Malware Analysis

Chapter 7: Analyzing Malicious Windows Programs

王志

zwang@nankai.edu.cn

updated on Otc. 31st 2021

College of Cyber Science
Nankai University

英国《2021年国防评论》

- 英国国防部国防情报局局长“在必要时首先在物理和虚拟环境中采取行动，以确保决策优势和行动优势”
- 英国总检察长“可造成与武装袭击同等规模的死亡或毁灭性损失的网络行动将触发《联合国宪章》第五十一条赋予的可采取自卫行动的固有权利。”

英国《2021年国防评论》

- “可造成与武装袭击同等规模的死亡或毁灭性损失的网络行动”，在网络空间适用核威慑（动用核武反击网络攻击）

攻击基础设施

- 2010 Stuxnet摧毁了大约1000台离心机
- 2016年恶意软件在乌克兰造成大停电
- 2017年 Triton/Trisis恶意软件攻击沙特Tasnee石化公司
- 2020年攻击以色列的水处理系统

Outline

- Windows API
- Windows Registry
- Networking APIs
- Following Running Malware
- Kernel Mode vs. User Mode
- Native API

Windows API

What is the API?

- A broad set of functionality
  - File operation
  - Network operation
  - ......
- API governs how programs interact with Windows OS

Windows API

- Concepts
  - Types and Hungarian Notation
  - Handles
  - File System Functions
  - Special Files

Data Types

- Windows API has its own names to represent data types
  - DWORD for 32-bit unsigned integers
  - WORD for 16-bit unsigned integers

Data Types

- WinDef.H

```
typedef   int            BOOL;
typedef   unsigned char       BYTE;
typedef        unsigned short        WORD;
typedef   unsigned long       DWORD;
```

- WinNT.H

```
typedef       BYTE            BOOLEAN;
typedef       PVOID           HANDLE;
#define       CALLBACK              __stdcall
```

Hungarian Notation

- Hungarian Notation
  - Clarity and consistency
  - Variable prefix notations
    - data type
  - Comment Block
  - Class Declaration Header
  - ……

Common API Data Types

Prefix	API Type	C Type
w	WORD	16-bit unsigned value
dw	DWORD	32-bit unsigned value
H	HANDLE	A reference to an object
LP	Long Pointer	A pointer to another type
Callback	Callback	A function called by API

Handles

- Items opened or created in the OS, like
  - window, process, menu, file, ...
- Handles are like pointers to those objects
  - They are not pointers, however
- The only thing you can do with a handle is store it and use it in a later function call to refer to the same object

Handle Example

- The CreateWindowEx function returns an HWND, a handle to the window
- To do anything to that window (such as DestroyWindow) , use the handle
- HWND can not be used as a pointer or arithmetic value.

File System Functions

- CreateFile, ReadFile, WriteFile
  - Normal file input/output
- CreateFileMapping, MapViewOfFile
  - Used by malware, loads file into RAM
  - Can be used to execute a file without using the Windows loader

Special Files

- **Shared files** like \\server\share
    - Or \\?\server\share
        - \\?\ Disables string parsing, allows longer filenames

Special Files

- **Namespaces**
  - Special folders in the Windows file system

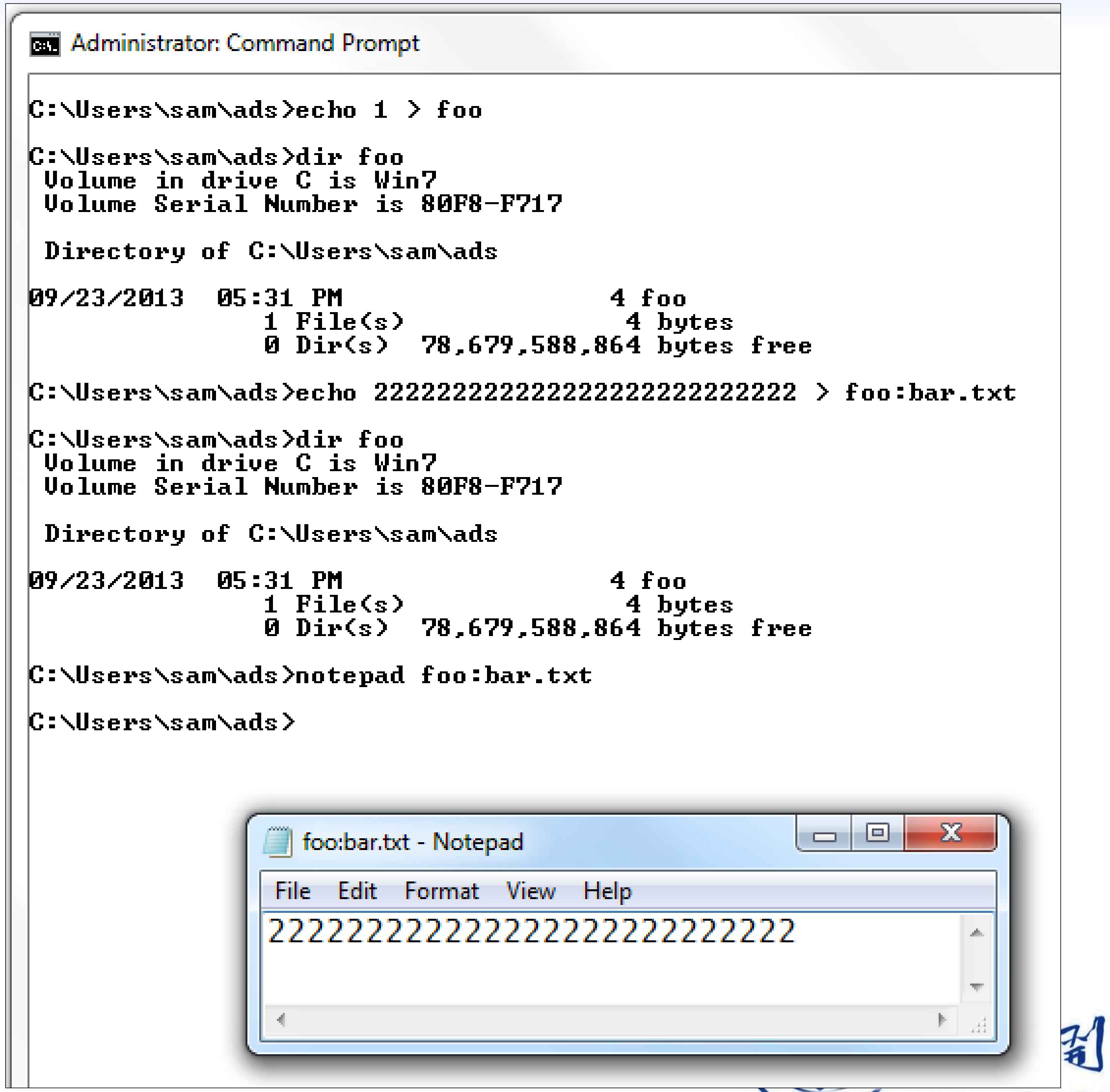
\ Lowest namespace, contains everything

\\\\.\ <span style="color:red">**Device namespace**</span> used for direct disk input/output

Witty worm wrote to \\\\.\\PhysicalDisk1 to corrupt the disk

Alternate Data Streams (ADS)

- Add one file to another file
- Not list in the directory
- Malware authors like ADS



```
C:\Users\sam\ads>echo 1 > foo

C:\Users\sam\ads>dir foo
 Volume in drive C is Win7
 Volume Serial Number is 80F8-F717

 Directory of C:\Users\sam\ads

09/23/2013  05:31 PM                 4 foo
               1 File(s)              4 bytes
               0 Dir(s)  78,679,588,864 bytes free

C:\Users\sam\ads>echo 222222222222222222222222222 > foo:bar.txt

C:\Users\sam\ads>dir foo
 Volume in drive C is Win7
 Volume Serial Number is 80F8-F717

 Directory of C:\Users\sam\ads

09/23/2013  05:31 PM                 4 foo
               1 File(s)              4 bytes
               0 Dir(s)  78,679,588,864 bytes free

C:\Users\sam\ads>notepad foo:bar.txt

C:\Users\sam\ads>
```

```
222222222222222222222222222
```

Windows Registry

Registry Purpose

- OS and program configuration settings
  - Desktop background, mouse preferences, etc.
- Malware uses the registry for **persistence**
  - Making malware re-start when the system reboots

Registry

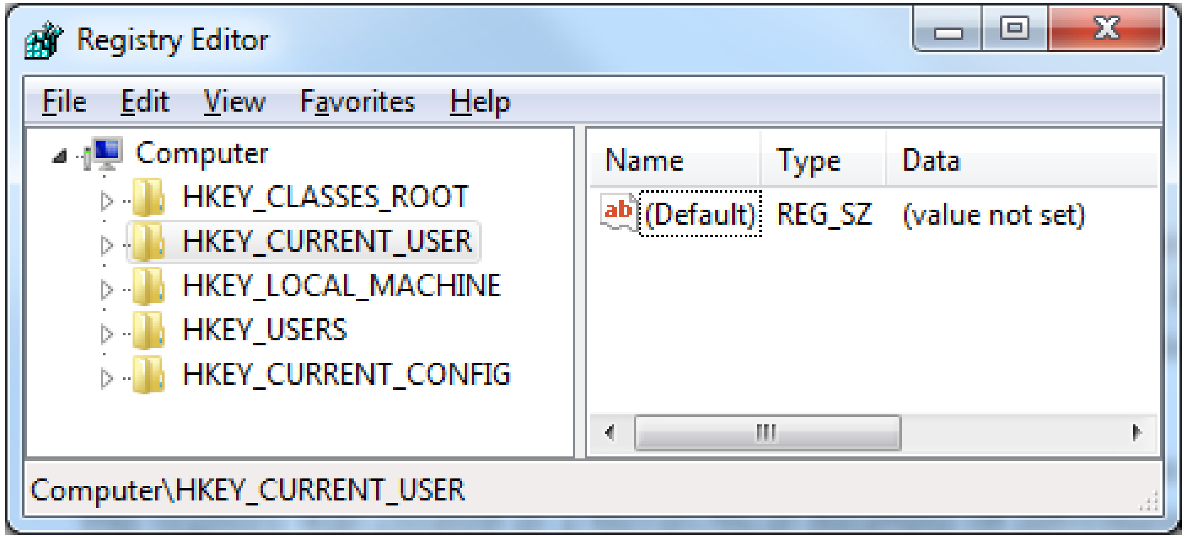
- Hierarchical Database
- Tow basic elements:
  - Key: Container Object similar to folder
  - Value: Non-Container Object similar to file
- Keys may contain values or further subkeys

Registry

- Structure:
  - Similar to Windows' path names
  - backslashes indicate levels of hierarchy
  - Registry keys can only be accessed from a root key.

Windows Registry

- **Root Keys** There are 5 root keys
- **Subkey** A folder within a folder
- **Key** A folder; can contain folders or values
- **Value Entry** Two parts: name and data
- **Value** or **Data** The data stored in a registry entry
- REGEDIT Tool to view/edit the Registry



Computer
- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG

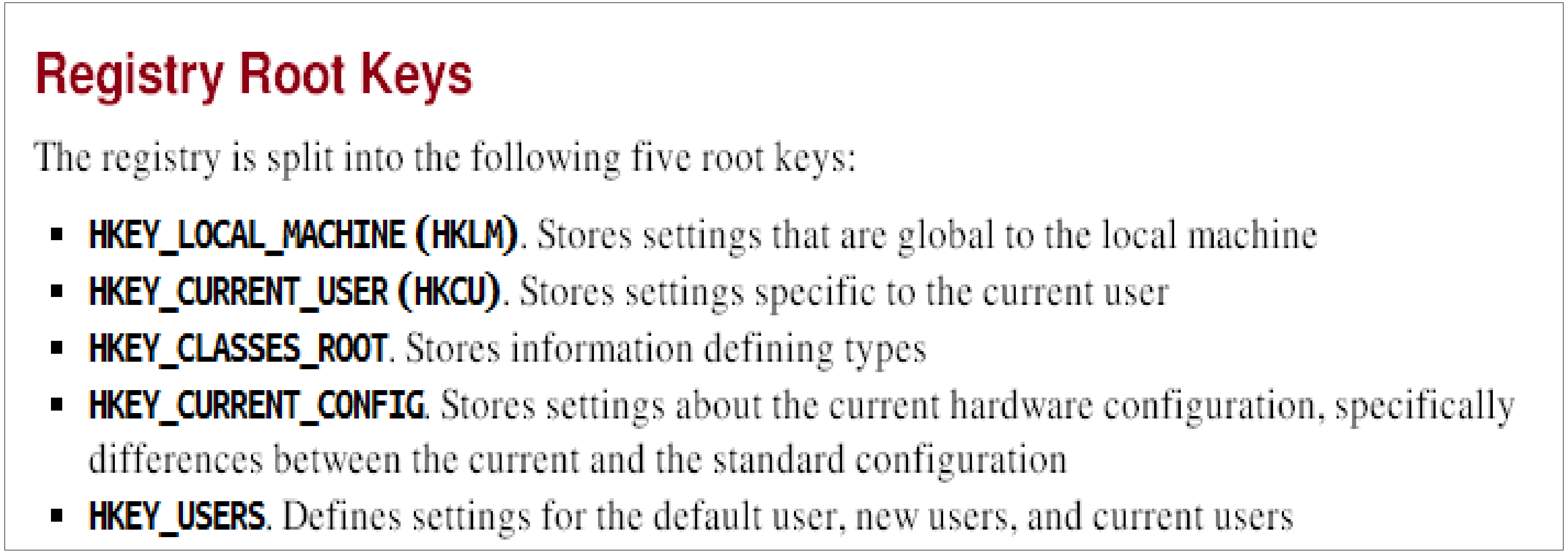
Name	Type	Data
(Default)	REG_SZ	(value not set)

Computer\HKEY_CURRENT_USER

Registry Example

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
  - refers to the subkey "Windows" of the subkey "Microsoft" of the subkey "Software" of the HKEY_LOCAL_MACHINE root key.

Root Keys



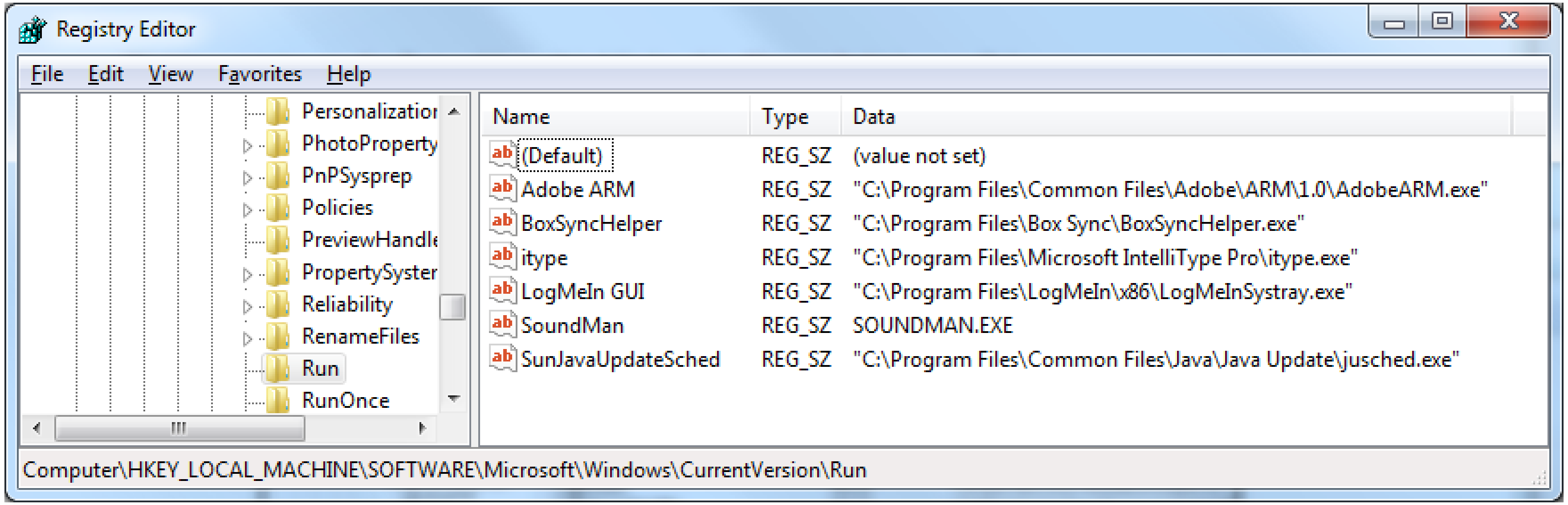
Registry Root Keys

The registry is split into the following five root keys:

- **HKEY_LOCAL_MACHINE (HKLM)**. Stores settings that are global to the local machine
- **HKEY_CURRENT_USER (HKCU)**. Stores settings specific to the current user
- **HKEY_CLASSES_ROOT**. Stores information defining types
- **HKEY_CURRENT_CONFIG**. Stores settings about the current hardware configuration, specifically differences between the current and the standard configuration
- **HKEY_USERS**. Defines settings for the default user, new users, and current users

Run Key

- **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**
  - Executables that start when a user logs on



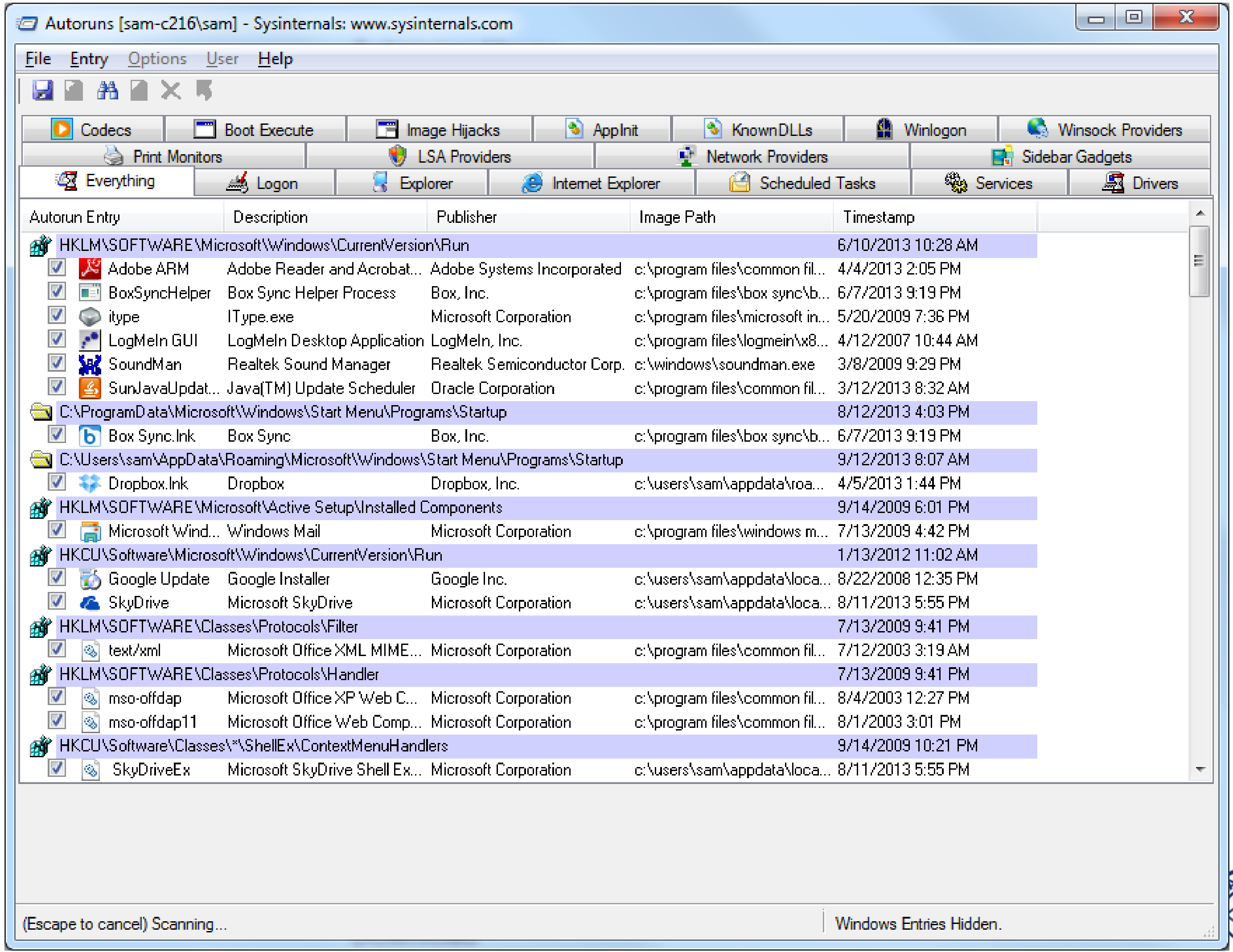
Name	Type	Data
(Default)	REG_SZ	(value not set)
Adobe ARM	REG_SZ	"C:\Program Files\Common Files\Adobe\ARM\1.0\AdobeARM.exe"
BoxSyncHelper	REG_SZ	"C:\Program Files\Box Sync\BoxSyncHelper.exe"
itype	REG_SZ	"C:\Program Files\Microsoft IntelliType Pro\itype.exe"
LogMeIn GUI	REG_SZ	"C:\Program Files\LogMeIn\x86\LogMeInSystray.exe"
SoundMan	REG_SZ	SOUNDMAN.EXE
SunJavaUpdateSched	REG_SZ	"C:\Program Files\Common Files\Java\Java Update\jusched.exe"

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Autoruns

- Sysinternals tool
- Lists code that will run automatically when system starts
  - Executables
  - DLLs loaded into IE and other programs
  - Drivers loaded into Kernel
  - It checks more than 25 registry locations

Autoruns



Autorun Entry	Description	Publisher	Image Path	Timestamp
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				6/10/2013 10:28 AM
☑ Adobe ARM	Adobe Reader and Acrobat...	Adobe Systems Incorporated	c:\program files\common fil...	4/4/2013 2:05 PM
☑ BoxSyncHelper	Box Sync Helper Process	Box, Inc.	c:\program files\box sync\b...	6/7/2013 9:19 PM
☑ itype	IType.exe	Microsoft Corporation	c:\program files\microsoft in...	5/20/2009 7:36 PM
☑ LogMeIn GUI	LogMeIn Desktop Application	LogMeIn, Inc.	c:\program files\logmein\x8...	4/12/2007 10:44 AM
☑ SoundMan	Realtek Sound Manager	Realtek Semiconductor Corp.	c:\windows\soundman.exe	3/8/2009 9:29 PM
☑ SunJavaUpdat...	Java(TM) Update Scheduler	Oracle Corporation	c:\program files\common fil...	3/12/2013 8:32 AM
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup				8/12/2013 4:03 PM
☑ Box Sync.lnk	Box Sync	Box, Inc.	c:\program files\box sync\b...	6/7/2013 9:19 PM
C:\Users\sam\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup				9/12/2013 8:07 AM
☑ Dropbox.lnk	Dropbox	Dropbox, Inc.	c:\users\sam\appdata\roa...	4/5/2013 1:44 PM
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components				9/14/2009 6:01 PM
☑ Microsoft Wind...	Windows Mail	Microsoft Corporation	c:\program files\windows m...	7/13/2009 4:42 PM
HKCU\Software\Microsoft\Windows\CurrentVersion\Run				1/13/2012 11:02 AM
☑ Google Update	Google Installer	Google Inc.	c:\users\sam\appdata\loca...	8/22/2008 12:35 PM
☑ SkyDrive	Microsoft SkyDrive	Microsoft Corporation	c:\users\sam\appdata\loca...	8/11/2013 5:55 PM
HKLM\SOFTWARE\Classes\Protocols\Filter				7/13/2009 9:41 PM
☑ text/xml	Microsoft Office XML MIME...	Microsoft Corporation	c:\program files\common fil...	7/12/2003 3:19 AM
HKLM\SOFTWARE\Classes\Protocols\Handler				7/13/2009 9:41 PM
☑ mso-offdap	Microsoft Office XP Web C...	Microsoft Corporation	c:\program files\common fil...	8/4/2003 12:27 PM
☑ mso-offdap11	Microsoft Office Web Comp...	Microsoft Corporation	c:\program files\common fil...	8/1/2003 3:01 PM
HKCU\Software\Classes\*\ShellEx\ContextMenuHandlers				9/14/2009 10:21 PM
☑ SkyDriveEx	Microsoft SkyDrive Shell Ex...	Microsoft Corporation	c:\users\sam\appdata\loca...	8/11/2013 5:55 PM

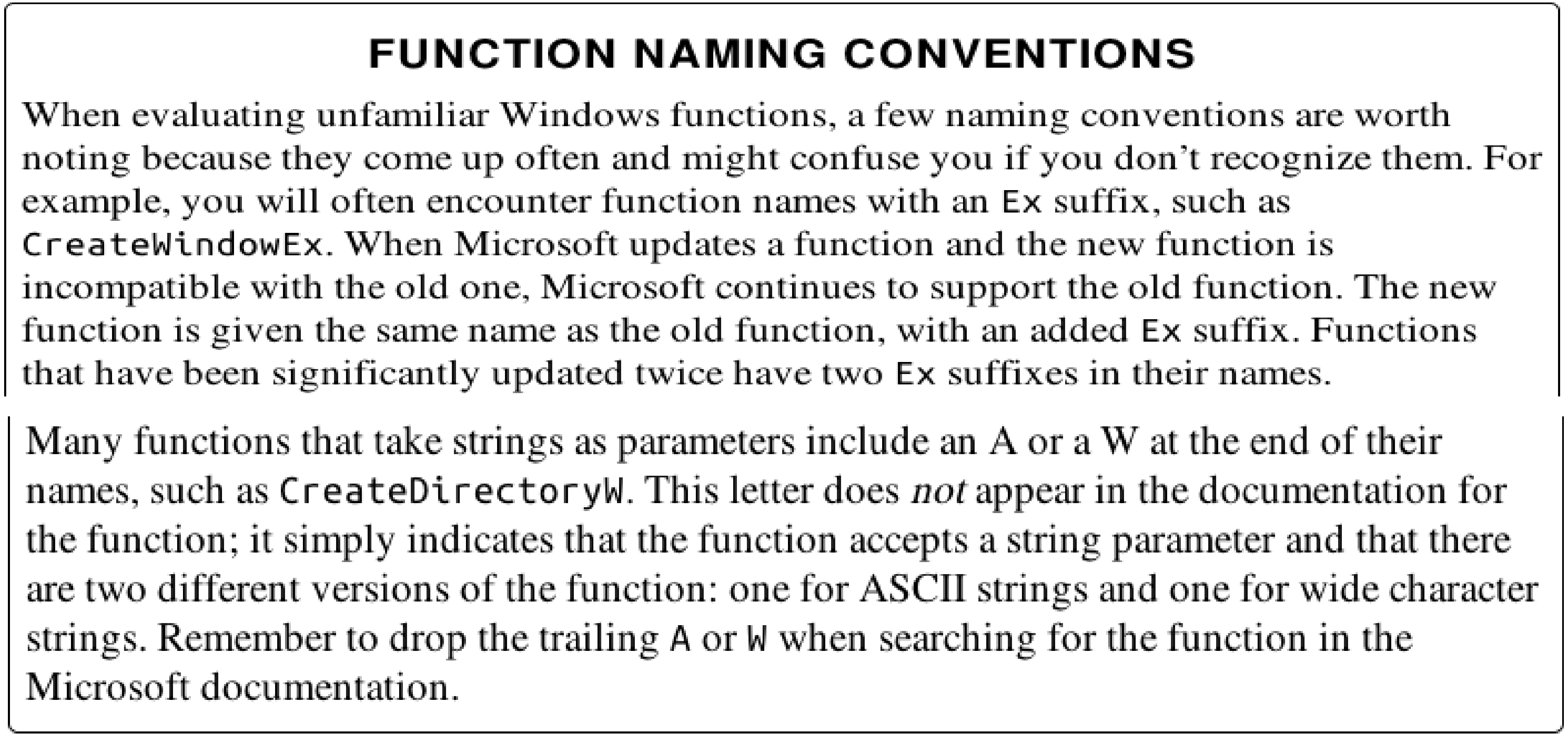
(Escape to cancel) Scanning...

Windows Entries Hidden.

Common Registry Functions

- RegOpenKeyEx
  - Opens a registry key for editing and querying
- RegSetValueEx
  - Adds a new value to the registry & sets its data
- RegGetValue
  - Returns the data for a value entry in the Registry
- Note: Documentation will omit the trailing W (wide) or A (ASCII) character in a call like RegOpenKeyExW

Ex, A, and W Suffixes

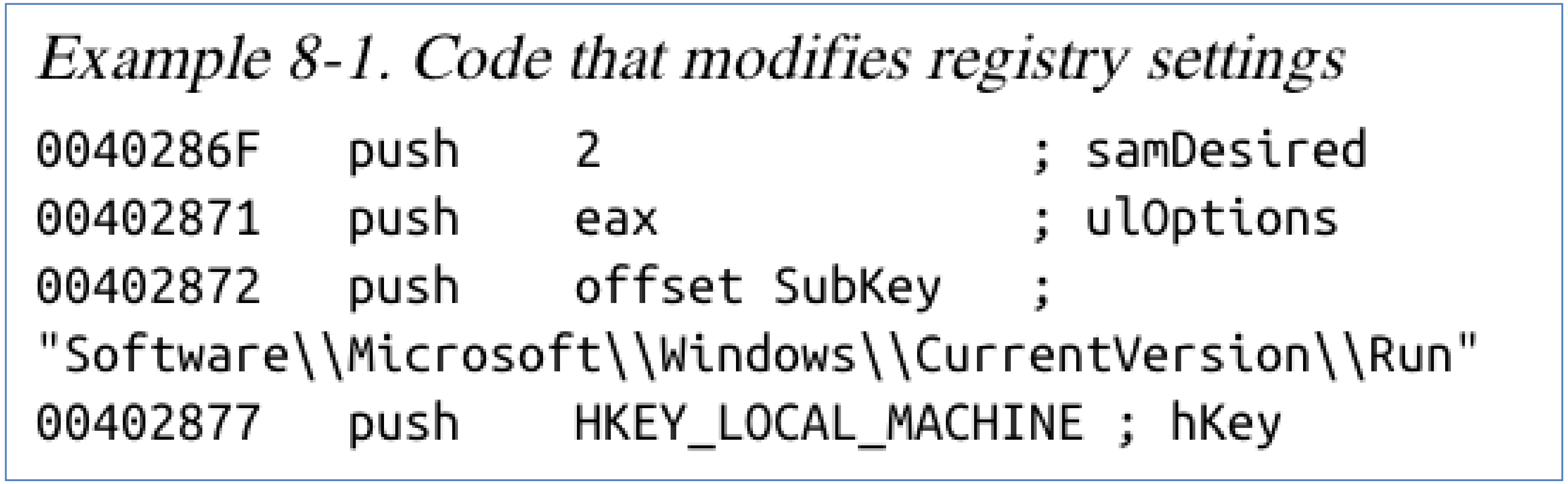


FUNCTION NAMING CONVENTIONS

When evaluating unfamiliar Windows functions, a few naming conventions are worth noting because they come up often and might confuse you if you don't recognize them. For example, you will often encounter function names with an `Ex` suffix, such as `CreateWindowEx`. When Microsoft updates a function and the new function is incompatible with the old one, Microsoft continues to support the old function. The new function is given the same name as the old function, with an added `Ex` suffix. Functions that have been significantly updated twice have two `Ex` suffixes in their names.

Many functions that take strings as parameters include an A or a W at the end of their names, such as `CreateDirectoryW`. This letter does *not* appear in the documentation for the function; it simply indicates that the function accepts a string parameter and that there are two different versions of the function: one for ASCII strings and one for wide character strings. Remember to drop the trailing `A` or `W` when searching for the function in the Microsoft documentation.

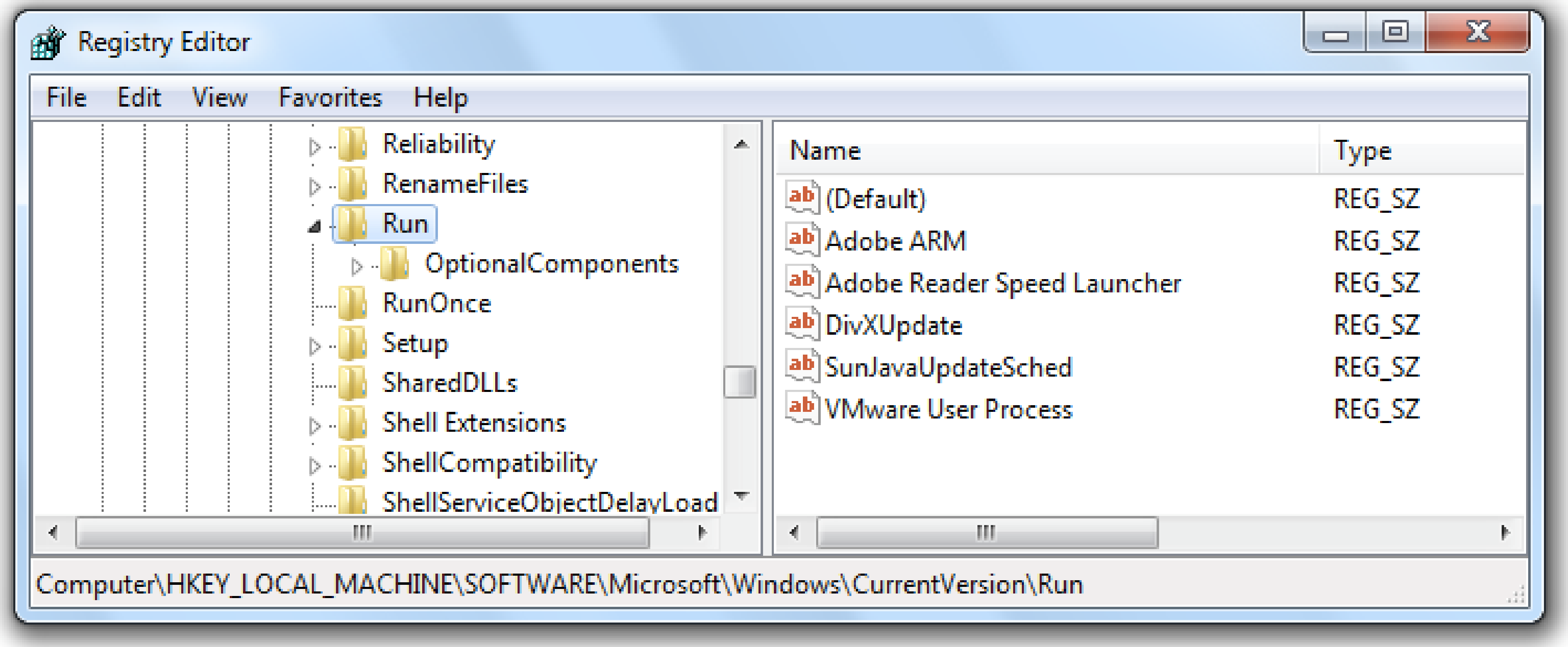
Registry Code



Example 8-1. Code that modifies registry settings

```
0040286F   push    2                   ; samDesired
00402871   push    eax                 ; ulOptions
00402872   push    offset SubKey       ;
"Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877   push    HKEY_LOCAL_MACHINE ; hKey
```

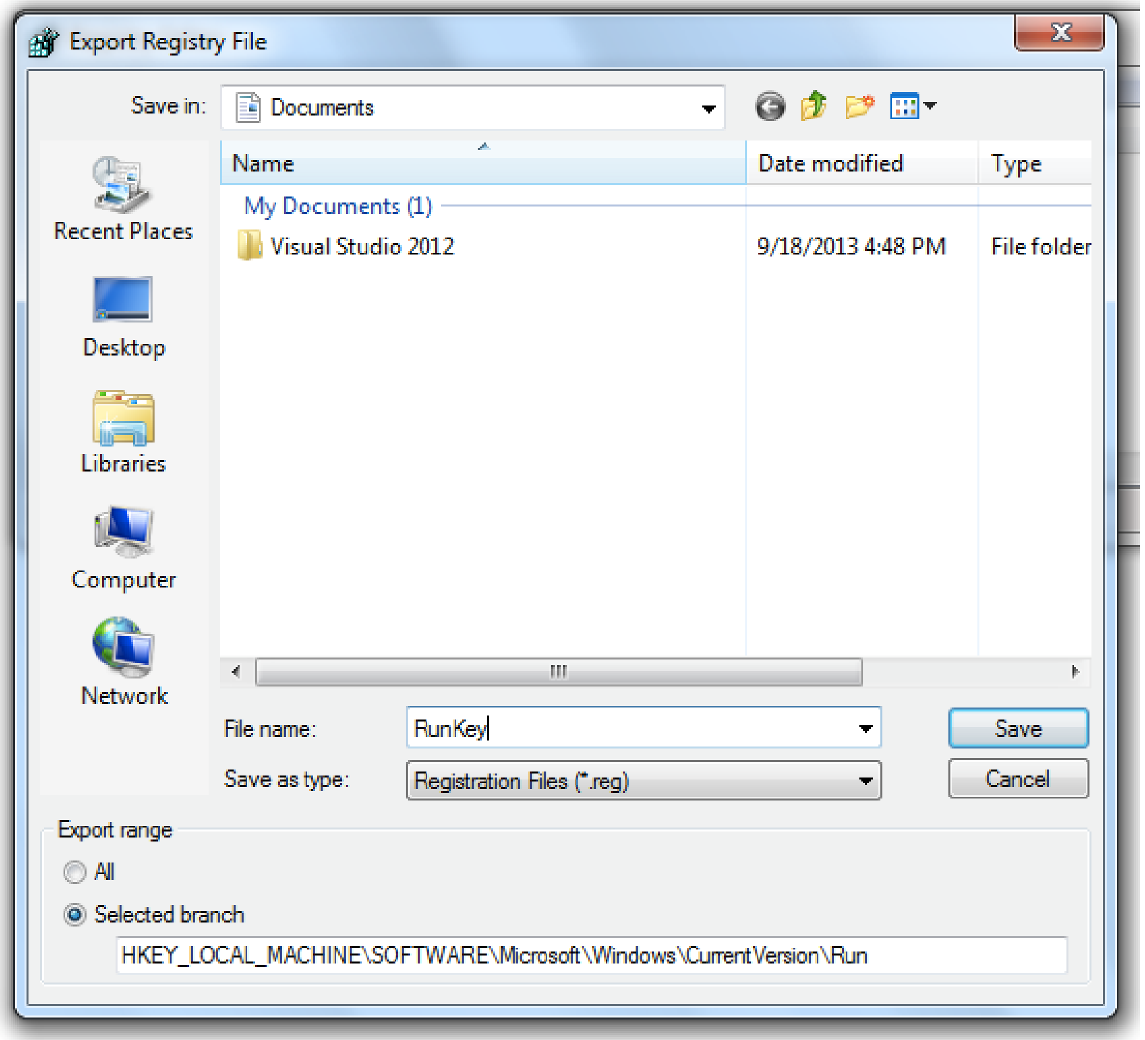
.REG Files



Name	Type
(Default)	REG_SZ
Adobe ARM	REG_SZ
Adobe Reader Speed Launcher	REG_SZ
DivXUpdate	REG_SZ
SunJavaUpdateSched	REG_SZ
VMware User Process	REG_SZ

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

.REG Files



Export Registry File

Save in: Documents

Name	Date modified	Type
My Documents (1)		
Visual Studio 2012	9/18/2013 4:48 PM	File folder

File name: RunKey

Save as type: Registration Files (*.reg)

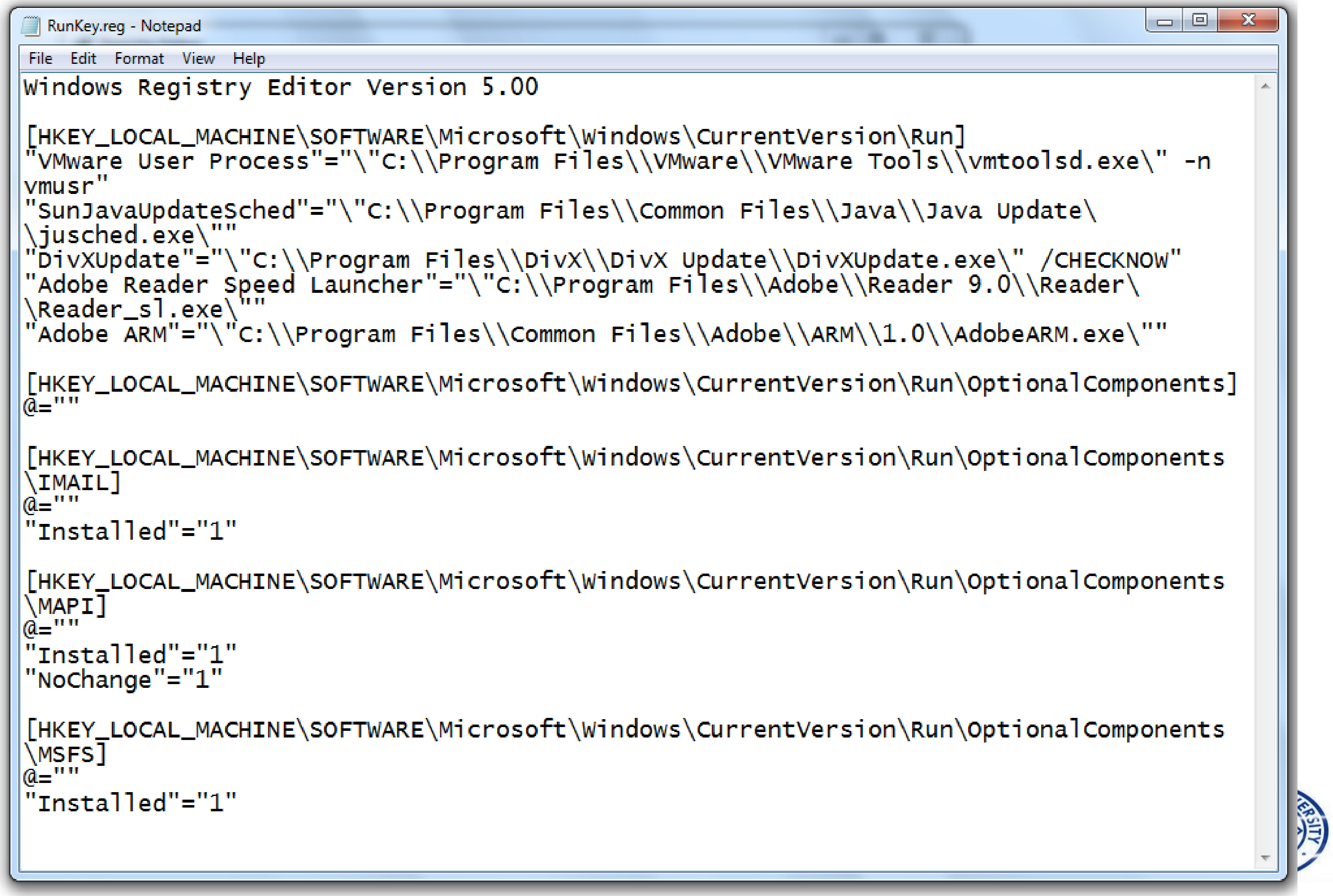
Export range

○ All

● Selected branch

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

.REG Files

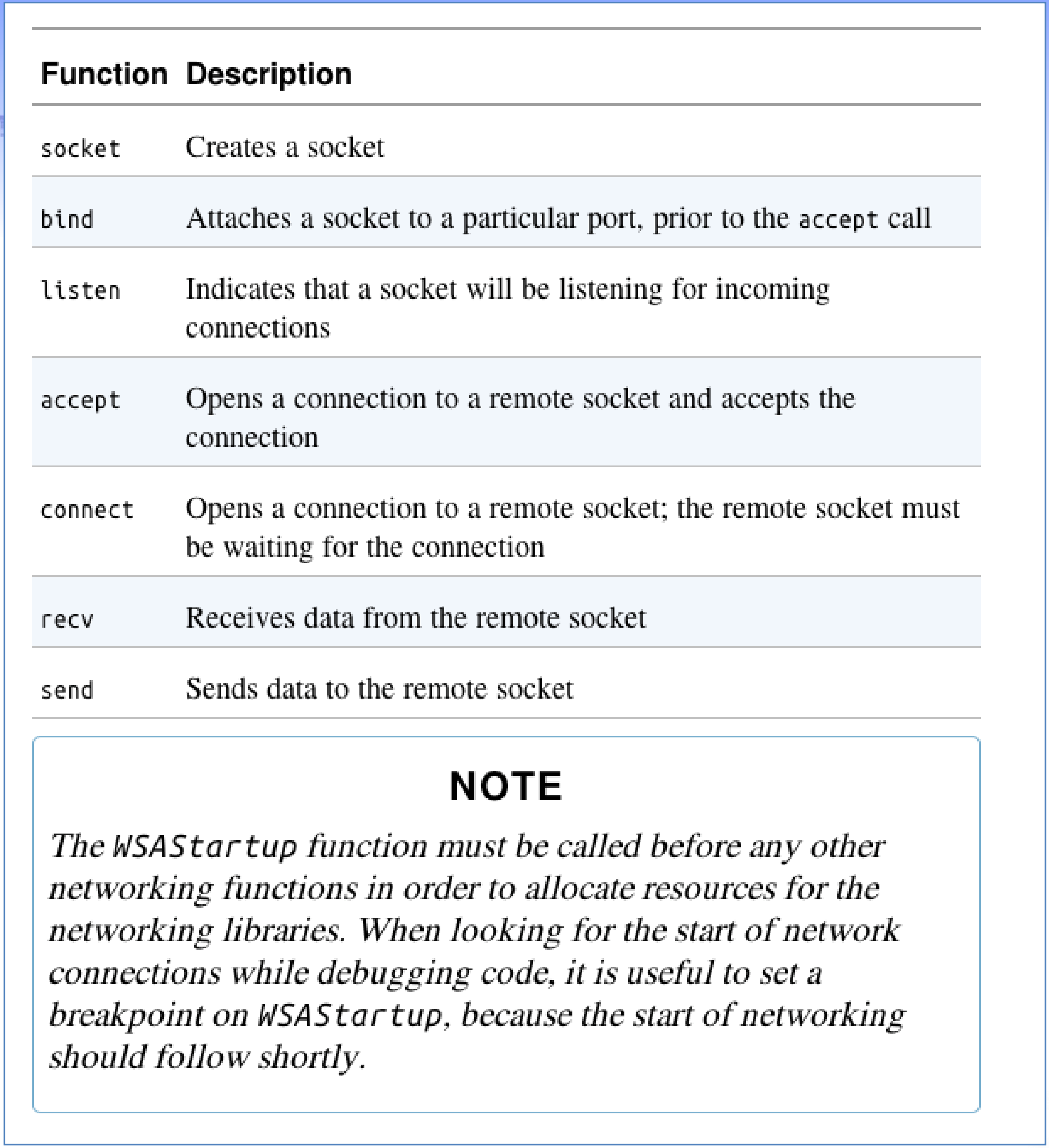


```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
"VMware User Process"="\"C:\\Program Files\\VMware\\VMware Tools\\vmtoolsd.exe\" -n
vmusr"
"SunJavaUpdateSched"="\"C:\\Program Files\\Common Files\\Java\\Java Update\
\jusched.exe\""
"DivXUpdate"="\"C:\\Program Files\\DivX\\DivX Update\\DivXUpdate.exe\" /CHECKNOW"
"Adobe Reader Speed Launcher"="\"C:\\Program Files\\Adobe\\Reader 9.0\\Reader\
\Reader_sl.exe\""
"Adobe ARM"="\"C:\\Program Files\\Common Files\\Adobe\\ARM\\1.0\\AdobeARM.exe\""

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OptionalComponents]
@=""

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OptionalComponents
\IMAIL]
@=""
"Installed"="1"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OptionalComponents
\MAPI]
@=""
"Installed"="1"
"NoChange"="1"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\OptionalComponents
\MSFS]
@=""
"Installed"="1"
```

Networking APIs

Berkeley Compatible Sockets

- Winsock libraries, primarily in *ws2_32.dll*
  - Almost identical in Windows and Unix



Function	Description
`socket`	Creates a socket
`bind`	Attaches a socket to a particular port, prior to the `accept` call
`listen`	Indicates that a socket will be listening for incoming connections
`accept`	Opens a connection to a remote socket and accepts the connection
`connect`	Opens a connection to a remote socket; the remote socket must be waiting for the connection
`recv`	Receives data from the remote socket
`send`	Sends data to the remote socket

NOTE

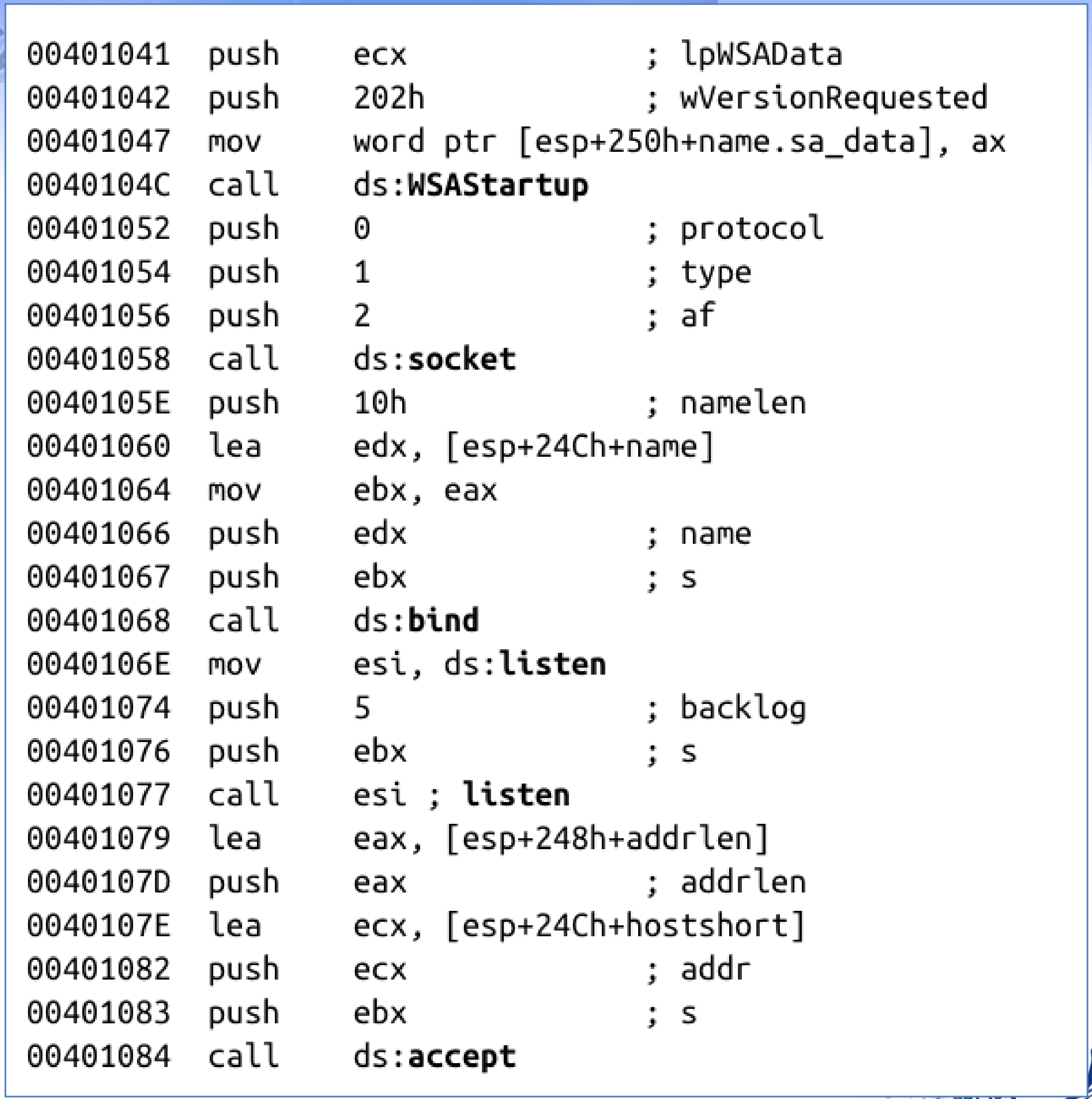
The `WSAStartup` function must be called before any other networking functions in order to allocate resources for the networking libraries. When looking for the start of network connections while debugging code, it is useful to set a breakpoint on `WSAStartup`, because the start of networking should follow shortly.

Server and Client Sides

- Server side
  - Maintains an open socket waiting for connections
  - Calls, in order, **socket**, **bind**, **listen**, **accept**
  - Then **send** and **recv** as necessary
- Client side
  - Connects to a waiting socket
  - Calls, in order, **socket**, **connect**
  - Then **send** and **recv** as necessary

Simplified Server Program

Realistic code would call **WSAGetLastError** many times



```
00401041  push    ecx                 ; lpWSAData
00401042  push    202h                ; wVersionRequested
00401047  mov     word ptr [esp+250h+name.sa_data], ax
0040104C  call    ds:WSAStartup
00401052  push    0                   ; protocol
00401054  push    1                   ; type
00401056  push    2                   ; af
00401058  call    ds:socket
0040105E  push    10h                 ; namelen
00401060  lea     edx, [esp+24Ch+name]
00401064  mov     ebx, eax
00401066  push    edx                 ; name
00401067  push    ebx                 ; s
00401068  call    ds:bind
0040106E  mov     esi, ds:listen
00401074  push    5                   ; backlog
00401076  push    ebx                 ; s
00401077  call    esi ; listen
00401079  lea     eax, [esp+248h+addrlen]
0040107D  push    eax                 ; addrlen
0040107E  lea     ecx, [esp+24Ch+hostshort]
00401082  push    ecx                 ; addr
00401083  push    ebx                 ; s
00401084  call    ds:accept
```

The WinINet API

- Higher-level API than Winsock
- Library name is "*Wininet.dll*"
- Implements Application-layer protocols like HTTP and FTP
  - **InternetOpen** – connects to Internet
  - **InternetOpenURL** –connects to a URL
  - **InternetReadFile** –reads data from a downloaded file

Following Running Malware

Transferring Execution

- **jmp** and **call** transfer execution to another part of code, but there are other ways
  - DLLs
  - Processes
  - Threads
  - Mutexes
  - Services
  - Component Object Model (COM)
  - Exceptions

DLLs (Dynamic Link Libraries)

- Share code among multiple applications
- DLLs export code that can be used by other applications
- Static libraries were used before DLLs
  - They still exist, but are much less common
  - They cannot share memory among running processes
  - Static libraries use more RAM than DLLs

DLL Advantages

- Using DLLs already included in Windows makes code smaller
- Software companies can also make custom DLLs
  - Distribute DLLs along with EXEs

How Malware Authors Use DLLs

- Store malicious code in DLL
  - Sometimes load malicious DLL into another process
- Using Windows DLLs
  - Nearly all malware uses basic Windows DLLs
- Using third-party DLLs
  - Use Firefox DLL to connect to a server, instead of Windows API

Basic DLL Structure

- DLLs are very similar to EXEs
- Same PE file format
- A single flag indicates that it's a DLL instead of an EXE
- DLLs have more exports & fewer imports
- **DllMain** is the main function, not exported, but specified as the entry point in the PE Header
    - Called when a function loads or unloads the library

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 run as an independent process
- Newer malware executes its code as part of another process

Many Processes Run at Once

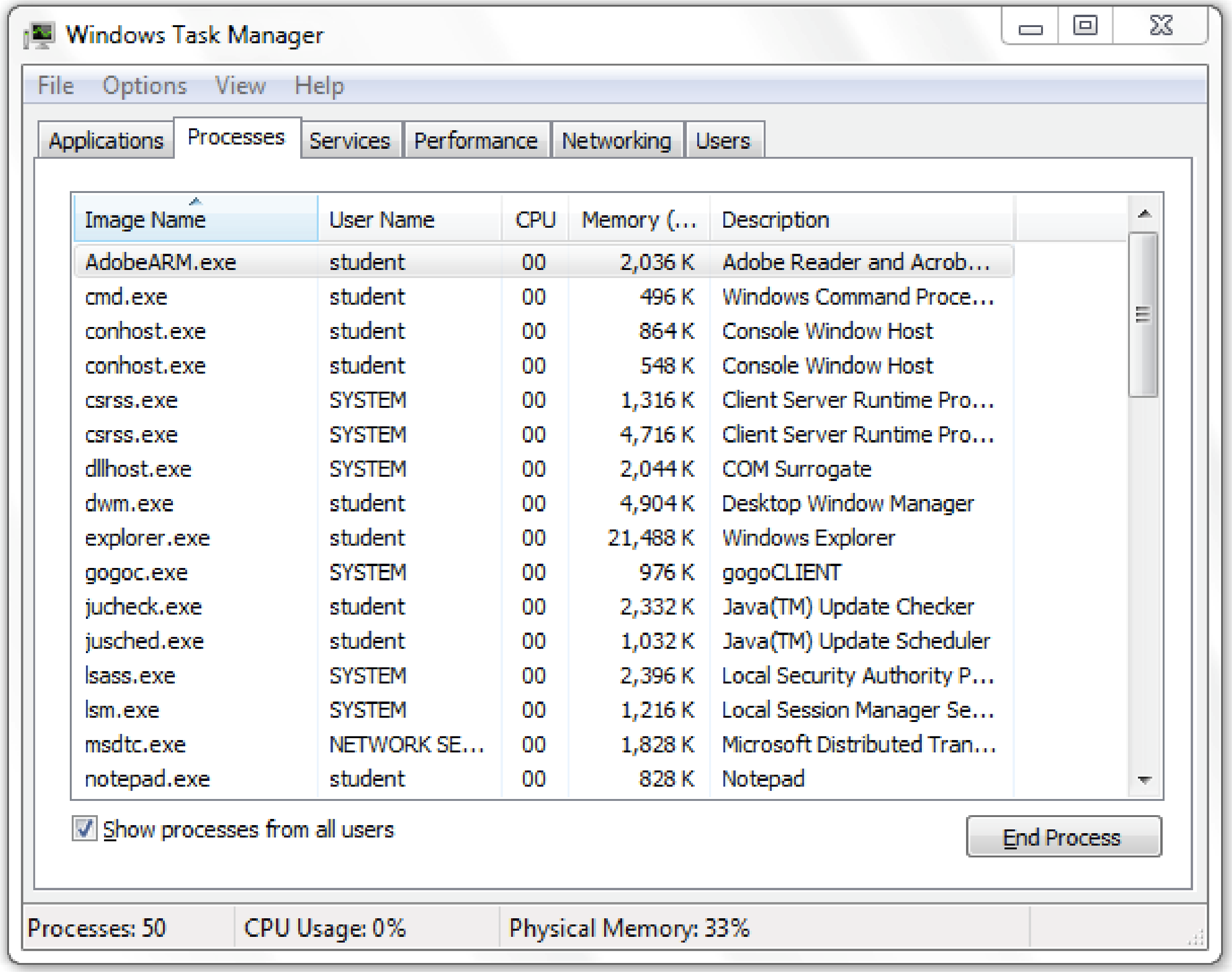


Image Name	User Name	CPU	Memory (...	Description
AdobeARM.exe	student	00	2,036 K	Adobe Reader and Acrob...
cmd.exe	student	00	496 K	Windows Command Proce...
conhost.exe	student	00	864 K	Console Window Host
conhost.exe	student	00	548 K	Console Window Host
csrss.exe	SYSTEM	00	1,316 K	Client Server Runtime 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%

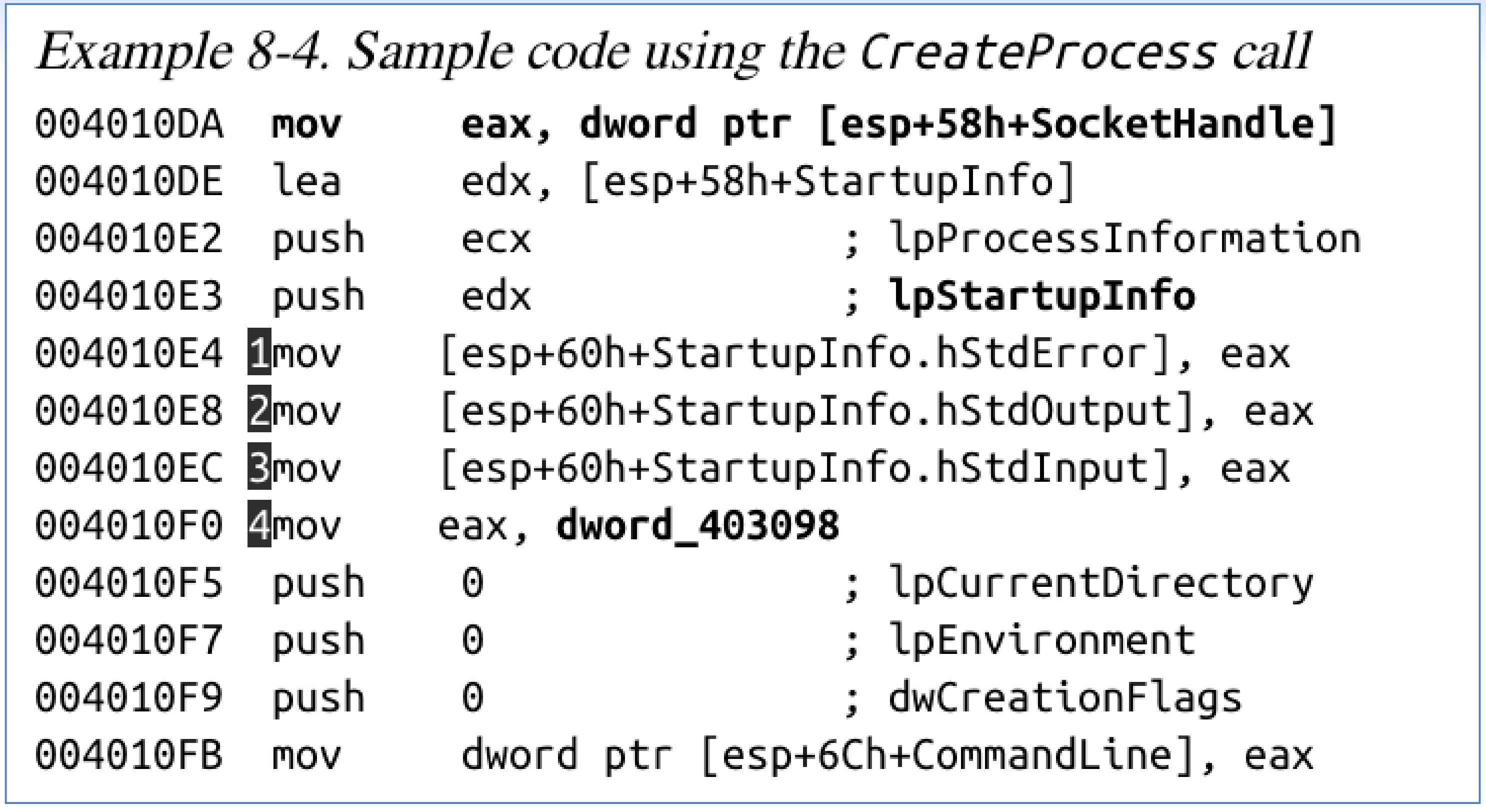
Memory Management

- 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

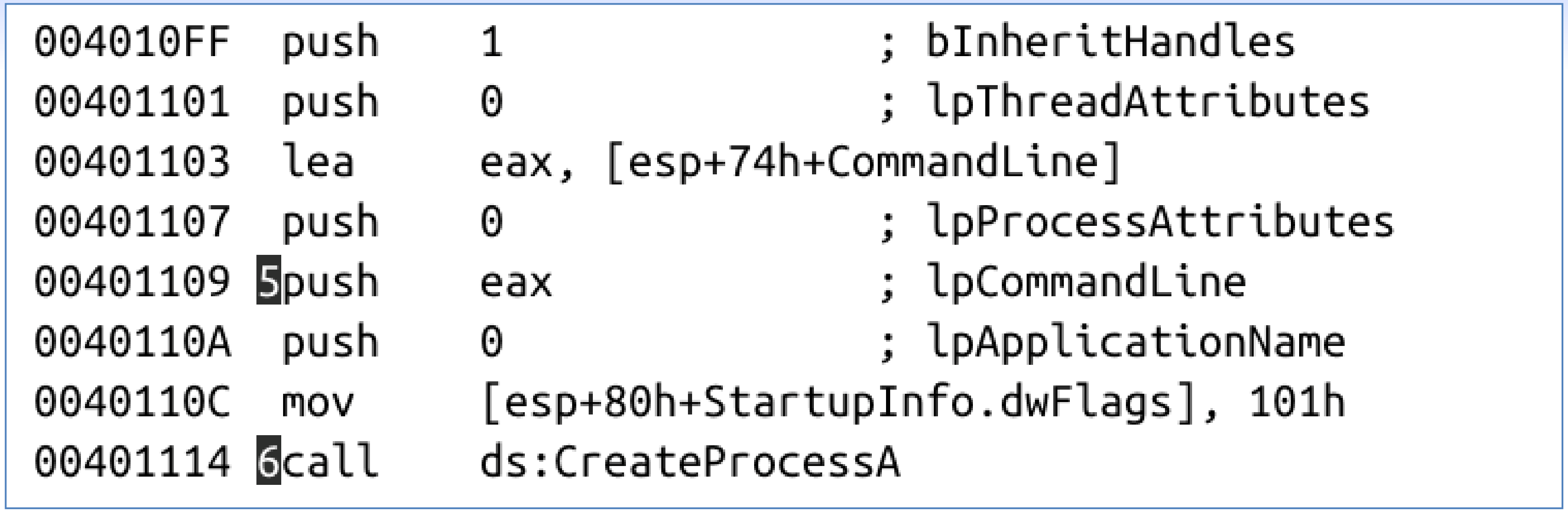
Code to Create a Shell



Example 8-4. Sample code using the `CreateProcess` call

```
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
```

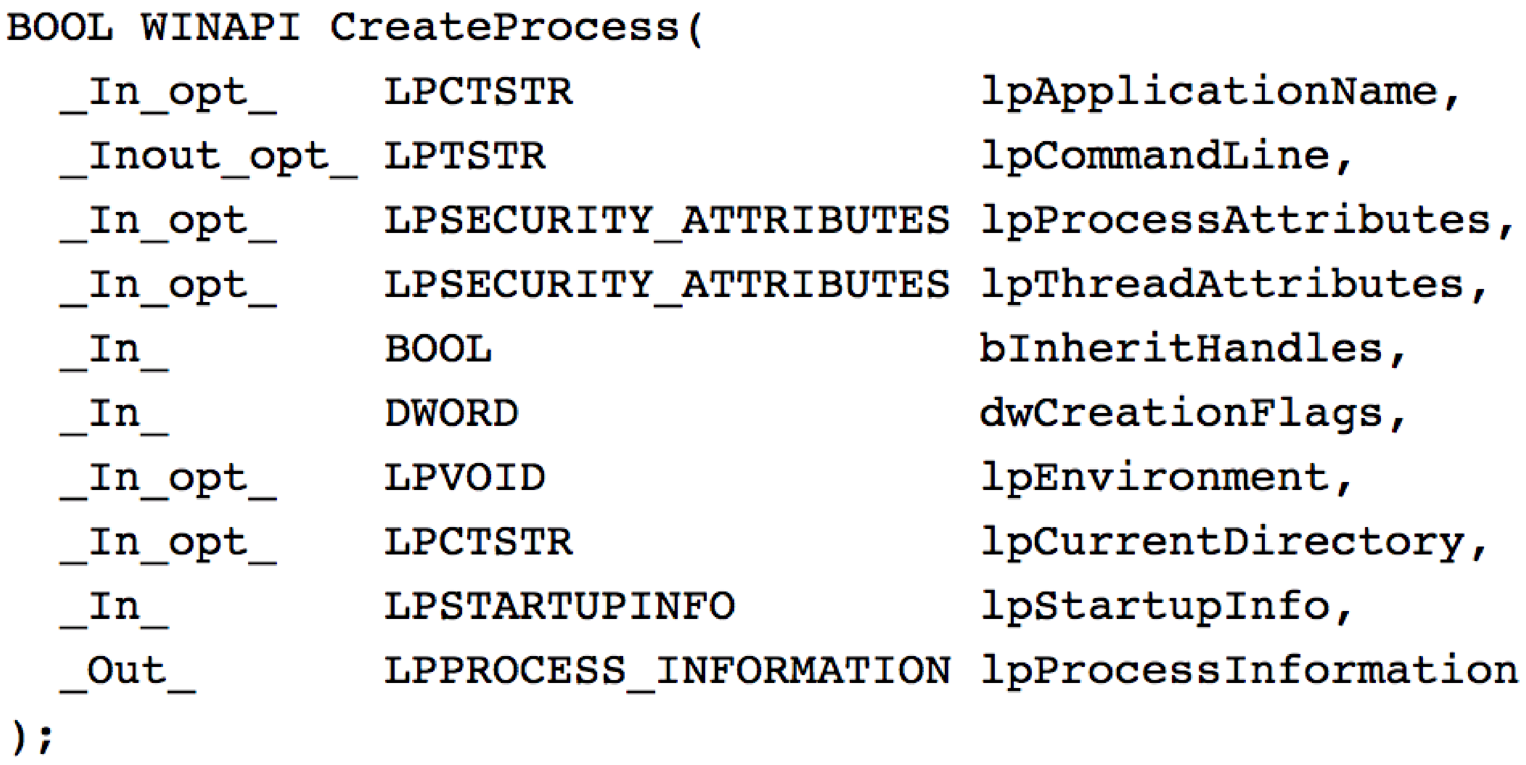
- Loads socket handles, StdError, StdOutput and StdInput into lpProcessInformation



```
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
```

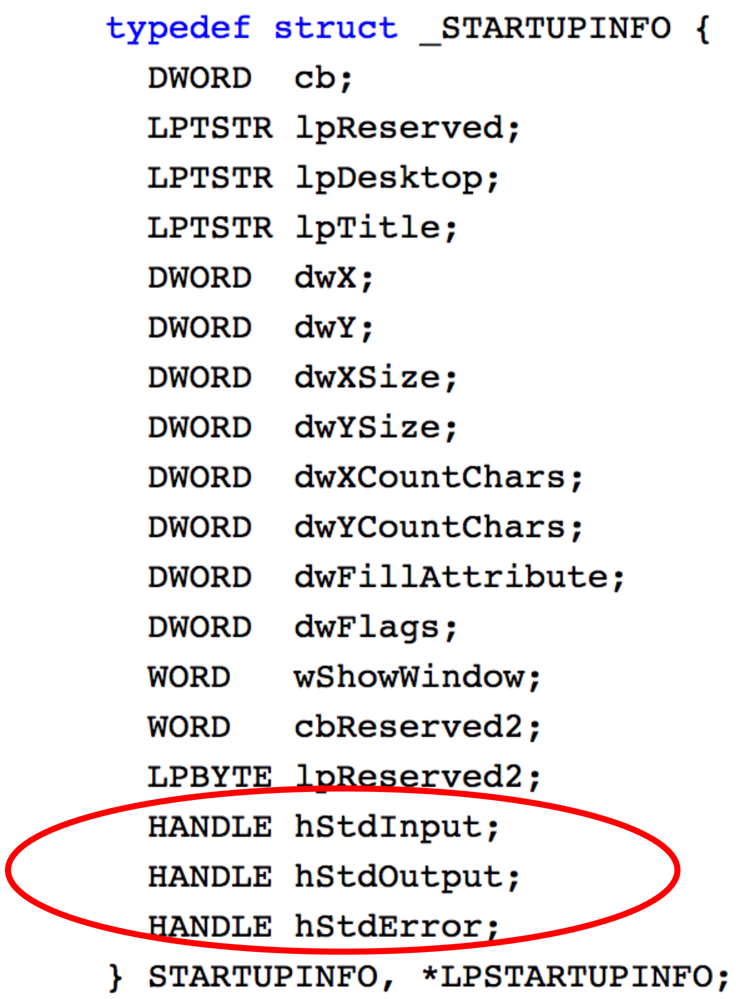
- CreateProcess has 10 parameters
  - lpCommandLine
  - lpProcessInformation
  - lpStartupInfo

Windows MSDN



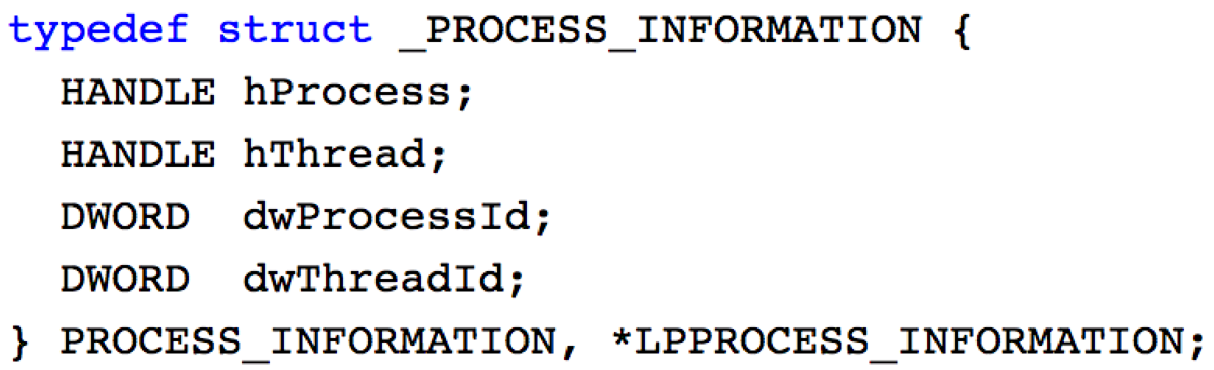
```
BOOL WINAPI CreateProcess(
  _In_opt_     LPCTSTR               lpApplicationName,
  _Inout_opt_  LPTSTR                lpCommandLine,
  _In_opt_     LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_     LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_         BOOL                  bInheritHandles,
  _In_         DWORD                 dwCreationFlags,
  _In_opt_     LPVOID                lpEnvironment,
  _In_opt_     LPCTSTR               lpCurrentDirectory,
  _In_         LPSTARTUPINFO         lpStartupInfo,
  _Out_        LPPROCESS_INFORMATION lpProcessInformation
);
```

STARTUPINFO



```
typedef struct _STARTUPINFO {
  DWORD  cb;
  LPTSTR lpReserved;
  LPTSTR lpDesktop;
  LPTSTR lpTitle;
  DWORD  dwX;
  DWORD  dwY;
  DWORD  dwXSize;
  DWORD  dwYSize;
  DWORD  dwXCountChars;
  DWORD  dwYCountChars;
  DWORD  dwFillAttribute;
  DWORD  dwFlags;
  WORD   wShowWindow;
  WORD   cbReserved2;
  LPBYTE lpReserved2;
  HANDLE hStdInput;
  HANDLE hStdOutput;
  HANDLE hStdError;
} STARTUPINFO, *LPSTARTUPINFO;
```

Process Information



```
typedef struct _PROCESS_INFORMATION {
  HANDLE hProcess;
  HANDLE hThread;
  DWORD  dwProcessId;
  DWORD  dwThreadId;
} PROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

Contains information about a newly created process and its primary thread.

Process and Thread

- Application
  - Consists of one or more processes
- Process
  - An executing program
  - One or more threads are running in the context of the process
- Thread
  - Basic unit to which the OS allocates CPU time

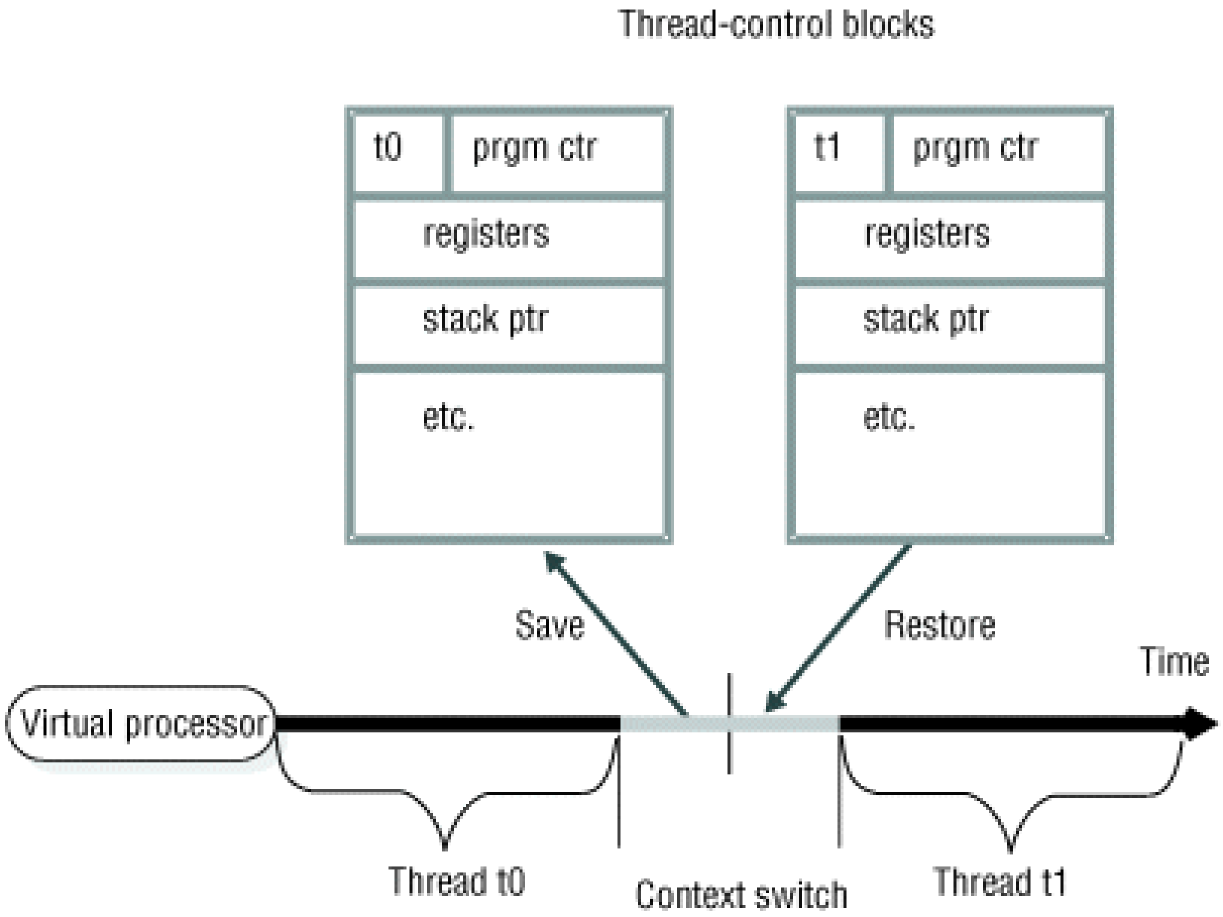
Threads

- 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

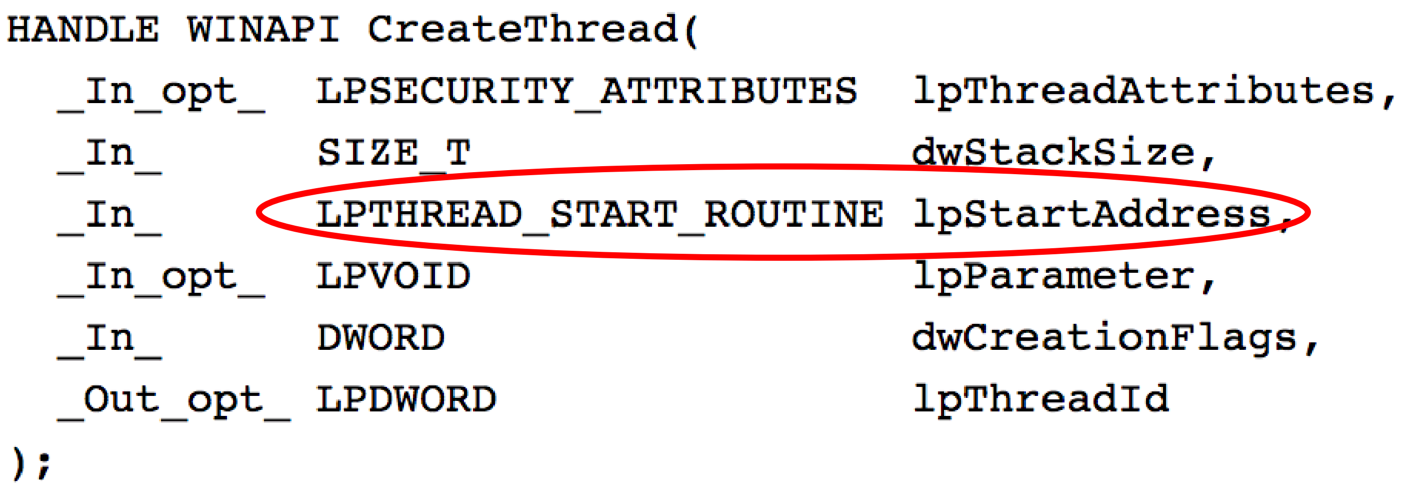
- When a thread is running, it has complete control of the CPU
- Other threads cannot affect the state of the CPU
- When a thread changes a register, it does not affect any other threads
- When the OS switches to another thread, it saves all CPU values in a structure called the **thread context**

Thread Context Switch



Creating a Thread

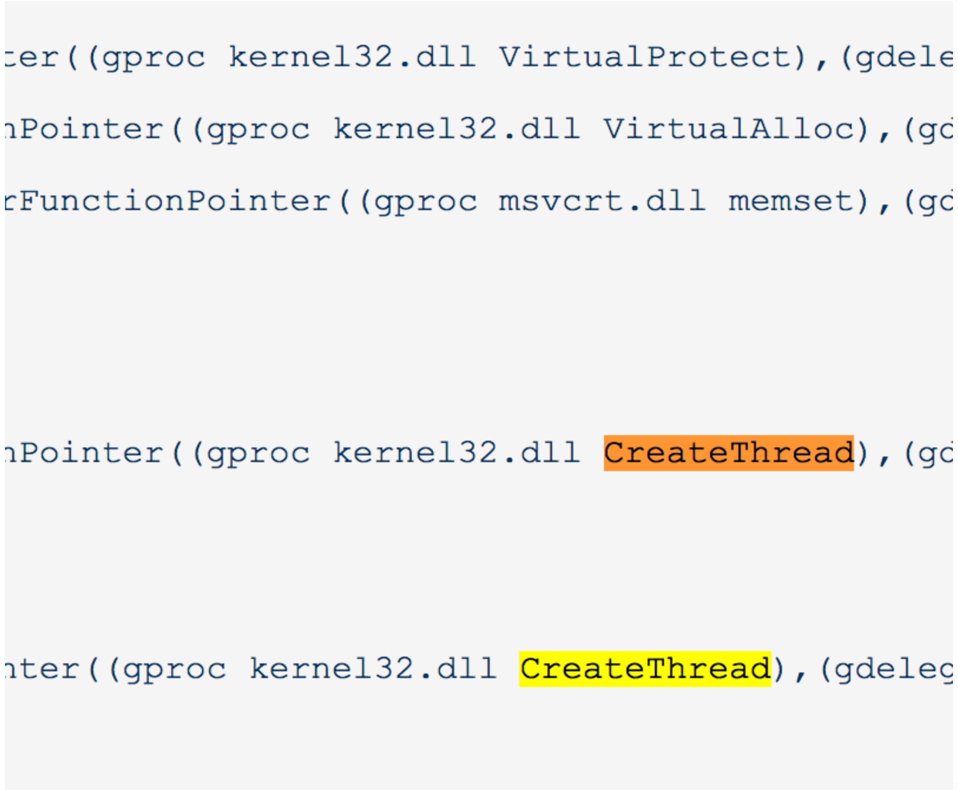
- **CreateThread**
  - Caller specified a **start** address, also called a **start** function



```
HANDLE WINAPI CreateThread(
  _In_opt_   LPSECURITY_ATTRIBUTES  lpThreadAttributes,
  _In_       SIZE_T                 dwStackSize,
  _In_       LPTHREAD_START_ROUTINE lpStartAddress,
  _In_opt_   LPVOID                 lpParameter,
  _In_       DWORD                  dwCreationFlags,
  _Out_opt_  LPDWORD                lpThreadId
);
```

How Malware Uses Threads

- Use **CreateThread** to load a malicious DLL into a process
  - Virtual Protect
  - VirtualAlloc
  - CreateThread



```
ter((gproc kernel32.dll VirtualProtect),(gdele
nPointer((gproc kernel32.dll VirtualAlloc),(gd
rFunctionPointer((gproc msvcrt.dll memset),(gd



nPointer((gproc kernel32.dll CreateThread),(gd


nter((gproc kernel32.dll CreateThread),(gdeleg
```

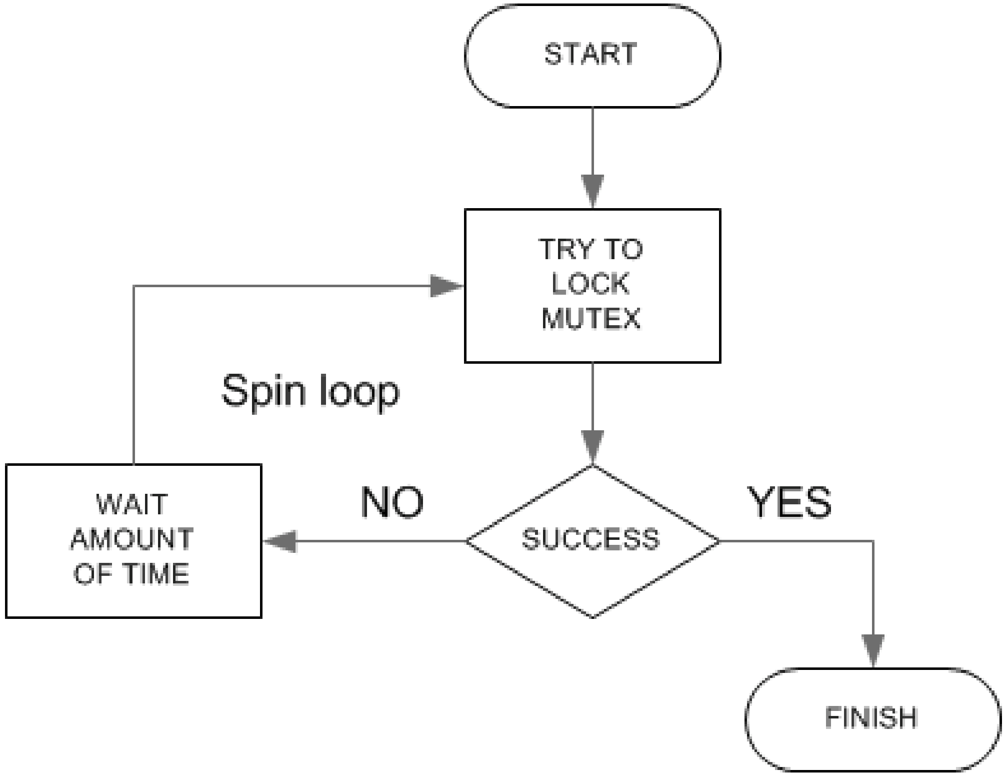
How Malware Uses Threads

- Create two threads, for input and output
  - Used to communicate with a running application
  - input: listen on a socket or pipe of a process
  - output: read from socket or pipe of a process

Interprocess Coordination with Mutexes

- **Mutexes** are global objects that coordinate multiple processes and threads
- Mutexes often use hard-coded names which can be used to identify malware

Mutex

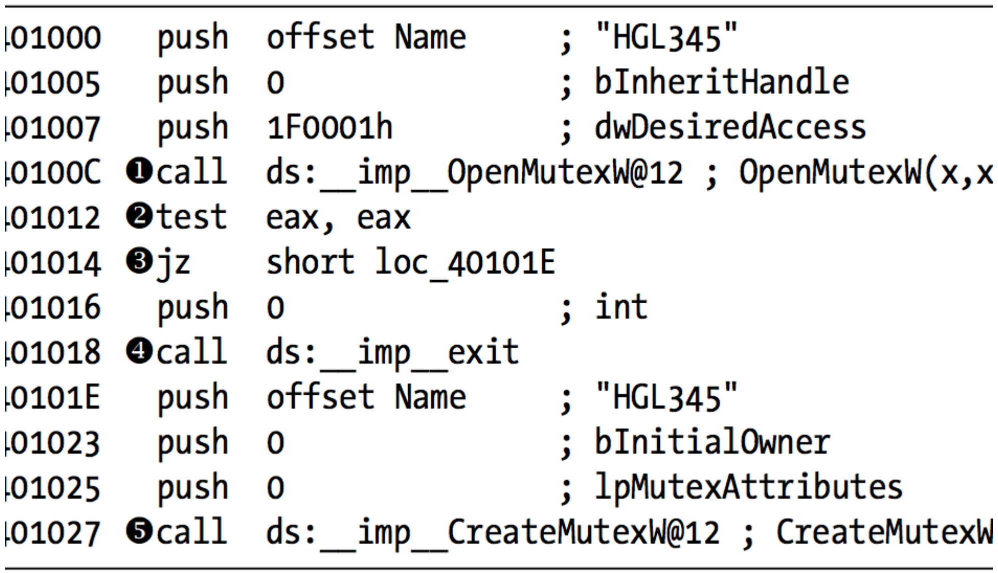

START
TRY TO LOCK MUTEX
Spin loop
WAIT AMOUNT OF TIME
NO
SUCCESS
YES
FINISH

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

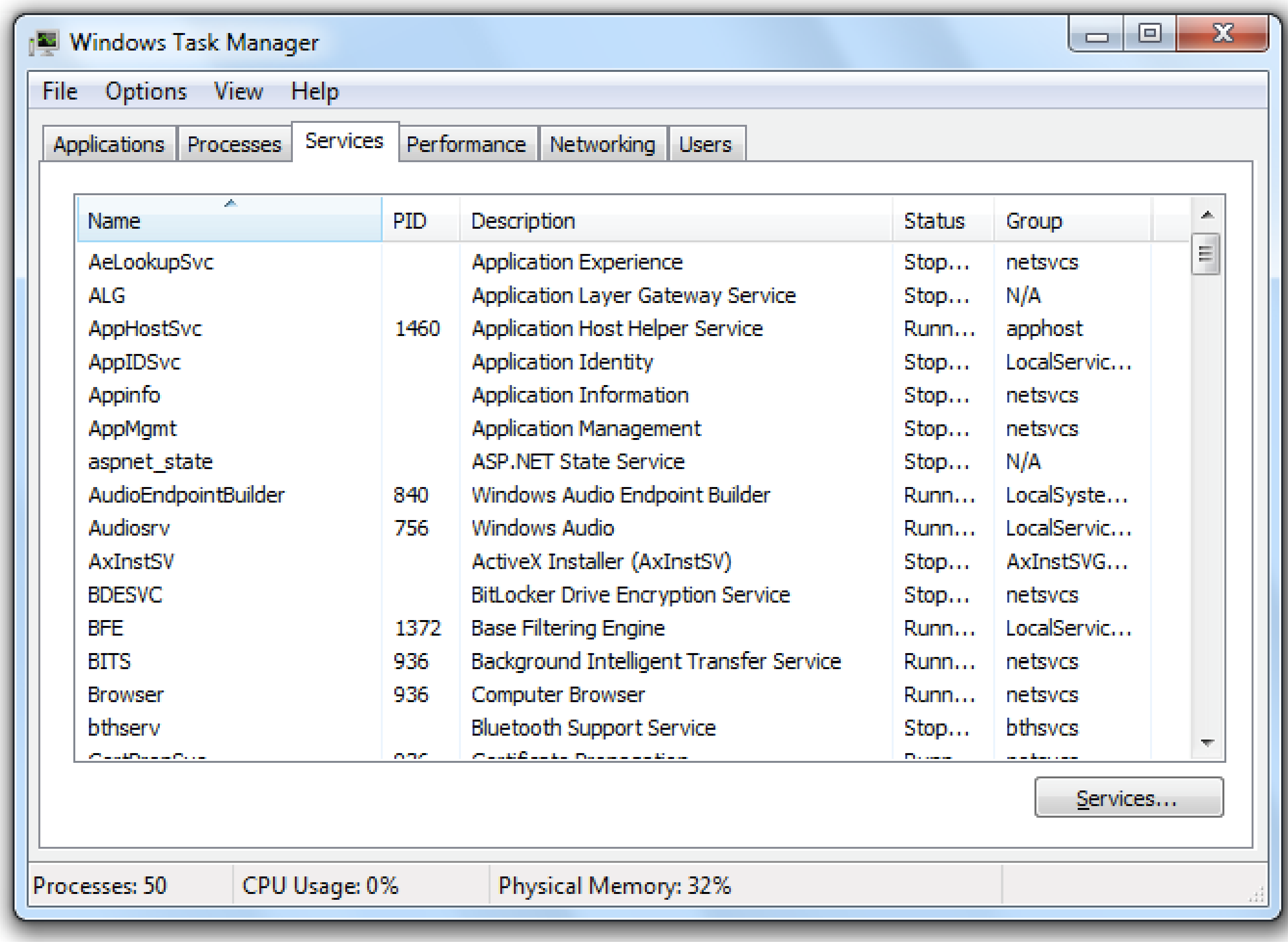
- **OpenMutex** checks if HGL345 exists
- If not, it is created with **CreateMutex**
- **test eax, eax**
  sets Z flag if eax is zero



```
401000     push   offset Name      ; "HGL345"
401005     push   0                ; bInheritHandle
401007     push   1F0001h          ; dwDesiredAccess
40100C  ❶call   ds:__imp__OpenMutexW@12 ; OpenMutexW(x,x
401012  ❷test   eax, eax
401014  ❸jz     short loc_40101E
401016     push   0                ; int
401018  ❹call   ds:__imp__exit
40101E     push   offset Name      ; "HGL345"
401023     push   0                ; bInitialOwner
401025     push   0                ; lpMutexAttributes
401027  ❺call   ds:__imp__CreateMutexW@12 ; CreateMutexW
```

Services

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



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
CertPropSvc	936	Certificate Propagation	Runn...	netsvcs

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

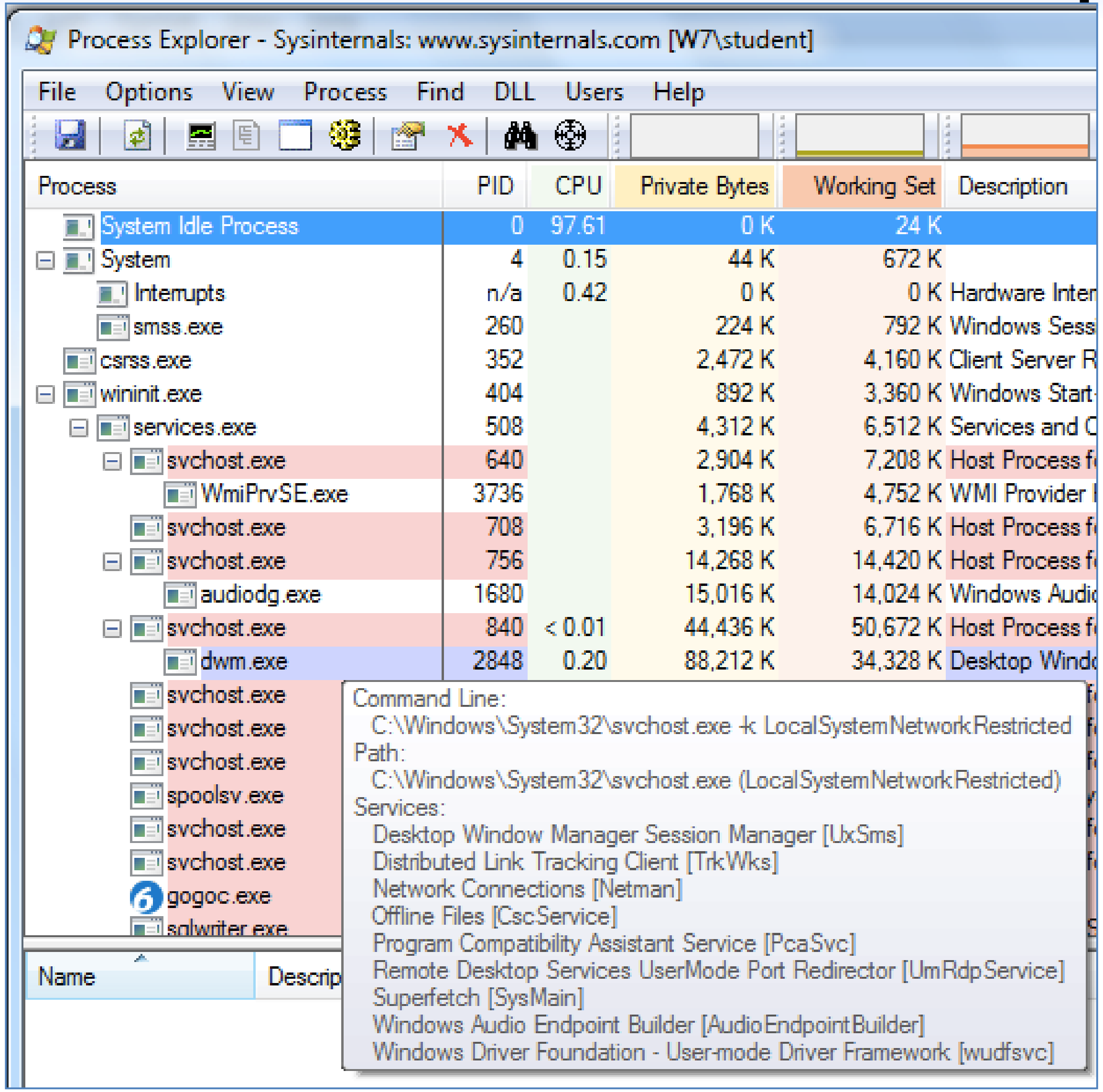
Service API Functions

- 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

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 I
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 Windo
svchost.exe					f
svchost.exe					f
svchost.exe					f
spoolsv.exe					
svchost.exe					f
svchost.exe					f
gogoc.exe					
sqlwriter.exe					S

Name	Descrip

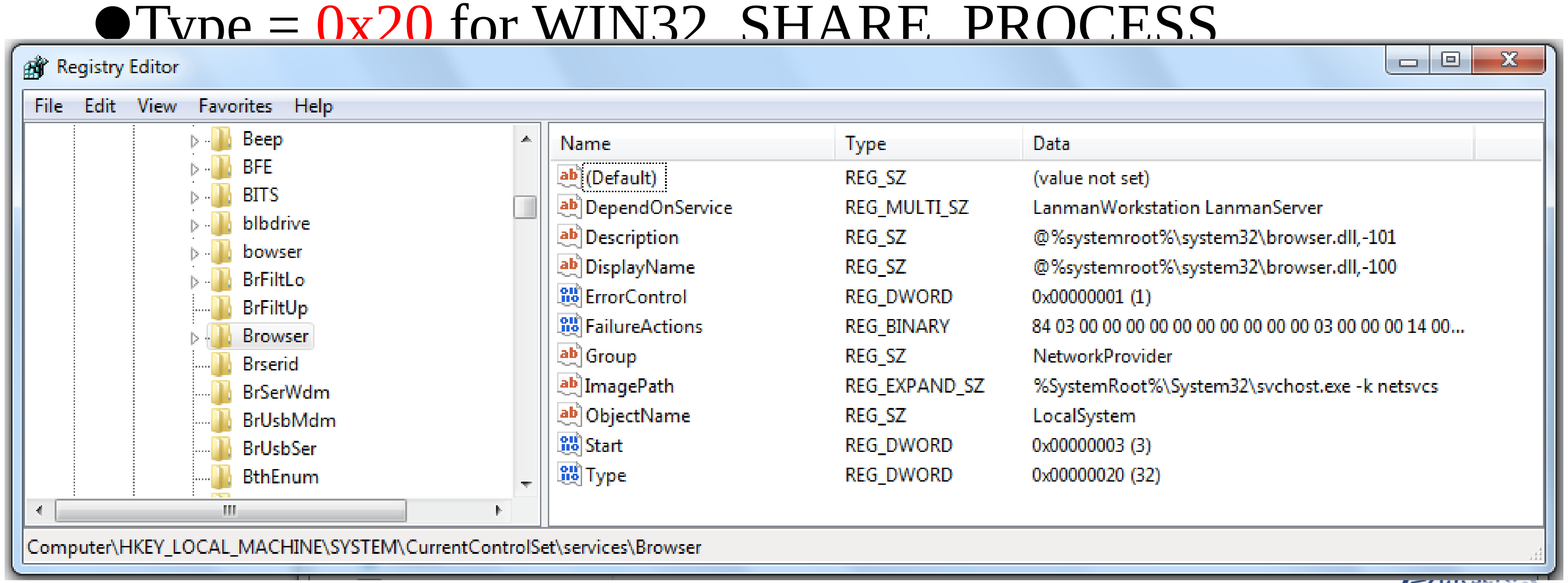
Command Line:
C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted
Path:
C:\Windows\System32\svchost.exe (LocalSystemNetworkRestricted)
Services:
Desktop Window Manager Session Manager [UxSms]
Distributed Link Tracking Client [TrkWks]
Network Connections [Netman]
Offline Files [CscService]
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]

Other Common Service Types

- WIN32_OWN_PROCESS
  - Runs as an EXE in an independent process
- KERNEL_DRIVER
  - Used to load code into the Kernel

Service Information in the Registry

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



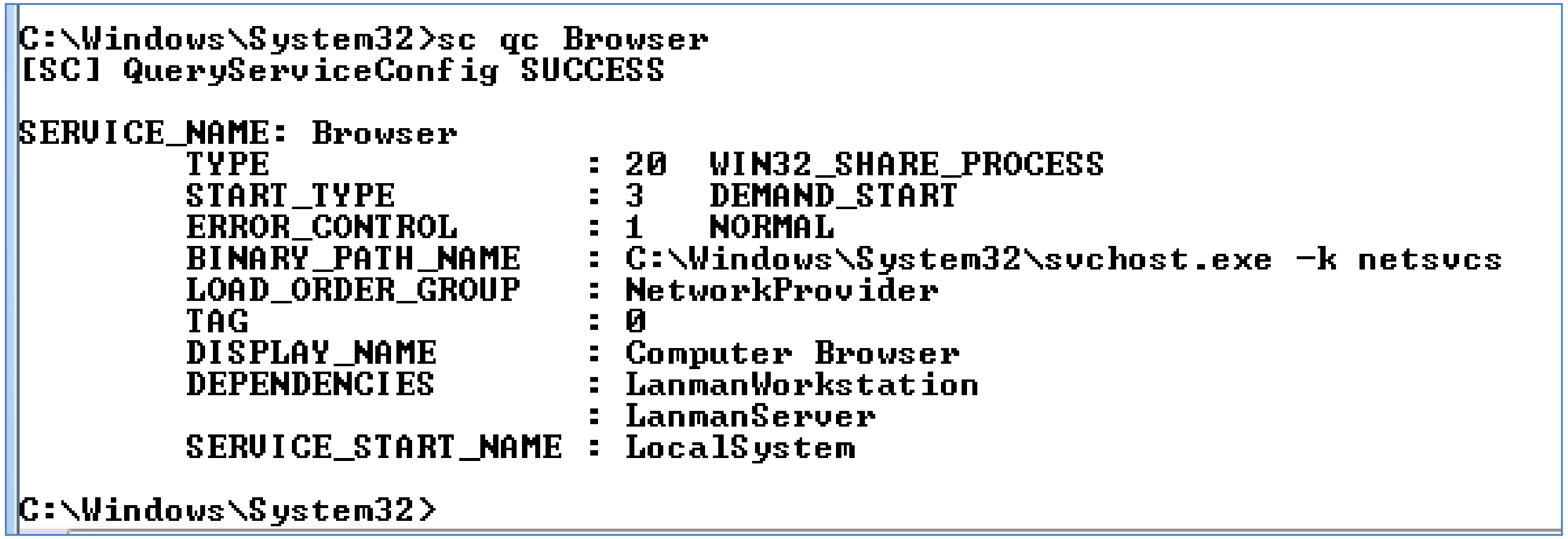
Beep
BFE
BITS
blbdrive
bowser
BrFiltLo
BrFiltUp
Browser
Brserid
BrSerWdm
BrUsbMdm
BrUsbSer
BthEnum

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)

- Microsoft COM allows different software components to share code
  - reuse software component
- Client/server framework
  - Client， programs using COM object
  - Server, reusable software component

Microsoft COM

- Microsoft provides a large number of COMs
  - ●Internet Explorer
  - ●Office Word

●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

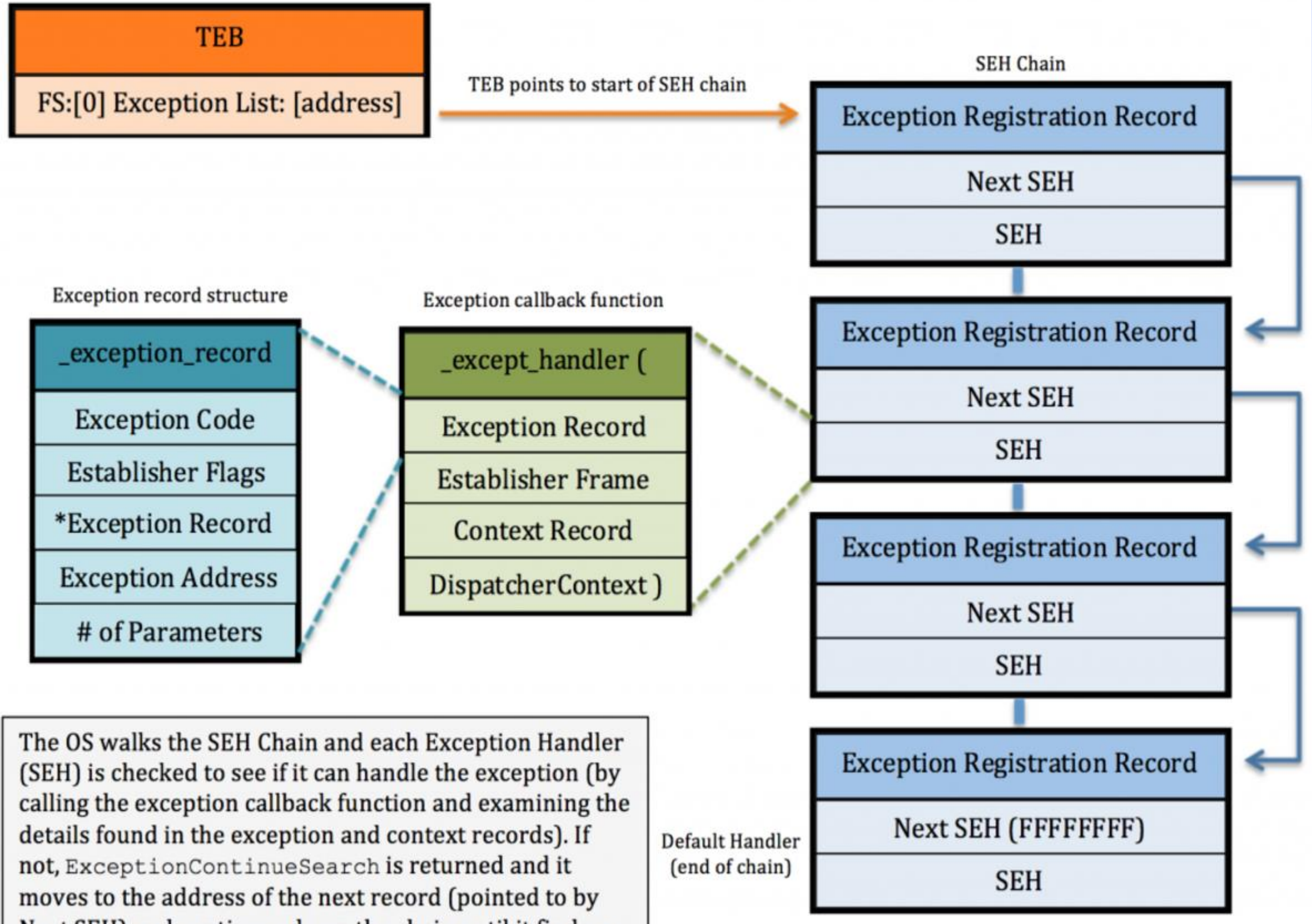
COM Server Malware

- Browser Helper Objects(BHOs)
  - third-party plug-ins for Internet Explorer
  - monitor Internet traffic
  - track browser usage
  - without running malware own process

Exceptions

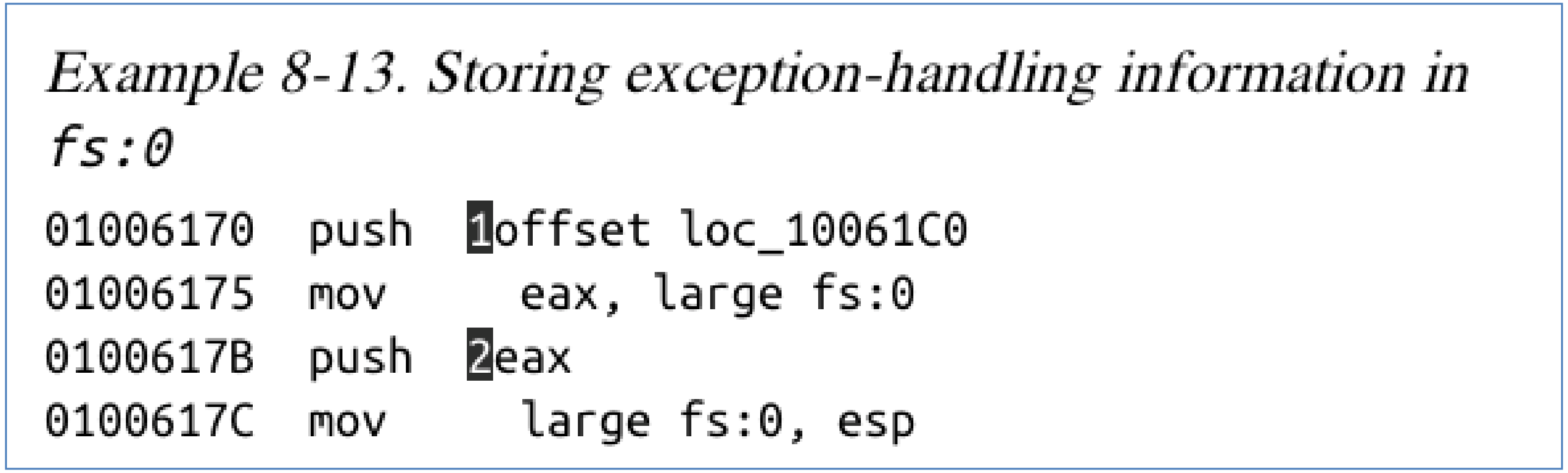
- Exceptions are caused by errors, such as division by zero or invalid memory access
- When an exception occurs, execution transfers to the **Structured Exception Handler**

Windows SEH Chain
(simplified)



The OS walks the SEH Chain and each Exception Handler (SEH) is checked to see if it can handle the exception (by calling the exception callback function and examining the details found in the exception and context records). If not, `ExceptionContinueSearch` is returned and it moves to the address of the next record (pointed to by Next SEH) and continues down the chain until it finds a suitable exception handler or hits the last, default handler (FFFFFFFF)

fs:0 Stores Exception Location



Example 8-13. Storing exception-handling information in fs:0

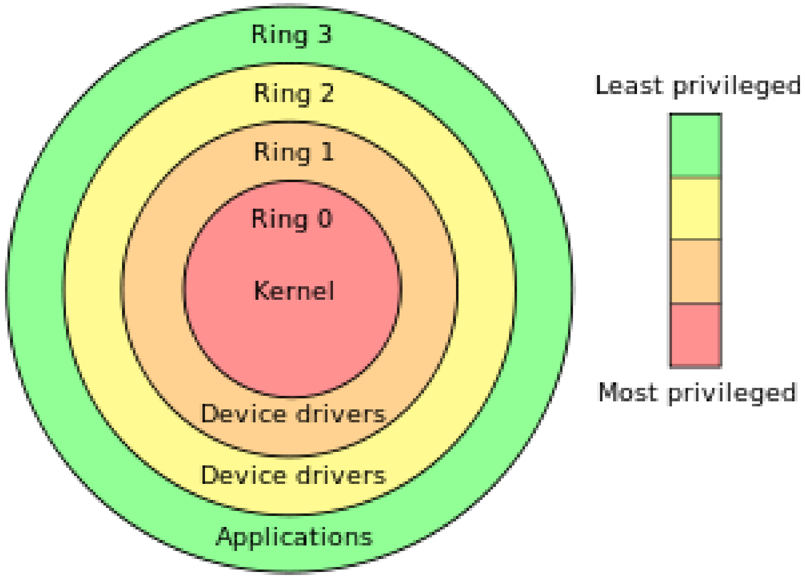
```
01006170  push  1offset loc_10061C0
01006175  mov     eax, large fs:0
0100617B  push  2eax
0100617C  mov     large fs:0, esp
```

- FS is one of six Segment Registers
- Structured Exception Handling(SEH)
    - MS mechanism for handling exceptions

Kernel Mode 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 nonmalicious programs
- Popular among malware writers

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

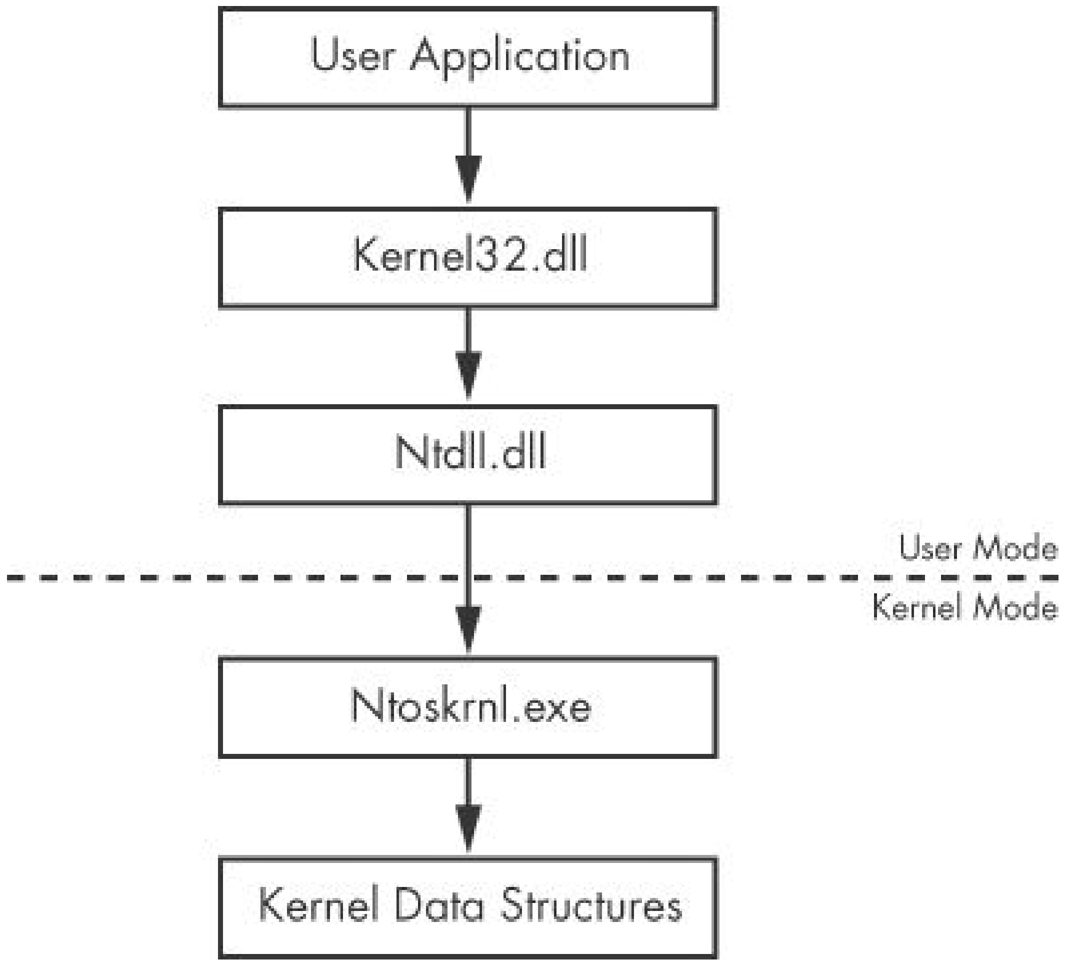


Figure 8-3. User mode and kernel mode

The Native API

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

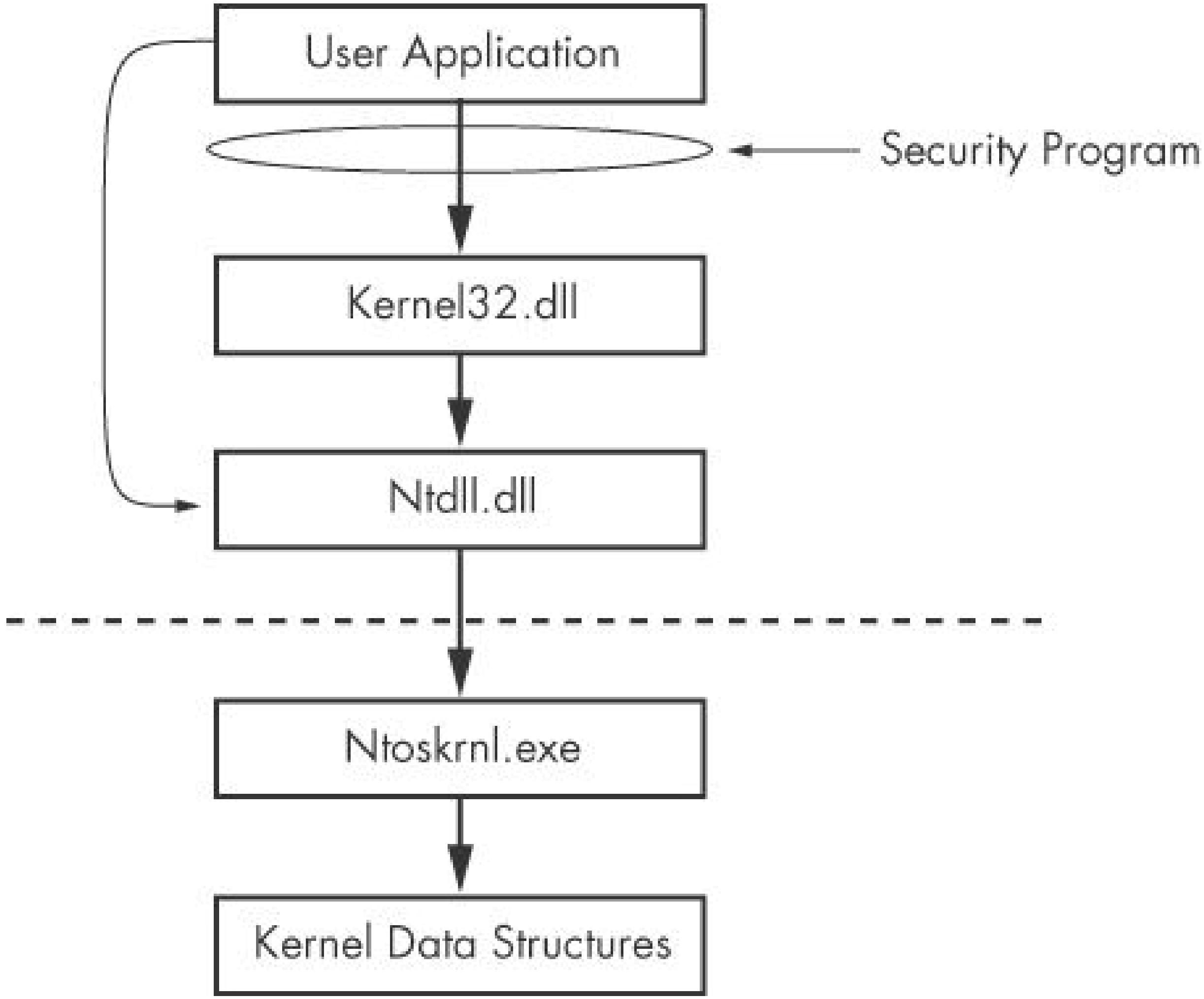


Figure 8-4. Using the Native API to avoid detection

Popular Native API Calls in Malware

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

Popular Native API Calls in Malware

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

Outline

- Windows API
- Windows Registry
- Networking APIs
- Following Running Malware
- Kernel Mode vs. User Mode
- Native API