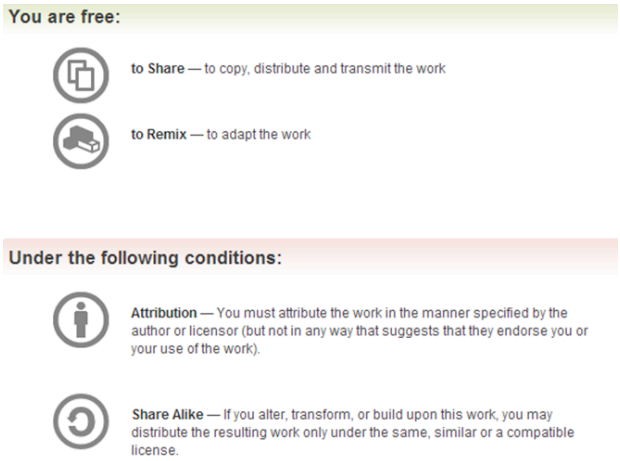
Malware Dynamic Analysis

Part 2

Veronica Kovah
vkovah.ost at gmail

All materials is licensed under a Creative Commons “Share Alike” license

http://creativecommons.org/licenses/by-sa/3.0/



Outline

- Part 1
  - Background concepts & tools
  - Observing an isolated malware analysis lab setup
  - Malware terminology
- Part 2
  - **RAT exploration - Poison IVY**
  - Persistence techniques
  - Maneuvering techniques
    (How malware strategically positions itself)

See notes for citation

Poison Ivy

- Freely available RAT, the latest version is v.2.3.2
- **Implant** (Server)
  - Customizable features: Encrypted communications, registry and file manager, screen capture, key logger, NTLM hash captures, etc.
  - No need to update for new features
  - Support 3rd party plugins
    - E.g. port scanner, wifi enumerator (“stumbler”), etc
- **Controller** (Client)
  - Once an implant is deployed, the implant connects to a controller, whose information is built into the implant.

See notes for citation

[References]

• http://www.poisonivy-rat.com/

[Image Sources]

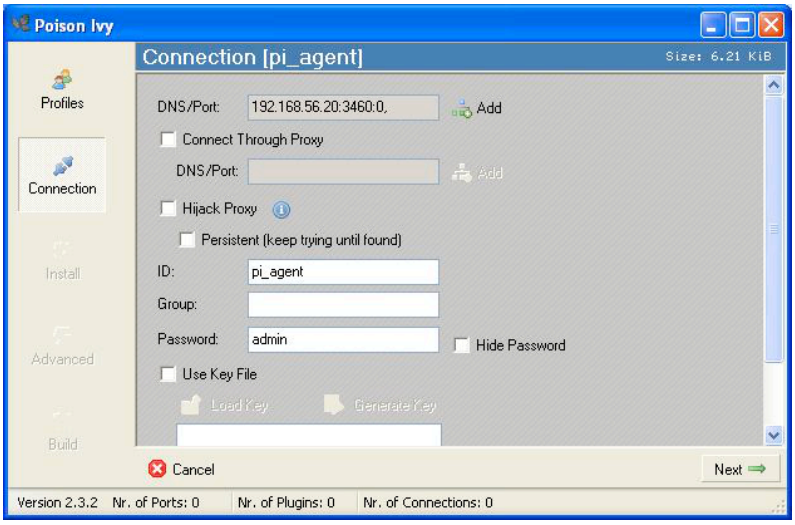
• http://25.media.tumblr.com/tumblr_m83rfveJWO1r6dcg4o1_500.jpg

Simple PI Server Creation

- On the *controller* VM
- Start Poison Ivy
  - MalwareClass/samples/PoisonIvy/Poison Ivy 2.3.2.exe
- File→New Server
- Create Profile with name “pi_agent”
- Connection: set DNS/Port to the controller VM’s IP and set port to 3460
  - 192.168.56.20:3460:0,

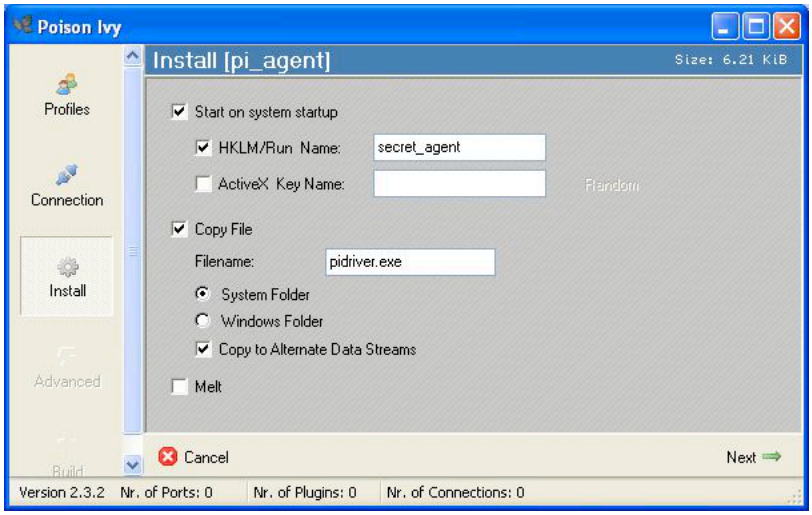
See notes for citation

Connection



See notes for citation

Install



See notes for citation

Creating pitest.exe

- Advanced: Leave as it is
- Build:
  - Click ‘Generate’ and save as “pitest.exe”
  - Then click ‘OK =>’
- We need to copy pitest.exe to the *victim* VM but will skip the step to save time

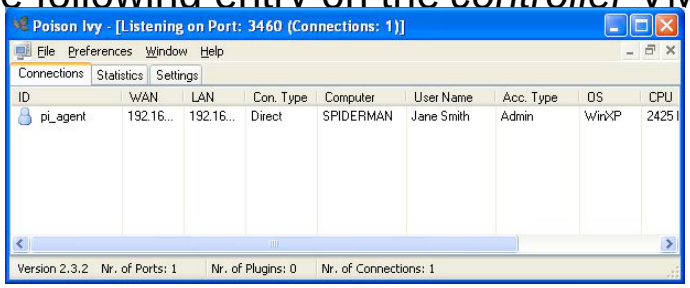
Client Creation

- On the *controller* VM
- File → New Client
- Verify ‘Listen on Port’ is set to 3460
- Click ‘Start’ button

See notes for citation

Executing Poison Ivy Implant

- On the *victim* VM
  - Execute the already prepared PI server (MalwareClass/samples/PoisonIvy/pi_agent.exe)
- Once a server connects to the client, you will see the following entry on the *controller* VM



See notes for citation

Think Evil!

- On the *controller* VM, double click on the ‘pi_agent’ line

Q1. Select ‘Remote Shell’ on the left panel, then on the right panel, click the right mouse button and select ‘Activate’, Can you start a calculator to surprise the victim? Hint: “cmd.exe /c ...”

Q2. Can you kill the calculator on the *victim* VM?

Q3. What’s in the registry value ‘secret_agent’ under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run? Anything special about it?

See notes for citation

Answers for PI Lab (1)

A1. C:\> cmd.exe /c c:\Windows\system32\calc.exe

A2. You can kill the calculator process using Managers→Processes left-side bar

See notes for citation

Answers for PI Lab (2)

A3. Alternate Data Stream (ADS) is attached to C:\WINDOWS\System32

- If you go to C:\WINDOWS\System32, you won't see anything named “pidriver.exe”. Let's find it with gmer
- Malware occasionally stores data in Alternate Data Stream (ADS). ADS is a mechanism for attaching metadata to files.
- If you use a colon in a filename, the part after the colon will be the metadata name/file, and the part before the colon will be the file it's being attached to
- Explorer doesn't show ADS files, but functions like CreateFile() can access them just fine, so the file still runs.

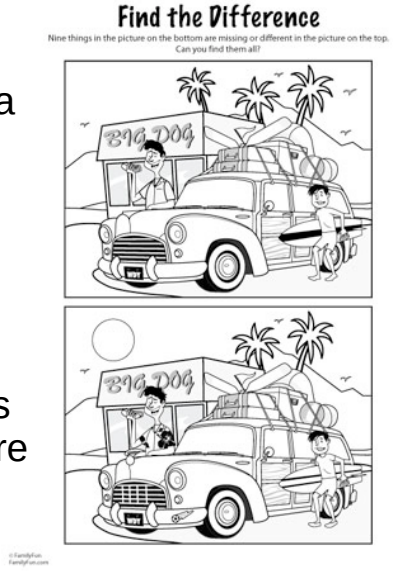
See notes for citation

Let's Start Behavioral Analysis!

See notes for citation

Diffing

- Take a snapshot of a clean system state and a snapshot of a compromised system state
- Compare before and after
- Pros: Artifacts can be observed easily
- Cons: Can miss evidence that is created during malware activities and erased purposely by malware
- Tools: regshot, autoruns



See notes for citation

[References]

• Regshot, http://code.google.com/p/regshot/
• Mark Russinovich et al., Autoruns, http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx

[Image Sources]

• http://familyfun.go.com/assets/cms/printables/0707c_findthedifference.jpg

System Monitoring

- From a clean system state, record every individual change on system and network traffic that appear after execution of made by the suspicious file
- Pro: Can collect all manifested changes
- Cons: Often too much information and need to weed out irrelevant data
- Tools: procmon, Wireshark

See notes for citation

[Image Sources]

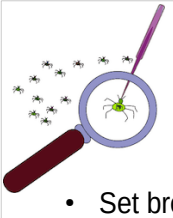
• http://i1.kym-cdn.com/entries/icons/original/000/007/195/im%20watching%20you%20-%20copia.jpg

API Tracing

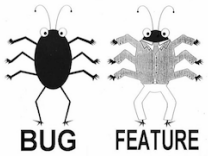
- Hook and record important API calls made by the suspicious process
- Pro: Can provide visibility into activity beyond the typical file/process/registry/network shown by other tools. Gets you a little closer to the type of interpretation that is required when doing static analysis.
- Cons: Often too much of information and need to weed out irrelevant data. API-specific interpretation can take a lot of time (but still less than static analysis ;))
- Tools: WinApiOverride, Rohitab API Monitor

[References]
• http://jacquelin.potier.free.fr/winapioverride32/
• http://www.rohitab.com/apimonitor

[Image Sources]
• Left, http://fc03.deviantart.net/fs39/f/2008/332/c/d/HAND_TURKEY_by_Bilious.jpg
• Right, http://dorpahdoo.files.wordpress.com/2010/11/foot-turkey.jpg



Debugging



- Set breakpoints inside the suspicious file to stop its execution at a given location and inspect its state. Can break when it calls to important APIs.
- Pro: Provides a superset of the functionality of an API monitor
- Cons: Typically must be be done in conjunction with some basic static analysis and assembly reading. Malware will often change its behavior or refuse to run when being debugged, which requires a work-around.
- Tools: IDA Pro Debugger, OllyDbg, Immunity Debugger, WinDbg
- We will ***NOT*** cover this in this class, because x86 assembly is not a prerequisite. See the Intro x86 and Intro Reverse Engineering classes to start working with debuggers

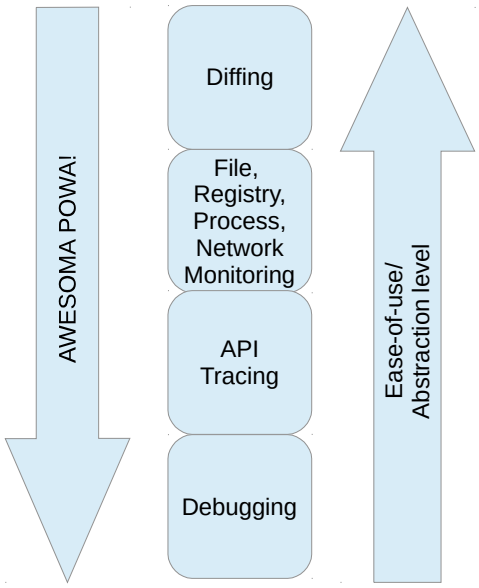
See notes for citation

[Image Sources]

• Top left, http://www.wpclipart.com/computer/humour/debugging.png
• Top right, http://www.phdcomics.com/comics/archive/phd011406s.gif
• Bottom, http://www.oraclealchemist.com/wp-content/uploads/2008/07/bug-feature.jpg

Behavioral Analysis Techniques

"Always use the easiest tool for the job" :)



See notes for citation

Outline

- Part 1
  - Background concepts & tools
  - Observing an isolated malware analysis lab setup
  - Malware terminology
- Part 2
  - RAT exploration - Poison IVY
  - **Persistence techniques**
  - Maneuvering techniques
    (How malware strategicallypositions itself)

See notes for citation

Persistence

- Techniques to survive after reboot
- Registry Key
- File System
  - Startup locations
  - DLL search order hijacking
  - Trojanizing system files
- MBR
- BIOS
- Uranium Enrichment Centrifuge PLCs :P

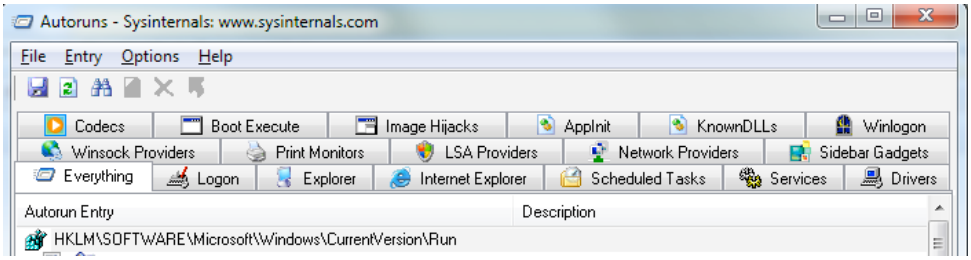
See notes for citation

[References]

• Michael Sikorski et al., Practical Malware Analysis
• Nick Harbour, https://blog.mandiant.com/archives/1207
• Nicolas Falliere et al., http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

autoruns.exe

- Provides comprehensive list of items which malware could use to be persistence



See notes for citation

[References]

Mark Russinovich et al., Autoruns, http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx

autoruns.exe

- On the *victim* VM
- Select Options → Filter Options... → Include Empty Locations, then press F5 to refresh
  - You can see all locations that autoruns.exe checks
  - Deselect the option to have cleaner view for the rest of the class
- Highlight a registry key, then double click
  - You can see the selected registry in Registry Editor
- Click the different category tabs and look around how they are grouped

See notes for citation

Frequently Used Registry Key (1)

Administrator privilege is required to update HKLM (The list is not comprehensive nor more important than others, which are not listed here)
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\”Shell” and “UserInit”
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\”Appinit_Dlls”
HKLM\System\CurrentControlSet\Control\Session Manager\KnownDlls
HKLM\System\CurrentControlSet\Services
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects

See notes for citation

Frequently Used Registry Key (2)

Without administrator privileges, malware can persist with the following registry keys (The list is not comprehensive nor more important than others, which are not listed here)
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logon
HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

See notes for citation

Observing “Image File Execution Options” registry key

- Start regedit on the *victim* VM
- Search the following registry key “HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options”
- Check if registry key *taskmgr.exe* exists
- Run procexp.exe and select Options→Replace Task Manager
- In the Registry Editor hit F5 to refresh the data
- How could malware use this to persist?

See notes for citation

Persistence Using File System

- Startup locations
  - For the logged-in user:
    %USERPROFILE%\Start Menu\Programs\Startup
  - For all users:
    %ALLUSERSPROFILE%\Start Menu\Programs\Startup
- Check the environment variables
  - C:\> set
  - To see the above two environment variables only
    - C:\> echo %USERPROFILE%
    - C:\> echo %ALLUSERSPROFILE%

How does IMworm persist?

- On the host machine, make sure inetsim is not running to observe the same results for this lab
  - $ sudo ps -ef | grep inetsim
  - $ sudo kill -9 {PID}
- Using Autoruns on the *victim* VM
  1) Start Autoruns, then File → save
  2) Run IMworm/malware.exe
  3) Press F5 to refresh Autoruns
  4) File → Compare

Q1. How does the malware persist?

- Observe what files are created in which directories
- Observe what registry keys are created/modified

See notes for citation

Answers for the IMworm Lab (1)

A1. Autoruns shows that malware persists by using the following registries and the Startup directory

- *lsass.exe* is created in c:\WINDOWS\system
- “c:\WINDOWS\system\lsass.exe” is added to HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
- “c:\WINDOWS\system\lsass.exe” is added to HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
- *msconfig.exe* is created in C:\Documents and Settings\All Users\Start Menu\Programs\Start up

See notes for citation

[References]

http://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

Answers for the IMworm Lab (2)

- lsass.exe and msconfig.exe are identical files.
- You cannot see the two files via Windows Explorer or the DOS prompt. We will have a lab to analyze how the malware hides these files
- Notice that the file names are chosen to impersonate existing MS files
  - lsass.exe: Local Security Authority Subsystem Service
  - msconfig.exe: System Configuration

Observing IMworm with Regshot

- In this lab, we will use Regshot to observe how the malware persists
- Using Regshot on the *victim* VM
  1) Start Regshot (MalwareClass/tools/v5_regshot_1.8.3...)
  2) Click *1st shot* button → Shot
  3) Run IMworm/malware.exe
  4) Click *2nd shot* button → Shot
  5) Click *Compare* button
- Compare the current results with the previous lab's results

See notes for citation

[References]

• Regshot, http://code.google.com/p/regshot/

How does Hydraq persist?

- Using Autoruns on the *victim* VM
  - Start Autoruns, then File→save
  - Run Hydraq/malware.exe
  - Press F5 to refresh Autoruns
  - File→Compare

Q1. How does the malware persist?

- Observe what files are created in which directories
- Observe what registry keys are created/modified

See notes for citation

Answers for the Hydraq lab

A1. Autoruns shows that malware persists by registering a service RaS???? (the last 4 characters are random)

- Double click the newly added RaS???? service
- ImagePath value's data is “%SystemRoot %\System32\svchost.exe -k netsvcs”
- RaS???? runs as part of *netsvcs* service group
- Parameters → ServiceDll value's data is “c:\windows\system32\rasmon.dll”
- Check if RaS???? is added to HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost\netsvcs

Observing Hydraq with Regshot (1)

- In this lab, we will use Regshot to observe how the malware persists
- Using Regshot on the *victim* VM
  1) Start Regshot (MalwareClass/tools/v5_regshot_1.8.3...)
  2) Click *1st shot* button → Shot
  3) Run Hydraq/malware.exe
  4) Click *2nd shot* button → Shot
  5) Click *Compare* button

See notes for citation

Observing Hydraq with Regshot (2)

- Compare the current results with the previous lab's results
- Note that HKLM\SYSTEM\CurrentControlSet is a pointer to HKLM\SYSTEM\ControlSet00X – check HKLM\System\Select

Outline

- Part 1
  - Background concepts & tools
  - Observing an isolated malware analysis lab setup
  - Malware terminology
- Part 2
  - RAT exploration - Poison IVY
  - Persistence techniques
  - **Maneuvering techniques (How malware strategically positions itself)**

See notes for citation

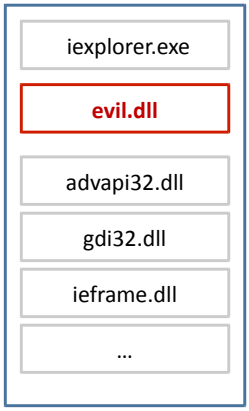
Maneuvering

- Direct code injection
- DLL injection
- DLL search order hijacking
- Asynchronous Procedure Call (APC) injection
- IAT/EAT hooking
- Inline hooking

See notes for citation

DLL/code Injection

- Load a malicious DLL into one or more processes
- Run malicious code on behalf of a legitimate process
- Bypass host-based security software
  - HIDS, Personal Firewall



IE process's memory

DLL Injection Methods (1)

- AppInit_DLLs
  - HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs is set to a space or comma-separated list of DLLs to load into processes that load user32.dll
  - On Windows Vista and newer you also have to set a few other values in that path like LoadAppInit_DLLs = 1 and RequireSignedAppInit_DLLs = 0

See notes for citation

[References]

• Michael Ligh et al., Malware Analyst's Cookbook and DVD
• AppInit_DLLs in Windows 7 and Windows Server 2008 R2, http://msdn.microsoft.com/en-us/library/windows/desktop/dd744762(v=vs.85).aspx

DLL Injection Methods (2)

- CreateRemoteThread Windows API
  - Manipulate a victim process to call LoadLibrary with the malicious DLL name
  - Malicious code is located in DllMain, which is called once a DLL is loaded into memory
  - A common API call pattern:
    - OpenProcess → VirtualAllocEx → WriteProcessMemory → GetModuleHandle → GetProcAddress → CreateRemoteThread

Refer to stand-alone DLL_Injection_APIs.pptx for details

See notes for citation

[References]

• Michael Sikorski et al., Practical Malware Analysis

DLL Injection Methods (3)

- SetWindowsHookEX Windows API
  - Monitor certain types of events (see e.g. keylogger)
  - Inject DLL into memory space of every process in the same Windows “desktop” (which is a memory organization term, not the desktop you see when looking at your computer)
    - For most intents and purposes you can think of it as injecting the DLL into every process at lesser or equal privilege
  - For the sake of simple DLL injection, use uncommon message type (e.g. WH_CBT)

See notes for citation

[References]

• Michael Sikorski et al., Practical Malware Analysis
• SetWindowsHookEx function, http://msdn.microsoft.com/en-us/library/windows/desktop/ms644990(v=vs.85).aspx

DLL Injection Methods (4)

- Codecave (*a redirection of program execution to another location and then returning back to the area where program execution had previously left.*)
  - Inject a snippet of code, which calls LoadLibrary, to a victim process
  - Suspend a thread in the victim process and restart the thread with the injected code
  - API call pattern
    - OpenProcess → VirtualAllocEx → WriteProcessMemory → SuspendThread → GetThreadContext → SetThreadContext → ResumeThread

See notes for citation

[References]

•Darawk, DLL Injection, http://www.blizzhackers.cc/viewtopic.php?p=2483118

Observing Parite's Maneuvering

- Using Regshot on the *victim* VM
  - Start Regshot (MalwareClass/tools/v5_regshot_1.8.3...)
  - Click *1st shot* button → Shot
  - Run parite/malware.exe
  - Click *2nd shot* button → Shot
  - Click *Compare* button

Q1. What is the maneuvering method?

Q2. Where is it maneuvering?

Q3. Open question: Any theories why it's maneuvering to there?

Answers for Parite Lab

A1. AppInit_DLLs is used

- “fmsiopcps.dll” is added to HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs

A2. All Windows applications, which uses user32.dll

Observing Onlinegames' Maneuvering (1)

- For this lab, we will use WinApiOverride (an API monitor) to analyze onlinegames/1/malware.exe

Q1. What is the maneuvering method?

Q2. Where is it maneuvering?

Q3. What's the path of DLL being injected?

- Take a dump of the process using Process Explorer.

Answers for Onlinegames 1 Lab

A1. Direct code injection

- OpenProcess → VirtualAllocEx → WriteProcessMemory → CreateRemoteThread

A2. Explorer.exe, OpenProcess takes PID as its parameter

A3. C:\Windows\System32\nmdfgds0.dll

- Process Explorer provides process memory dump. In order to open the dump file, use windbg's File → Open Dump menu option

See notes for citation

Observing Onlinegames' Maneuvering (2)

- Use WinApiOverride to analyze onlinegames/2/malware.exe
- Hint: new process will be invoked

Q1. What is the maneuvering method?

Q2. Where is it maneuvering to?

Q3. What's the path of the DLL being injected?

Answers for Onlinegames 2 Lab

A1. LoadLibrary call

- GetProcAddress → OpenProcess → VirtualAllocEx → WriteProcessMemory → CreateRemoteThread

A2. Explorer.exe, OpenProcess takes PID as its parameter

A3. C:\WINDOWS\system32\ailin.dll

See notes for citation

Maneuvering

- Direct code injection
- DLL injection
- DLL search order hijacking
- Asynchronous Procedure Call (APC) injection
- IAT/EAT hooking
- Inline hooking

See notes for citation

DLL Search order hijacking (1)

- (default) DLL search order in Windows XP SP3
  1. KnownDLLs and its dependent DLLs
     HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
  2. Directory from which the application loaded
  3. System directory (e.g. c:\WINDOWS\system32)
  4. 16-bit System Directory (e.g. c:\WINDOWS\system)
  5. Windows Directory
  6. Current working directory
  7. Directories in %Path%

See notes for citation

[References]

• Dynamic-Link Library Search Order (Windows), http://msdn.microsoft.com/en-us/library/windows/desktop/ms682586(v=vs.85).aspx

DLL Search order hijacking (2)

- Also an obfuscated method to be persistent
- A malware can make a legitimate looking DLL
  - Loaded by an application
  - In the directory where the application is located or the current working directory
  - Which is not listed in KnownDLLs and its dependent DLLs
  - Identically named dll as the one in system32 directory

See notes for citation

[References]

Nick Harbour, Malware Persistence without the Windows Registry, https://www.mandiant.com/blog/malware-persistence-windows-registry/

Asynchronous Procedure Call (APC) Injection

- A function executed asynchronously when a thread is in an alertable state
- A thread enters to alertable states when it calls some functions such as SleepEx, WaitForSingleObjectEx, WaitForMultipleObjectEx
- Each thread has a queue of APCs
- Kernel-mode APC is generated by the system
- User-mode APC is generated by an application
- API call pattern
  - OpenThread → QueueUserAPC
  - From kernel-space to run user-mode code: KeInitializeAPC → KeInsertQueueApc

See notes for citation

[References]

• Michael Sikorski et al., Practical Malware Analysis

Checking KnownDLLs

- Use Regedit
  - Start →Run.. →regedit
  - Search for the following registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl Set\Control\Session Manager\KnownDLLs
- Use Winobj.exe to see all dependent DLLs of KnownDLL
  - On desktop, SysinternalSuite\Winobj.exe
  - Check \KnownDlls

Observing Nitol's Maneuvering

- For this lab, we will use Process Monitor to analyze nitol/malware.exe

Q1. What is the maneuvering method?

Q2. Where is it maneuvering to?

Q3. Open question: Any theories why it's maneuvering to there?

Q4. Bonus question: How does it persist?

See notes for citation

[References]

• Microsoft Digital Crimes Unit, Operation b70, http://blogs.technet.com/cfs-file.ashx/__key/communityserver-blogs-components-weblogfiles/00-00-00-80-54/3755.Microsoft-Study-into-b70.pdf
• Rex Plantado, MSRT October '12 - Nitol: Counterfeit code isn't such a great deal after all, http://blogs.technet.com/b/mmpc/archive/2012/10/15/msrt-october-12-nitol-counterfeit-code-isn-t-such-a-great-deal-after-all.aspx

Answers for Nitol Lab

A1. DLL search order hijacking

- lpk.dll was written to multiple directories where executable files exist
  - C:\Program Files\Internet Explorer\lpk.dll
  - C:\Program Files\Messenger\lpk.dll etc.
- Compare where lpk.dll is loaded from with iexplorer.exe

A2. All executable which has lpk.dll in the same directory and uses lpk.dll

Just for fun, 基础类应用程序 means "Foundation Classes application" according Google Translation

See notes for citation

Maneuvering

- Direct code injection
- DLL injection
- DLL search order hijacking
- Asynchronous Procedure Call (APC) injection
- IAT/EAT hooking
- Inline hooking

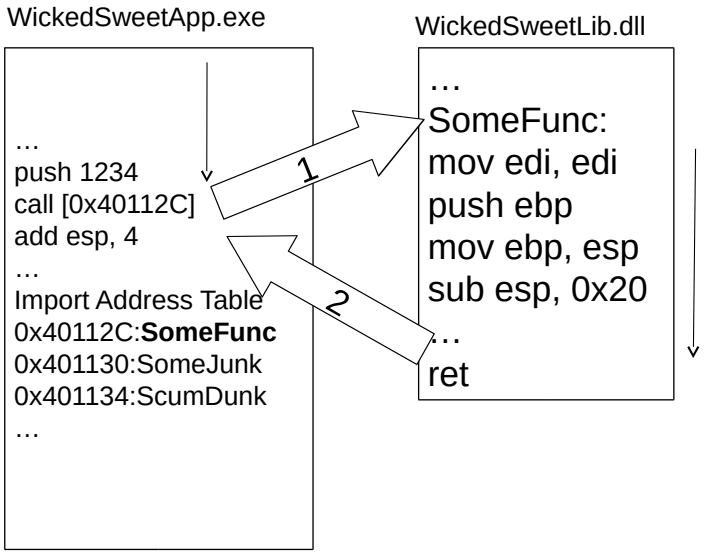
See notes for citation

IAT/EAT Hooking

- Import Address Table (IAT) holds addresses of dynamically linked library functions
- Export Address Table (EAT) holds addresses of functions a DLL allows other code to call
- Overwrite one or more IAT/EAT entries to redirect a function call to the attacker controlled code
- IAT hooking only affects a module
- EAT hooking affects all modules loaded after EAT hooking takes place
- IAT & EAT hooking only affect one process memory space

See notes for citation

Normal Inter-Module Function Call

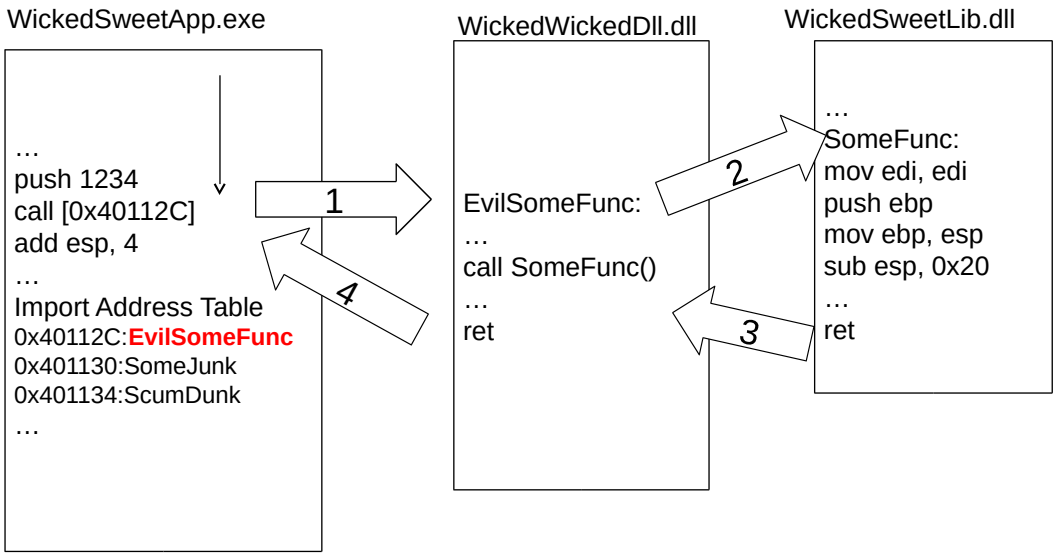


From the Rootkits class

[References]

• Xeno Kovah, Rookits: What they are, and how to find them, http://opensecuritytraining.info/Rootkits.html

IAT Hooked Inter-Module Function Call



From the Rootkits class

Inline Hooking

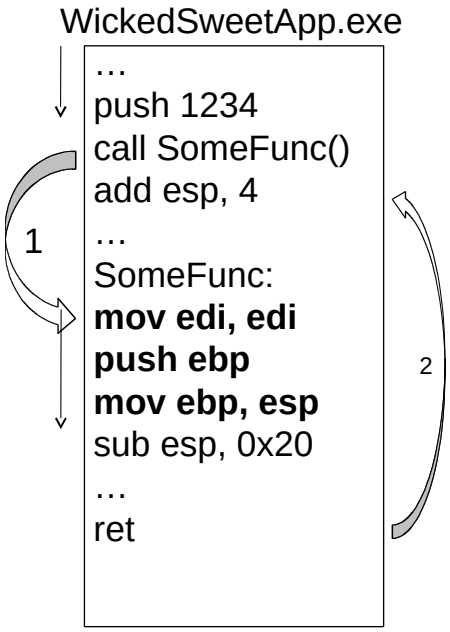
- There are a few first meaningless bytes at the beginning of a function for hooking if it is compiled with /hotpatch option
- Overwrite the first 5 or so bytes of a function with jump to the attacker's code
- This redirect the program control from the called function to the malicious code
- Execute any instructions overwritten in the first 5 bytes as the last part of the malicious code before jumping back to wherever it came from

See notes for citation

[References]

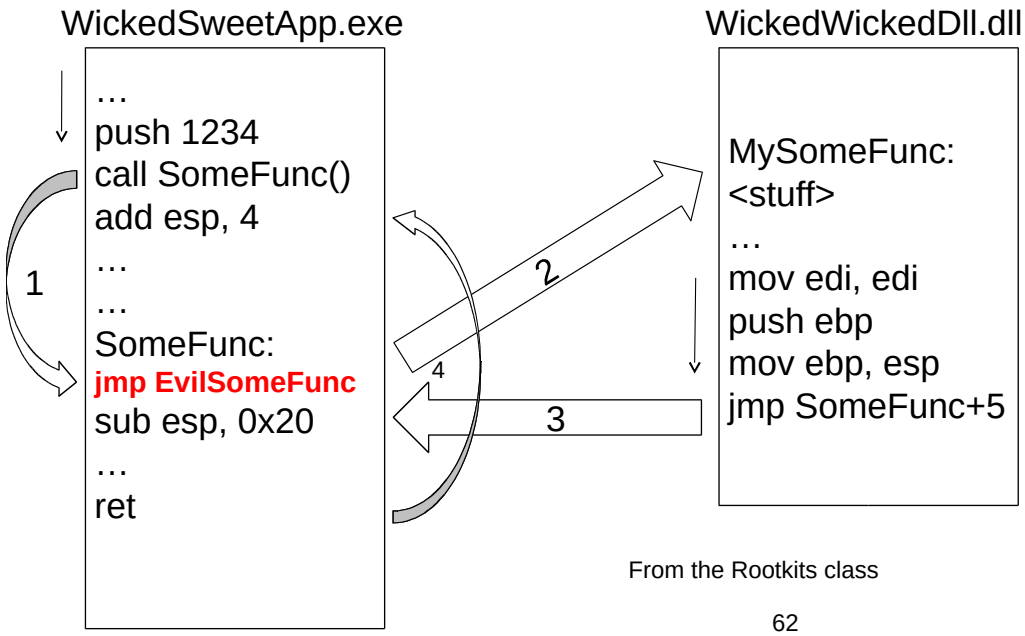
• /hotpatch (Create Hotpatchable Image), http://msdn.microsoft.com/en-us/library/ms173507.aspx
• Greg Hoglund et al., Rootkits

Normal Intra-Module Function Call



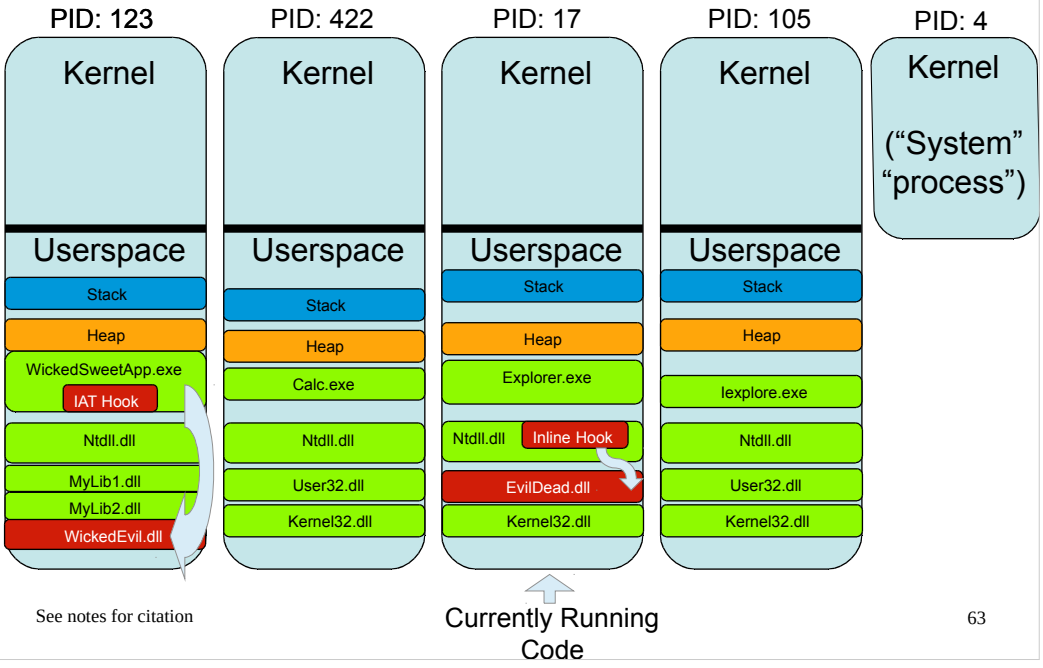
From the Rootkits class

Inline Hooked Intra-Module Function Call



From the Rootkits class

Many processes, each with their own view of memory, and the kernel schedules different ones to run at different times



See notes for citation