CYBR 473 T1 2023

Malware and Reverse Engineering

Analysing Malicious Windows Programs (B)

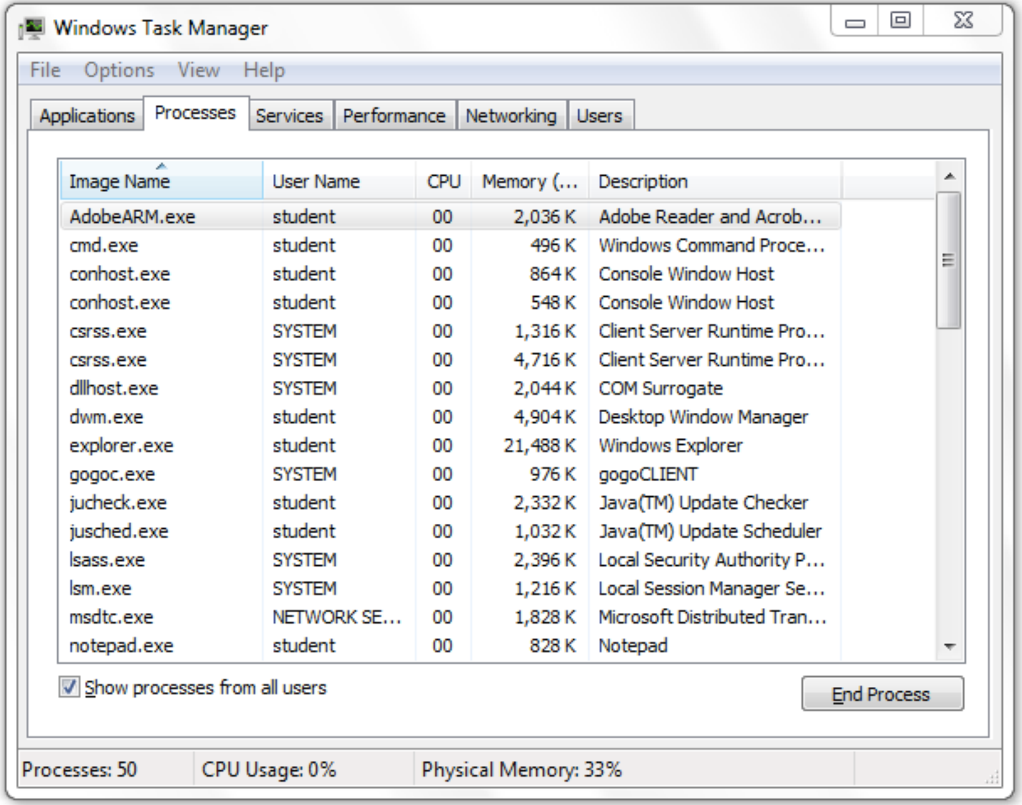
Chapter 7: "*Practical Malware Analysis: The Hands-on Guide to Dissecting Malicious Software*", Michael Sikorski and Andrew Honig, 2012

PROCESSES

Processes

- Every program being executed by Windows is a **process**
- Each process has its **own resources**
    - Handles, memory
- Each process has one or more **threads**
- Older malware ran as an independent process
- Newer malware executes its code as part of another process

Many Processes Run at Once



Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users

Image Name	User Name	CPU	Memory (...	Description
AdobeARM.exe	student	00	2,036 K	Adobe Reader and Acrob...
cmd.exe	student	00	496 K	Windows Command Proce...
conhost.exe	student	00	864 K	Console Window Host
conhost.exe	student	00	548 K	Console Window Host
csrss.exe	SYSTEM	00	1,316 K	Client Server Runtime Pro...
csrss.exe	SYSTEM	00	4,716 K	Client Server Runtime Pro...
dllhost.exe	SYSTEM	00	2,044 K	COM Surrogate
dwm.exe	student	00	4,904 K	Desktop Window Manager
explorer.exe	student	00	21,488 K	Windows Explorer
gogoc.exe	SYSTEM	00	976 K	gogoCLIENT
jucheck.exe	student	00	2,332 K	Java(TM) Update Checker
jusched.exe	student	00	1,032 K	Java(TM) Update Scheduler
lsass.exe	SYSTEM	00	2,396 K	Local Security Authority P...
lsm.exe	SYSTEM	00	1,216 K	Local Session Manager Se...
msdtc.exe	NETWORK SE...	00	1,828 K	Microsoft Distributed Tran...
notepad.exe	student	00	828 K	Notepad

☑ Show processes from all users

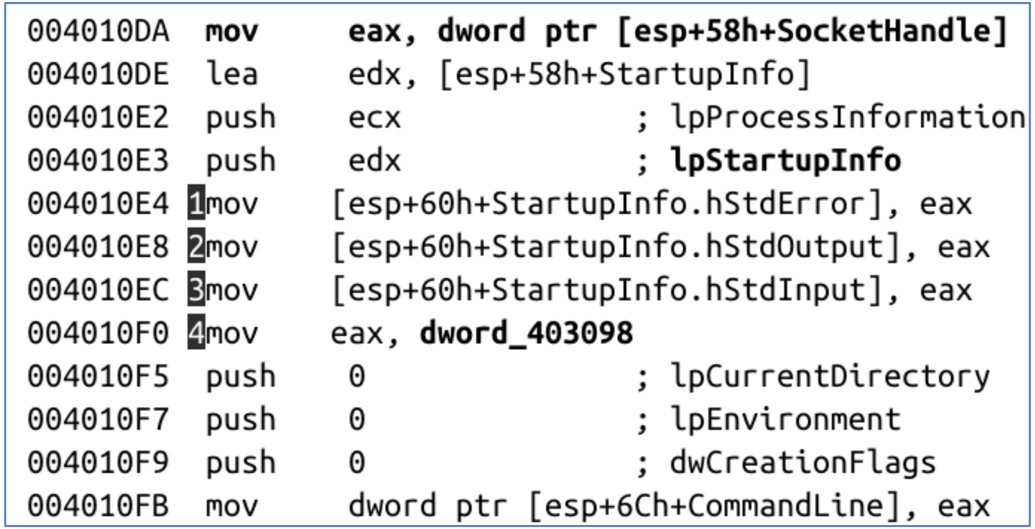
Processes: 50 CPU Usage: 0% Physical Memory: 33%

Memory Management

- Each process uses resources, like CPU, file system, and memory
- OS allocates memory to each process
- Two processes accessing the **same memory address** actually access **different locations** in RAM
  - **Virtual address space**

Creating a New Process

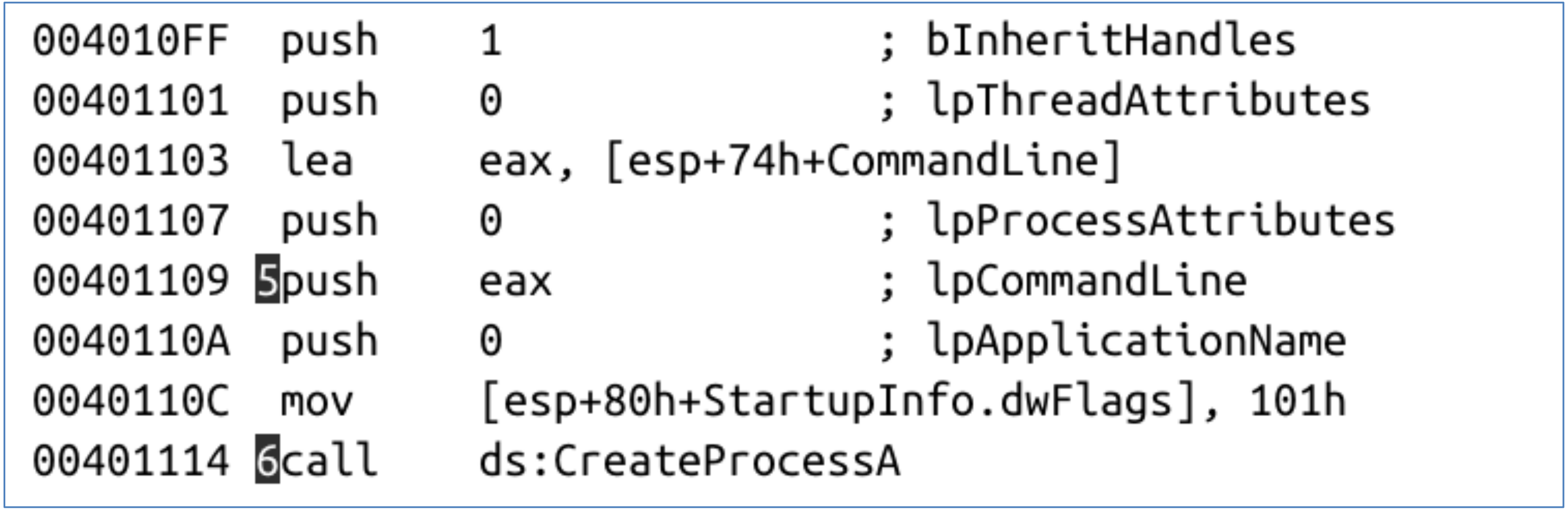
- **CreateProcess**
  - Can create a simple remote shell with one function call
  - **STARTUPINFO** parameter contains handles for **standard input**, **standard output**, and **standard error streams**
    - Can be set to a **socket**, creating a remote shell
- Example: **create a Shell**
  - Loads socket handle, StdError, StdOutput and StdInput into lpProcessInformation



```
004010DA   mov       eax, dword ptr [esp+58h+SocketHandle]
004010DE   lea       edx, [esp+58h+StartupInfo]
004010E2   push      ecx                  ; lpProcessInformation
004010E3   push      edx                  ; lpStartupInfo
004010E4  1mov      [esp+60h+StartupInfo.hStdError], eax
004010E8  2mov      [esp+60h+StartupInfo.hStdOutput], eax
004010EC  3mov      [esp+60h+StartupInfo.hStdInput], eax
004010F0  4mov      eax, dword_403098
004010F5   push      0                    ; lpCurrentDirectory
004010F7   push      0                    ; lpEnvironment
004010F9   push      0                    ; dwCreationFlags
004010FB   mov       dword ptr [esp+6Ch+CommandLine], eax
```

Code to Create a Shell (cont.)

- CommandLine contains the command line
- It's executed when CreateProcess is called



```
004010FF  push    1                       ; bInheritHandles
00401101  push    0                       ; lpThreadAttributes
00401103  lea     eax, [esp+74h+CommandLine]
00401107  push    0                       ; lpProcessAttributes
00401109 5push    eax                     ; lpCommandLine
0040110A  push    0                       ; lpApplicationName
0040110C  mov     [esp+80h+StartupInfo.dwFlags], 101h
00401114 6call    ds:CreateProcessA
```

THREADS

Threads

- Processes are containers
  - Each process contains one or more threads

- Threads are what Windows actually executes

- Threads
  - Independent **sequences of instructions**
  - Executed by CPU without waiting for other threads
  - Threads within a process **share** the same memory space
  - Each thread has its own registers and stack

Thread Context

- When a thread is running, it has **complete control** of the CPU
- Other threads **cannot affect** the state of the CPU
- When a thread changes a register, it does not affect any other threads
- When the OS switches to another thread, it saves all CPU values in a structure called the **thread context**

- **Creating a thread**
  - CreateThread
    - The **caller** specified a **start** address, also called a **start** function

How Malware Uses Threads

- Use **CreateThread** to **load a malicious DLL** into a process

- Create two threads, for input and output
  - Used to communicate with a running application

Interprocess Coordination with Mutexes

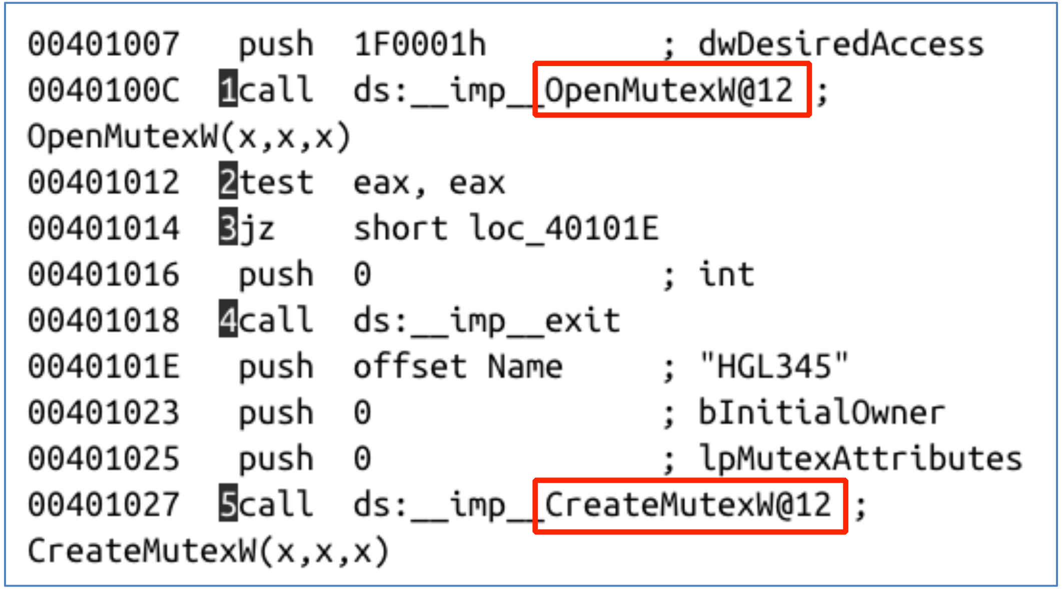
- **Mutexes** are global objects that coordinate multiple processes and threads
- In the kernel, they are called **mutants**
- Mutexes often use **hard-coded names** which can be used to identify malware

Functions for Mutexes

- **WaitForSingleObject**
  - Gives a thread **access** to the mutex
  - Any subsequent threads attempting to gain access to it must wait
- **ReleaseMutex**
  - Called when a thread is **done** using the mutex
- **CreateMutex**
- **OpenMutex**
  - Gets a handle to another process's mutex

Making Sure Only One Copy of Malware is Running

- **OpenMutex** checks if HGL345 exists
- If not, it is created with **CreateMutex**
- **`test eax, eax`**
  sets Z flag if eax is zero

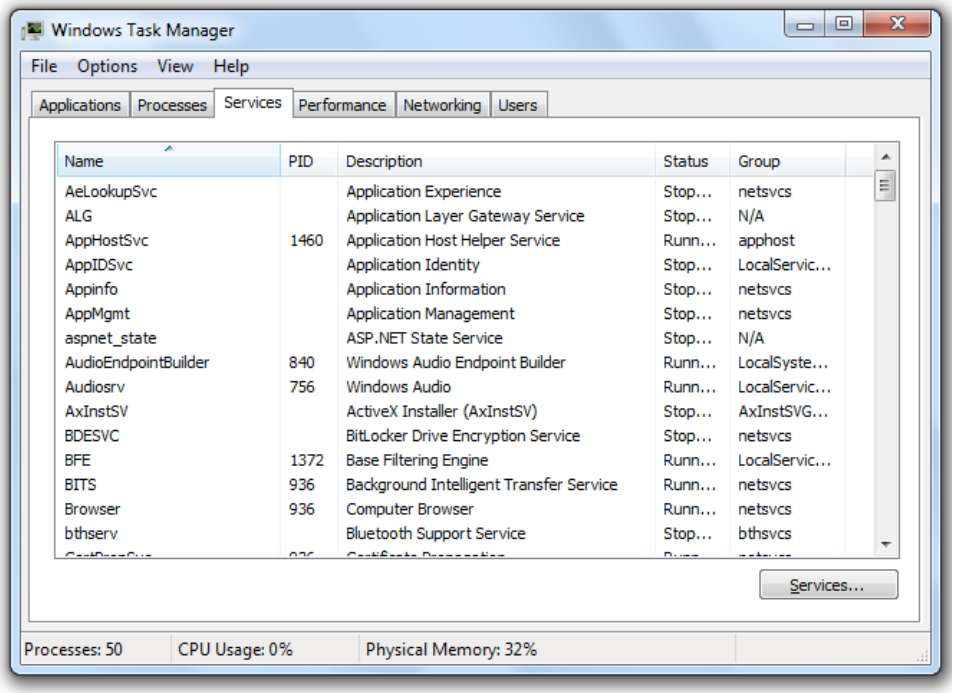


```
00401007    push  1F0001h            ; dwDesiredAccess
0040100C   1call  ds:__imp__OpenMutexW@12 ;
OpenMutexW(x,x,x)
00401012   2test  eax, eax
00401014   3jz    short loc_40101E
00401016    push  0                  ; int
00401018   4call  ds:__imp__exit
0040101E    push  offset Name        ; "HGL345"
00401023    push  0                  ; bInitialOwner
00401025    push  0                  ; lpMutexAttributes
00401027   5call  ds:__imp__CreateMutexW@12 ;
CreateMutexW(x,x,x)
```

SERVICES

Services

- Services run in the background **without user input**



Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Services...

Name	PID	Description	Status	Group
AeLookupSvc		Application Experience	Stop...	netsvcs
ALG		Application Layer Gateway Service	Stop...	N/A
AppHostSvc	1460	Application Host Helper Service	Runn...	apphost
AppIDSvc		Application Identity	Stop...	LocalServic...
Appinfo		Application Information	Stop...	netsvcs
AppMgmt		Application Management	Stop...	netsvcs
aspnet_state		ASP.NET State Service	Stop...	N/A
AudioEndpointBuilder	840	Windows Audio Endpoint Builder	Runn...	LocalSyste...
Audiosrv	756	Windows Audio	Runn...	LocalServic...
AxInstSV		ActiveX Installer (AxInstSV)	Stop...	AxInstSVG...
BDESVC		BitLocker Drive Encryption Service	Stop...	netsvcs
BFE	1372	Base Filtering Engine	Runn...	LocalServic...
BITS	936	Background Intelligent Transfer Service	Runn...	netsvcs
Browser	936	Computer Browser	Runn...	netsvcs
bthserv		Bluetooth Support Service	Stop...	bthsvcs

Processes: 50 CPU Usage: 0% Physical Memory: 32%

SYSTEM Account

- Services often run as **SYSTEM**, which is even more powerful than the **Administrator**

- Services can run **automatically** when Windows starts
  - An easy way for malware to maintain **persistence**
  - Persistent malware survives a restart

Service API Functions

- **OpenSCManager**
  - Returns a handle to the Service Control Manager

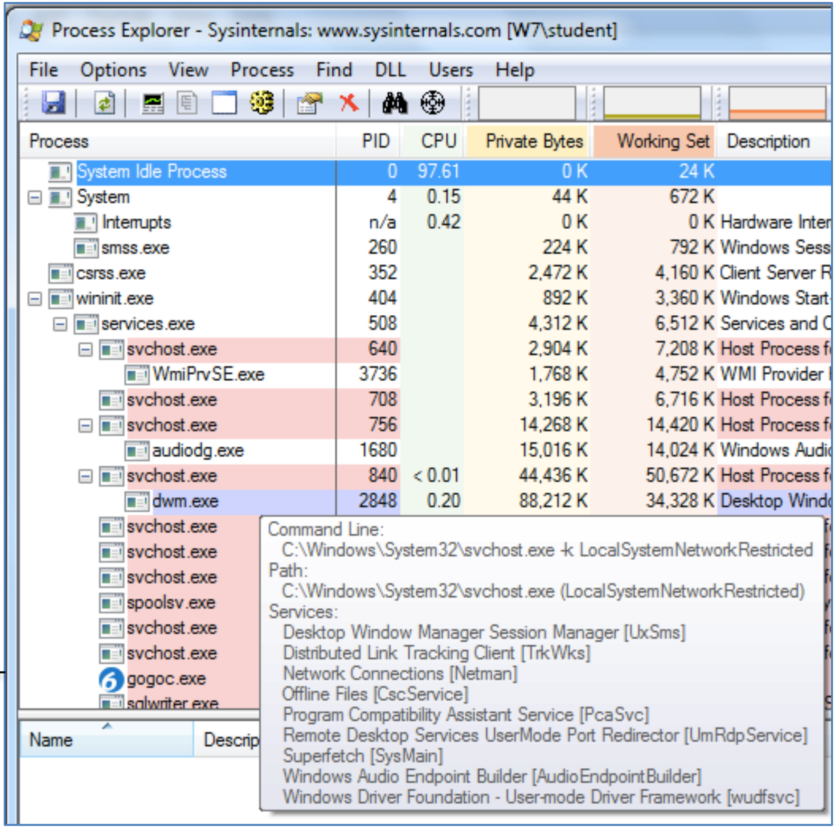
- **CreateService**
  - Adds a new service to the Service Control Manager
  - Can specify whether the service will start automatically at boot time

- **StartService**
  - Only used if the service is set to start manually

Svchost.exe

- WIN32_SHARE_PROCESS
  - Most common type of service used by malware
  - Stores code for service in a DLL
  - Combines **several services** into a **single shared process** named **`svchost.exe`**

svchost.exe in Process Explorer

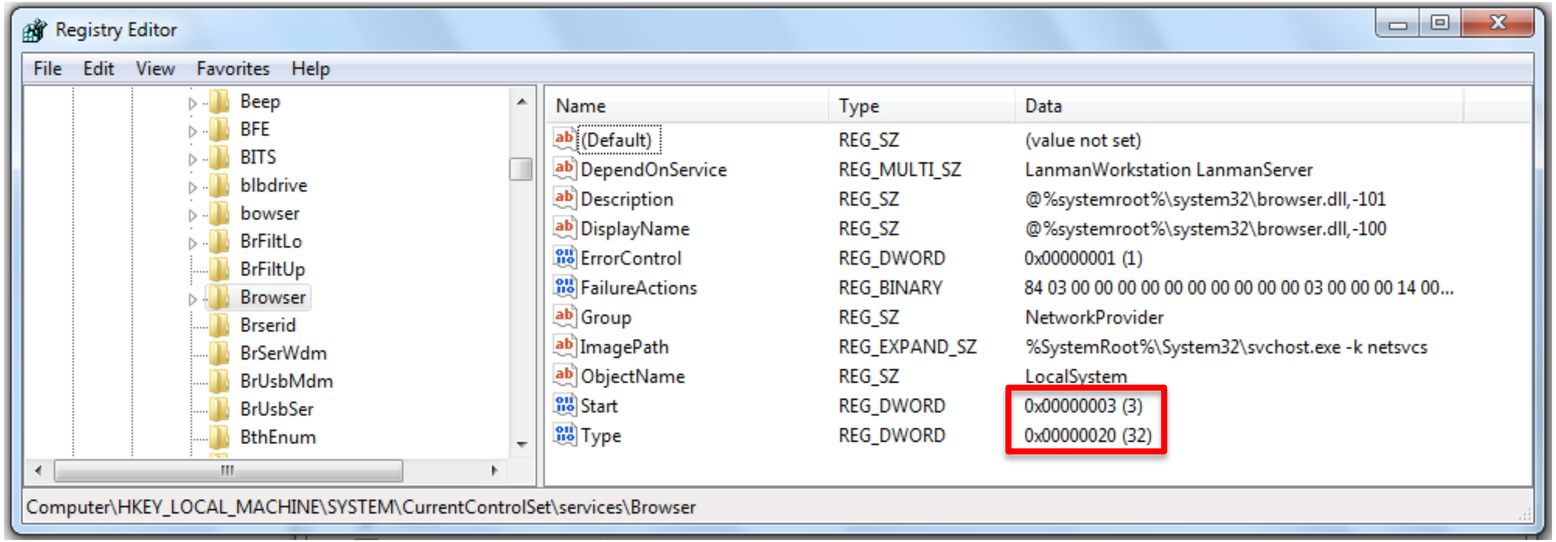

Process Explorer - Sysinternals: www.sysinternals.com [W7\student]
File
Options
View
Process
Find
DLL
Users
Help
Name
Descrip

Process	PID	CPU	Private Bytes	Working Set	Description
System Idle Process	0	97.61	0 K	24 K	
System	4	0.15	44 K	672 K	
Interrupts	n/a	0.42	0 K	0 K	Hardware Inter
smss.exe	260		224 K	792 K	Windows Sess
csrss.exe	352		2,472 K	4,160 K	Client Server R
wininit.exe	404		892 K	3,360 K	Windows Start
services.exe	508		4,312 K	6,512 K	Services and C
svchost.exe	640		2,904 K	7,208 K	Host Process f
WmiPrvSE.exe	3736		1,768 K	4,752 K	WMI Provider
svchost.exe	708		3,196 K	6,716 K	Host Process f
svchost.exe	756		14,268 K	14,420 K	Host Process f
audiodg.exe	1680		15,016 K	14,024 K	Windows Audi
svchost.exe	840	< 0.01	44,436 K	50,672 K	Host Process f
dwm.exe	2848	0.20	88,212 K	34,328 K	Desktop Wind
svchost.exe					
svchost.exe					
svchost.exe					
spoolsv.exe					
svchost.exe					
svchost.exe					
gogoc.exe					
sqlwriter.exe					

Command Line:
C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted
Path:
C:\Windows\System32\svchost.exe (LocalSystemNetworkRestricted)
Services:
Desktop Window Manager Session Manager [UxSms]
Distributed Link Tracking Client [TrkWks]
Network Connections [Netman]
Offline Files [CscService]
Program Compatibility Assistant Service [PcaSvc]
Remote Desktop Services UserMode Port Redirector [UmRdpService]
Superfetch [SysMain]
Windows Audio Endpoint Builder [AudioEndpointBuilder]
Windows Driver Foundation - User-mode Driver Framework [wudfsvc]

Service Information in the Registry

- HKLM\System\CurrentControlSet\Services
  - ○ Start value = 0x03 for "Load on Demand"
  - ○ Type = 0x20 for WIN32_SHARE_PROCESS

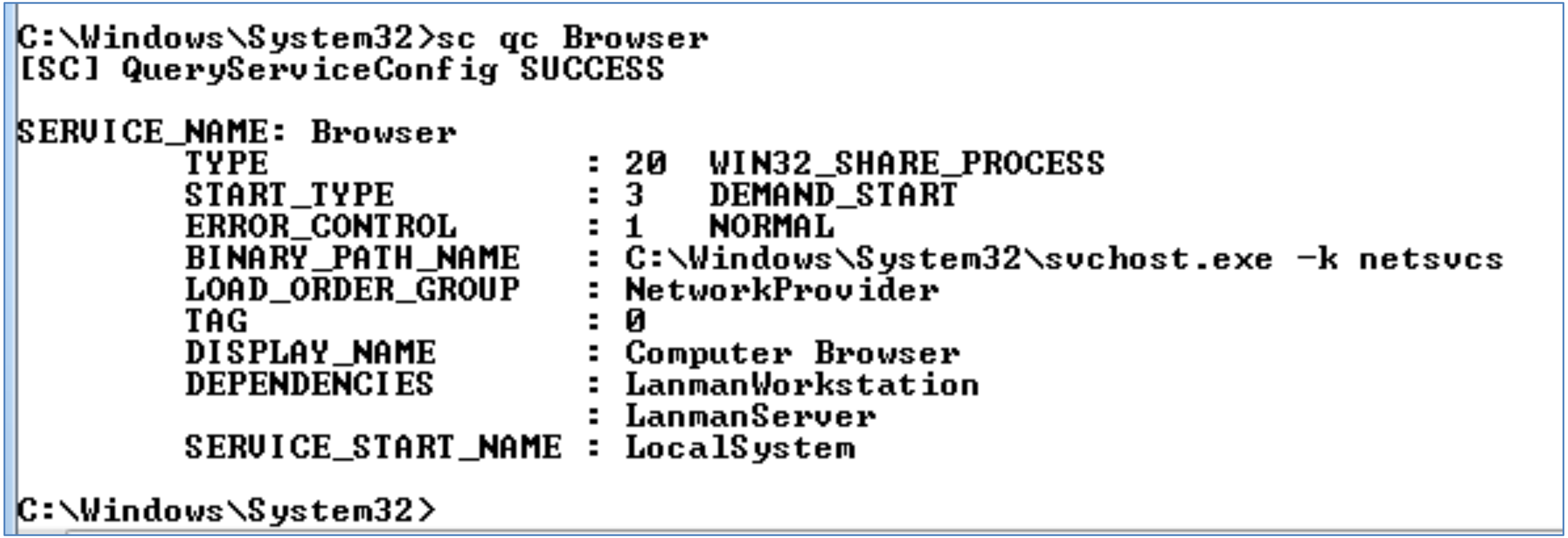


Name	Type	Data
(Default)	REG_SZ	(value not set)
DependOnService	REG_MULTI_SZ	LanmanWorkstation LanmanServer
Description	REG_SZ	@%systemroot%\system32\browser.dll,-101
DisplayName	REG_SZ	@%systemroot%\system32\browser.dll,-100
ErrorControl	REG_DWORD	0x00000001 (1)
FailureActions	REG_BINARY	84 03 00 00 00 00 00 00 00 00 00 00 03 00 00 00 14 00...
Group	REG_SZ	NetworkProvider
ImagePath	REG_EXPAND_SZ	%SystemRoot%\System32\svchost.exe -k netsvcs
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000003 (3)
Type	REG_DWORD	0x00000020 (32)

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Browser

SC Command

- Included in Windows
- Gives information about Services



```
C:\Windows\System32>sc qc Browser
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Browser
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Windows\System32\svchost.exe -k netsvcs
        LOAD_ORDER_GROUP   : NetworkProvider
        TAG                : 0
        DISPLAY_NAME       : Computer Browser
        DEPENDENCIES       : LanmanWorkstation
                           : LanmanServer
        SERVICE_START_NAME : LocalSystem

C:\Windows\System32>
```

COMPONENT OBJECT MODEL (COM)

Component Object Model (COM)

- Allows different software components to share code

- Every thread that uses COM must call **OleInitialize** or **CoInitializeEx** before calling other COM libraries

GUIDs, CLSIDs, IIDs

- COM objects are accessed via Globally Unique Identifiers (GUIDs)

- There are several types of GUIDs, including
  - Class Identifiers (**CLSIDs**)
    - in Registry at HKEY_CLASSES_ROOT\CLSID
  - Interface Identifiers (**IIDs**)
    - in Registry at HKEY_CLASSES_ROOT\Interface

EXCEPTIONS

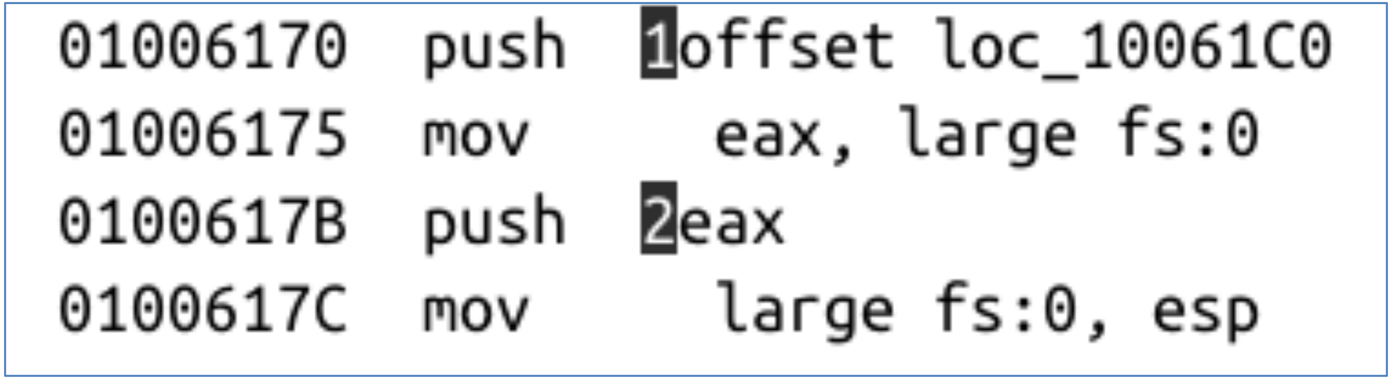
Exceptions

- Exceptions are caused by errors, such as division by zero (hardware) or invalid memory access (software)

- When an exception occurs, **execution transfers** to the **Structured Exception Handler**

fs:0 Stores Exception Location

- FS is one of six Segment Registers

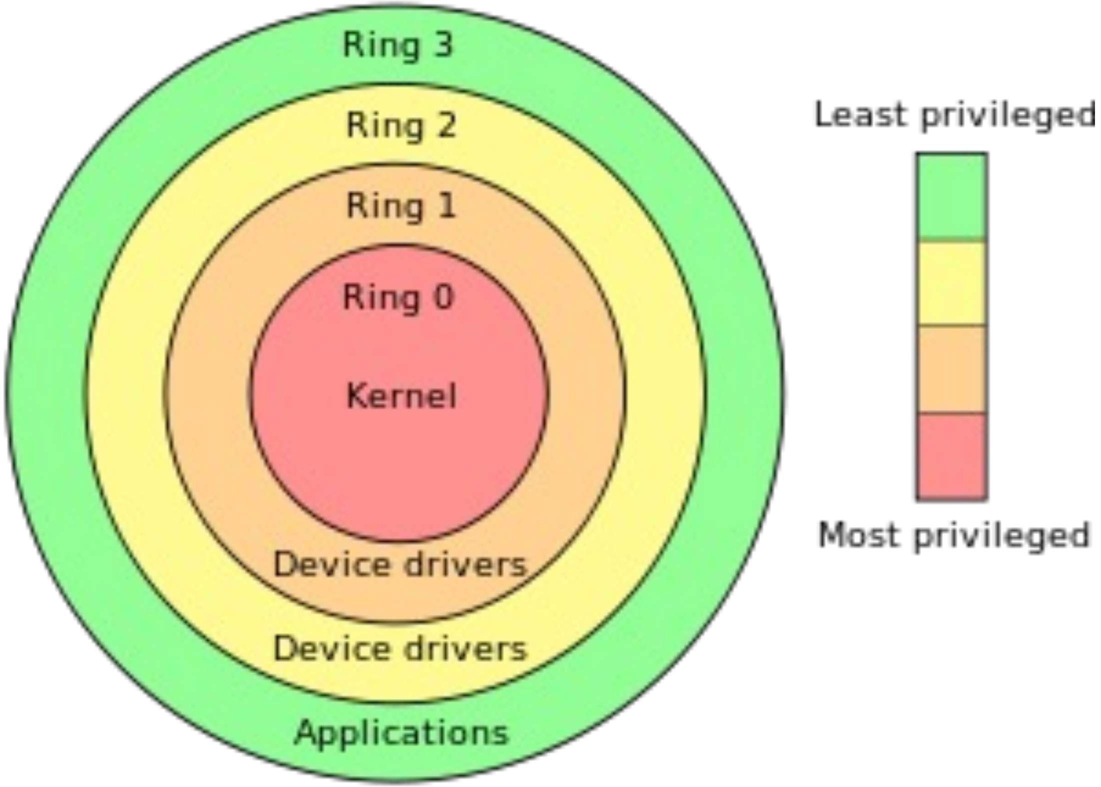


```
01006170  push  1offset loc_10061C0
01006175  mov     eax, large fs:0
0100617B  push  2eax
0100617C  mov     large fs:0, esp
```

KERNEL VS. USER MODE

Two Privilege Levels

- Ring 0: Kernel Mode
- Ring 3: User mode
- Rings 1 and 2 are not used by Windows



User Mode

- Nearly all code runs in user mode
  - Except OS and hardware drivers, which run in kernel mode
- User mode cannot access hardware **directly**
- Restricted to a subset of CPU instructions
- Can only manipulate hardware through the Windows API

- User mode processes
  - Each process has **its own memory**, **security permissions**, and **resources**
  - If a user-mode program executes an **invalid instruction** and crashes, Windows can reclaim the resources and terminate the program

Calling the Kernel

- It's not possible to jump directly from user mode to the kernel
- **SYSENTER**, **SYSCALL**, or **INT 0x2E** instructions use lookup tables to locate predefined functions

Kernel Processes

- All kernel processes share resources and memory addresses
- Fewer security checks
- If kernel code executes an **invalid instruction**, the OS crashes with the Blue Screen of Death
- Antivirus software and firewalls run in Kernel mode

Malware in Kernel Mode

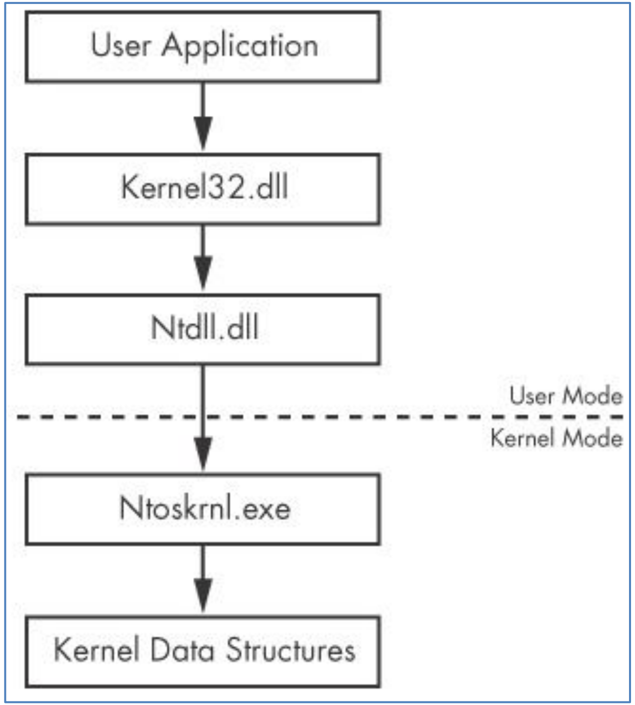
- More powerful than user-mode malware
- Auditing doesn't apply to kernel
- Almost all **rootkits** use kernel code
- Most malware ***does not*** use kernel mode

THE NATIVE API

The Native API

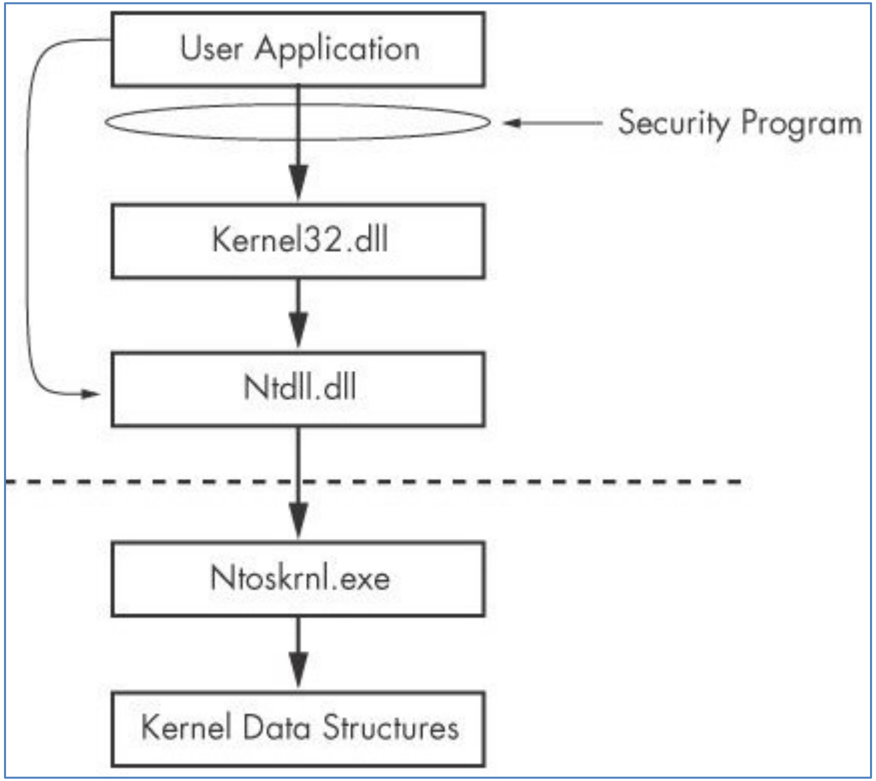
- **Lower-level interface** for interacting with Windows
- **Rarely** used by non-malicious programs
- **Popular** among malware writers

- **Ntdll.dll** manages interactions between **user space** and the **kernel**
- **Ntdll** functions make up the **Native API**



The Native API (cont.)

- Undocumented
- Intended for internal Windows use
- Can be used by programs
- Native API calls can be more powerful and **stealthier** than Windows API calls



Popular Native API Calls in Malware

- NTtQuerySystemInformation
- NTtQueryInformationProcess
- NTtQueryInformationThread
- NTtQueryInformationFile
- NTtQueryInformationKey
  - Provide much more information than any available Win32 calls

- NtContinue
  - Returns from an exception
  - Can be used to <u>transfer execution</u> in complicated ways
  - Used to confuse analysts and make a program <u>more difficult to debug</u>

END OF LECTURE. THANK YOU.