Malware Analysis

Analyzing Malicious Windows Program

COMPILED BY: DHARMESH DAVE | ASST. PROF. | NATIONAL FORENSIC SCIENCES UNIVERSITY

Introduction

- Non-malicious programs are generally well formed by compilers and follow Microsoft guidelines, but malware is typically poorly formed and tends to perform unexpected actions.
- Focus on the functionality most relevant to malware analysis.
- Overview of some common Windows API terminology, and then discuss the ways that malware can modify the host system.

The Windows API

- The Windows API is a broad set of functionality that governs the way that malware interacts with the Microsoft libraries. The Windows API is so extensive that developers of Windows-only applications have little need for third-party libraries.
- The Windows API uses certain terms, names, and conventions that you should become familiar with before turning to specific functions.
- Much of the Windows API uses its own names to represent C types.
- Windows generally uses Hungarian notation for API function identifiers.

The Windows API Types

Type and Prefix	Description
WORD (w)	A 16-bit unsigned value.
DWORD (dw)	A double-WORD, 32-bit unsigned value.
Handles (H)	A reference to an object. The information stored in the handle is not documented, and the handle should be manipulated only by the Windows API. Examples include HModule, HInstance, and HKey.
Long Pointer (LP)	A pointer to another type. For example, LPByte is a pointer to a byte, and LPCSTR is a pointer to a character string. Strings are usually prefixed by LP because they are actually pointers. Occasionally, you will see Pointer (P)... prefixing another type instead of LP; in 32-bit systems, this is the same as LP. The difference was meaningful in 16-bit systems.
Callback	Represents a function that will be called by the Windows API. For example, the InternetSetStatusCallback function passes a pointer to a function that is called whenever the system has an update of the Internet status.

Handles

- Handles are **items** that have been opened or created in the OS, such as a window, process, module, menu, file, and so on. Handles are like **pointers** in that they refer to an object or memory location somewhere else.
- However, unlike pointers, handles cannot be used in arithmetic operations, and they do not always represent the object's address.
- 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.
- Example: The **CreateWindowEx** function has a simple example of a handle. It returns an **HWND**, which is a handle to a window. Whenever you want to do anything with that window, such as call **DestroyWindow**, you'll need to use that handle.

File System Functions

- One of the most common ways that malware interacts with the system is by creating or modifying files, and distinct filenames or changes to existing filenames can make good host-based indicators.

- Microsoft provides several functions for accessing the file system, as follows:

  - **CreateFile**
  - **ReadFile and WriteFile**
  - **CreateFileMapping and MapViewOfFile**

File System Functions

- **CreateFile:** This function is used to create and open files. It can open existing files, pipes, streams, and I/O devices, and create new files. The parameter dwCreationDisposition controls whether the CreateFile function creates a new file or opens an existing one.

- **ReadFile and WriteFile:** These functions are used for reading and writing to files. Both operate on files as a stream. When you first call ReadFile, you read the next several bytes from a file; the next time you call it, you read the next several bytes after that.

- *For example, if you open a file and call ReadFile with a size of 40, the next time you call it, it will read beginning with the forty-first byte.*

File System Functions

- **CreateFileMapping and MapViewOfFile:** File mappings are commonly used by malware writers because they allow a file to be loaded into memory and manipulated easily. The CreateFileMapping function loads a file from disk into memory. The MapViewOfFile function returns a pointer to the base address of the mapping, which can be used to access the file in memory.
- The program calling these functions can use the pointer returned from MapViewOfFile to read and write anywhere in the file. This feature is extremely handy when parsing a file format, because you can easily jump to different memory addresses.
- *Note: File mappings are commonly used to replicate the functionality of the Windows loader. After obtaining a map of the file, the malware can parse the PE header and make all necessary changes to the file in memory, thereby causing the PE file to be executed as if it had been loaded by the OS loader.*

File System Functions

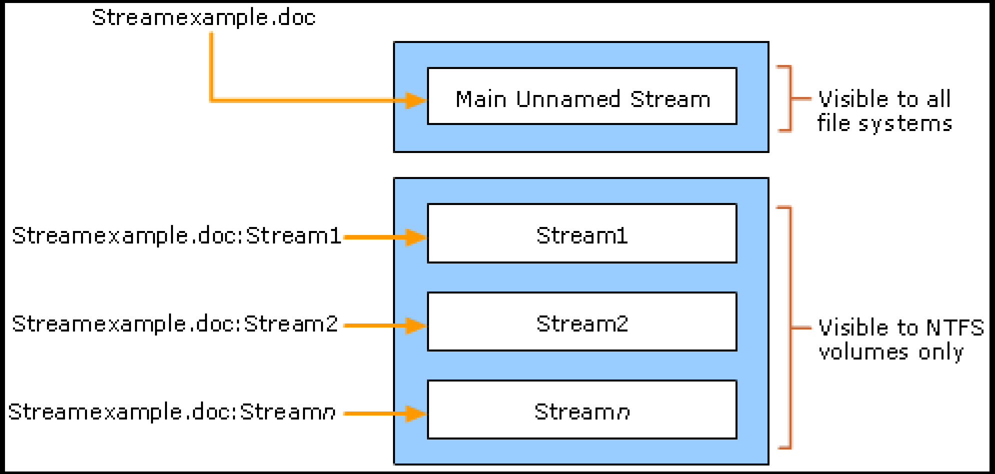
- **Special Files:** Some special files can be stealthier than regular ones because they don't show up in directory listings. Certain special files can provide greater access to system hardware and internal data.
- Special files can be passed as strings to any of the file-manipulation functions, and will operate on a file as if it were a normal file.
- **Shared Files**
- Shared files are special files with names that start with **\\serverName\share** or **\\?\serverName\share**. They access directories or files in a shared folder stored on a network. The \\?\ prefix tells the OS to disable all string parsing, and it allows access to longer filenames.

File System Functions

- **Files Accessible via Namespaces:** Namespaces can be thought of as a fixed number of folders, each storing different types of objects.
- The lowest level namespace is the NT namespace with the prefix \\. The NT namespace has access to all devices, and all other namespaces exist within the NT namespace.
- The Win32 device namespace, with the prefix \\\\.\\, is often used by malware to access physical devices directly, and read and write to them like a file.
- For example, a program might use the \\\\ .\\PhysicalDisk1 to directly access PhysicalDisk1 while ignoring its file system, thereby allowing it to modify the disk in ways that are not possible through the normal API.

File System Functions

- Alternate Data Stream (ADS): The Alternate Data Streams (ADS) feature allows **additional data to be added to an existing file within NTFS**, essentially adding one file to another.
- The extra data does not show up in a directory listing, and it is not shown when displaying the contents of the file; it's visible only when you access the stream.
- ADS data is named according to the convention **normalFile.txt:Stream:$DATA**, which allows a program to read and write to a stream.


Streamexample.doc
Main Unnamed Stream
Visible to all file systems
Streamexample.doc:Stream1
Stream1
Streamexample.doc:Stream2
Stream2
Streamexample.doc:Streamn
Streamn
Visible to NTFS volumes only

Windows Registry

- The Windows registry is used to store OS and program configuration information, such as settings and options.
- Nearly all Windows configuration information is stored in the registry, including networking, driver, startup, user account, and other information.
- Malware often uses registry functions that are part of the Windows API in order to modify the registry to run automatically when the system boots.
- **RegOpenKeyEx** Opens a registry for editing and querying. There are functions that allow you to query and edit a registry key without opening it first, but most programs use RegOpenKeyEx anyway.
- **RegSetValueEx** Adds a new value to the registry and sets its data.
- **RegGetValue** Returns the data for a value entry in the registry.

Windows Registry

```
0040286F push 2 ; samDesired
00402871 push eax ; ulOptions
00402872 push offset SubKey ; "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877 push HKEY_LOCAL_MACHINE ; hKey
0040287C call 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 call ds:lstrlenW
0040288F lea edx, [eax+eax+2]
00402893 push edx ; cbData
00402894 mov edx, [esp+428h+hKey]
00402898 lea eax, [esp+428h+Data]
0040289C push eax ; lpData
0040289D push 1 ; dwType
0040289F push 0 ; Reserved
004028A1 lea ecx, [esp+434h+ValueName]
004028A8 push ecx ; lpValueName
004028A9 push edx ; hKey
004028AA call ds:RegSetValueExW
```

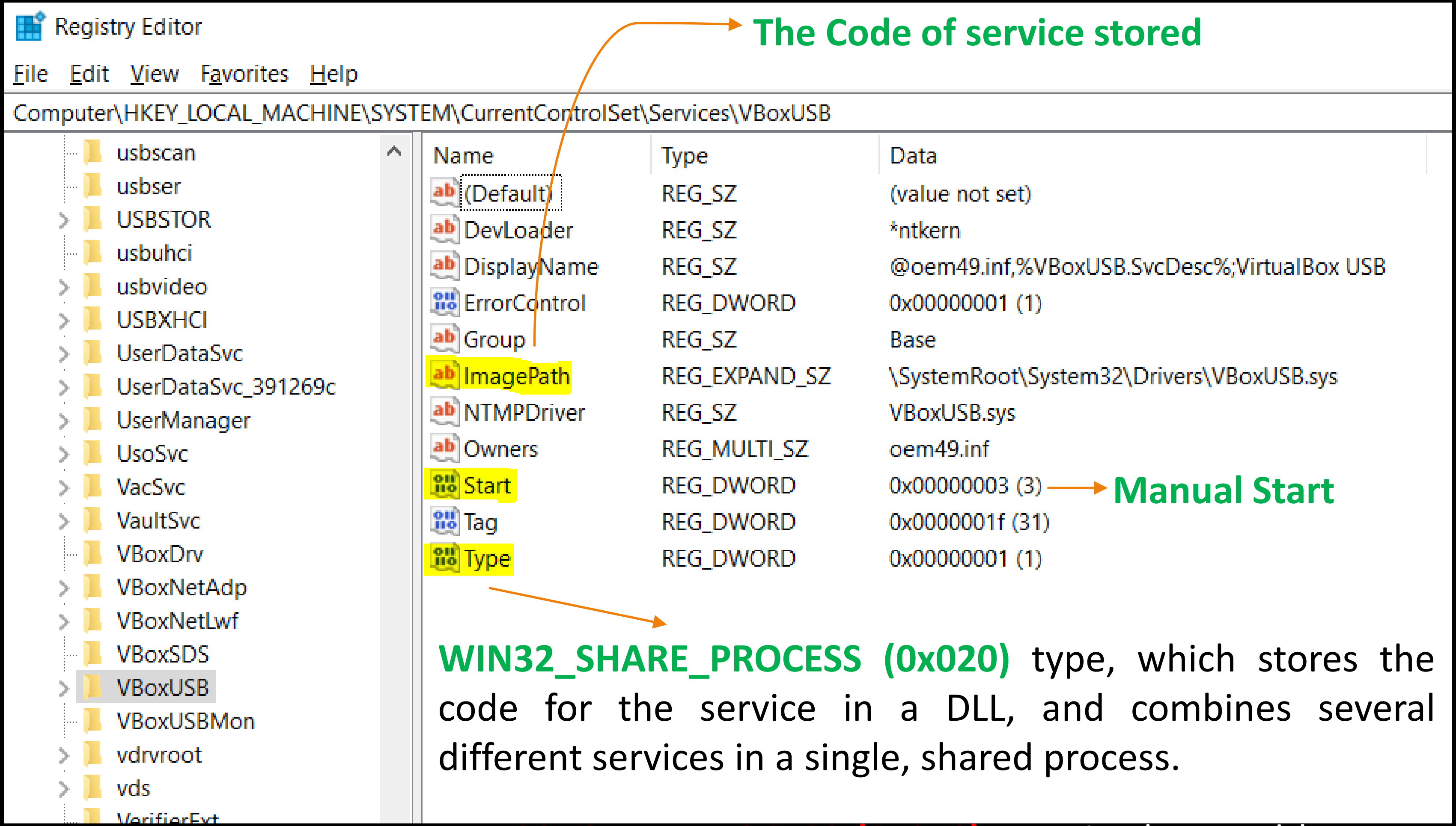
Services

- Another way for malware to execute additional code is by **installing it as a service**. Windows allows tasks to run without their own processes or threads by using services that run as background applications.
- Code is scheduled and run by the Windows service manager without user input.
- Malware Writer uses services because they are normally run as **SYSTEM** or another privileged account, and the **SYSTEM** account has more access than administrator or user accounts.
- Services also provide another way to maintain persistence on a system, because they can be set to run automatically when the OS starts, and may not even show up in the Task Manager as a process.

Services

- Services can be installed and manipulated via a few Windows API functions, which are prime targets for malware.
- **OpenSCManager** Returns a handle to the service control manager, which is used for all subsequent service-related function calls. All code that will interact with services will call this function.
- **CreateService** Adds a new service to the service control manager, and allows the caller to specify whether the service will start automatically at boot time or must be started manually.
- **StartService** Starts a service, and is used only if the service is set to be started manually.

Services


Registry Editor

File Edit View Favorites Help

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VBoxUSB

```
usbscan
usbser
USBSTOR
usbuhci
usbvideo
USBXHCI
UserDataSvc
UserDataSvc_391269c
UserManager
UsoSvc
VacSvc
VaultSvc
VBoxDrv
VBoxNetAdp
VBoxNetLwf
VBoxSDS
VBoxUSB
VBoxUSBMon
vdrvroot
vds
VerifierExt
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
DevLoader	REG_SZ	*ntkern
DisplayName	REG_SZ	@oem49.inf,%VBoxUSB.SvcDesc%;VirtualBox USB
ErrorControl	REG_DWORD	0x00000001 (1)
Group	REG_SZ	Base
ImagePath	REG_EXPAND_SZ	\SystemRoot\System32\Drivers\VBoxUSB.sys
NTMPDriver	REG_SZ	VBoxUSB.sys
Owners	REG_MULTI_SZ	oem49.inf
Start	REG_DWORD	0x00000003 (3)
Tag	REG_DWORD	0x0000001f (31)
Type	REG_DWORD	0x00000001 (1)

The Code of service stored

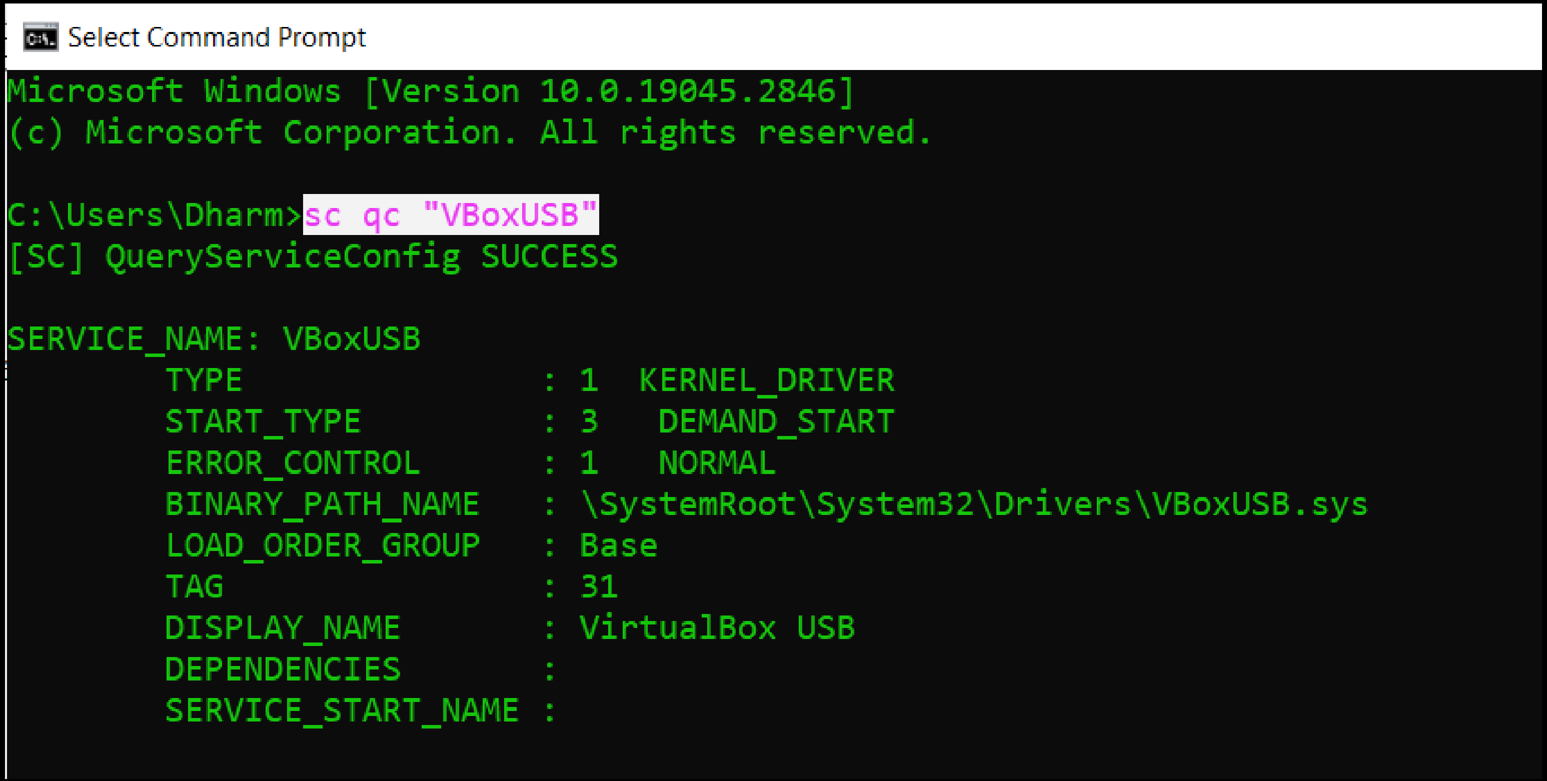
Manual Start

WIN32_SHARE_PROCESS (0x020) type, which stores the code for the service in a DLL, and combines several different services in a single, shared process.

WIN32_OWN_PROCESS (0x010) type is also used because it stores the code in an *.exe* file and runs as an independent process.

Common service type is **KERNEL_DRIVER**, which is used for loading code into the kernel.

Service Start


Select Command Prompt

```
Microsoft Windows [Version 10.0.19045.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dharm>sc qc "VBoxUSB"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: VBoxUSB
        TYPE               : 1  KERNEL_DRIVER
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : \SystemRoot\System32\Drivers\VBoxUSB.sys
        LOAD_ORDER_GROUP   : Base
        TAG                : 31
        DISPLAY_NAME       : VirtualBox USB
        DEPENDENCIES       :
        SERVICE_START_NAME :
```

Service Start

Value	Description (Start)
0	Boot: Loaded by kernel loader. Components of the driver stack for the boot (startup) volume must be loaded by the kernel loader.
1	System: Loaded by I/O subsystem. Specifies that the driver is loaded at kernel initialization.
2	Automatic: Loaded by Service Control Manager. Specifies that the service is loaded or started automatically.
3	Manual: The service does not start until the user starts it manually, such as by using Services or Devices in Control Panel.
4	Disabled: Specifies that the service should not be started.

Service Type

Value	Description (Type)
1	A kernel-mode device driver.
2	A file system driver.
4	A set of arguments for an adapter.
8	A file system driver service, such as a file system recognizer.
16 (0x10)	A Win32 program that runs in a process by itself. This type of Win32 service.can be started by the Service Controller.
32 (0x20)	A Win32 program that shares a process. This type of Win32 service can be started by the Service Controller.
272 (0x110)	A Win32 program that runs in a process by itself (like Type16) and can interact with users.
288 (0x120)	A Win32 program that shares a process and can interact with users. s

References

- Practical Malware Analysis by Michael Sikorski