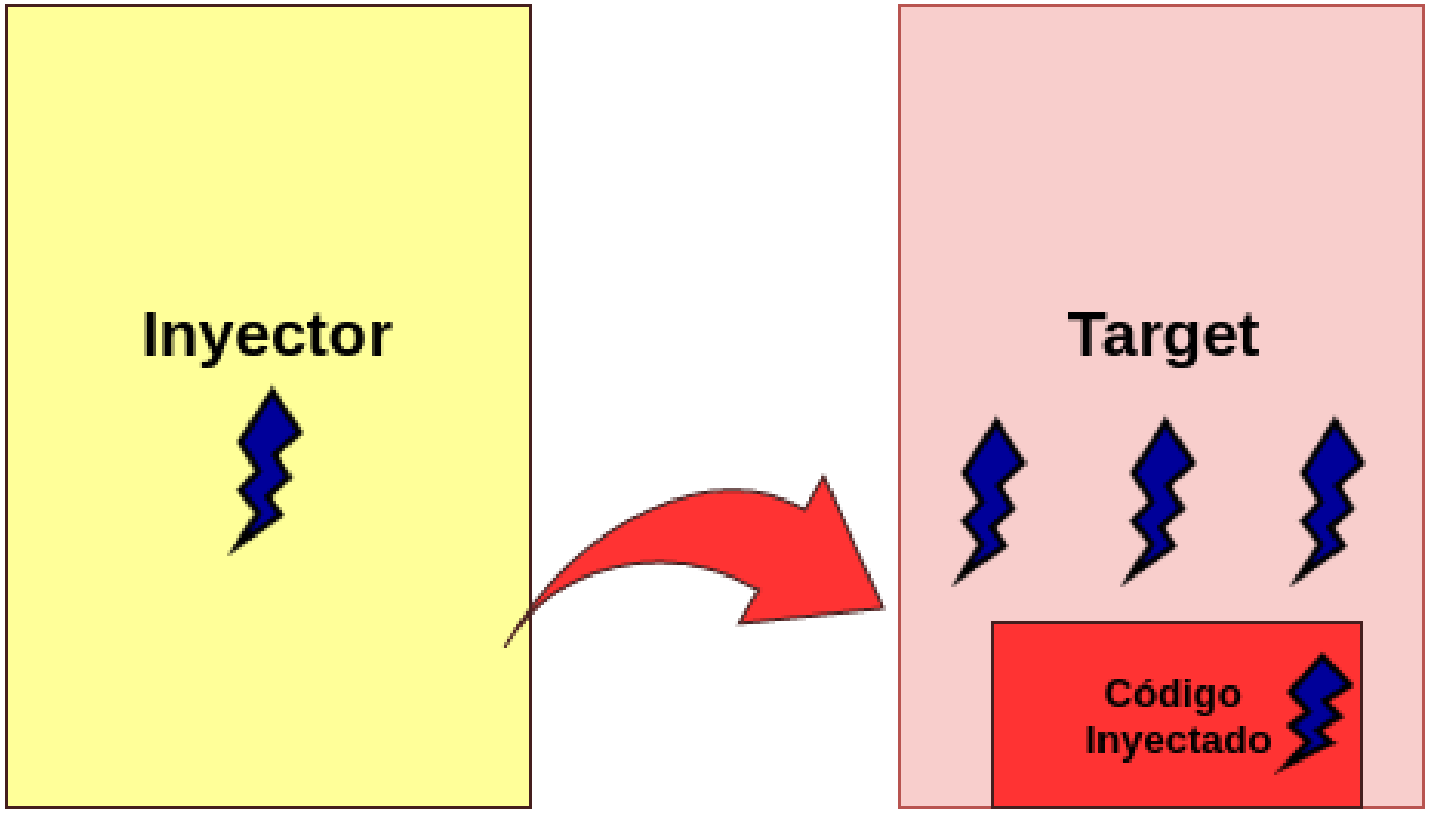
Análisis de Malware

- Inyección en procesos
  - Esconderse y evitar trazas de auditoría
  - Bypassear firewalls de endpoint con filtrado por aplicación
  - Robarle información al proceso inyectado
  - Cambiar de sesión (sesión no-interactiva → sesión interactiva)
    - Screenshots

Análisis de Malware



Análisis de Malware



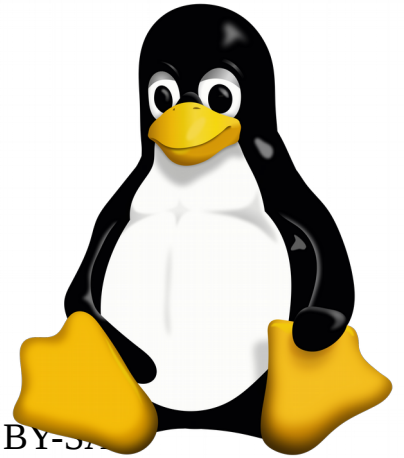
Análisis de Malware

- Objetivos
  - Pasar el código malicioso a otro proceso
  - Crear un thread en el proceso que ejecute el código malicioso
  - Evitar interrupciones o corrupción de datos en el proceso atacado (continuación del proceso)

Análisis de Malware

- Técnicas de inyección en procesos (Windows)
  - CreateRemoteThread
  - Asynchronous Procedure Call
  - Debugging API

- Técnicas de inyección en procesos (Linux)
  - Debugging API (ptrace)



Análisis de Malware

- CreateRemoteThread (Windows)
  - OpenProcess
    - Handle para controlar un proceso remoto
  - VirtualAllocEx
    - Alocarle memoria al proceso remoto
      - Permisos de escritura y ejecución

Análisis de Malware

- CreateRemoteThread (Windows)
  - WriteProcessMemory
    - Escribir la memoria del proceso remoto
      - Código a inyectar
  - CreateRemoteThread
    - Crear un thread en el proceso remoto y ponerlo a ejecutar la dirección de memoria escrita anteriormente

Demo 5.1

Inyección en procesos por CreateRemoteThread (Windows)

Análisis de Malware

- Asynchronous Procedure Call (Windows)
  - API de Windows que permite encolar llamadas asincrónicas a threads
    - Cuando el thread está en estado “alertable” (ej. durmiendo), ejecutará la llamada
    - La llamada será a una dirección arbitraria del proceso elegida por el que la encola
    - Al terminar la llamada, se restaura automáticamente el contexto original del thread

Análisis de Malware

- Asynchronous Procedure Call (Windows)
  - Obtenemos un handle al proceso víctima y a un thread cualquiera en él: OpenProcess, CreateToolhelp32Snapshot, OpenThread
  - Alocamos memoria en el proceso y la escribimos con el código ejecutable: VirtualAllocEx y WriteProcessMemory

Análisis de Malware

- Asynchronous Procedure Call (Windows)
  - Encolamos una llamada por APC con QueueUserAPC
  - Es importante que el código llamado cree un nuevo thread para continuar la ejecución y que el thread que atiende el APC pueda retornar a su ejecución normal
    - Si le interrumpimos de forma definitiva un thread a una aplicación, podemos causar inestabilidad

Análisis de Malware

- Debugging API (Windows)
  - Alocar memoria y escribir código a inyectar en el proceso (remotamente)
  - Debuggear el proceso target
    - El debugger se attachea a 1 thread
    - DebugActiveProcess / WaitForDebugEvent / ContinueDebugEvent
  - Salvar el contexto del thread attachado (ej. salvar valor de los registros en el inyector)

Análisis de Malware

- Debugging API (Windows)
  - El thread attachado se pone a ejecutar el código inyectado
    - Crear un nuevo thread en el comienzo del código inyectado
  - Se vuelve al debugger y se restaura el contexto al thread original

Análisis de Malware

- Inyección de procesos en Linux
  - Técnica similar a la descripta para Windows
  - API de debugging en Linux (Unix): ptrace
  - Leer / escribir el contexto (registros) de un thread
  - Leer / escribir memoria del proceso
  - Interceptar todas las señales para el proceso debuggeado
  - Ejecutar el proceso paso a paso

Análisis de Malware

- Inyección de procesos en Linux
  - Problema:
    - No podemos alocarle remotamente memoria al proceso
    - No podemos crearle remotamente un thread al proceso
  - Solución:
    - Secuestrar un thread y que lo haga por nosotros

Análisis de Malware

- Algunas primitivas de ptrace
  - PTRACE_ATTACH
  - PTRACE_PEEKDATA
  - PTRACE_POKEDATA
  - PTRACE_SYSCALL
  - PTRACE_CONT
  - PTRACE_GETREGS
  - PTRACE_SETREGS

Análisis de Malware

- Inyección de procesos en Linux
  - Attacharse al proceso a inyectar
  - Resolver la dirección virtual de las funciones mmap y __clone (libc)
    - Base de libc en /proc/<PID>/maps
    - Resolución navegando la memoria (estructura ELF)
    - Workaround: resolver el offset con dlsym y dladdr en el inyector y aplicarlo en el proceso remoto

Análisis de Malware

- Inyección de procesos en Linux
  - Salvar el contexto del thread attachado para restaurarlo cuando termine la inyección
    - Los valores son salvados en la memoria del inyector
    - Queremos que el proceso inyectado continúe su ejecución con normalidad

Análisis de Malware

- Inyección de procesos en Linux
  - Modificar el contexto del thread attachado (secuestro):
    - RIP → dirección de mmap / __clone
    - Otros registros → parámetros de las funciones (según la ABI x86_64 en este caso)
    - Modificar el stack: alineación a 16 bytes (ABI) y return address = 0x0

Análisis de Malware

- Inyección de procesos en Linux
  - Continuar el proceso y dejar que se ejecute la función llamada. Al retornar, se intentará ejecutar la instrucción en la dirección 0x0 y se enviará una señal al proceso (dirección inválida)
    - Como el inyector es debugger, recibe la señal primero y puede handlearla
      - La señal se descarta en lugar de pasarla al proceso debuggeado
    - Modificar el contexto del thread attachado para que ejecute otra función o restaurarlo

Análisis de Malware

- Inyección de procesos en Linux
  - Llamadas a funciones en el proceso remoto:
    - Alocar memoria para el buffer ejecutable (instrucciones inyectadas), con mmap
    - Alocar memoria para el stack del thread que ejecutará el buffer inyectado, con mmap
    - Crear un nuevo thread, con __clone

Análisis de Malware

- Inyección de procesos en Linux
  - ¿Cómo tienen que setearse los registros o la memoria para cada llamada?
    - Application Binary Interface (ABI), según la arquitectura
    - Ayuda: debuggear un ejemplo simple que utilice la API de la libc y seguirlo hasta el momento de ejecutar la syscall

Demo 5.2

Inyección en procesos por ptrace (Linux)

Análisis de Malware

- Limitaciones de las técnicas anteriores
  - Modelo de seguridad en Windows
    - Access Token – objeto que describe el contexto de seguridad de un proceso o thread (GetTokenInformation)
      - Cuando el usuario se loggea en el sistema, se le asigna un access token. Todos los procesos que ejecutan tienen este token
      - Contiene identidad de la cuenta de usuario y sus grupos: SIDs (Security Identifiers)
      - Contiene privilegios para tareas administrativas (ej. reiniciar sistema, cambiar hora, cargar drivers, etc.)
      - Un thread puede eventualmente impersonar a otro usuario y usar su access token

Análisis de Malware

- Limitaciones de las técnicas anteriores
  - Modelo de seguridad en Windows
    - Security Descriptors: información de seguridad asociada a cada Securable Object
      - Owner y grupo primario
      - DACL – acceso discrecional (acceso a usuarios o grupos específicos)

Análisis de Malware

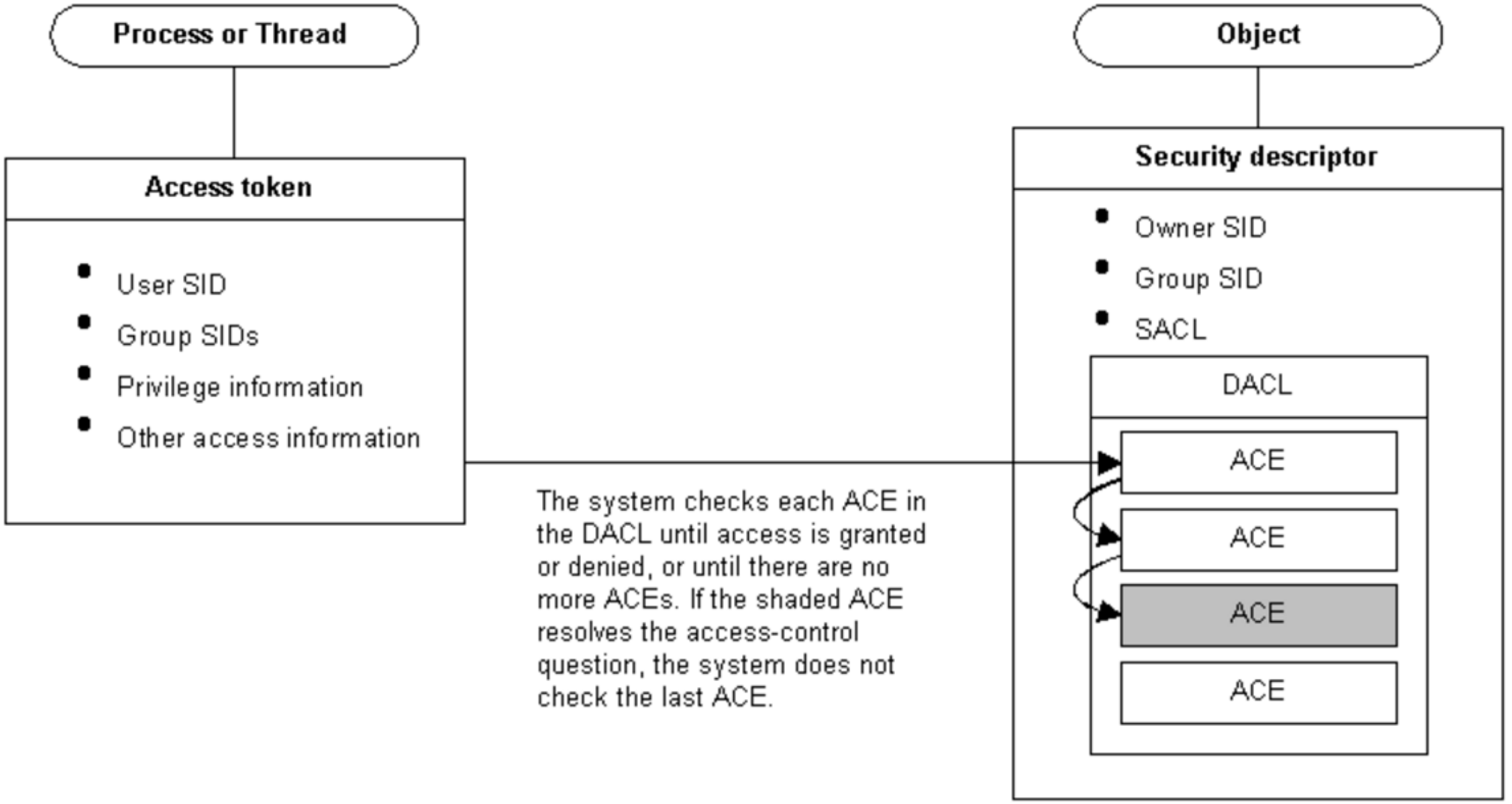
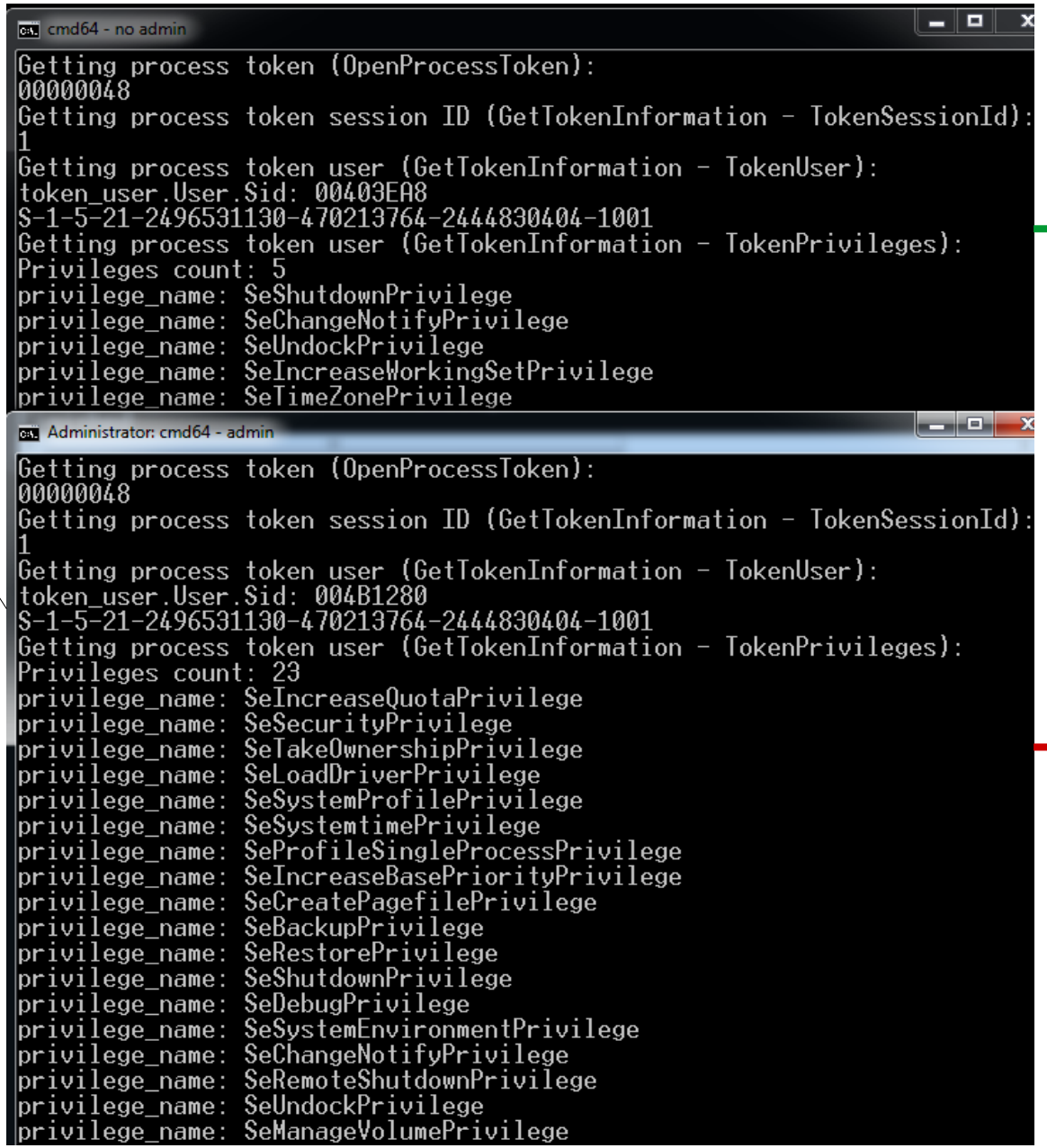


Imagen de https://msdn.microsoft.com/en-us/library/windows/desktop/aa378890(v=vs.85).aspx

Análisis de Malware

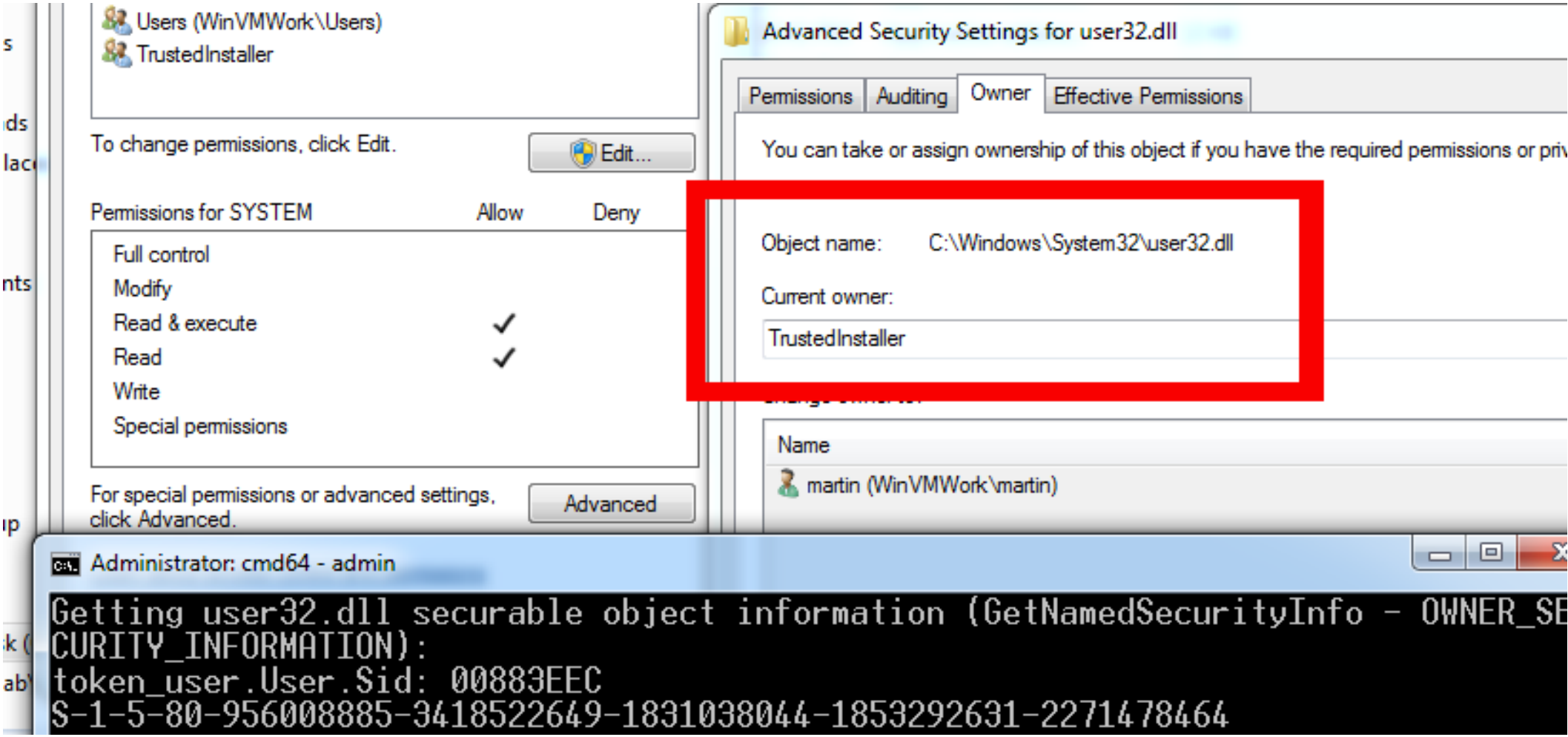


Mismo SID

No elevado

Elevado

Análisis de Malware



Análisis de Malware

- Si queremos invocar OpenProcess y otras APIs de debugging en un proceso de otro usuario, necesitamos habilitar el privilegio “SeDebugPrivilege” en el Access Token
  - Solo cuentas de administrador deberían tener este privilegio disponible para ser habilitado
  - Para debuggear procesos del mismo usuario no necesitamos este privilegio
  - Desde una perspectiva defensiva, esto recuerda la importancia de no ejecutar con cuentas administrativas o, en ese caso, impersonar usuarios no-privilegiados
  - El modelo provee granularidad para asignar a cuentas no privilegiadas los “privilegios” requeridos (principio de menor privilegio)

Análisis de Malware

- Limitaciones de las técnicas anteriores
  - Modelo de seguridad en Linux
    - Antes del kernel 2.2, el modelo de seguridad en procesos consistía en privilegiado y no-privilegiado. Un proceso privilegiado tenía control completo del sistema
    - Hay software que legítimamente requiere privilegios. Ejemplo: un servidor de DNS tiene que escuchar en un puerto bajo (53)

Análisis de Malware

- Limitaciones de las técnicas anteriores
  - Modelo de seguridad en Linux
    - Sin embargo, ante la hipótesis de que el proceso sea explotado, se necesita contener el daño
    - Las “capabilities” permiten mayor granularidad en los privilegios de los procesos
    - Las “capabilities” están asociadas a los binarios ejecutables
    - Un proceso que en tiempo de ejecución resigna “capabilities”, no las puede re-adquirir después

Análisis de Malware

```
[martin@vmlinwork 5]$ ./run.sh
[sudo] password for martin:
./bin/main = cap_net_raw+ep
Start
Creating socket...
socket_fd: 3
Dropping CAP_NET_RAW capability
Creating socket...
Socket couldn't be created
Trying to re-acquire CAP_NET_RAW capability
Capability couldn't be re-acquired. Error: Operation not permitted
Finished
```

```
#include <linux/capability.h>
#include <sys/capability.h>
```

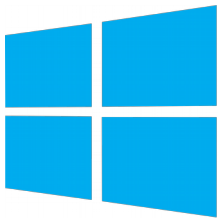
```
cap_get_proc(...);
cap_set_flag(...);
cap_set_proc(...);
```

Análisis de Malware

- Limitaciones de las técnicas anteriores
  - Modelo de seguridad en Linux
    - Para debuggear arbitrariamente procesos (de otros usuarios), se requiere la capability CAP_SYS_PTRACE

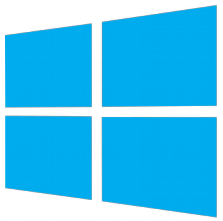
Análisis de Malware

- Analizar las librerías y funciones externas que utiliza el malware puede darnos una idea de su comportamiento
  - Es muy probable que el uso de APIs de debug este relacionado a la inyección en procesos



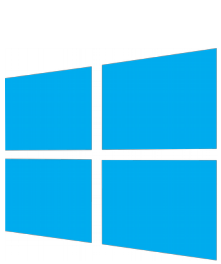
Análisis de Malware

- Conociendo DLLs de Windows
  - Kernel32.dll
    - DLL base. Manipulación de memoria, archivos, hardware. Todos los binarios ejecutables la importan
  - Advapi32.dll
    - Acceso al Service Manager y al Registry
  - User32.dll
    - Componentes de interfaz gráfica (botones, scroll bars, áreas de texto, etc.)



Análisis de Malware

- Conociendo DLLs de Windows
  - Gdi32.dll
    - Gráficos. Librería en espacio de usuario. Win32k.sys en kernel
  - WSock32.dll, Ws2_32.dll y Wininet.dll
    - Librerías de networking (sockets, conexiones HTTP, etc.)
  - Msvcrt.dll
    - Runtime de C/C++. Capa de abstracción por encima de la API de Windows. Alocación de memoria, archivos, strings, etc.
  - Ntdll.dll
    - No documentada pero presente en todos los procesos. Interfaz al kernel



Análisis de Malware

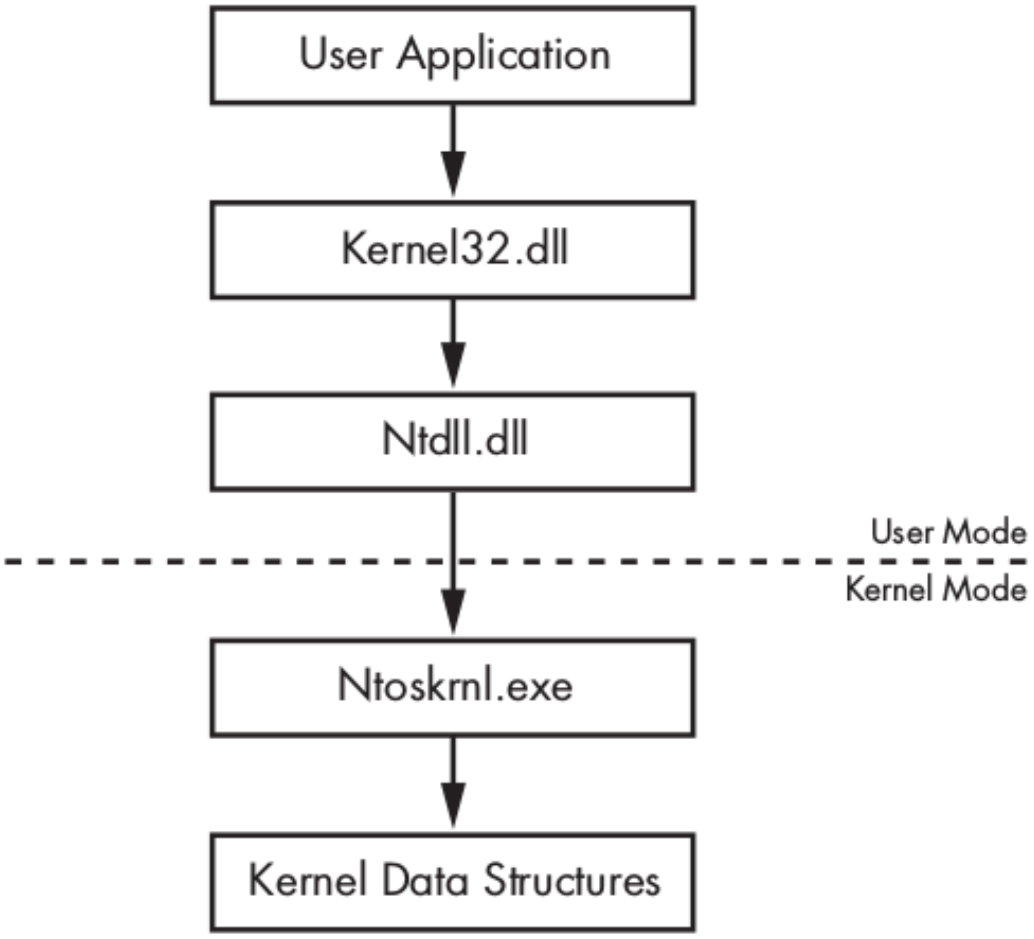


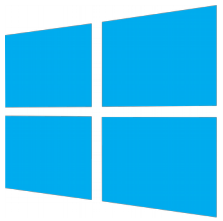
Figure 7-3: User mode and kernel mode

Imagen de “Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software”

Ntdll.dll es interesante porque incluye:

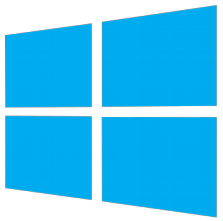
- Estructuras de kernel
- APIs no documentadas, que permiten funcionalidad extra (o bypassear restricciones de APIs de más alto nivel)
- Evitar importar funciones “sospechosas”

Eventualmente el malware puede realizar syscalls directo al kernel, basado en lo aprendido de ntdll.dll (ya que las syscalls no están documentadas)



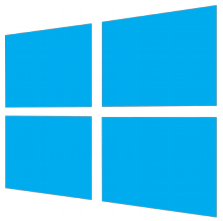
Análisis de Malware

- ¿Cómo funciona un keylogger?
  - Desafío: manejar una gran cantidad de información
  - API SetWindowsHookEx
    - El malware installa un hook (callback) para un cierto tipo de evento
      - En el caso de un keylogger, ese evento es WH_KEYBOARD_LL
    - Los hooks pueden ser globales o acotados a un cierto thread
    - Esta técnica se puede utilizar para inyectar DLLs en procesos: el callback (implementado en una DLL) es llamado en el contexto del proceso que genera el evento



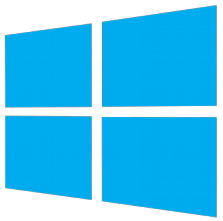
Análisis de Malware

- ¿Cómo funciona un keylogger?
  - Limitaciones
    - La seguridad discrecional (por usuario o grupo) es insuficiente: ¿qué sucede si un malware bajado de Internet es ejecutado por un usuario administrador? ¿qué sucede si el navegador de Internet es explotado remotamente?
    - Seguridad mandatoria: los secure objects y los procesos tienen un nivel de integridad asignado
      - Un proceso de baja integridad no puede leer o escribir un objeto de alta integridad
      - Un proceso de baja integridad no puede instalar un hook para un keylogger



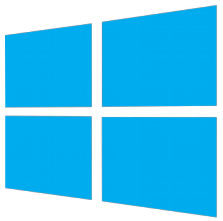
Análisis de Malware

- COM – Component Object Model
  - Framework de comunicación orientado a objetos
    - Comunicación en el mismo proceso, entre procesos o entre procesos de hosts distribuidos (DCOM)
    - Bindings para diferentes lenguajes. Ejemplo: desde VBAScript puedo invocar funcionalidad de una DLL desarrollada en C++
      - Usado por Internet Explorer y Microsoft Office entre otros
    - Marshalling de parámetros. Normalización de los tipos de datos. Reference counting de objetos



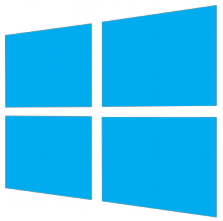
Análisis de Malware

- COM – Component Object Model
  - ABI estable, independiente del lenguaje y del compilador
    - La comunicación ocurre sobre mecanismos de más bajo nivel
      - Por ejemplo, DCOM puede utilizar SMB y TCP/IP como transporte



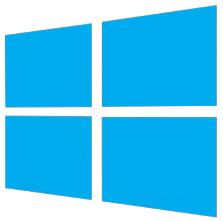
Análisis de Malware

- COM – Component Object Model
  - Funciona en modalidad cliente-servidor
  - El servidor “expone” un objeto (componente reusable) a diferentes clientes
  - El objeto implementa una o más interfaces (IIDs). Ej. IWebBrowser2. La implementación concreta del objeto (clase, identificada por un CLSID) puede ser una DLL o un binario ejecutable. Ej. Internet Explorer
  - El cliente consume los servicios ofrecidos por el objeto llamando a métodos y propiedades



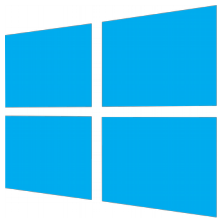
Análisis de Malware

- COM – Component Object Model
  - Un cliente localiza a un objeto expuesto a través del Registry
    - Las interfaces y clases se identifican por GUIDs (numeros únicos de 128 bits)
    - HKLM\SOFTWARE\Classes\CLSID\ y HKCU\SOFTWARE\Classes\CLSID
    - OleInitialize, CoInitializeEx, CoCreateInstance
  - COM está implementado en DLLs como Ole32.dll, Oleauto32.dll y tecnologías como ActiveX



Análisis de Malware

- COM – Component Object Model
  - Los métodos retornan siempre HRESULT para indicar el resultado de la llamada
  - Los valores de salida van por parámetros que son punteros. Se especifica el tipo de parámetro como [IN] y [OUT] en la documentación
  - Un objeto siempre implementa la interfaz IUnknown. Esta interfaz permite:
    - Modificar el reference counting a un objeto (AddRef, Release)
    - Obtener punteros a otras interfaces implementadas por el objeto (“casting”)

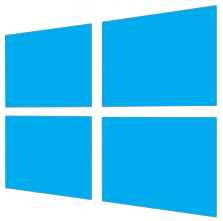


Análisis de Malware

```
mov     edx, [ebp+bstrString]
push    edx
mov     eax, [ebp+var_4]
mov     ecx, [eax]
EIP
mov     edx, [ebp+var_4]
push    edx
mov     eax, [ecx+2Ch]
call    eax
mov     [ebp+var_10], eax
cmp     [ebp+var_10], 0
```

```
EAX 002F5AF8 ↳ debug036:002F5AF8
EBX 7EFDE000 ↳ debug150:7EFDE000
ECX 6CD01D74 ↳ ieproxy:ieproxy_Dll
EDX 0094474C ↳ debug162:0094474C
ESI 00148D0C ↳ .data:dword_148D0C
EDI 00148D10 ↳ .data:dword_148D10
EBP 004CFEC8 ↳ debug045:004CFEC8
ESP 004CFE94 ↳ debug045:004CFE94
EIP 001310F2 ↳ _main+F2
EFL 00000246
```

```
IWebBrowser2* pObjBrowser2;
CoCreateInstance(...);
pObjBrowser2->Navigate();
```



Análisis de Malware

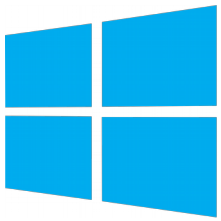
```
EAX 002F5AF8 ↳ debug036:002F5AF8
```

EAX = puntero al objeto (heap)

En la primeros bytes de la memoria del objeto hay un pointer a la vtable de la clase.

La vtable es una tabla de punteros a la implementación de los métodos de la clase.

Después del puntero a la vtable están los atributos del objeto.

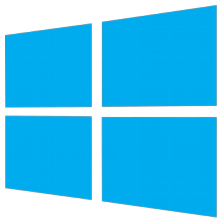


Análisis de Malware

```
ECX 6CD01D74 ↳ ieproxy:ieproxy_Dll
```

ECX = Puntero a la vtable de la clase del objeto (interfaz IWebBrowser2)

```
6CD01D74   6CD02EB0   ieproxy:ieproxy_DllGetClassObject+4760
6CD01D78   6CD0A6F0   ieproxy:ieproxy_GetProxyDllInfo+2160
6CD01D7C   6CD01F70   ieproxy:ieproxy_DllGetClassObject+3820
6CD01D80   6CD0A9D0   ieproxy:ieproxy_GetProxyDllInfo+2440
6CD01D84   6CD0A980   ieproxy:ieproxy_GetProxyDllInfo+23F0
6CD01D88   6CD0A8E0   ieproxy:ieproxy_GetProxyDllInfo+2350
6CD01D8C   6CD0AB20   ieproxy:ieproxy_GetProxyDllInfo+2590
6CD01D90   6CD0AA20   ieproxy:ieproxy_GetProxyDllInfo+2490
6CD01D94   6CD0AA60   ieproxy:ieproxy_GetProxyDllInfo+24D0
6CD01D98   6CD0AAA0   ieproxy:ieproxy_GetProxyDllInfo+2510
6CD01D9C   6CD0AAE0   ieproxy:ieproxy_GetProxyDllInfo+2550
6CD01DA0   6CD0AB80   ieproxy:ieproxy_GetProxyDllInfo+25F0
```

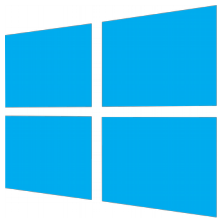


Análisis de Malware

- Debido a que la DLL que implementa la clase del objecto se puede localizar en diferentes direcciones virtuales, la vtable no se encuentra necesariamente en una dirección fija
- Los valores de la vtable (punteros a la implementación de los métodos) pueden variar de proceso en proceso por la misma razón

Demo 5.3

Llamada a objeto COM (Windows)



Análisis de Malware

```
typedef struct tagVARIANT {
  union {
    struct __tagVARIANT {
      VARTYPE vt;
      WORD    wReserved1;
      WORD    wReserved2;
      WORD    wReserved3;
      union {
        LONGLONG          llVal;
        LONG              lVal;
        BYTE              bVal;
        SHORT             iVal;
        FLOAT             fltVal;
        DOUBLE            dblVal;
        ...
      }
      ...
    }
  }
} VARIANT, …;
```

Estructura para representar parámetros de tipo “genérico”. Tienen mayor overhead pero la ventaja de que su tipo no tiene por qué ser conocido en tiempo de compilación.

El valor VARTYPE vt permite identificar el tipo del parámetro e interpretar el valor de forma correcta.

Los objetos que implementan la interfaz IDispatch permiten introspección: consultar los métodos y propiedades en tiempo de ejecución e invocarlos. Esta interfaz necesita parámetros y valores de retorno genéricos, porque dependen de cada implementación.

Análisis de Malware

- Rootkits
  - Malware que logra escalar privilegios y ejecutar en ring0 (ej. cargar driver)
  - Es necesario debuggear kernel para detectarlo
  - Puede modificar todas las estructuras del kernel para ocultarse de espacio de usuario (ej.: removerse de la lista de procesos u ocultar los puertos donde está escuchando)
    - Evade anti-virus

Análisis de Malware

- Rootkits
  - Tiene visibilidad global de todo lo que sucede en el sistema: memoria de los procesos y syscalls
    - Hooking en la sys_call_table, SSDT o vector de interrupciones
    - Puede escribir memoria read-only (el procesador está en modo privilegiado cuando ejecuta el rootkit)
  - Puede intentar persistirse dentro de algún firmware (y resistir formateos del disco)

Lab

Lab 5.1: Modificar el código de la Demo 5.1 (inyección por Create Remote Thread) para llamar a la función “GetCommandLine” en el proceso inyectado y guardar el resultado en un archivo.

Lab 5.2: Modificar el código de la Demo 5.2 (inyección por ptrace) para llamar a la función “getpid” en el proceso inyectado y guardar el resultado en un archivo.

Lab

Lab 5.3: Modificar el código de la Demo 5.2 (inyección por ptrace) para interceptar las llamadas que la aplicación inyectada haga a una función elegida y registrarlas en un archivo.

Referencias

- http://resources.infosecinstitute.com/using-createremotethread-for-dll-injection-on-windows
- https://msdn.microsoft.com/es-es/library/windows/desktop/ms682437(v=vs.85).aspx
- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software