Reverse Engineering
Class 5

Malware Analysis

Malware Analysis

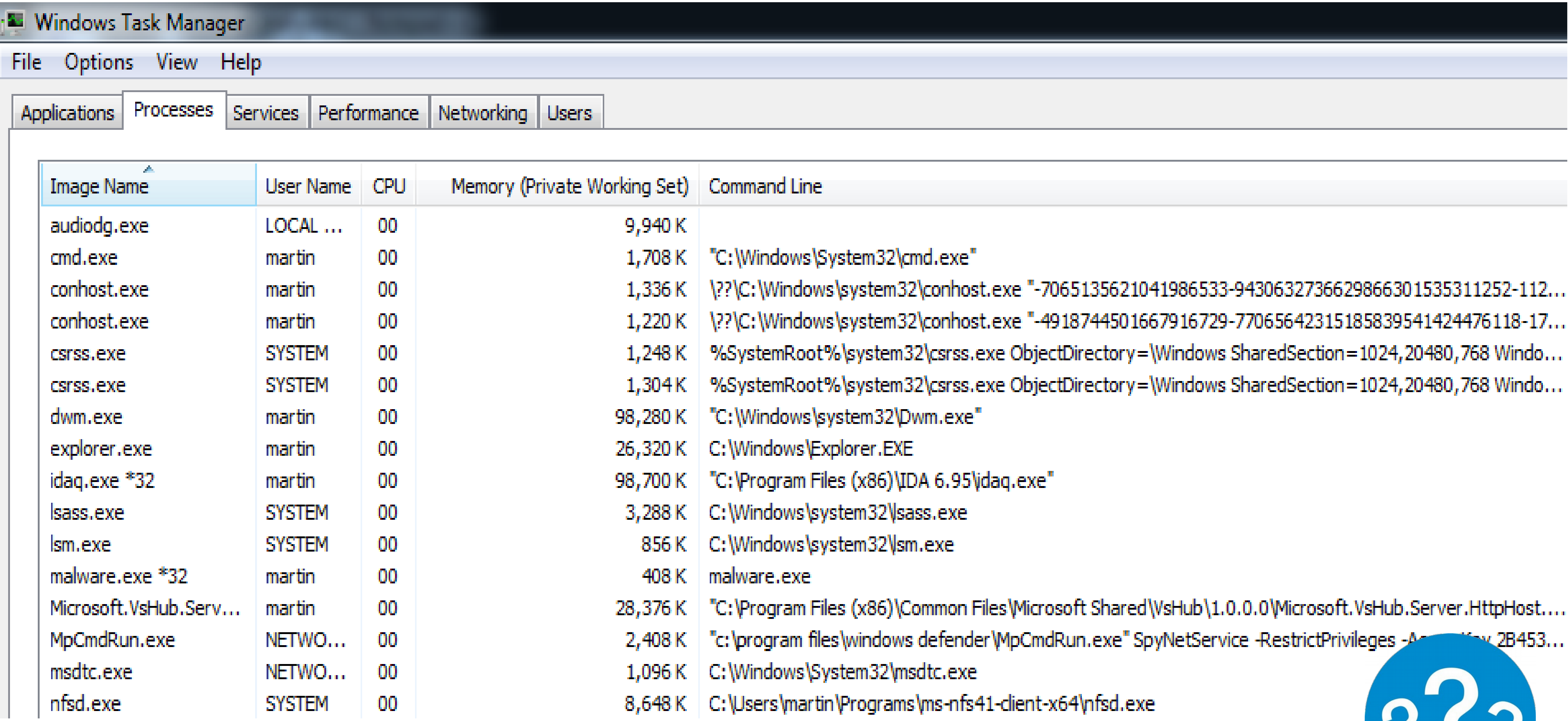


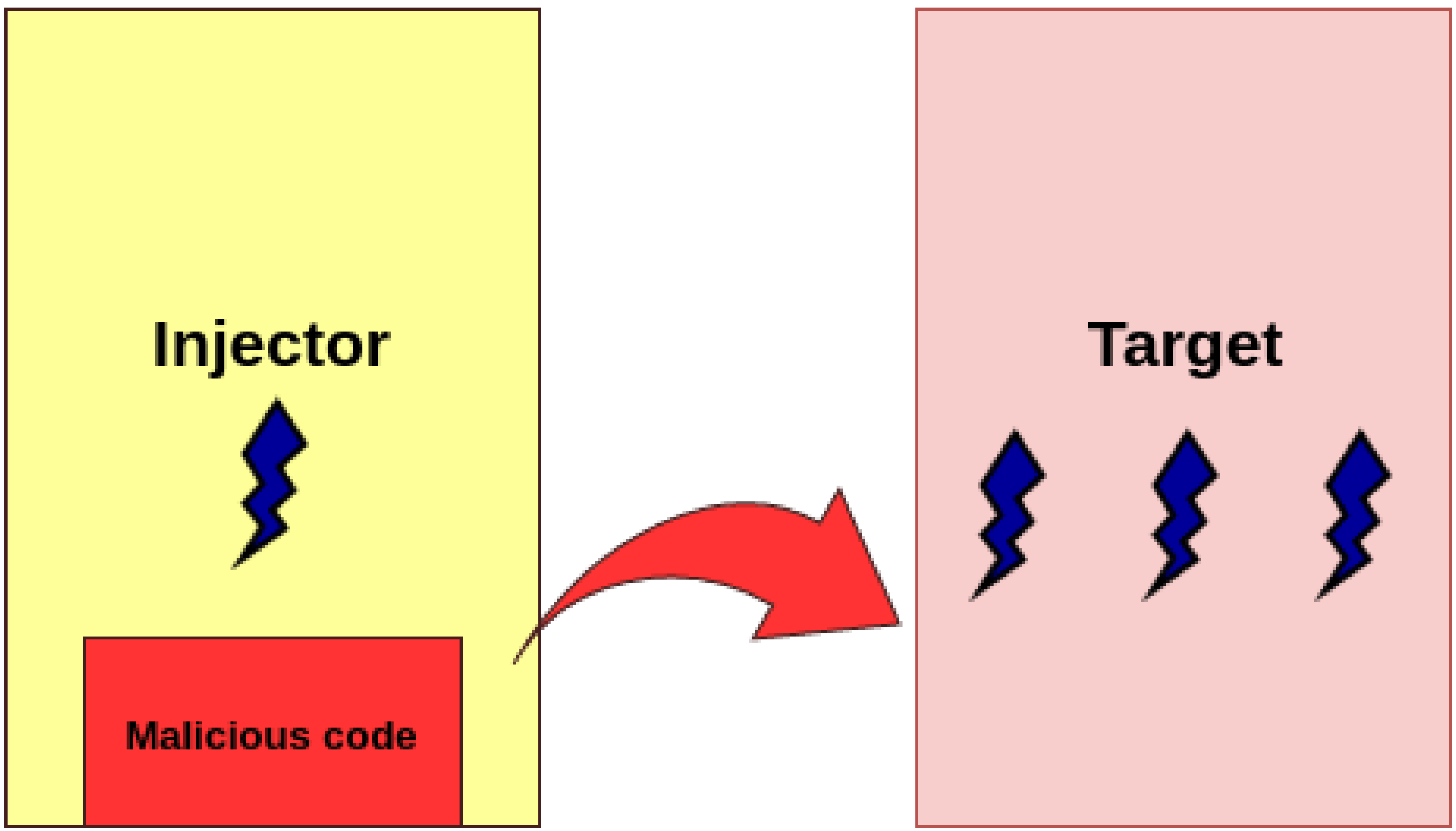
Image Name	User Name	CPU	Memory (Private Working Set)	Command Line
audiodg.exe	LOCAL ...	00	9,940 K	
cmd.exe	martin	00	1,708 K	"C:\Windows\System32\cmd.exe"
conhost.exe	martin	00	1,336 K	\??\C:\Windows\system32\conhost.exe "-7065135621041986533-9430632736629866301535311252-112...
conhost.exe	martin	00	1,220 K	\??\C:\Windows\system32\conhost.exe "-4918744501667916729-77065642315185839541424476118-17...
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[illegible] 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

Something odd?

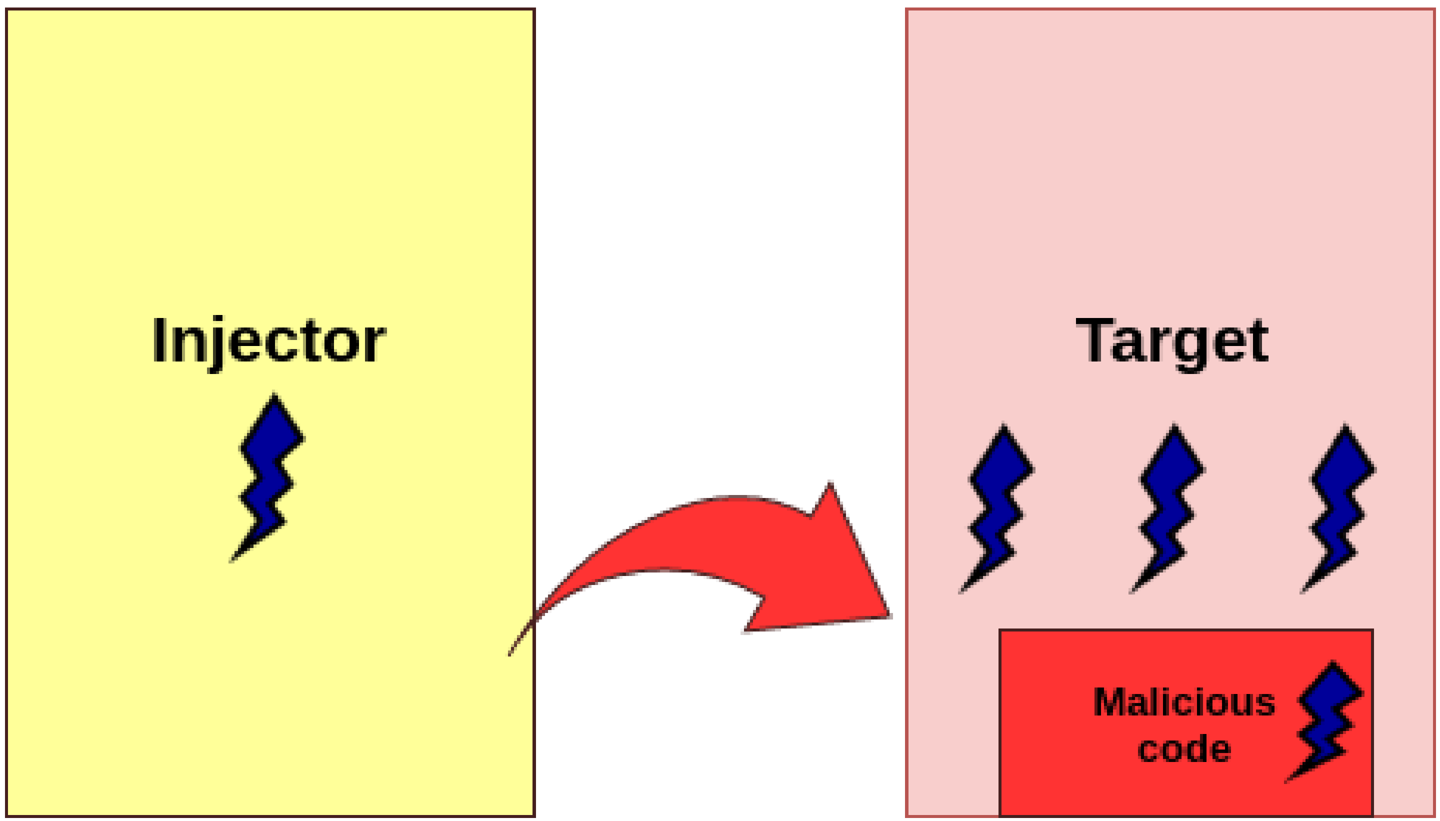
Malware Analysis

- Process injection
  - Hide and evade audit logs
  - Bypass endpoint firewalls with application filtering
  - Steal information from the injected process
  - Change session (non-interactive session → interactive session)
    - Screenshots

Malware Analysis



Malware Analysis



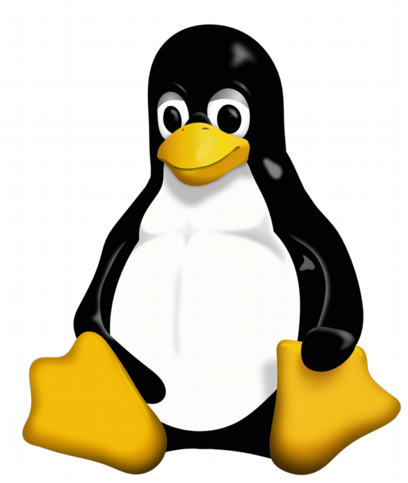
Malware Analysis

- Goals
  - Move malicious code to the target process
  - Create a new thread in the target process that executes malicious code
  - Avoid interruptions or data corruption in the target process (process continuation)

Malware Analysis

- Process injection techniques (Windows)
  - CreateRemoteThread
  - Asynchronous Procedure Call
  - Debugging API

- Process injection techniques (Linux)
  - Debugging API (ptrace)



Malware Analysis

- CreateRemoteThread (Windows)
  - OpenProcess
    - Handle to manage a remote process
  - VirtualAllocEx
    - Allocate memory to the remote process
      - Write and execution permissions

Malware Analysis

- CreateRemoteThread (Windows)
  - WriteProcessMemory
    - Write remote process memory
      - Malicious code to inject
  - CreateRemoteThread
    - Create a new thread in the remote process and make it execute previously written memory

Demo 5.1

Process injection with CreateRemoteThread (Windows)

Malware Analysis

- Asynchronous Procedure Call (Windows)
  - Windows API to queue asynchronous calls to threads
    - When thread is in “alertable” state (I.e. sleeping), it will handle the call
    - Call is to an arbitrary process address, chosen by the one who enqueues it
    - When call ends, thread context is automatically restored

Malware Analysis

- Asynchronous Procedure Call (Windows)
  - A handle to the victim process and to a thread in it are obtained: OpenProcess, CreateToolhelp32Snapshot, OpenThread
  - Memory is allocated in the process and executable code is written: VirtualAllocEx y WriteProcessMemory

Malware Analysis

- Asynchronous Procedure Call (Windows)
  - An APC call is enqueued with QueueUserAPC
  - It’s important that injected code creates a new thread to continue execution: thread that handles the APC has to return to its normal execution
    - If an application thread is definitely interrupted, instability may be caused

Malware Analysis

- Debugging API (Windows)
  - Allocate memory and write code into the target process (remotely)
  - Debug the target process
    - Debugger is attached to a thread
    - DebugActiveProcess / WaitForDebugEvent / ContinueDebugEvent
  - Save the attached thread context (I.e. save registers values in the injector process)

Malware Analysis

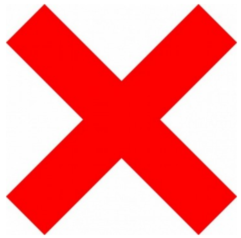
- Debugging API (Windows)
  - Attached thread is set to execute injected code
    - A new thread has to be created at the beginning of the injected code
  - Control returns to debugger and the attached thread original context is restored

Malware Analysis

- Process injection in Linux
  - Similar to the technique described for Windows
  - Debugging API in Linux (Unix): ptrace
  - Read / write the thread context (registers)
  - Read / write process memory
  - Intercept every signal to the debugged process
  - Run the process step-by-step

Malware Analysis

- Process injection in Linux
  - Problem:
    - It’s not possible to remotely allocate memory to a process
    - It’s not possible to remotely create a thread on a process



  - Solution:
    - Hijack a thread and make it do it on our behalf

Malware Analysis

- Some ptrace primitives
  - PTRACE_ATTACH
  - PTRACE_PEEKDATA
  - PTRACE_POKEDATA
  - PTRACE_SYSCALL
  - PTRACE_CONT
  - PTRACE_GETREGS
  - PTRACE_SETREGS

Malware Analysis

- Process injection in Linux
  - Attach to the target process
  - Resolve mmap and __clone virtual addresses (libc)
    - Libc base in /proc/<PID>/maps
    - Resolution reading memory (ELF format)
    - Workaround: resolve offset with dlsym and dladdr inside the injector and use it on the injected process

Malware Analysis

- Process injection in Linux
  - Save attached thread context to restore it at the end of injection
    - Values are saved on the injector memory
    - We want the injected process to continue execution normally

Malware Analysis

- Process injection in Linux
  - Modify attached thread context (hijacking):
    - RIP → mmap / __clone address
    - Other registers → function parameters (according to x86_64 ABI)
    - Modify stack: 16 bytes alignment (ABI) and return address = 0x0

Malware Analysis

- Process injection in Linux
  - Continue the process and let the called function execute. When returning, instruction at address 0x0 will be executed and a signal sent to the process (invalid address)
    - Given that the injector is a debugger, receives the signal first and can handle it
      - Signal is discarded, instead of forwarding it to the debugged process
    - Modify the attached thread context to execute another function or restore it to continue normal execution

Malware Analysis

- Process injection in Linux
  - Function calls in the remote process:
    - Allocate memory for the executable buffer (injected instructions), with mmap
    - Allocate memory for the stack of the thread that is going to execute the injected buffer, with mmap
    - Create a new thread, with __clone

Malware Analysis

- Process injection in Linux
  - How registers or memory have to be set for each call?
    - Application Binary Interface (ABI), according to the architecture
    - Tip: debug a simple example that uses libc API and follow it until syscall is executed

Demo 5.2

Process injection with ptrace (Linux)

Malware Analysis

- Limits of previous techniques
  - Security model in Windows
    - Access Token – object that describes the security context of a process or thread (GetTokenInformation)
      - When user logs into the system, an access token is assigned. Every process that the user executes have this token
      - Contains user account identity and its groups: SIDs (Security Identifiers)
      - Contains privileges for administrative tasks (I.e. reboot the system, change date, load drivers, etc.)
      - One thread may eventually impersonate a different user and use its access token

Malware Analysis

- Limits of previous techniques
  - Security model in Windows
    - Security Descriptors: security information associated to each Securable Object
      - Owner and primary group
      - DACL – discretionary access (access to specific users or groups)

Malware Analysis

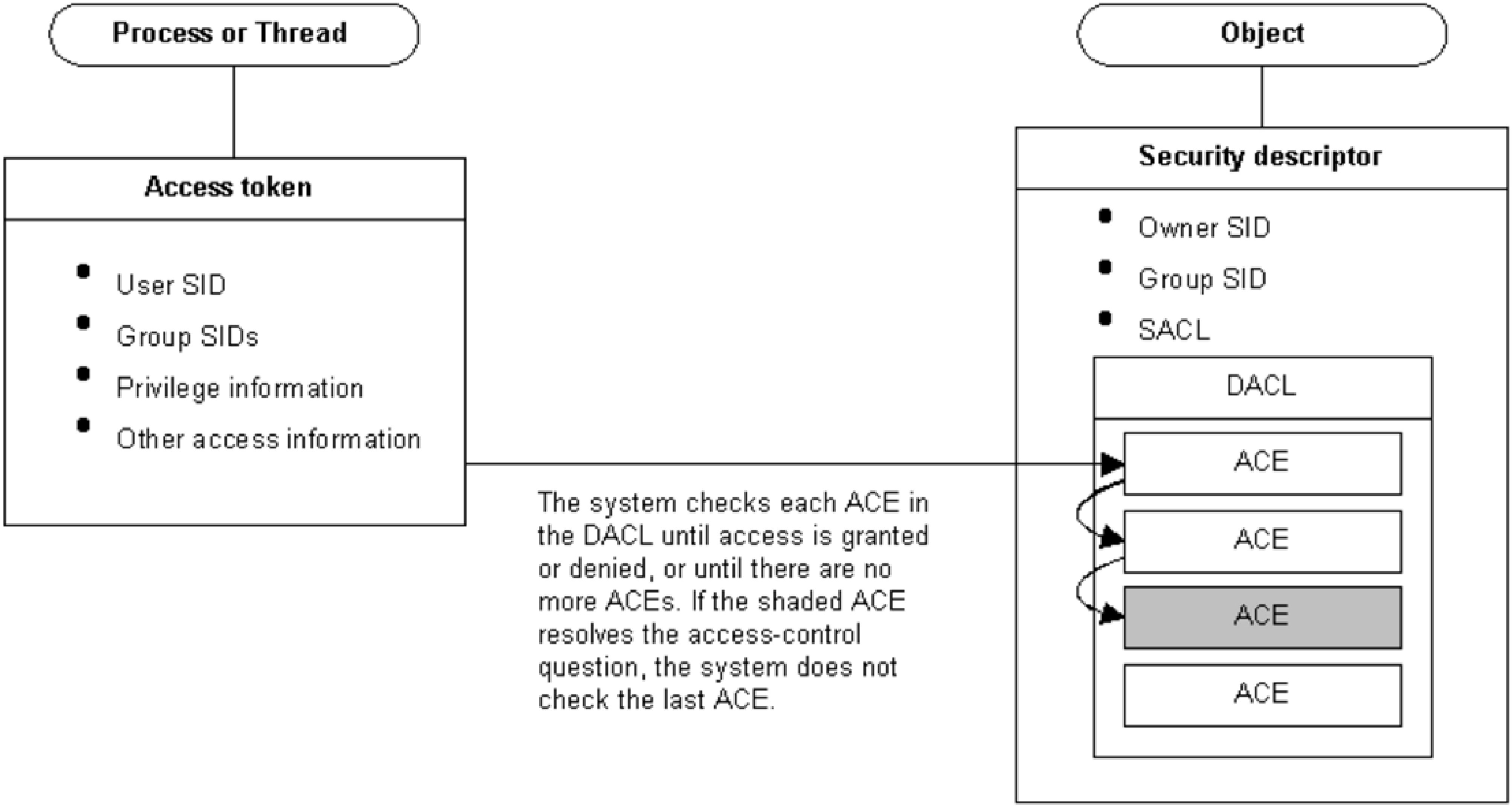


Image from https://msdn.microsoft.com/en-us/library/windows/desktop/aa378890(v=vs.85).aspx

Malware Analysis

cmd64 - no admin

```
Getting process token (OpenProcessToken):
00000048
Getting process token session ID (GetTokenInformation - TokenSessionId):
1
Getting process token user (GetTokenInformation - TokenUser):
token_user.User.Sid: 00403EA8
S-1-5-21-2496531130-470213764-2444830404-1001
Getting process token user (GetTokenInformation - TokenPrivileges):
Privileges count: 5
privilege_name: SeShutdownPrivilege
privilege_name: SeChangeNotifyPrivilege
privilege_name: SeUndockPrivilege
privilege_name: SeIncreaseWorkingSetPrivilege
privilege_name: SeTimeZonePrivilege
```

Not elevated

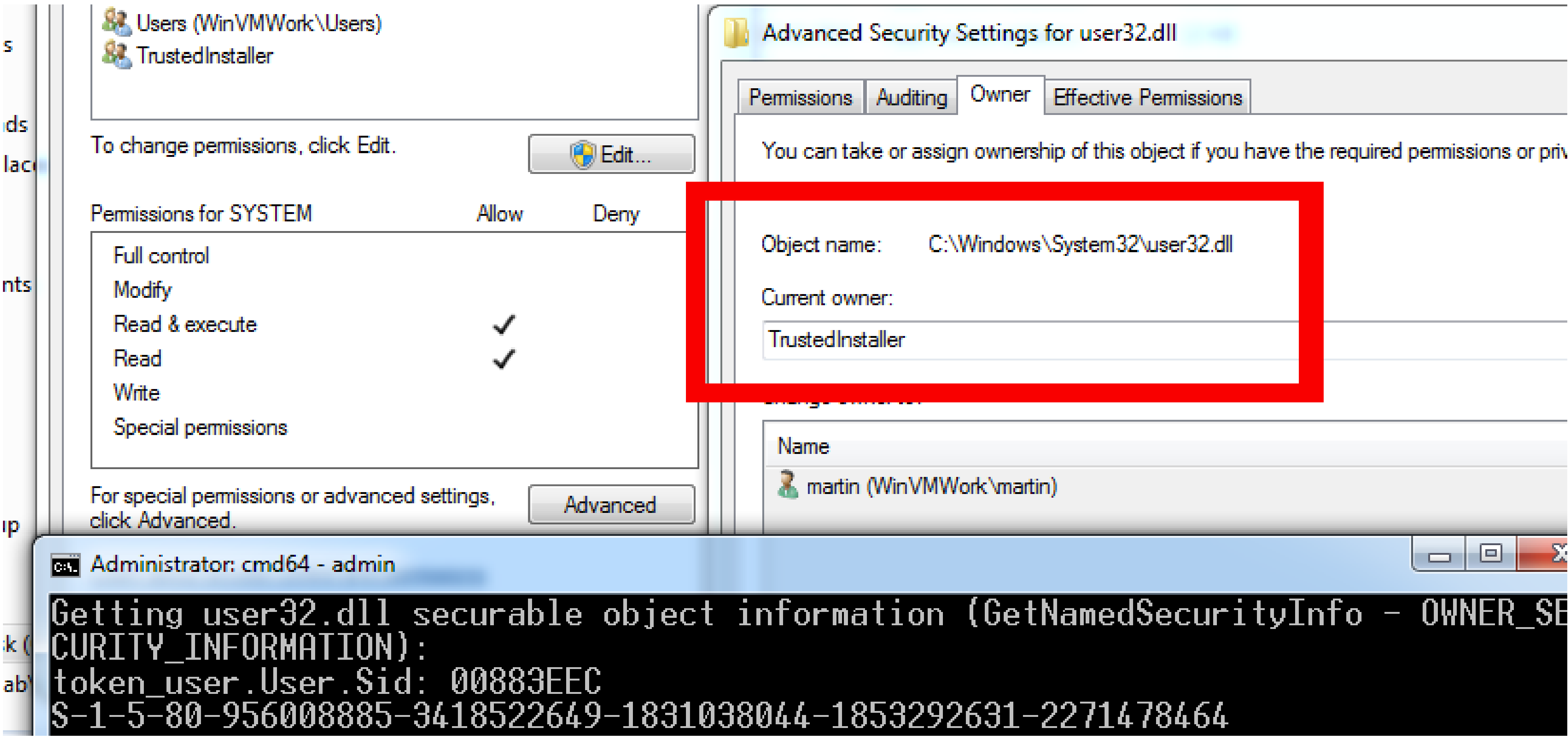
Same SID

Administrator: cmd64 - admin

```
Getting process token (OpenProcessToken):
00000048
Getting process token session ID (GetTokenInformation - TokenSessionId):
1
Getting process token user (GetTokenInformation - TokenUser):
token_user.User.Sid: 004B1280
S-1-5-21-2496531130-470213764-2444830404-1001
Getting process token user (GetTokenInformation - TokenPrivileges):
Privileges count: 23
privilege_name: SeIncreaseQuotaPrivilege
privilege_name: SeSecurityPrivilege
privilege_name: SeTakeOwnershipPrivilege
privilege_name: SeLoadDriverPrivilege
privilege_name: SeSystemProfilePrivilege
privilege_name: SeSystemtimePrivilege
privilege_name: SeProfileSingleProcessPrivilege
privilege_name: SeIncreaseBasePriorityPrivilege
privilege_name: SeCreatePagefilePrivilege
privilege_name: SeBackupPrivilege
privilege_name: SeRestorePrivilege
privilege_name: SeShutdownPrivilege
privilege_name: SeDebugPrivilege
privilege_name: SeSystemEnvironmentPrivilege
privilege_name: SeChangeNotifyPrivilege
privilege_name: SeRemoteShutdownPrivilege
privilege_name: SeUndockPrivilege
privilege_name: SeManageVolumePrivilege
```

Elevated

Malware Analysis



Malware Analysis

- To invoke OpenProcess and other debugging APIs in a process from a different user, “SeDebugPrivilege” privilege has to be enabled in the Access Token
  - Only administrative accounts should have this privilege available to be enabled
  - To debug processes from the same user this privilege is not needed
  - From a defensive point of view, this remarks the importance of not executing with administrative accounts or, in that case, impersonate non-privileged users
  - Model brings granularity to assign non-privileged accounts the privileges needed (least privilege principle)

Malware Analysis

- Limits of previous techniques
  - Security model in Linux
    - Before kernel 2.2, security model consisted of privileged and non-privileged. A privileged process had unrestricted control of the whole system
    - Some software legitimately requires privileges. In example, a DNS server has to listen incoming connections in a low port (53)

Malware Analysis

- Limits of previous techniques
  - Security model in Linux
    - Under the assumption that the process might be exploited, damage mitigation is needed
    - “Capabilities” bring privilege granularity to processes
    - “Capabilities” are associated to executable binaries.
    - A process that drops “capabilities” in run time, cannot re-acquire them later

Malware Analysis

```
[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(...);
```

Malware Analysis

- Limits of previous techniques
  - Security model in Linux
    - To arbitrary debug processes (from different users), CAP_SYS_PTRACE capability is required

Malware Analysis

- Analyzing external libraries and functions that malware uses may give an idea of its behavior
  - I.e if debugging APIs are used, it’s likely that the malware has process injection capabilities

Malware Analysis

- Getting familiar with DLLs in Windows
  - Kernel32.dll
    - Base DLL. Memory, files and hardware management. Imported by every executable binary
  - Advapi32.dll
    - Access to Service Manager and Registry
  - User32.dll
    - Graphic interface components (buttons, scroll bars, text areas, etc.)

Malware Analysis

- Getting familiar with DLLs in Windows
  - Gdi32.dll
    - Graphics management. User space library. Win32k.sys in kernel
  - WSock32.dll, Ws2_32.dll y Wininet.dll
    - Networking libraries (sockets, HTTP connections, etc.)
  - Msvcrt.dll
    - C/C++ runtime. Abstraction layer on top of Windows API. Memory allocation, files, strings, etc.
  - Ntdll.dll
    - Not documented but present in every process. Interface to kernel

Malware Analysis

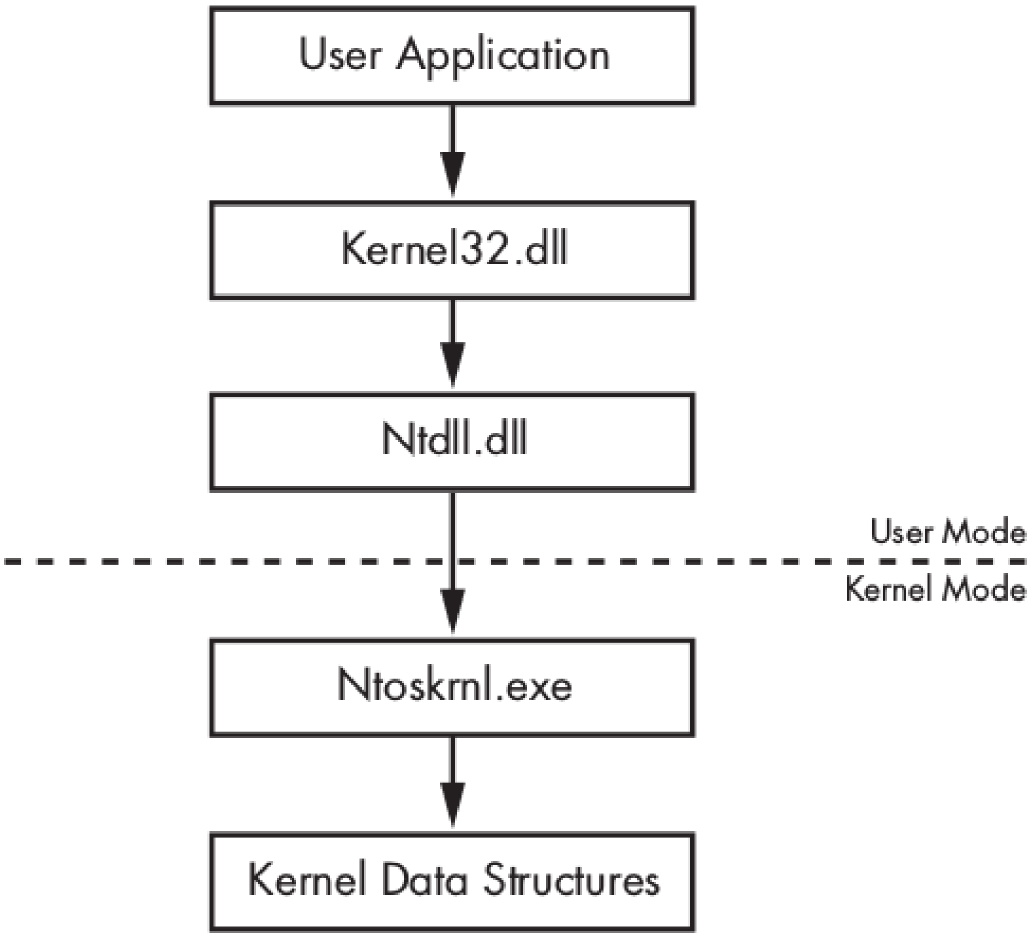


Figure 7-3: User mode and kernel mode

Image from "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software"

Ntdll.dll is interesting because it includes:

- Kernel structures
- Not documented APIs, that enable extra functionality (or high level APIs restrictions bypassing)
- Avoid importing "suspicious" functions

Malware can eventually execute direct kernel syscalls, based on what ntdll.dll does (syscalls are not documented)

Malware Analysis

- How does a keylogger work?
  - Challenge: mange a huge amount of data
  - API SetWindowsHookEx
    - Malware installs a hook (callback) for a specific event
      - In case of a keylogger, that event is WH_KEYBOARD_LL
    - Hooks can be global or thread-specific
    - This technique can be used to inject DLLs in processes: callback (implemented in a DLL) is called in the context of the process that generates the event

Malware Analysis

- How does a keylogger work?
  - Limits
    - Discretionary security (per user or group) is not enough: what happens if a malware downloaded from the Internet gets executed by an administrative user? What happens if the Internet browser is remotely exploited?
    - Mandatory security: securable objects and processes have an assigned integrity level
      - A low integrity process cannot read or write a high integrity object
      - A low integrity process cannot install a keylogger hook

Malware Analysis

- COM – Component Object Model
  - Object oriented communication framework
    - Communication within the same process, between processes or between processes on distributed hosts (DCOM)
    - Bindings for different languages. Example: from VBAScript a function on a DLL (developed in C++) can be invoked
      - Used by Internet Explorer and Microsoft Office among others
    - Parameters marshalling. Data types normalization. Objects reference counting

Malware Analysis

- COM – Component Object Model
  - Stable ABI, independent from the language and compiler
    - Communication happens on top of low level mechanisms
      - In example, DCOM can use SMB and TCP/IP as transport

Malware Analysis

- COM – Component Object Model
  - Works in client-server mode
  - Server exposes an object (reusable component) to be used by different clients
  - Object implements one or more interfaces (IIDs). I.e. IWebBrowser2. Object concrete implementation (class, identified by a CLSID) can be a DLL or an executable binary. I.e. Internet Explorer
  - Client consumes services offered by the object calling its methods or properties

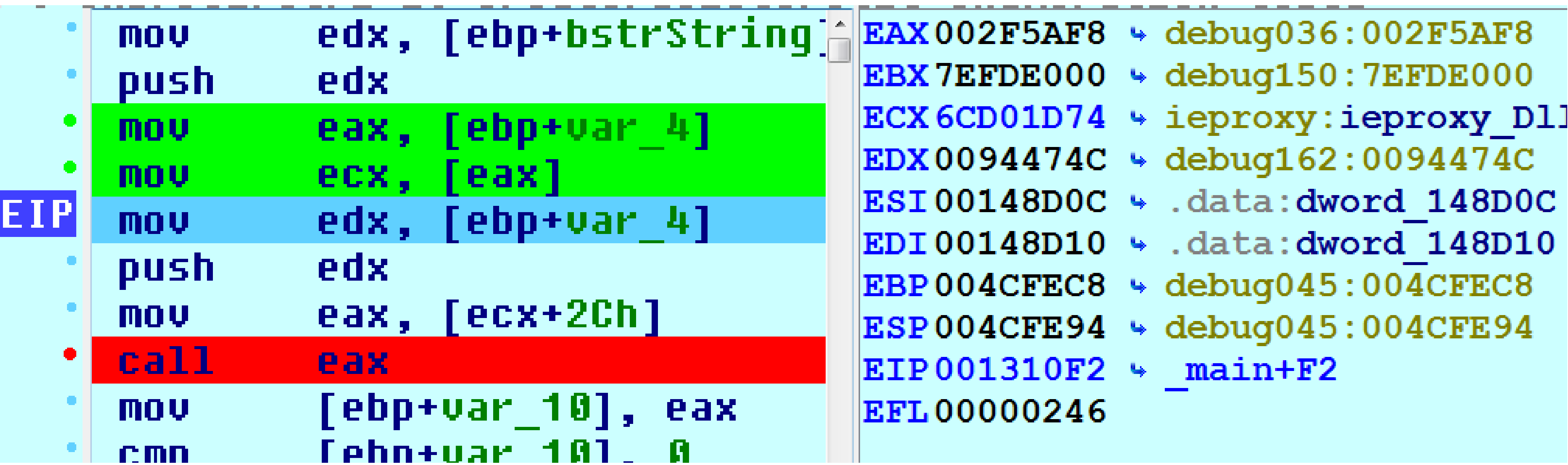
Malware Analysis

- COM – Component Object Model
  - A client locates an object published in the Registry
    - Interfaces and classes are identified by GUIDs (unique numbers 128 bits long)
    - HKLM\SOFTWARE\Classes\CLSID\ and HKCU\SOFTWARE\Classes\CLSID
    - OleInitialize, CoInitializeEx, CoCreateInstance
  - COM is implemented in DLLs like Ole32.dll, Oleauto32.dll and technologies like ActiveX

Malware Analysis

- COM – Component Object Model
  - Methods always return HRESULT to indicate the result of the call
  - Return values go through pointer parameters. Parameter types are specified with [IN] and [OUT] in documentation
  - An object always implements IUnknown interface. This interface allows to:
    - Modify the object reference counter (AddRef, Release)
    - Obtain pointers to other interfaces implemented by the object (“casting”)

Malware Analysis




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

Malware Analysis

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

EAX = pointer to the object (heap)

In the first bytes of the object memory there is a pointer to the class vtable.

vtable is a table of pointers to the implementation of class methods.

After vtable pointer, object attributes are located.

Malware Analysis

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

ECX = pointer to object's class vtable (IWebBrowser2 interface)

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

Malware Analysis

- vtable is not necessarily in a fixed address because the DLL that implements the object class may be located at any virtual address
- vtable values (pointers to method implementations) may change from process to process for the same reason

Demo 5.3

COM object call (Windows)

Malware Analysis

```
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, …;
```

Structure to represent “generic” parameter types. Has more overhead but the advantage of data type being unknown in compile time.

VARTYPE vt value allows to identify the parameter type and correctly interpret the value.

Objects that implement IDispatch interface allow introspection: query methods and properties in run time and invoke them. This interface requires generic parameters and return values, because they depend on each implementation.

Malware Analysis

- Rootkits
  - Malware that manages to escalate privileges and execute in ring0 (I.e. load a driver)
  - It’s necessary to debug kernel to detect it
  - May modify kernel structures to hide from user space (I.e.: remove itself from processes list or hide listening ports)
    - Evades anti-virus

Malware Analysis

- Rootkits
  - Has global system visibility: processes memory and syscalls
  - Hooks sys_call_table, SSDT or interruption vector
  - May write read-only memory (processor is in privileged mode when executing the rootkit)
  - May try to persist in a firmware (and resist disk formatting)

Lab

Lab 5.1: Modify Demo 5.1 code (Create Remote Thread injection) to call “GetCommandLine” function in the injected process and save the result to a file.

Lab 5.2: Modify Demo 5.2 code (ptrace injection) to call “getpid” function in the injected process and save the result to a file.

Lab

Lab 5.3: Modify Demo 5.2 code (ptrace injection) to intercept calls that the injected application does to a chosen function and log them to a file.

References

- 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