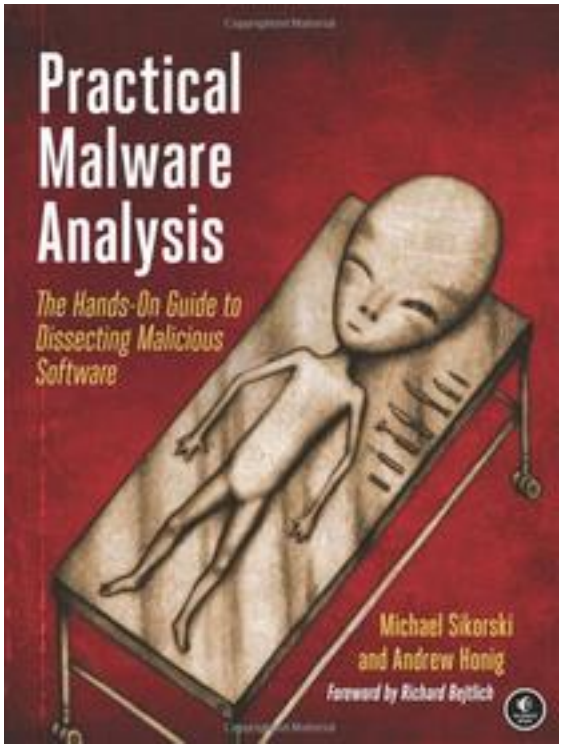
Chapter 2
Malware Behavior

Mohd Zaki Mas'ud

Practical Malware Analysis

Ch 11: Malware Behavior



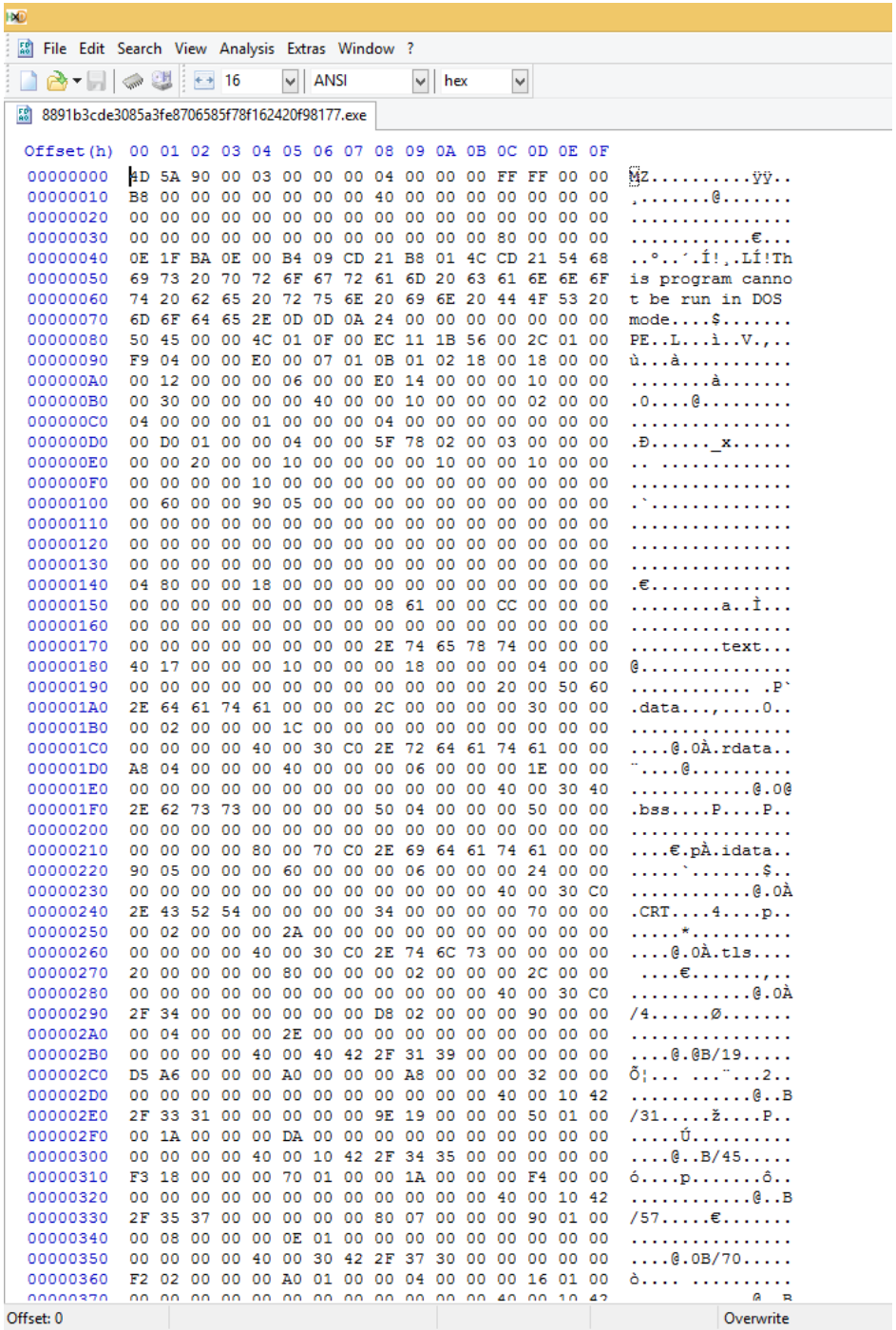
Topic

- Downloaders and Launchers
- Backdoor
- Credential Stealers
- Persistence Mechanism
- Privilege Escalation
- Network Connection (CNC)
- Covering Tracks

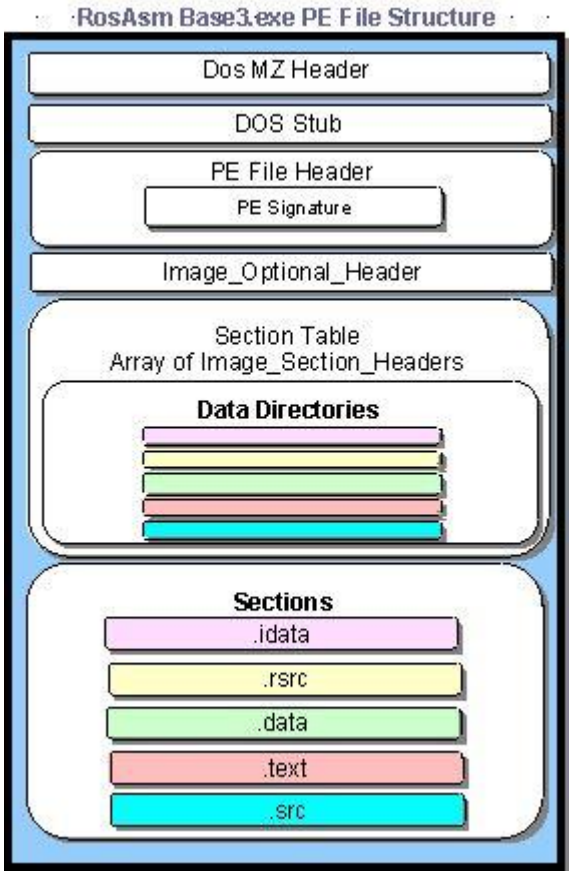
INTRODUCTION

INTRODUCTION

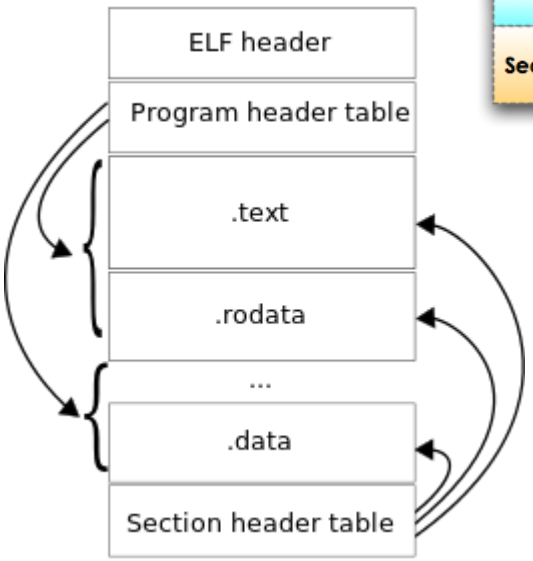
- This chapter discuss the general behavior of a malware.
- These behavior must be observed or investigated in the malware analysis
- Through the static analysis any library or class imported to the program can be use as the sign of existence of the behavior
- Through Dynamic analysis any suspicious activity can be observed through the log file.

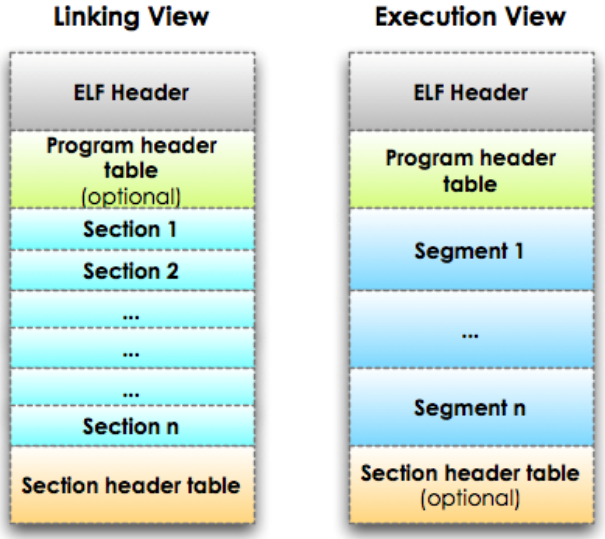

File Edit Search View Analysis Extras Window ?
16
ANSI
hex
8891b3cde3085a3fe8706585f78f162420f98177.exe
Offset (h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....ÿÿ..
00000010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ¸.......@.......
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000030 00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 ....€...
00000040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..º..´.Í!¸.LÍ!Th
00000050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$.......
00000080 50 45 00 00 4C 01 0F 00 EC 11 1B 56 00 2C 01 00 PE..L...ì..V.,..
00000090 F9 04 00 00 E0 00 07 01 0B 01 02 18 00 18 00 00 ù...à...........
000000A0 00 12 00 00 00 06 00 00 E0 14 00 00 00 10 00 00 ....à.......
000000B0 00 30 00 00 00 00 40 00 00 10 00 00 00 02 00 00 .0....@.........
000000C0 04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00 ....
000000D0 00 D0 01 00 00 04 00 00 5F 78 02 00 03 00 00 00 .Ð......_x......
000000E0 00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00 .. ..
000000F0 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ....
00000100 00 60 00 00 90 05 00 00 00 00 00 00 00 00 00 00 .`..............
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000140 04 80 00 00 18 00 00 00 00 00 00 00 00 00 00 00 .€..............
00000150 00 00 00 00 00 00 00 00 00 00 08 61 00 00 CC 00 00 00 ....a..Ì...
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000170 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 ....text...
00000180 40 17 00 00 00 10 00 00 00 18 00 00 00 04 00 00 @...............
00000190 00 00 00 00 00 00 00 00 00 00 00 00 20 00 50 60 .... .P`
000001A0 2E 64 61 74 61 00 00 00 2C 00 00 00 00 30 00 00 .data...,....0..
000001B0 00 02 00 00 00 1C 00 00 00 00 00 00 00 00 00 00 ....
000001C0 00 00 00 00 40 00 30 C0 2E 72 64 61 74 61 00 00 ....@.0À.rdata..
000001D0 A8 04 00 00 00 40 00 00 00 06 00 00 00 1E 00 00 ¨....@..........
000001E0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 40 ....@.0@
000001F0 2E 62 73 73 00 00 00 00 50 04 00 00 00 50 00 00 .bss....P....P..
00000200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000210 00 00 00 00 80 00 70 C0 2E 69 64 61 74 61 00 00 ....€.pÀ.idata..
00000220 90 05 00 00 00 60 00 00 00 06 00 00 00 24 00 00 .....`.......$..
00000230 00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 C0 ....@.0À
00000240 2E 43 52 54 00 00 00 00 34 00 00 00 00 70 00 00 .CRT....4....p..
00000250 00 02 00 00 00 2A 00 00 00 00 00 00 00 00 00 00 .....*..........
00000260 00 00 00 00 40 00 30 C0 2E 74 6C 73 00 00 00 00 ....@.0À.tls....
00000270 20 00 00 00 00 80 00 00 00 02 00 00 00 2C 00 00 ....€.......,..
00000280 00 00 00 00 00 00 00 00 00 00 00 00 40 00 30 C0 ....@.0À
00000290 2F 34 00 00 00 00 00 00 D8 02 00 00 00 90 00 00 /4......Ø.......
000002A0 00 04 00 00 00 2E 00 00 00 00 00 00 00 00 00 00 ....
000002B0 00 00 00 00 40 00 40 42 2F 31 39 00 00 00 00 00 ....@.@B/19.....
000002C0 D5 A6 00 00 00 A0 00 00 00 A8 00 00 00 32 00 00 Õ¦... ...¨...2..
000002D0 00 00 00 00 00 00 00 00 00 00 00 00 40 00 10 42 ....@..B
000002E0 2F 33 31 00 00 00 00 00 9E 19 00 00 00 50 01 00 /31.....ž....P..
000002F0 00 1A 00 00 00 DA 00 00 00 00 00 00 00 00 00 00 .....Ú..........
00000300 00 00 00 00 40 00 10 42 2F 34 35 00 00 00 00 00 ....@..B/45.....
00000310 F3 18 00 00 00 70 01 00 00 1A 00 00 00 F4 00 00 ó....p.......ô..
00000320 00 00 00 00 00 00 00 00 00 00 00 00 40 00 10 42 ....@..B
00000330 2F 35 37 00 00 00 00 00 80 07 00 00 00 90 01 00 /57.....€.......
00000340 00 08 00 00 00 0E 01 00 00 00 00 00 00 00 00 00 ....
00000350 00 00 00 00 40 00 30 42 2F 37 30 00 00 00 00 00 ....@.0B/70.....
00000360 F2 02 00 00 00 A0 01 00 00 04 00 00 00 16 01 00 ò.... ....
Offset: 0
Overwrite

Any Executable file has a file structure

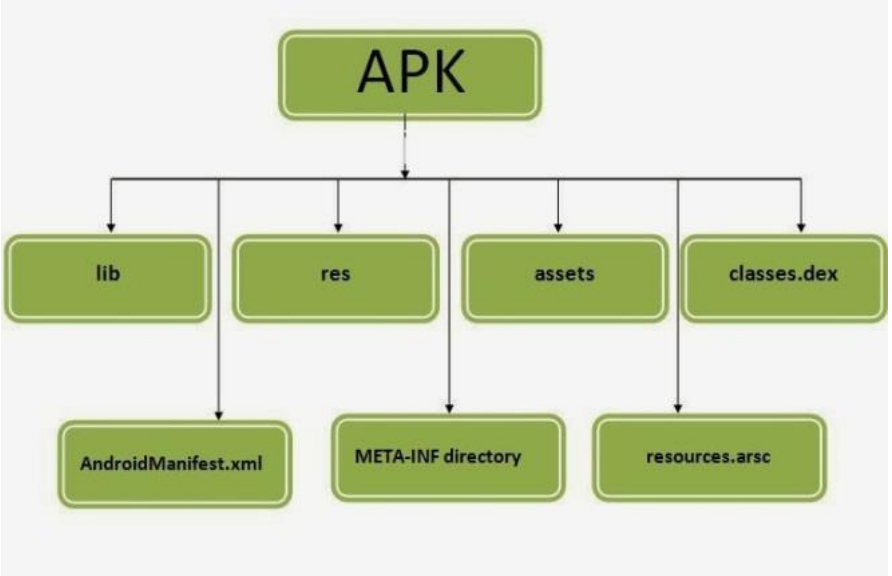


PE (win file)





ELF (linux/unix)

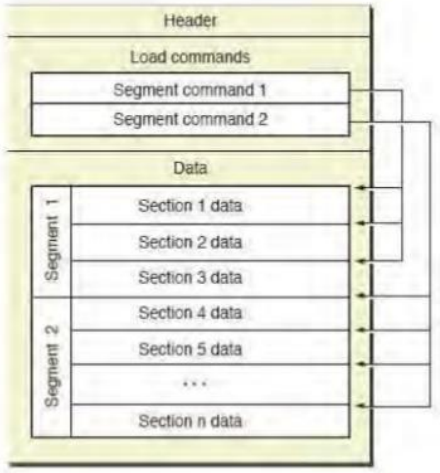


Android

iOS apps

nccgroup

- Application binaries are in the Mach-O file format
- Three parts:
  - Header
  - Load
  - Data
- Each app gets unique identifier (GUID) with corresponding home directory, inside **/var/mobile/Applications/**



Ios apps

Key Definitions

Variants : New strains of viruses that borrow code, to varying degrees, directly from other known viruses.

Source: Symantec Security Response Glossary

Family: a set of variants with a common code base.

The Malware Problem

- Malware writers use any and all techniques to evade detection.
  - Obfuscation / packing / encryption
  - Remote code updates
  - Rootkit-based hiding

- Detectors use technology from 15 years ago: signature-based detection.

Signature-Based Detection

```
lea     eax, [ebp+Data]
push    offset aServices_exe
push    eax
call    _strcat
pop     ecx
lea     eax, [ebp+Data]
pop     ecx
push    edi
push    eax
lea     eax, [ebp+ExistingFileName]
push    eax
call    ds:CopyFileA
```

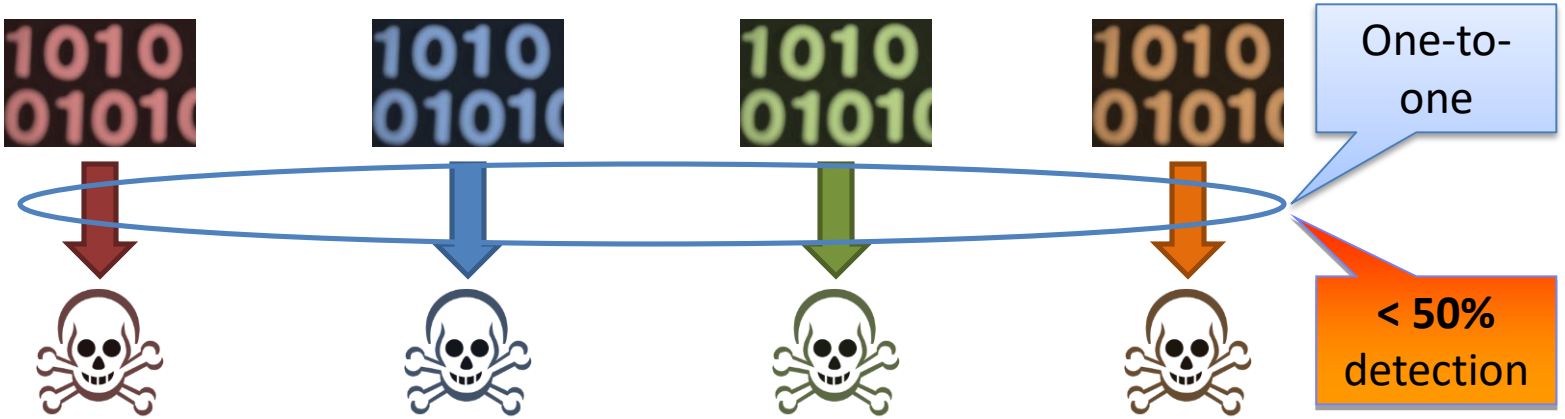
```
8D 85 D8 FE FF FF
68 78 8E 40 00
50
E8 69 06 00 00
59
8D 85 D8 FE FF FF
59
57
50
8D 85 D4 FD FF FF
50
FF 15 C0 60 40 00
```

Signature

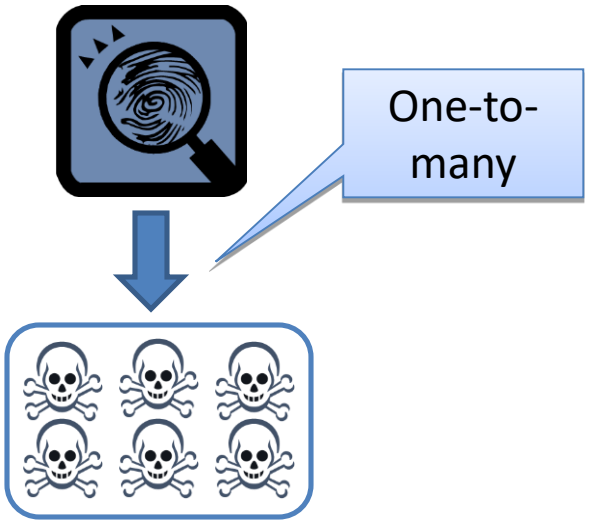
- *Signatures (aka scan-strings)* are the most common malware detection mechanism.

Behavior-Based Detection

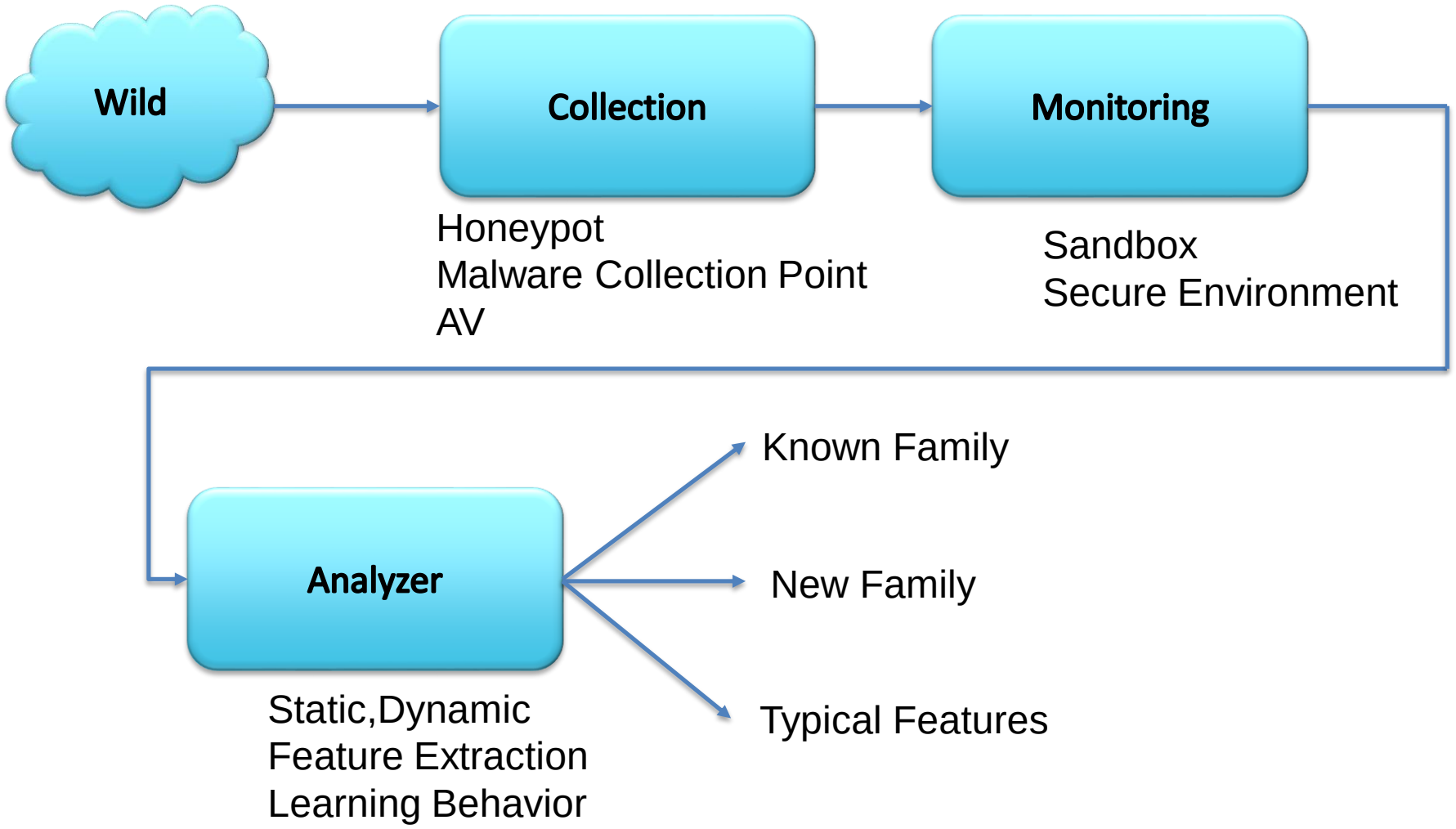
The old way – match syntactic signatures:



The new way – examine underlying behavior:



Analysing Malware Behavior Process



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

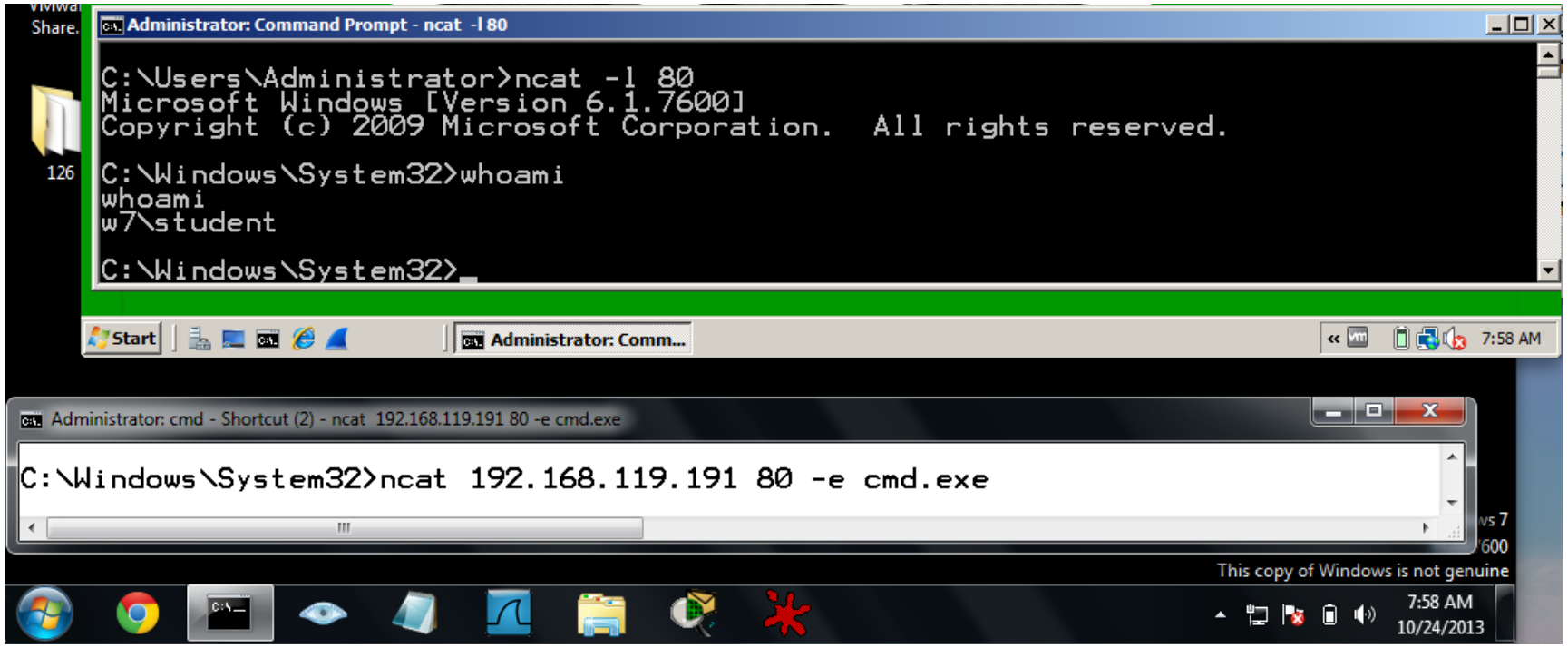
Backdoors

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



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)

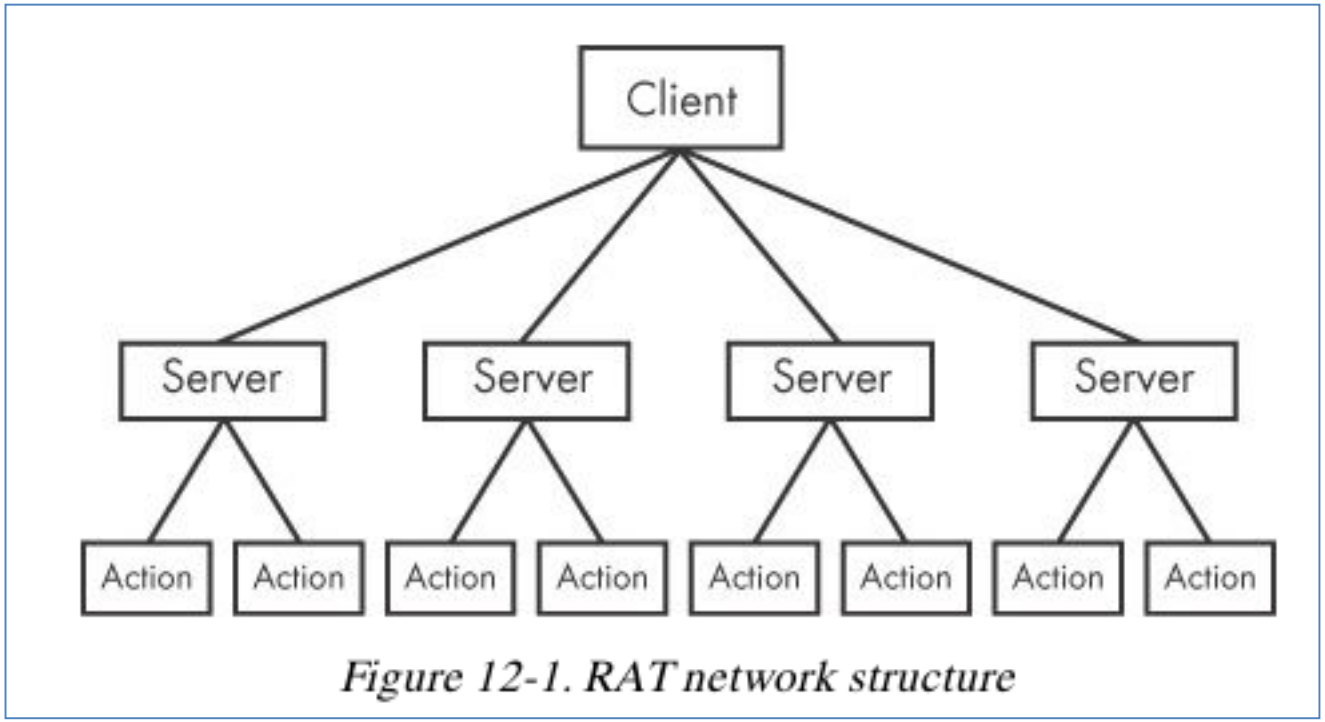


Figure 12-1. RAT network structure

- 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 the string **`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
```

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" 1
10001244        call    esi ; LoadLibraryA
10001248        push    offset aAdvapi32_dll_0  ; "advapi32.dll" 2
...
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" 3
100012D4        push    edi                     ; hModule
100012DA        call    esi ; GetProcAddress
100012DC        push    offset aSystemfuni_0  ; "SystemFunction027" 4
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 1
...
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 2
```

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
    - Polling
      - Uses **`GetAsyncKeyState`** *`(pressed or depressed)`* & **`GetForegroundWindow`** *`(foreground windows-one that is active)`* to constantly poll the state of the keys

Polling Keyloggers

- **`GetAsyncKeyState`**
  - Identifies whether a key is pressed or unpressed
- **`GetForegroundWindow`**
  - Identifies the foreground window

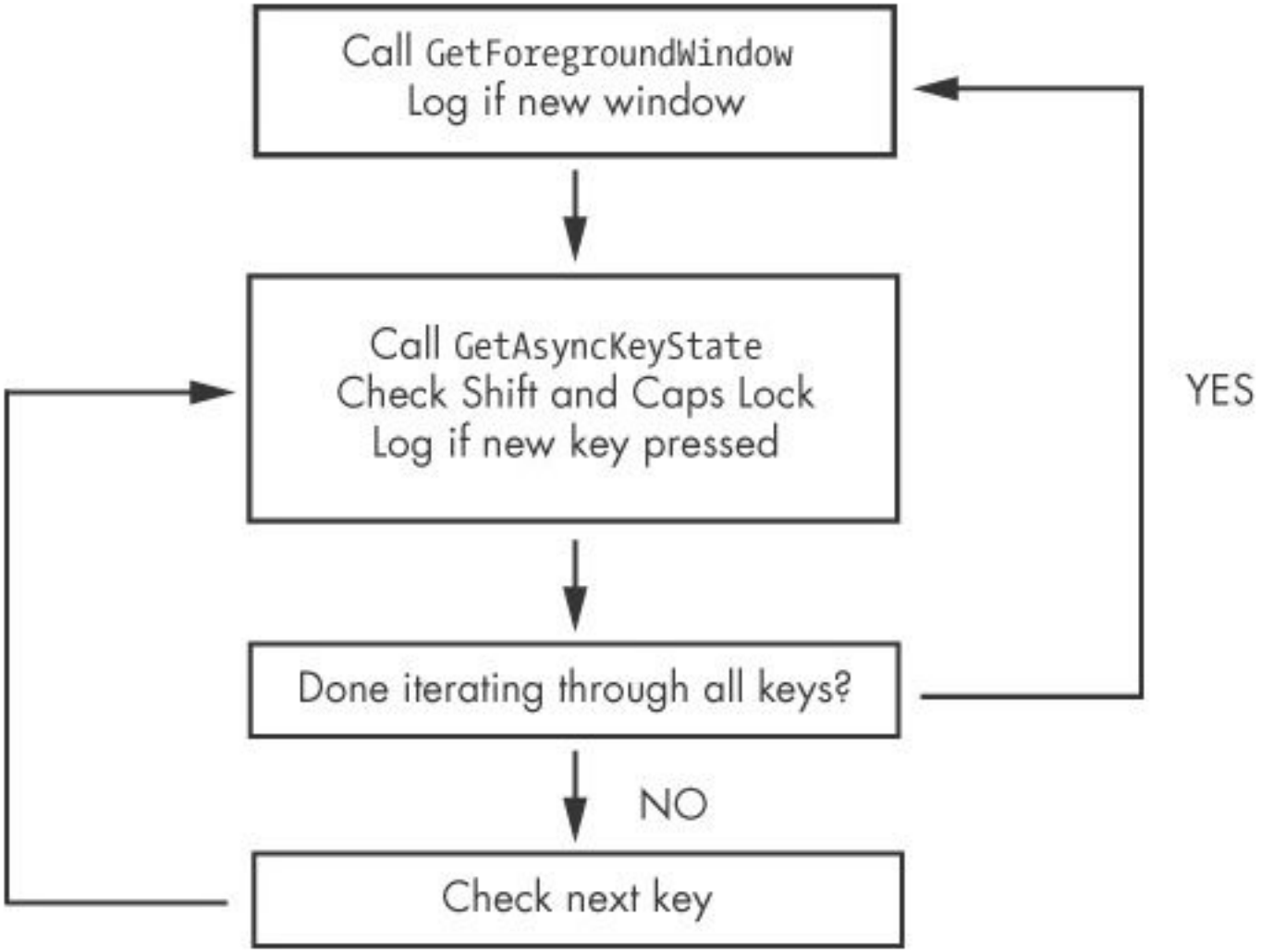


Figure 12-3. Loop structure of `GetAsyncKeyState` and `GetForegroundWindow` keylogger

Identifying Keyloggers in Strings Listings

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

Persistence Mechanisms

Three Persistence Mechanisms

- Once malware gain access it will stay for a long time.
- This is called as persistence
- If it is unique it can be used as a malware fingerprint
- Registry modifications, such as Run key
- Other important registry entries:
  - AppInit_DLLs
  - Winlogon Notify
  - ScvHost DLLs

Registry Modifications

- Run key
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Run
  - Many others, as revealed by Autoruns
- ProcMon shows registry modifications
- AppInit_DLLs

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

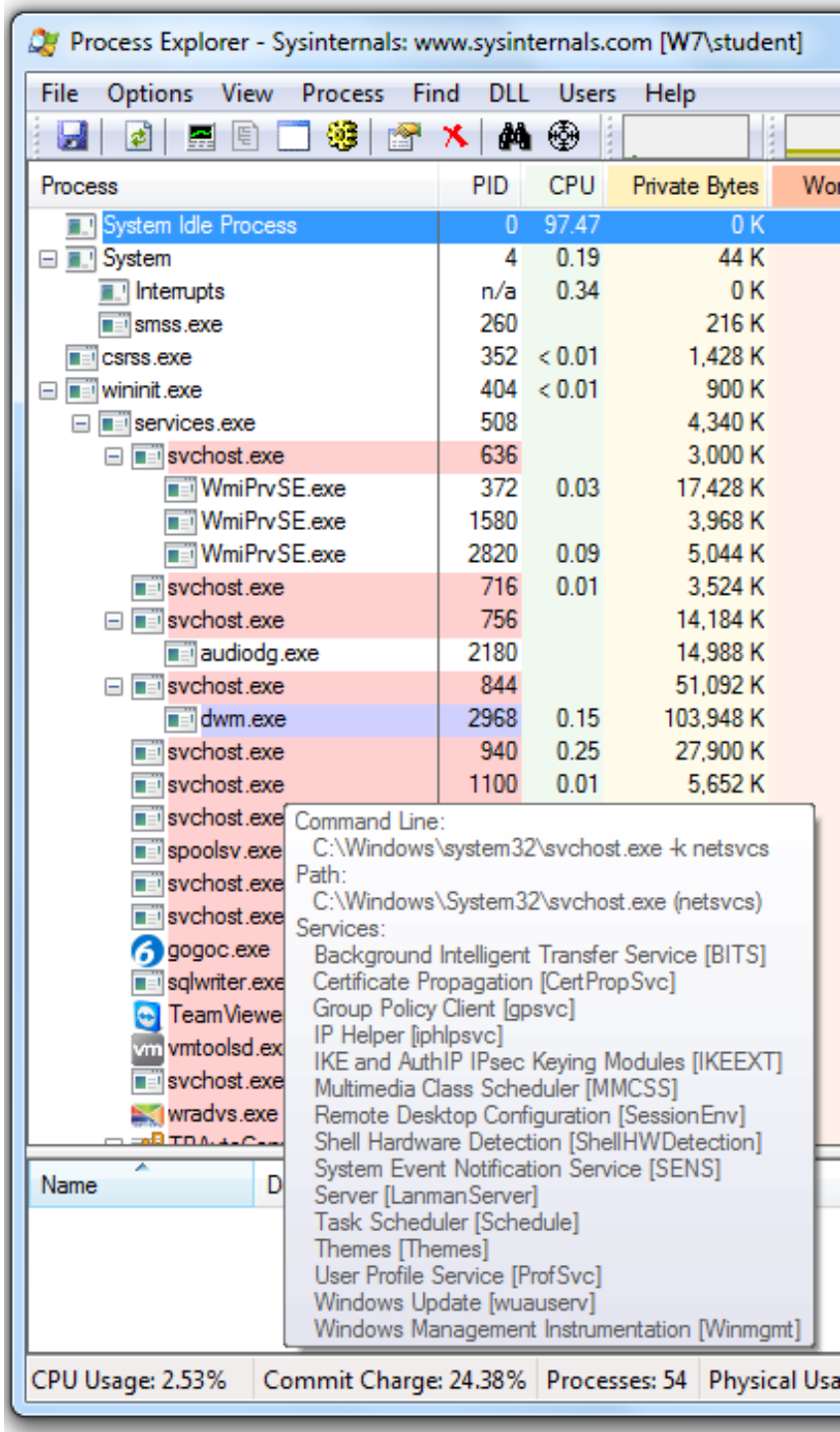
Winlogon Notify

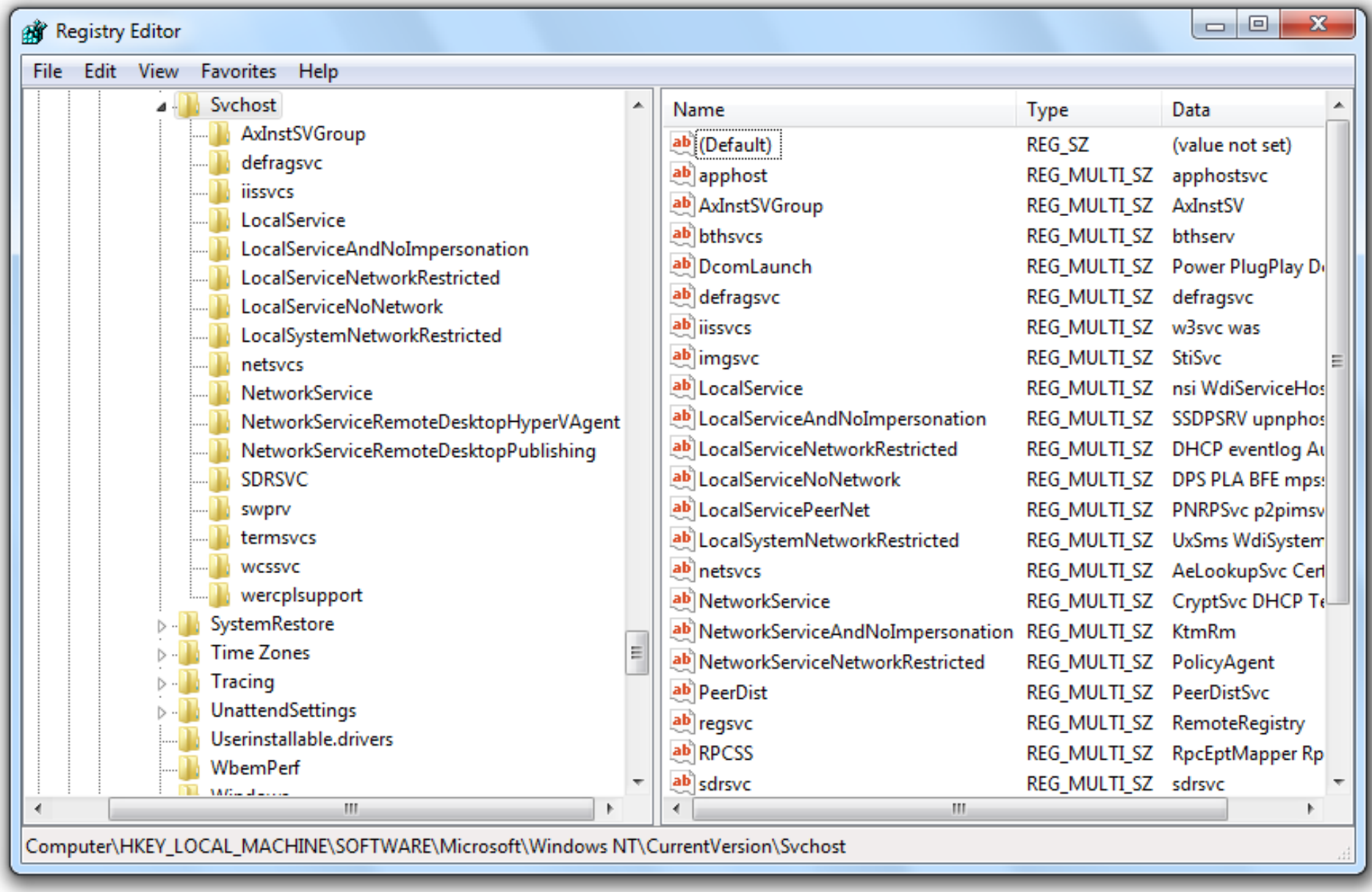
- Notify value in
  - HKEY_LOCAL_MACHINE\ SOFTWARE\ Microsoft\ Windows
  - These DLLs handle *winlogon.exe* events
  - Malware tied to an event like logon, startup, lock screen, etc.
  - It can even launch in Safe Mode

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

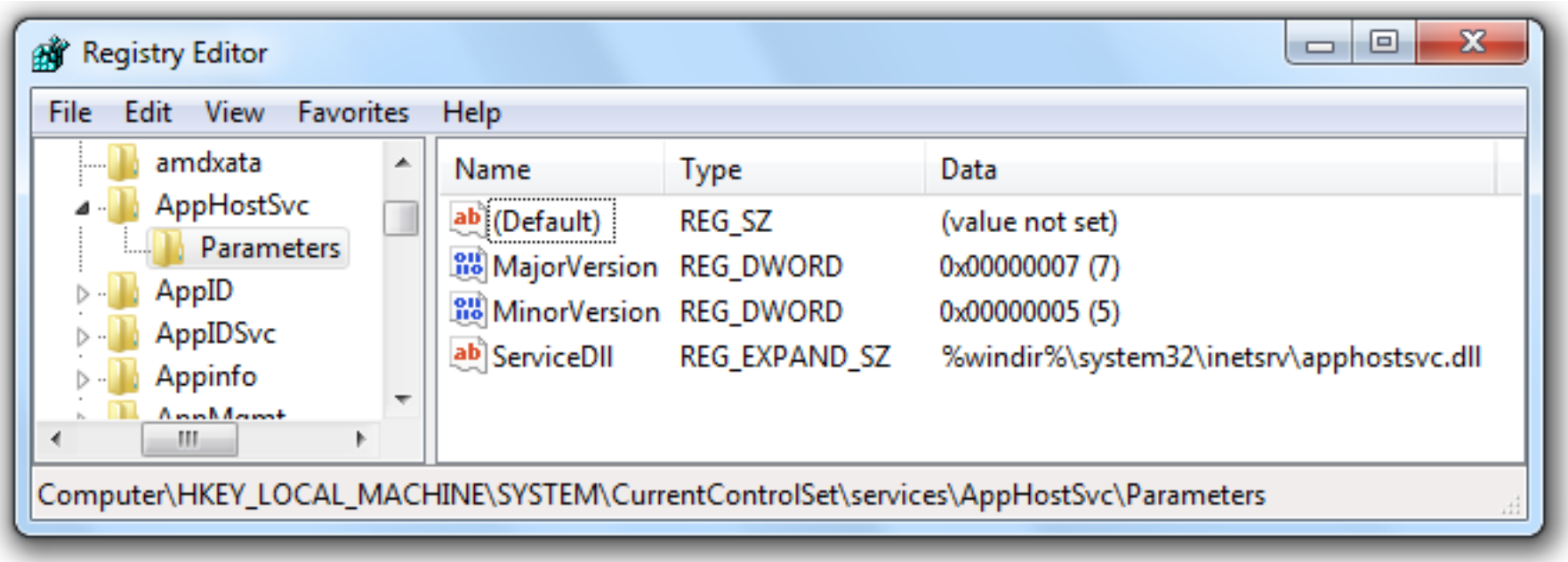
Process Explorer




Registry Editor
File Edit View Favorites Help
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 WdiServiceHos
LocalServiceAndNoImpersonation REG_MULTI_SZ SSDPSRV upnphos
LocalServiceNetworkRestricted REG_MULTI_SZ DHCP eventlog Au
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 Cert
NetworkService REG_MULTI_SZ CryptSvc DHCP Te
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 kety with a ServiceDLL value
  - Malware sets ServiceDLL to location of malicious DLL



Groups

- Malware usually adds itself to an existing group
  - Or overwrites a nonvital service
  - Often a rarelyused 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)`
 	
`mov edi, edi`
`push ebp`
`mov ebp, esp`
`push ebx`
`mov ebx, [ebp+8]`
`push esi`
`mov esi, [ebp+0Ch]`
	`DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)`
	
	`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
- 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 Known DLLs

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

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
- 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`.

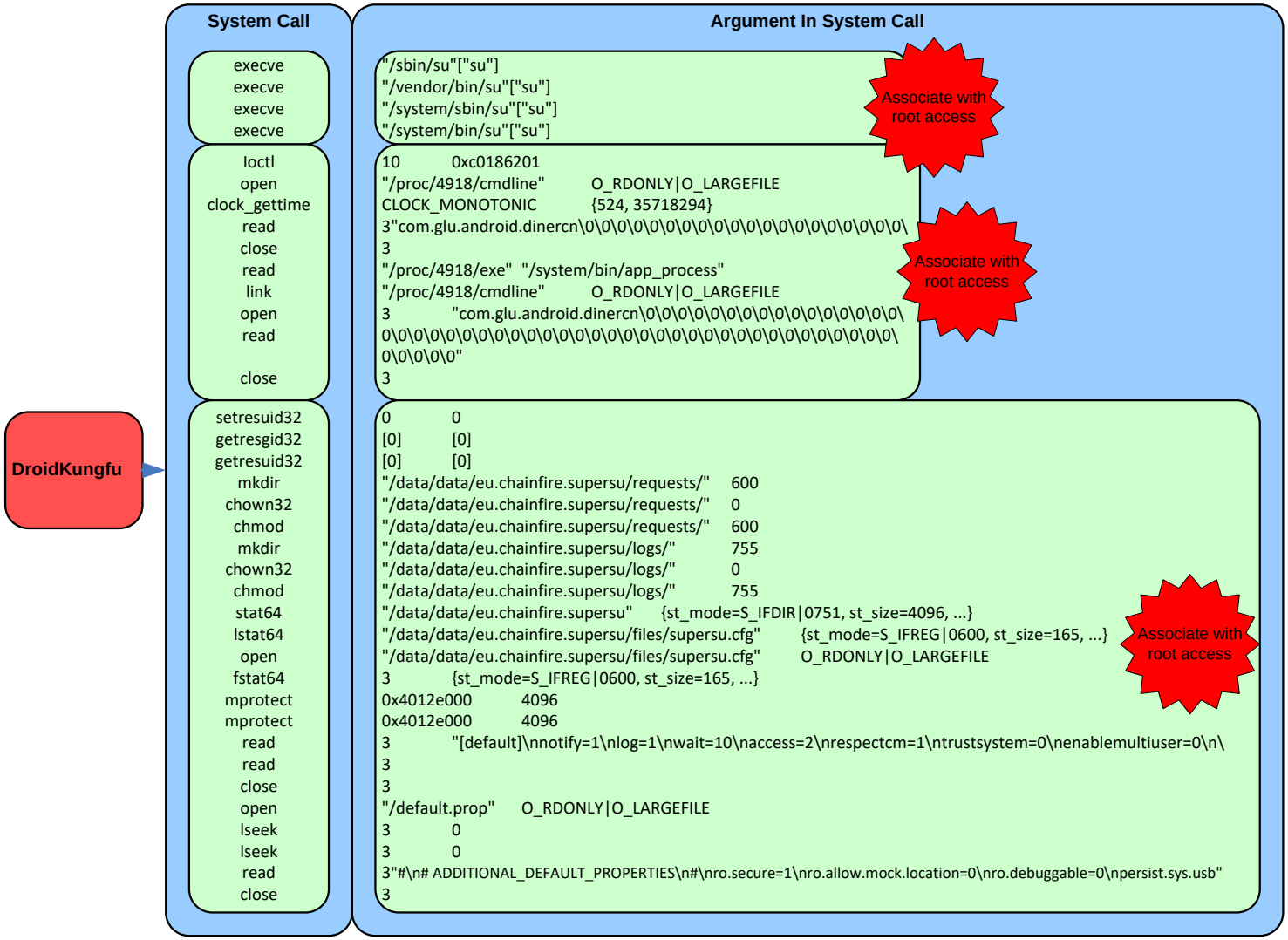
Example 12-6. Setting the access token to `SeDebugPrivilege`

```
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


DroidKungfu
System Call
execve
execve
execve
execve
Ioctl
open
clock_gettime
read
close
read
link
open
read
close
setresuid32
getresgid32
getresuid32
mkdir
chown32
chmod
mkdir
chown32
chmod
stat64
lstat64
open
fstat64
mprotect
mprotect
read
read
close
open
lseek
lseek
read
close
Argument In System Call
"/sbin/su"["su"]
"/vendor/bin/su"["su"]
"/system/sbin/su"["su"]
"/system/bin/su"["su"]
Associate with root access
10 0xc0186201
"/proc/4918/cmdline" O_RDONLY|O_LARGEFILE
CLOCK_MONOTONIC {524, 35718294}
3"com.glu.android.dinercn\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
3
"/proc/4918/exe" "/system/bin/app_process"
"/proc/4918/cmdline" O_RDONLY|O_LARGEFILE
3 "com.glu.android.dinercn\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0"
3
Associate with root access
0 0
[0] [0]
[0] [0]
"/data/data/eu.chainfire.supersu/requests/" 600
"/data/data/eu.chainfire.supersu/requests/" 0
"/data/data/eu.chainfire.supersu/requests/" 600
"/data/data/eu.chainfire.supersu/logs/" 755
"/data/data/eu.chainfire.supersu/logs/" 0
"/data/data/eu.chainfire.supersu/logs/" 755
"/data/data/eu.chainfire.supersu" {st_mode=S_IFDIR|0751, st_size=4096, ...}
"/data/data/eu.chainfire.supersu/files/supersu.cfg" {st_mode=S_IFREG|0600, st_size=165, ...}
"/data/data/eu.chainfire.supersu/files/supersu.cfg" O_RDONLY|O_LARGEFILE
3 {st_mode=S_IFREG|0600, st_size=165, ...}
0x4012e000 4096
0x4012e000 4096
3 "[default]\nnotify=1\nlog=1\nwait=10\naccess=2\nrespectcm=1\ntrustsystem=0\nenablemultiuser=0\n\
3
3
"/default.prop" O_RDONLY|O_LARGEFILE
3 0
3 0
3"#\n# ADDITIONAL_DEFAULT_PROPERTIES\n#\nro.secure=1\nro.allow.mock.location=0\nro.debuggable=0\npersist.sys.usb"
3
Associate with root access

Network Connection

- Other than monitoring the activity on the file system, and on the registry.
- Tracing network traffic to and from the system is also very interesting.
- Capturing network behavior could be useful to identify which connection has been made by the malware, which services are requested, and which data are sent or retrieved from servers.
- Malware usually interacts with external servers to gather further exploits, or gather other malware from remote sources or could signify an interaction with command and control servers

- Monitoring network activity is carried out by API Hooking in user-mode.
- Examples of traced API are:
  - DNS Query
  - Connect
  - Bind
  - Send
  - Recv

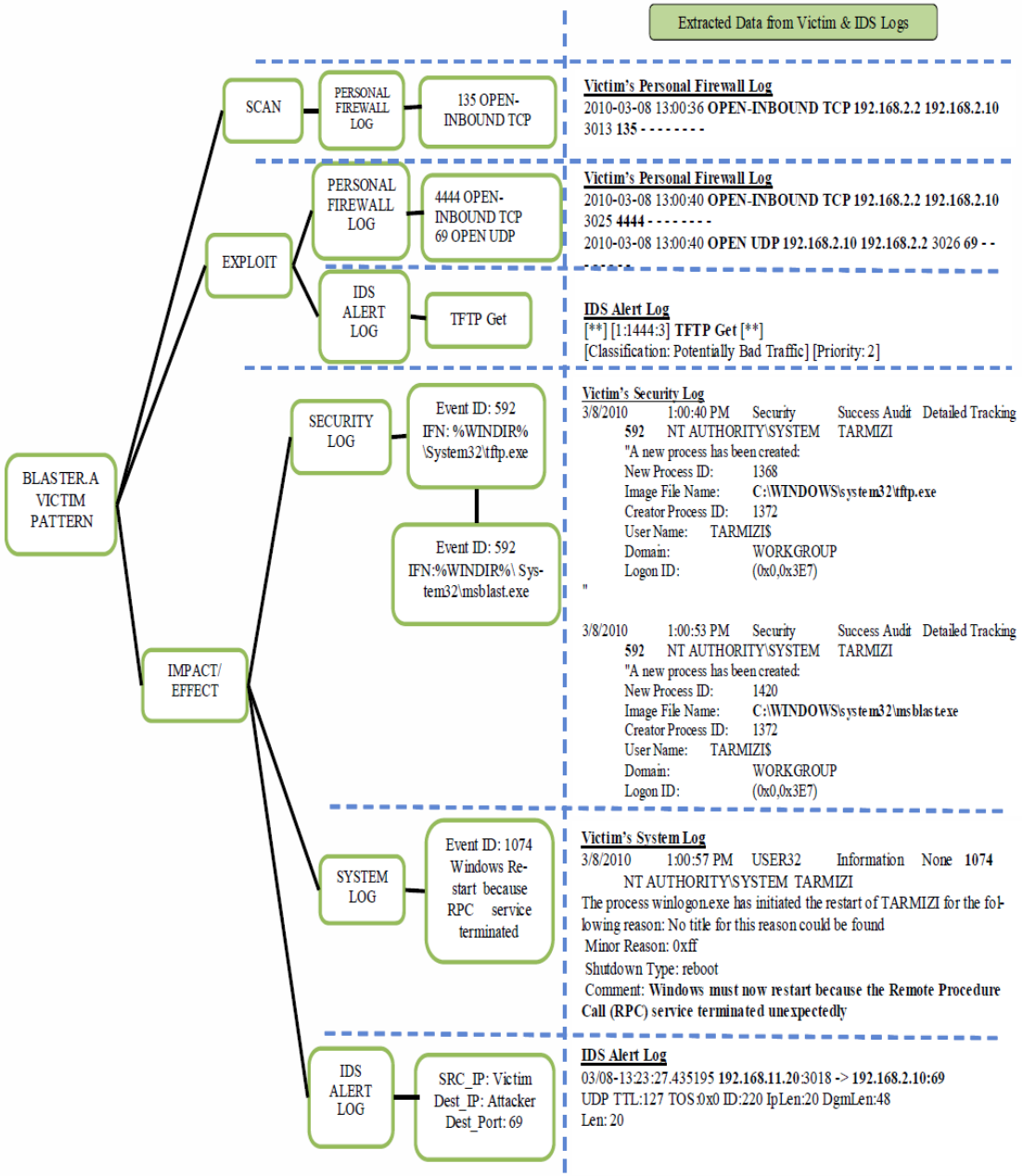
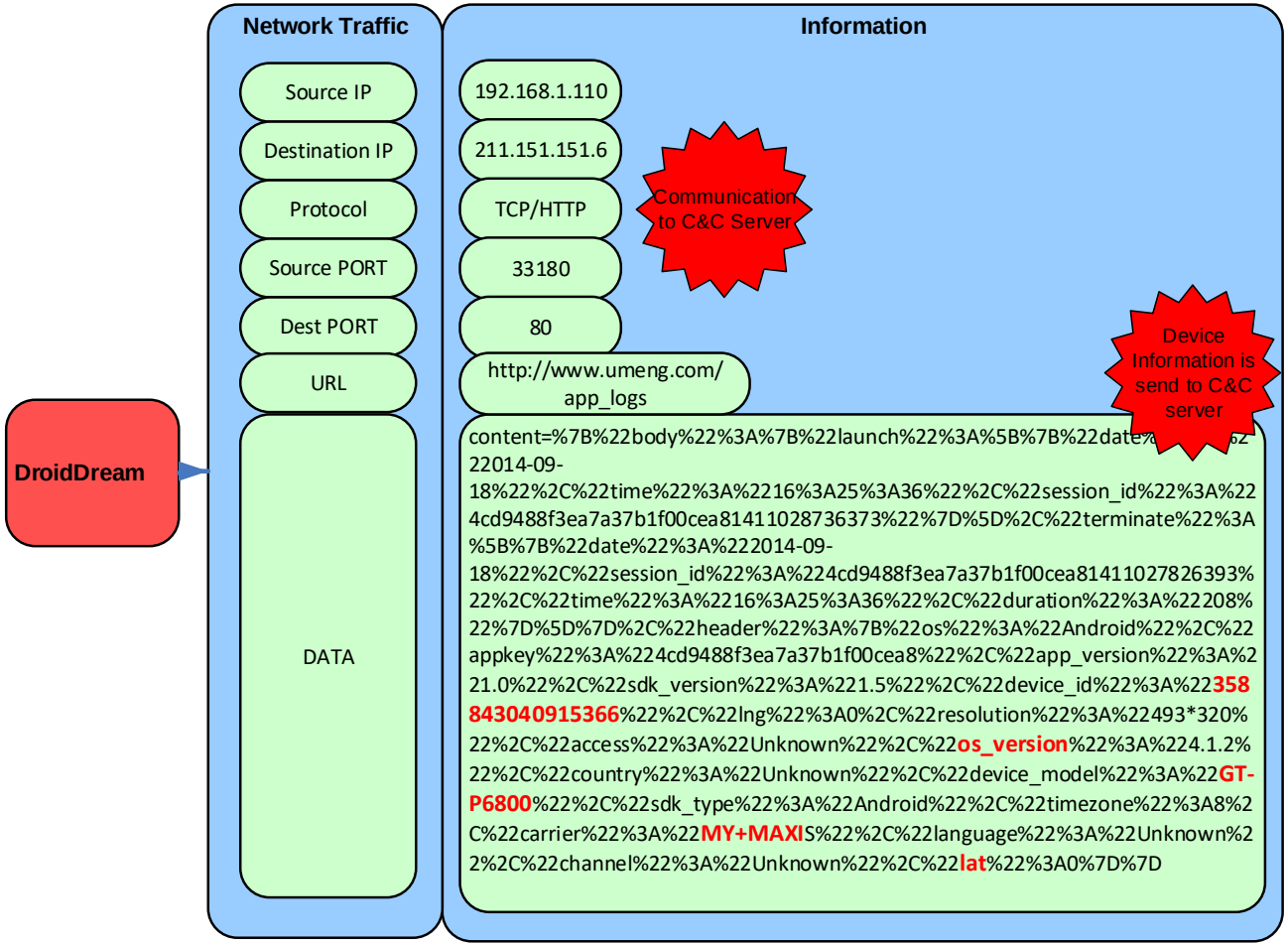
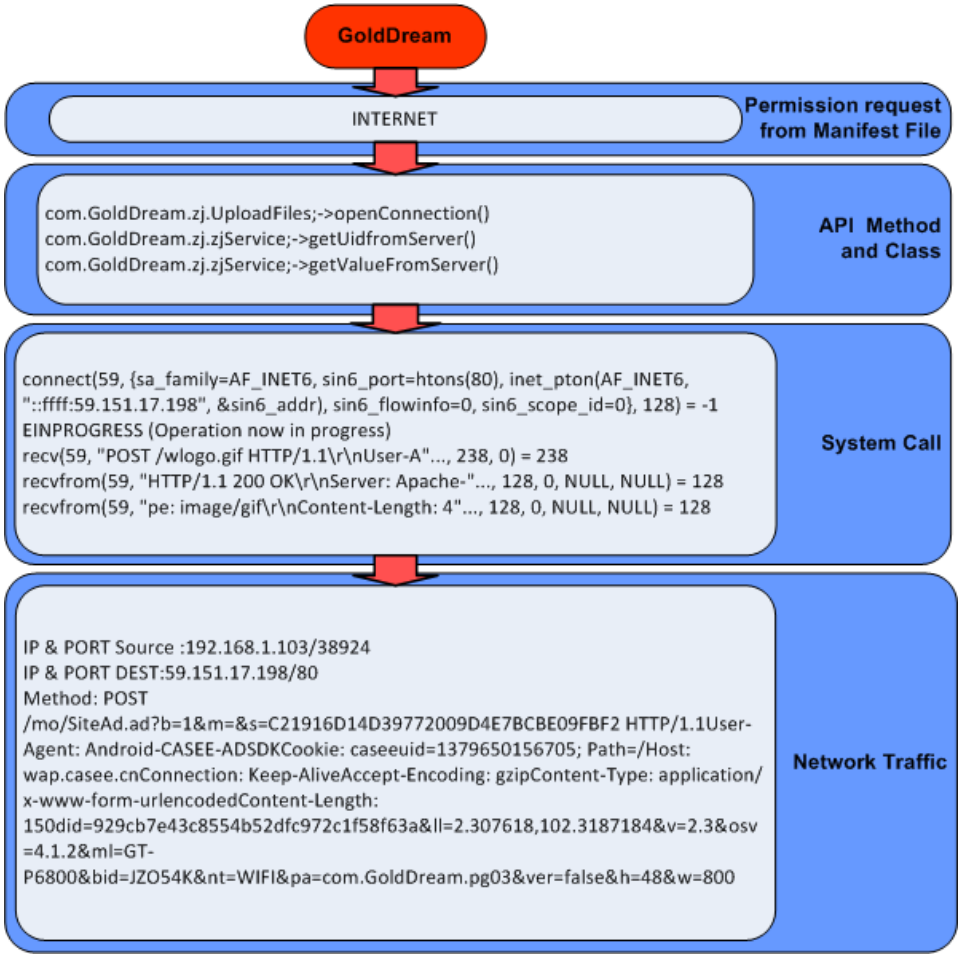


Figure 4.8 Blaster.A Victim Pattern-Scenario A

A network malware behavior Investigation




GoldDream
INTERNET
Permission request from Manifest File
com.GoldDream.zj.UploadFiles;->openConnection()
com.GoldDream.zj.zjService;->getUidfromServer()
com.GoldDream.zj.zjService;->getValueFromServer()
API Method and Class
connect(59, {sa_family=AF_INET6, sin6_port=htons(80), inet_pton(AF_INET6, "::ffff:59.151.17.198", &sin6_addr), sin6_flowinfo=0, sin6_scope_id=0}, 128) = -1 EINPROGRESS (Operation now in progress)
recv(59, "POST /wlogo.gif HTTP/1.1\r\nUser-A"..., 238, 0) = 238
recvfrom(59, "HTTP/1.1 200 OK\r\nServer: Apache-"..., 128, 0, NULL, NULL) = 128
recvfrom(59, "pe: image/gif\r\nContent-Length: 4"..., 128, 0, NULL, NULL) = 128
System Call
IP & PORT Source :192.168.1.103/38924
IP & PORT DEST:59.151.17.198/80
Method: POST
/mo/SiteAd.ad?b=1&m=&s=C21916D14D39772009D4E7BCBE09FBF2 HTTP/1.1User-Agent: Android-CASEE-ADSDKCookie: caseeuid=1379650156705; Path=/Host: wap.casee.cnConnection: Keep-AliveAccept-Encoding: gzipContent-Type: application/x-www-form-urlencodedContent-Length: 150did=929cb7e43c8554b52dfc972c1f58f63a&ll=2.307618,102.3187184&v=2.3&osv=4.1.2&ml=GT-P6800&bid=JZO54K&nt=WIFI&pa=com.GoldDream.pg03&ver=false&h=48&w=800
Network Traffic

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

Inline Hooking

- Overwrites the API function code
- Contained in the imported DLLs
- Changes actual function code, not pointers

Summary

- The general behavior of a malware is presented
- Among the behavior are:-
  - Downloaders and Launchers
  - Backdoor
  - Credential Stealers
  - Persistence Mechanism
  - Privilege Escalation
  - Network Connection
  - Covering Tracks
- These behaviour can be observed through it library call, system call or network Comm.
- This Information can be acquired from the static and dynamic analysis.