School of

Engineering and Computer Science

Te Kura Mātai Pūkaha, Pūrorohiko

CYBR 473 T1 2023

Malware and Reverse Engineering

Analysing Malicious Windows Programs (A)

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

PART I:
THE WINDOWS APPLICATION PROGRAMMING INTERFACE (API)

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 Notion

- Windows API has <u>its own names</u> 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 <u>32-bit unsigned integer</u> 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 <u>pointers</u> 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**
- Examples
  - 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, direct disk input/output

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

Special Files (cont.)

- **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 2222222222222222222222222222 > 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

2222222222222222222222222222

PART II:
THE WINDOWS REGISTRY

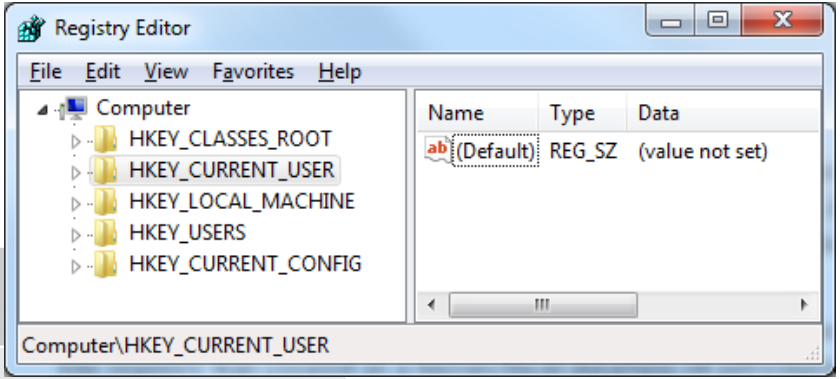
Registry Purpose

- 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

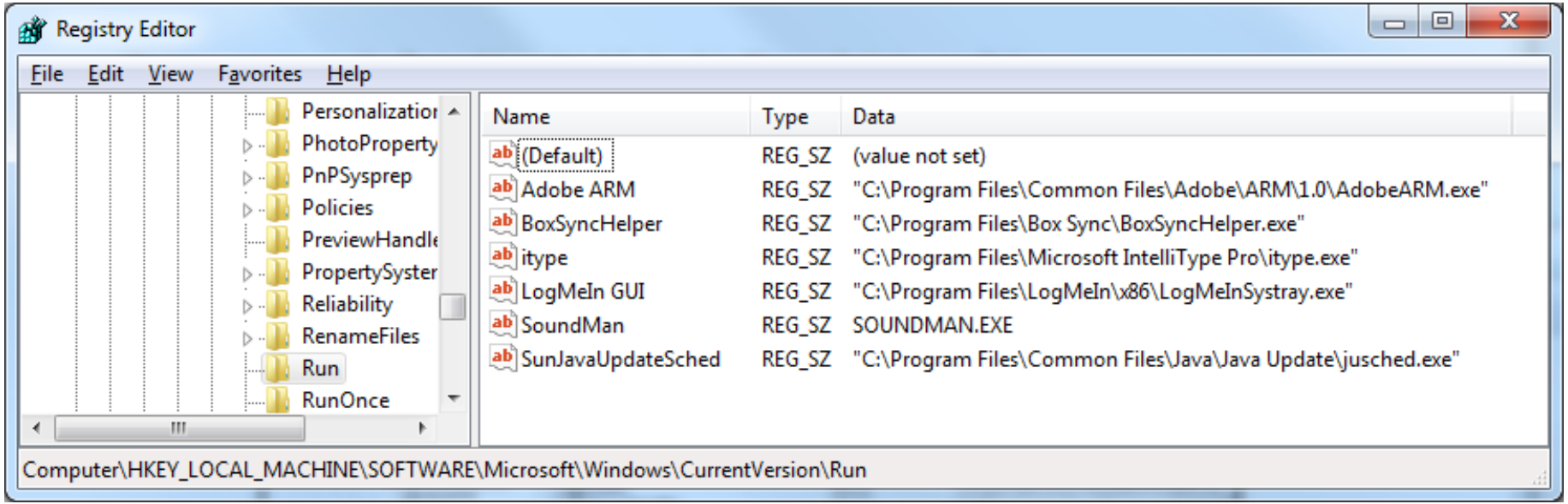
Registry Root Keys

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

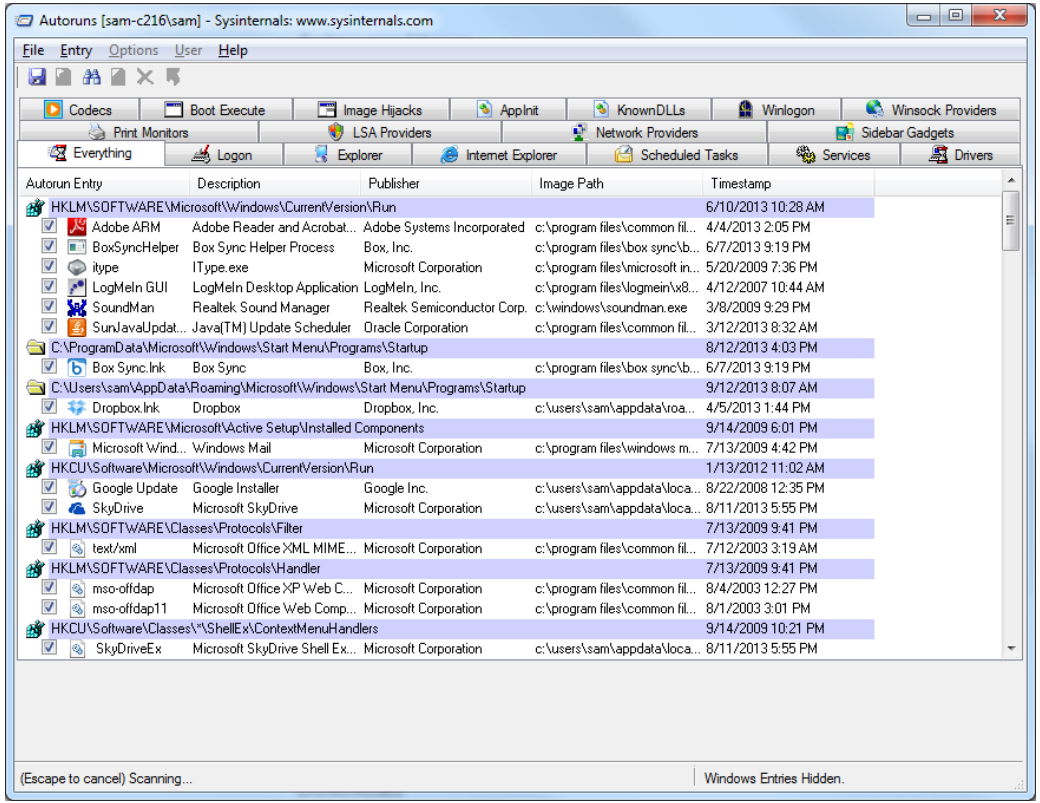
Run Key

- 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*



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 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.

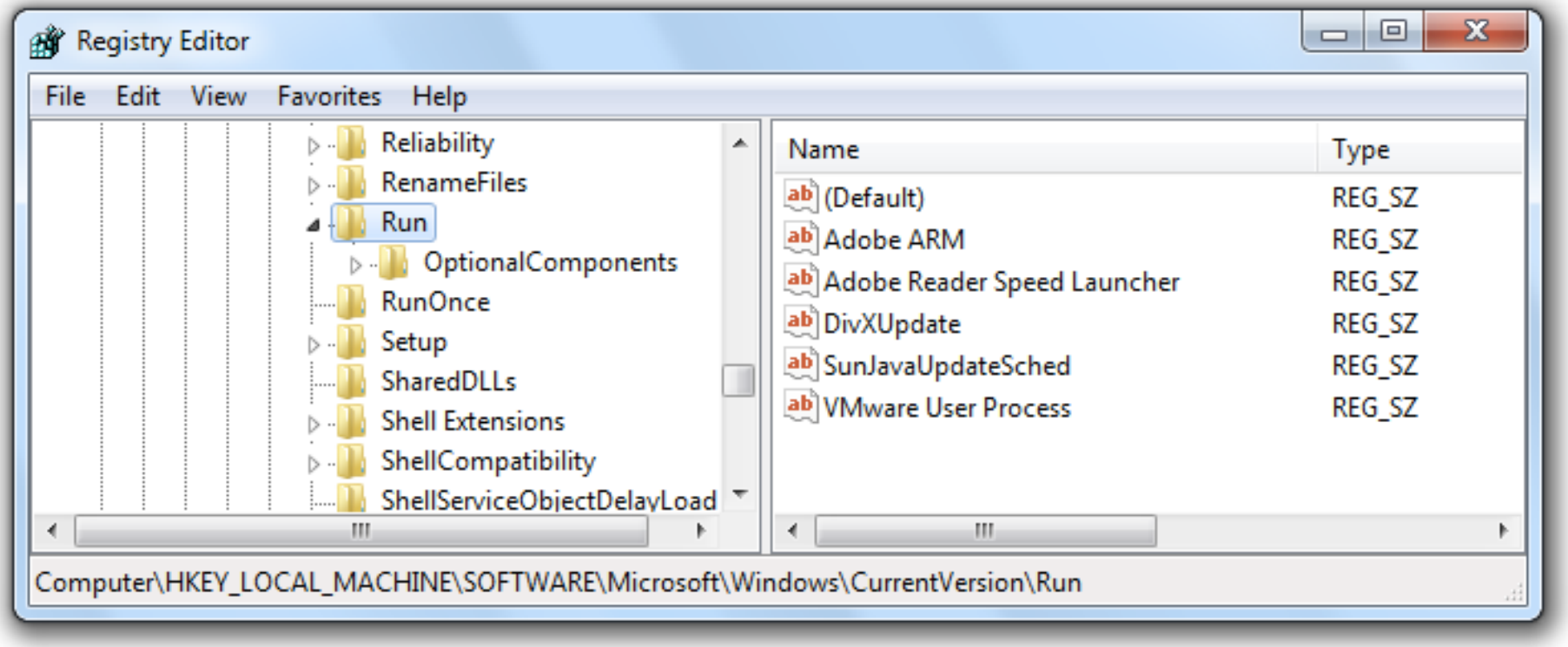
Many functions that take strings as parameters include an A or a W at the end of their names, such as `CreateDirectoryW`. This letter does *not* appear in the documentation for the function; it simply indicates that the function accepts a string parameter and that there are two different versions of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

Example

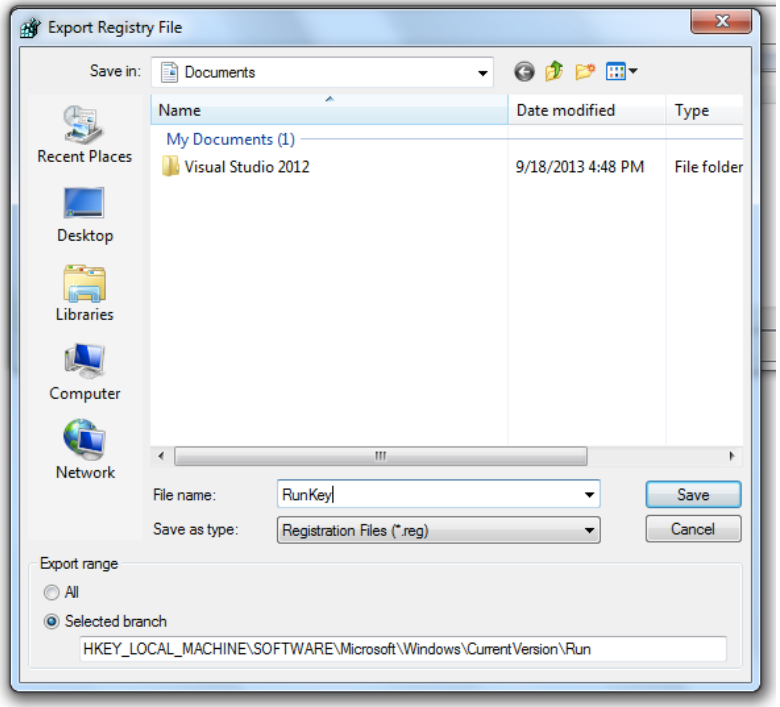
- Code that modifies registry settings

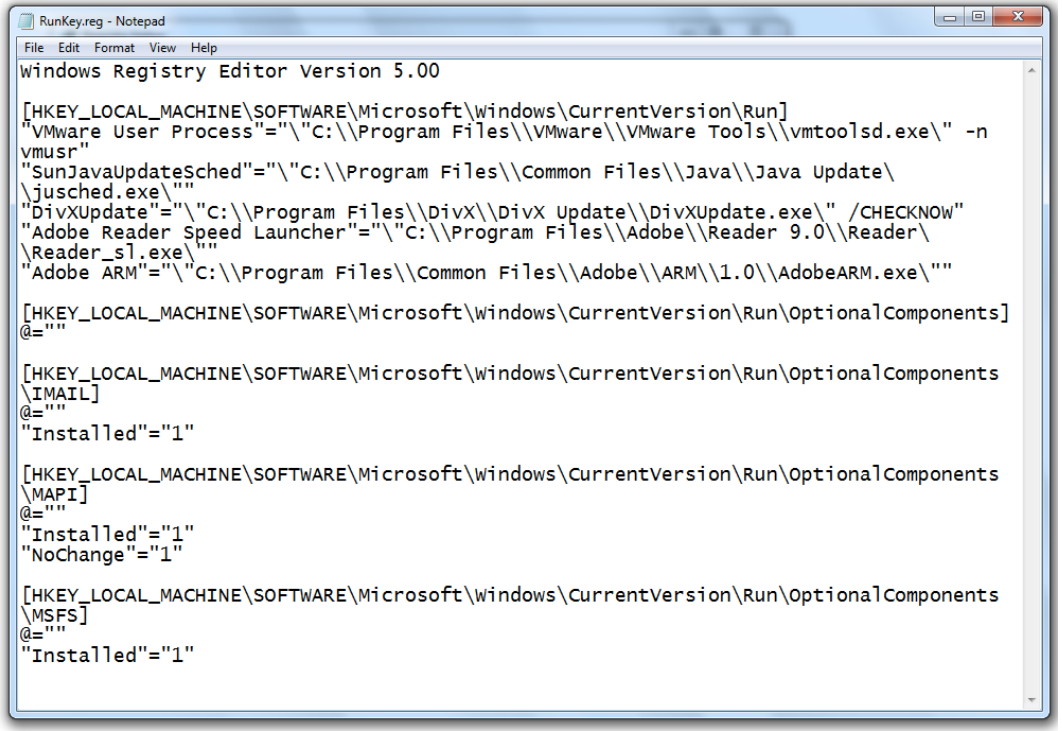
```
0040286F   push    2                ; samDesired
00402871   push    eax              ; ulOptions
00402872   push    offset SubKey    ;
"Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877   push    HKEY_LOCAL_MACHINE ; hKey
0040287C  1call    esi ; RegOpenKeyExW
0040287E   test    eax, eax
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 (cont.)





PART III:
NETWORKING APIS

Berkeley Compatible Sockets

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

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.

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

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

This pattern is common to both **malicious** and **non-malicious** programs.

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 (e.g.HTTP and FTP)
- **InternetReadFile** –reads data from a downloaded file

PART IV:
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

DYNAMIC LINK LIBRARIES (DLL)

Dynamic Link Libraries (DLLs)

- **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 <u>more exports</u> & <u>fewer imports</u>
- **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

END OF LECTURE. THANK YOU.