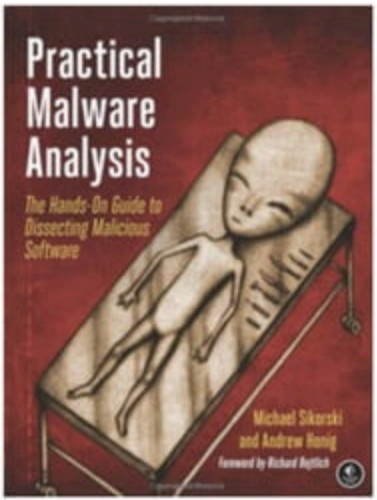
Practical Malware Analysis

Ch 7: Analyzing Malicious Windows Programs



Rev. 2-27-17

The Windows API
(Application Programming Interface)

What is the API?

- Governs how programs interact with Microsoft libraries
- Concepts
  - Types and Hungarian Notation
  - Handles
  - File System Functions
  - Special Files

Types and Hungarian Notation

- Windows API has its own names to represent C data types
  - Such as DWORD for 32-bit unsigned integers and WORD for 16-bit unsigned integers
- Hungarian Notation
  - Variables that contain a 32-bit unsigned integer start with the prefix **dw**

Common API Types

Type (Prefix)	Meaning
WORD (w)	16-bit unsigned value
DWORD (dw)	32-bit unsigned value
Handle (H)	A reference to an object
Long Pointer (LP)	Points to another type

Handles

- Items opened or created in the OS, like
  - Window, process, menu, file, ...
- Handles are like pointers to those objects
  - They 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 that handle

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

    \ Lowest namespace, contains everything

    \\.\Device namespace used for direct disk input/output

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

    Link Ch 7a

Special Files

- **Alternate Data Streams**
  - Second stream of data attached to a filename
  - File.txt:otherfile.txt

Administrator: Command Prompt

```
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 2222222222222222222222222 > 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>
```

foo:bar.txt - Notepad

File Edit Format View Help

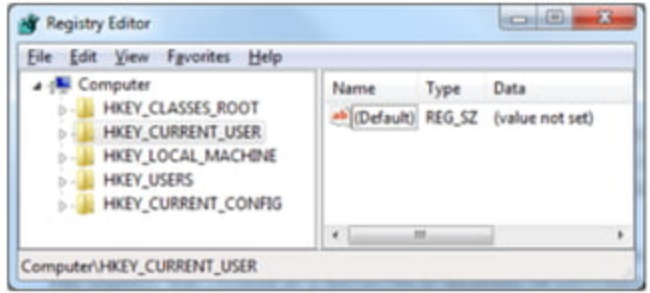
2222222222222222222222222

The Windows Registry

Registry Purpose

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

Registry Terms



- **Root keys** These 5

- **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

Root Keys

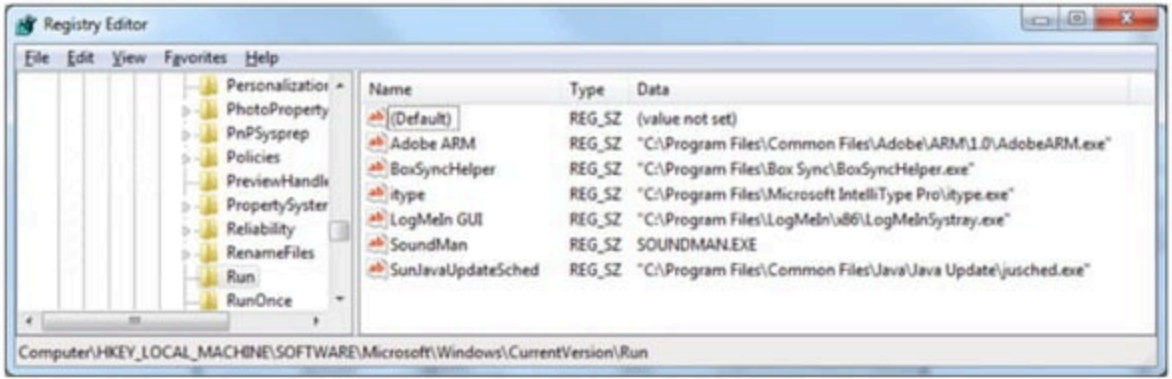
Registry Root Keys

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

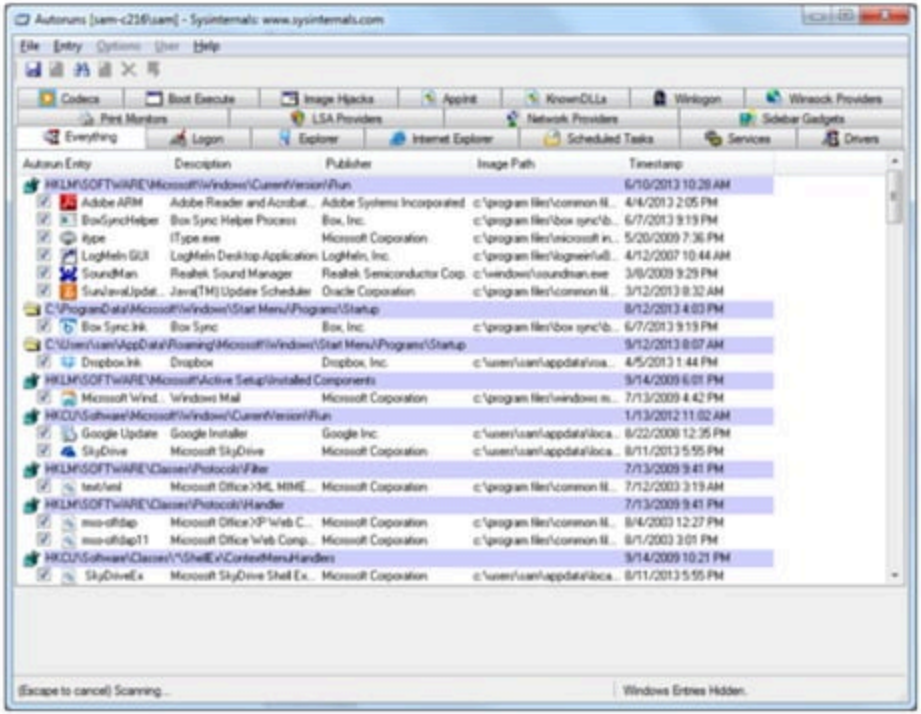
- **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**
  - Executables that start when a user logs on



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 25 to 30 registry locations
  - Won't necessarily find all automatically running code
- Link Ch 7b

Autoruns


Autoruns [sam-c216\sam] - Sysinternals: www.sysinternals.com
File Entry Options User Help
Codecs
Boot Execute
Image Hijacks
AppInit
KnownDLLs
Winlogon
Winsock Providers
Print Monitors
LSA Providers
Network Providers
Sidebar Gadgets
Everything
Logon
Explorer
Internet Explorer
Scheduled Tasks
Services
Drivers
Autorun Entry
Description
Publisher
Image Path
Timestamp
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Adobe ARM
Adobe Reader and Acrobat...
Adobe Systems Incorporated
BoxSyncHelper
Box Sync Helper Process
Box, Inc.
iype
iType.exe
Microsoft Corporation
LogMeIn GUI
LogMeIn Desktop Application
LogMeIn, Inc.
SoundMan
Realtek Sound Manager
Realtek Semiconductor Corp.
SunJavaUpdat...
Java(TM) Update Scheduler
Oracle Corporation
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup
Box Sync.lnk
Box Sync
Box, Inc.
C:\Users\sam\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
Dropbox.lnk
Dropbox
Dropbox, Inc.
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components
Microsoft Wind... \Windows Mail
Microsoft Corporation
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
Google Update
Google Installer
Google Inc.
SkyDrive
Microsoft SkyDrive
Microsoft Corporation
HKLM\SOFTWARE\Classes\Protocols\Filter
text/wsl
Microsoft Office XML MIME...
Microsoft Corporation
HKLM\SOFTWARE\Classes\Protocols\Handler
mso-offdap
Microsoft Office XP Web C...
Microsoft Corporation
mso-offdap11
Microsoft Office Web Comp...
Microsoft Corporation
HKCU\Software\Classes*\ShellEx\ContextMenuHandlers
SkyDriveEx
Microsoft SkyDrive Shell Ex...
Microsoft Corporation
(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.

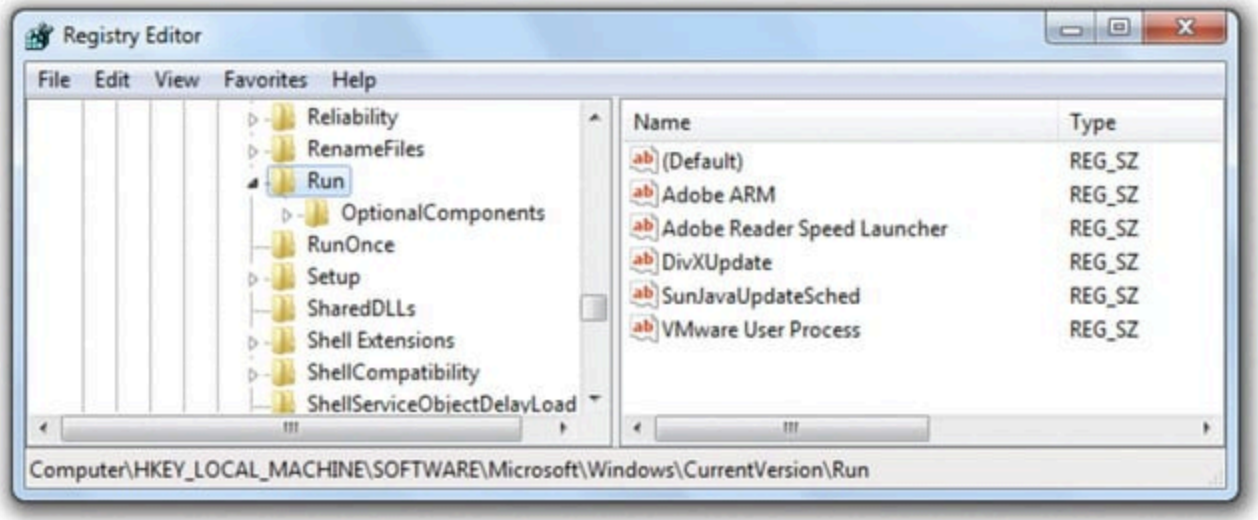
Many functions that take strings as parameters include an A or a W at the end of their names, such as `CreateDirectoryW`. This letter does *not* appear in the documentation for the function; it simply indicates that the function accepts a string parameter and that there are two different versions of 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.

- From Ch 2

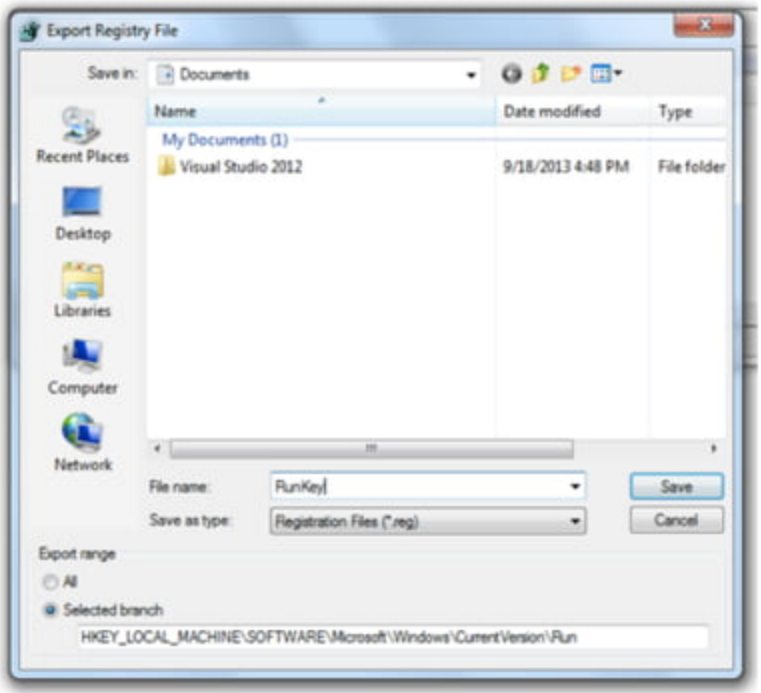
Example 8-1. Code that modifies registry settings

```
0040286F   push    2                ; samDesired
00402871   push    eax              ; ulOptions
00402872   push    offset SubKey    ;
"Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877   push    HKEY_LOCAL_MACHINE ; hKey
0040287C  1call    esi ; RegOpenKeyExW
0040287E   test    eax, eax
00402880   jnz     short loc_4028C5
00402882
00402882 loc_402882:
00402882   lea     ecx, [esp+424h+Data]
00402886   push    ecx              ; lpString
00402887   mov     bl, 1
00402889  2call    ds:lstrlenW
0040288F   lea     edx, [eax+eax+2]
00402893  3push    edx              ; cbData
00402894   mov     edx, [esp+428h+hKey]
00402898  4lea     eax, [esp+428h+Data]
0040289C   push    eax              ; lpData
0040289D   push    1                ; dwType
0040289F   push    0                ; Reserved
004028A1  5lea     ecx, [esp+434h+ValueName]
004028A8   push    ecx              ; lpValueName
004028A9   push    edx              ; hKey
004028AA   call    ds:RegSetValueExW
```

.REG Files


Registry Editor
File Edit View Favorites Help
Reliability
RenameFiles
Run
OptionalComponents
RunOnce
Setup
SharedDLLs
Shell Extensions
ShellCompatibility
ShellServiceObjectDelayLoad
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
Recent Places
Desktop
Libraries
Computer
Network
File name:
RunKey
Save as type:
Registration Files (*.reg)
Save
Cancel
Export range
All
Selected branch
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

.REG Files

RunKey.reg - Notepad

File Edit Format View Help

```
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
  - Berkeley compatible sockets

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
- Functions in *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

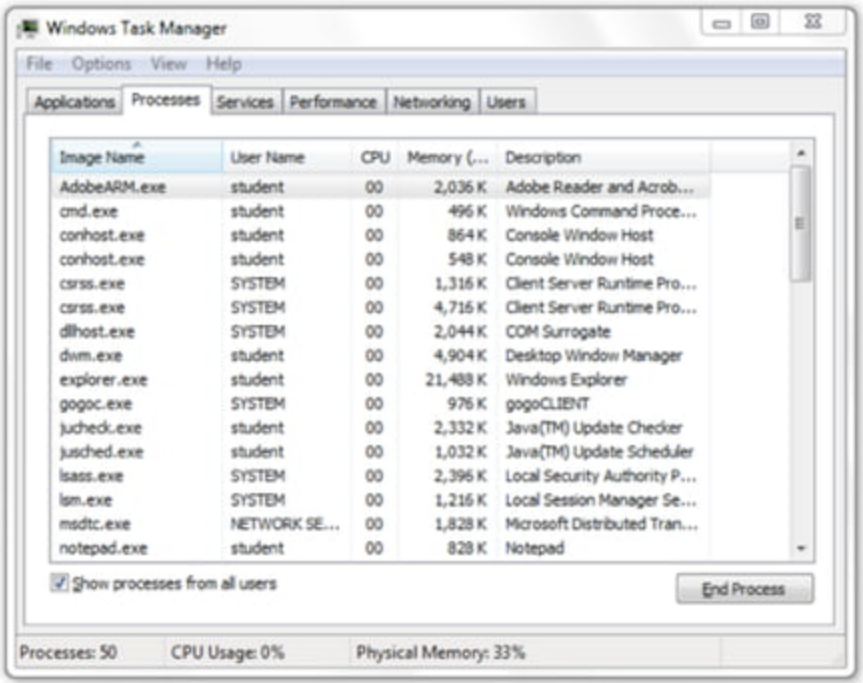
Basic DLL Structure

- DLLs are very similar to EXEs
- 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 ran as an independent process
- Newer malware executes its code as part of another process

Many Processes Run at Once



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** (link Ch 7c)

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 handle, 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
```

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

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**
  - 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

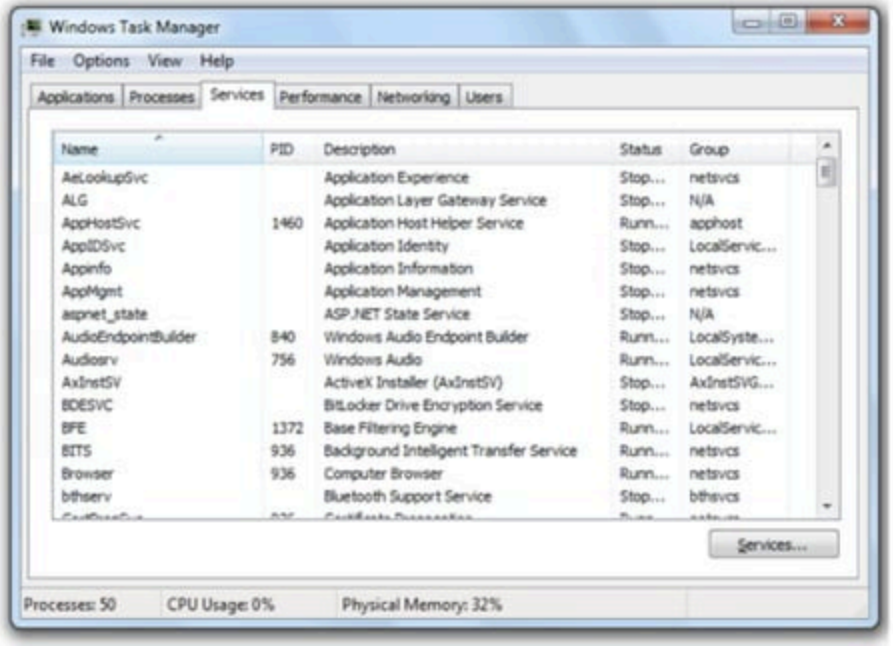
Making Sure Only One Copy of Malware is Running

- **OpenMutex** checks if HGL345 exists
- If not, it is created with **CreateMutex**
- **test eax, eax** sets Z flag if eax is zero (link Ch 7d)

```
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 run in the background without user input



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

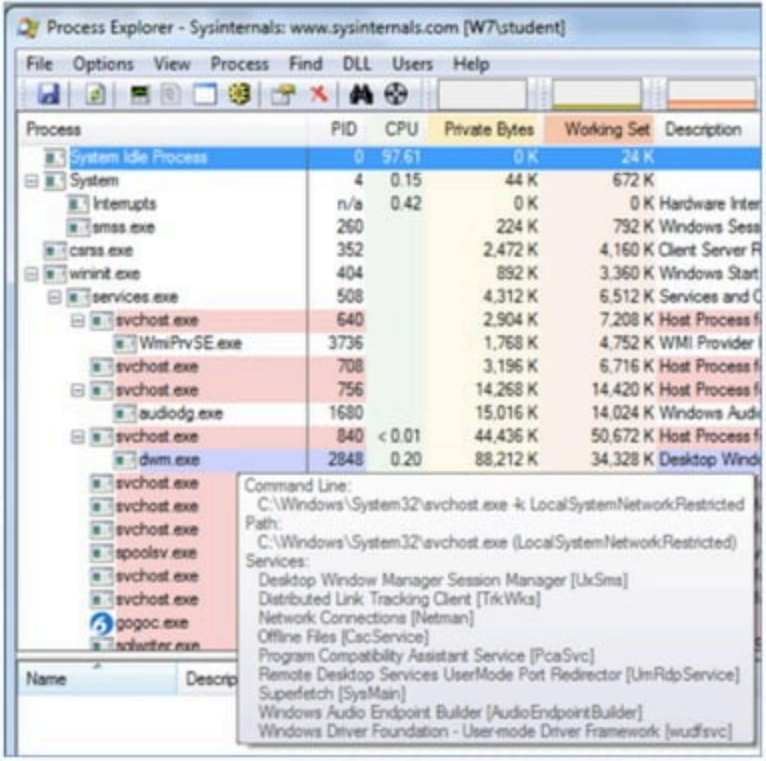
Service API Functions

- 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

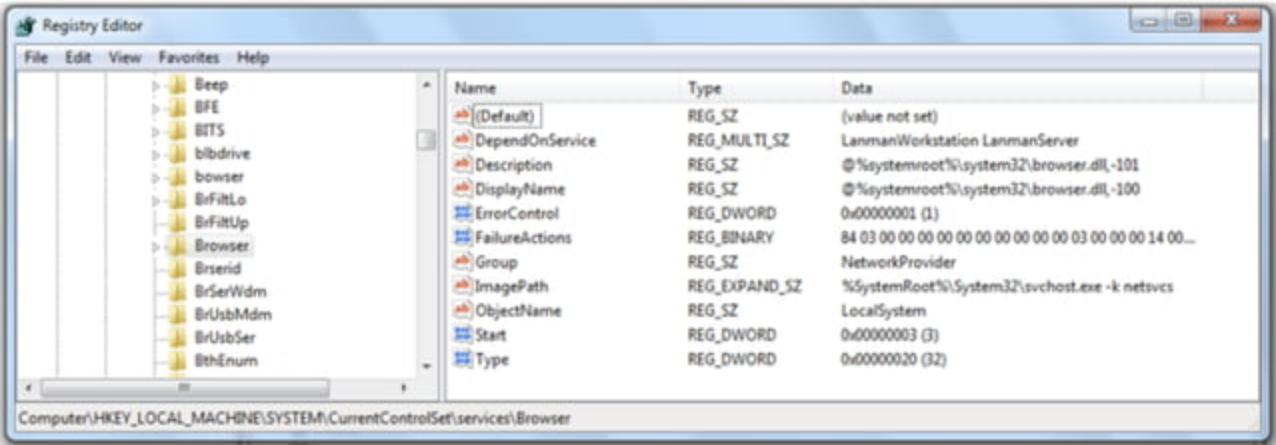


Other Common Service Types

- 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
    - Link Ch 7e



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)

- 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
- Link Ch 7f

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**

fs:0 Stores Exception Location

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

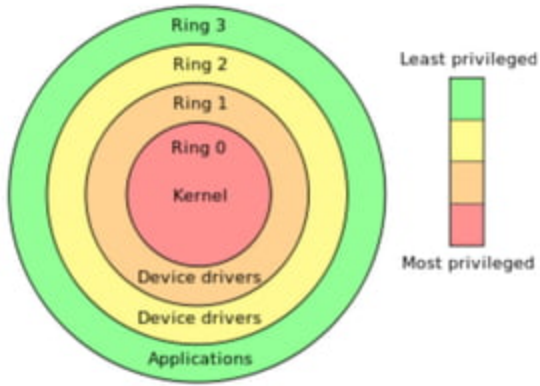
```
01006170  push  1offset loc_10061C0
01006175  mov     eax, large fs:0
0100617B  push  2eax
0100617C  mov     large fs:0, esp
```

- FS is one of six Segment Registers
- Link Ch 7g-i

Kernel v. User Mode

Two Privilege Levels

- Ring 0: Kernel Mode
- Ring 3: User mode
- Rings 1 and 2 are not used by Windows
  - Link Ch 7j



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

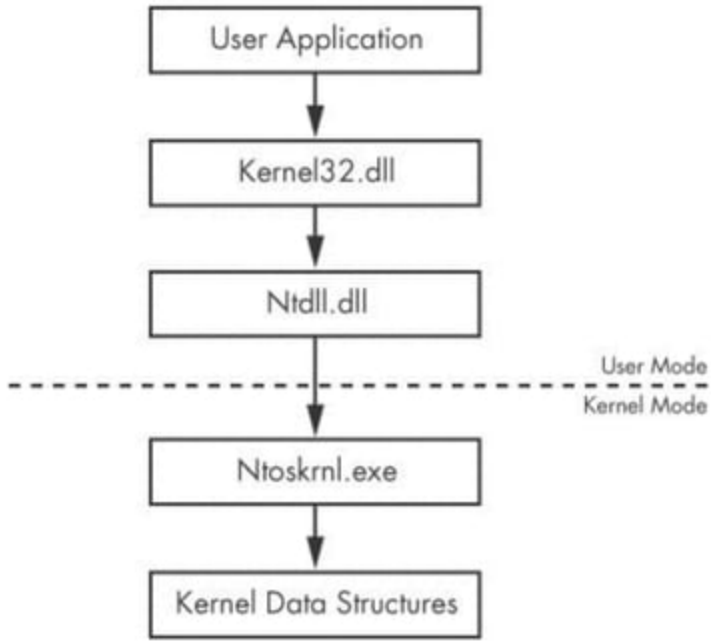


Figure 8-3. User mode and kernel mode

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

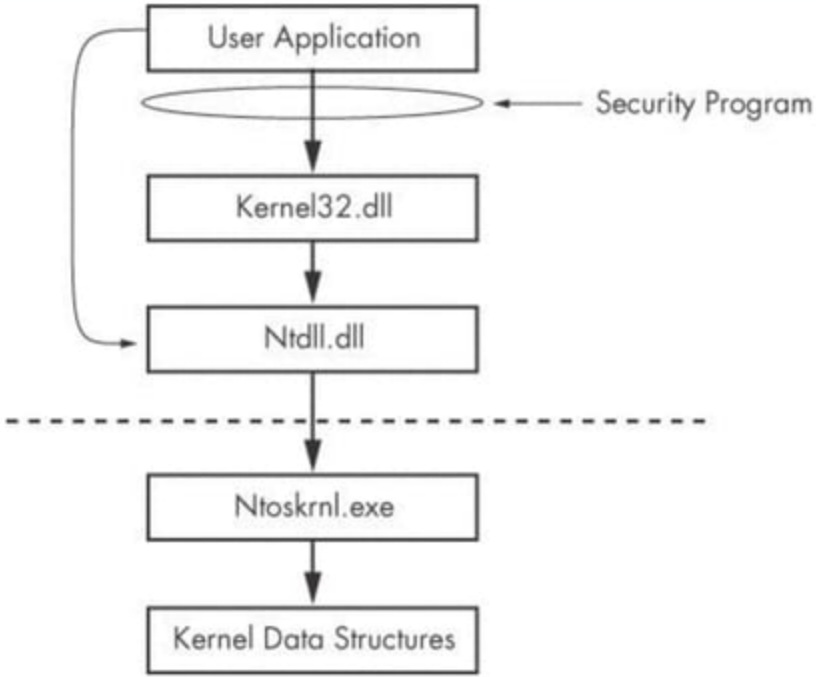


Figure 8-4. Using the Native API to avoid detection

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