The Goals of Malware Analysis
Incident Response

• Case history
  – A medical clinic with 10 offices found malware on one of their workstations
  – Hired a consultant to clean & re-image that machine

• All done—case closed?
Incident Response

- After malware is found, you need to know
  - Did an attacker implant a rootkit or trojan on your systems?
  - Is the attacker really gone?
  - What did the attacker steal or add?
  - How did the attack get in
    - Root-cause analysis
Breach clean-up cost LinkedIn nearly $1 million, another $2-3 million in upgrades

**Summary:** LinkedIn executives reveal on quarterly earnings call just what the June theft of 6.5 million passwords cost the company in forensic work and on-going security updates.

By John Fontana for Identity Matters | August 3, 2012 -- 17:10 GMT (10:10 PDT)

LinkedIn spent nearly $1 million investigating and unraveling the theft of 6.5 million passwords in June and plans to spend up to $3 million more updating security on its social networking site.

- Link Ch 1a
Malware Analysis

• Dissecting malware to understand
  – How it works
  – How to identify it
  – How to defeat or eliminate it

• A critical part of incident response
The Goals of Malware Analysis

- Information required to respond to a network intrusion
  - Exactly what happened
  - Ensure you’ve located all infected machines and files
  - How to measure and contain the damage
  - Find signatures for intrusion detection systems
Signatures

- **Host-based signatures**
  - Identify files or registry keys on a victim computer that indicate an infection
  - Focus on what the malware did to the system, not the malware itself
    - Different from antivirus signature

- **Network signatures**
  - Detect malware by analyzing network traffic
  - More effective when made using malware analysis
City College Of San Francisco Computer
Lab Security Breached

January 13, 2012 1:56 PM

SAN FRANCISCO (KCBS) – The personal banking data from thousands of City College of San Francisco students, faculty and staff may be at risk because of a virus that infiltrated one computer lab – perhaps years ago.

Incredibly, the breach was only discovered recently – over the Thanksgiving holiday weekend.

KCBS’ Holly Quan Reports:

Click here to play audio
What’s most disturbing isn’t that the IP addresses identified as receiving transmissions belong to the Russian Mafia –

Sponsored Links
$28/Hr Data Entry Jobs At Home
$28/hr Part-Time Job Openi...
Malware Analysis Techniques
Static v. Dynamic Analysis

- **Static Analysis**
  - Examines malware without running it
  - Tools: VirusTotal, strings, a disassembler like IDA Pro

- **Dynamic Analysis**
  - Run the malware and monitor its effect
  - Use a virtual machine and take snapshots
  - Tools: RegShot, Process Monitor, Process Hacker, CaptureBAT
  - RAM Analysis: Mandant Redline and Volatility
Basic Analysis

• Basic static analysis
  – View malware without looking at instructions
  – Tools: VirusTotal, strings
  – Quick and easy but fails for advanced malware and can miss important behavior

• Basic dynamic analysis
  – Easy but requires a safe test environment
  – Not effective on all malware
Advanced Analysis

• Advanced static analysis
  – Reverse-engineering with a disassembler
  – Complex, requires understanding of assembly code

• Advanced Dynamic Analysis
  – Run code in a debugger
  – Examines internal state of a running malicious executable
Types of Malware
Types of Malware

• Backdoor
  – Allows attacker to control the system

• Botnet
  – All infected computers receive instructions from the same Command-and-Control (C&C) server

• Downloader
  – Malicious code that exists only to download other malicious code
  – Used when attacker first gains access
Types of Malware

• Information-stealing malware
  – Sniffers, keyloggers, password hash grabbers

• Launcher
  – Malicious program used to launch other malicious programs
  – Often uses nontraditional techniques to ensure stealth or greater access to a system

• Rootkit
  – Malware that conceals the existence of other code
  – Usually paired with a backdoor
Types of Malware

- Scareware
  - Frightens user into buying something
  - Link Ch 1b
Types of Malware

• Spam-sending malware
  – Attacker rents machine to spammers

• Worms or viruses
  – Malicious code that can copy itself and infect additional computers
Mass v. Targeted Malware

- **Mass malware**
  - Intended to infect as many machines as possible
  - Most common type

- **Targeted malware**
  - Tailored to a specific target
  - Very difficult to detect, prevent, and remove
  - Requires advanced analysis
  - Ex: Stuxnet
General Rules for Malware Analysis
General Rules for Malware Analysis

• Don’t Get Caught in Details
  – You don’t need to understand 100% of the code
  – Focus on key features

• Try Several Tools
  – If one tool fails, try another
  – Don’t get stuck on a hard issue, move along

• Malware authors are constantly raising the bar
Ch 2: Basic Static Analysis
Techniques

• Antivirus scanning
• Hashes
• A file’s strings, functions, and headers
Antivirus Scanning
Only a First Step

- Malware can easily change its signature and fool the antivirus
- VirusTotal is convenient, but using it may alert attackers that they’ve been caught

– Link Ch 2a
Hashing

A fingerprint for malware
Hashes

• MD5 or SHA-1

• Condenses a file of any size down to a fixed-length fingerprint

• Uniquely identifies a file well in practice
  – There are MD5 collisions but they are not common
  – Collision: two different files with the same hash
### HashCalc

#### Data Format:
- **File**

#### Data:
- C:\Users\student\Desktop\p3.pcap

#### Key Format:
- **Text string**

#### Key:
- MD5: 52583b5e2c99d19c046915181fd7b29b
- SHA1: 991d4e880832dd6aaebadb8040798a6b9f163194
Hash Uses

• Label a malware file
• Share the hash with other analysts to identify malware
• Search the hash online to see if someone else has already identified the file
Finding Strings
Strings

• Any sequence of printable characters is a string
• Strings are terminated by a null (0x00)
• ASCII characters are 8 bits long
  – Now called ANSI
• Unicode characters are 16 bits long
  – Microsoft calls them "wide characters"
Figure 2-2. ASCII representation of the string BAD

Figure 2-3. Unicode representation of the string BAD
The strings Command

• Native in Linux, also available for Windows
• Finds all strings in a file 3 or more characters long
The strings Command

• Bold items can be ignored
• GetLayout and SetLayout are Windows functions
• GDI32.DLL is a Dynamic Link Library

```
C:\>strings bp6.ex_
VP3
VW3
t$@
D$4
99.124.221 4
e-@
GetLayout 1
GDI32.DLL 3
SetLayout 2
M}C
Mail system DLL is invalid! Send Mail failed to send message.
```
BinText

• Link Ch 2i
Packed and Obfuscated Malware
Packing Files

• The code is compressed, like Zip file
• This makes the strings and instructions unreadable
• All you'll see is the wrapper - small code that unpacks the file when it is run
Detecting Packers with PEiD

Figure 2-5. The PEiD program
Demo: UPX

```bash
root@kali:~/126# cat chatty.c
#include <stdio.h>
main()
{
   char name[10];
   printf("This program contains readable strings\n");
   printf("Enter your name: ");
   scanf("%s", name);
   printf("Hello %s\n", name);
}

root@kali:~/126# gcc -static chatty.c -o chatty
root@kali:~/126# upx -o chatty-packed chatty
```

UPX 3.08  Markus Oberhummer, Laszlo Molnar & John Reiser  Dec 12th 2011

<table>
<thead>
<tr>
<th>File size</th>
<th>Ratio</th>
<th>Format</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>592800</td>
<td>272588</td>
<td>linux/elf386</td>
<td>chatty-packed</td>
</tr>
</tbody>
</table>

Packed 1 file.

```bash
root@kali:~/126# ls -l
total 852
-rwxr-xr-x 1 root root 592800 Aug 16 20:34 chatty
-rw-r--r-- 1 root root  174 Aug 16 20:27 chatty.c
-rwxr-xr-x 1 root root 272588 Aug 16 20:34 chatty-packed
root@kali:~/126# ```
Packing Obfuscates Strings

```
root@kali:~/126# strings chatty | wc
  1962  4498  33817
root@kali:~/126# strings chatty-packed | wc
  3950  4290  23623
root@kali:~/126#  
```
NOTE

Many PEiD plug-ins will run the malware executable without warning! (See Chapter 3 to learn how to set up a safe environment for running malware.) Also, like all programs, especially those used for malware analysis, PEiD can be subject to vulnerabilities. For example, PEiD version 0.92 contained a buffer overflow that allowed an attacker to execute arbitrary code. This would have allowed a clever malware writer to write a program to exploit the malware analyst’s machine. Be sure to use the latest version of PEiD.
Portable Executable File Format
PE Files

• Used by Windows executable files, object code, and DLLs
• A data structure that contains the information necessary for Windows to load the file
• Almost every file executed on Windows is in PE format
PE Header

- Information about the code
- Type of application
- Required library functions
- Space requirements
LordPE Demo
Main Sections

[Image of PE Editor software interface showing Basic PE Header Information and Section Table with various entries like `.text`, `.data`, `.rsrc`, `.reloc`, etc.]
There are a lot more sections

• But the main ones are enough for now
• Link Ch 2c
Linked Libraries and Functions
Imports

- Functions used by a program that are stored in a different program, such as library
- Connected to the main EXE by Linking
- Can be linked three ways
  - Statically
  - At Runtime
  - Dynamically
Static Linking

• Rarely used for Windows executables
• Common in Unix and Linux
• All code from the library is copied into the executable
• Makes executable large in size
Runtime Linking

• Unpopular in friendly programs
• Common in malware, especially packed or obfuscated malware
• Connect to libraries only when needed, not when the program starts
• Most commonly done with the LoadLibrary and GetProcAddress functions
Dynamic Linking

- Most common method
- Host OS searches for necessary libraries when the program is loaded

![Image of system error message]

The program can't start because MSVCR100D.dll is missing from your computer. Try reinstalling the program to fix this problem.
Clues in Libraries

• The PE header lists every library and function that will be loaded
• Their names can reveal what the program does
• URLDownloadToFile indicates that the program downloads something
Dependency Walker
Shows Dynamically Linked Functions

- Normal programs have a lot of DLLs
- Malware often has very few DLLs
Services.exe
Services.ex_ (malware)
Imports & Exports in Dependency Walker
<table>
<thead>
<tr>
<th>DLL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel32.dll</td>
<td>This is a very common DLL that contains core functionality, such as access</td>
</tr>
<tr>
<td></td>
<td>and manipulation of memory, files, and hardware.</td>
</tr>
<tr>
<td>Advapi32.dll</td>
<td>This DLL provides access to advanced core Windows components such as the</td>
</tr>
<tr>
<td></td>
<td>Service Manager and Registry.</td>
</tr>
<tr>
<td>User32.dll</td>
<td>This DLL contains all the user-interface components, such as buttons, scroll</td>
</tr>
<tr>
<td></td>
<td>bars, and components for controlling and responding to user actions.</td>
</tr>
<tr>
<td>Gdi32.dll</td>
<td>This DLL contains functions for displaying and manipulating graphics.</td>
</tr>
<tr>
<td>DLL</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Ntdll.dll</strong></td>
<td>This DLL is the interface to the Windows kernel. Executables generally do not import this file directly, although it is always imported indirectly by <strong>Kernel32.dll</strong>. If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs. Some tasks, such as hiding functionality or manipulating processes, will use this interface.</td>
</tr>
<tr>
<td><strong>WSock32.dll</strong> and <strong>Ws2_32.dll</strong></td>
<td>These are networking DLLs. A program that accesses either of these most likely connects to a network or performs network-related tasks.</td>
</tr>
<tr>
<td><strong>Wininet.dll</strong></td>
<td>This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.</td>
</tr>
</tbody>
</table>
Exports

• DLLs export functions
• EXEs import functions
• Both exports and imports are listed in the PE header
Notepad.exe
Advapi32.dll

<table>
<thead>
<tr>
<th>pFile</th>
<th>Data</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00037F20</td>
<td>0008606A</td>
<td>Forwarded Name RVA</td>
<td>0001 1_ScGetCurrentGroupIdW</td>
</tr>
<tr>
<td>00037F24</td>
<td>0003EC70</td>
<td>Forwarded Name RVA</td>
<td>0002 A_SHAfinal -&gt; NTDLL_A_SHAfinal</td>
</tr>
<tr>
<td>00037F28</td>
<td>0003EC81</td>
<td>Forwarded Name RVA</td>
<td>0003 A_SHAInit -&gt; NTDLL_A_SHAInit</td>
</tr>
<tr>
<td>00037F2C</td>
<td>0003EC91</td>
<td>Forwarded Name RVA</td>
<td>0004 A_SHAUpdate -&gt; NTDLL_A_SHAUpdate</td>
</tr>
<tr>
<td>00037F30</td>
<td>00081746</td>
<td>Function RVA</td>
<td>0005 AbortSystemShutdownA</td>
</tr>
<tr>
<td>00037F34</td>
<td>000816F2</td>
<td>Function RVA</td>
<td>0006 AbortSystemShutdownW</td>
</tr>
<tr>
<td>00037F38</td>
<td>000341F0</td>
<td>Function RVA</td>
<td>0007 AccessCheck</td>
</tr>
<tr>
<td>00037F3C</td>
<td>000630A1</td>
<td>Function RVA</td>
<td>0008 AccessCheckAndAuditAlarmA</td>
</tr>
<tr>
<td>00037F40</td>
<td>0011BA7</td>
<td>Function RVA</td>
<td>0009 AccessCheckAndAuditAlarmW</td>
</tr>
<tr>
<td>00037F44</td>
<td>0002B4F9</td>
<td>Function RVA</td>
<td>000A AccessCheckByType</td>
</tr>
<tr>
<td>00037F48</td>
<td>0006318B</td>
<td>Function RVA</td>
<td>000B AccessCheckByTypeAndAuditAlarmA</td>
</tr>
<tr>
<td>00037F4C</td>
<td>00062E8F</td>
<td>Function RVA</td>
<td>000C AccessCheckByTypeAndAuditAlarmW</td>
</tr>
<tr>
<td>00037F50</td>
<td>00062E21</td>
<td>Function RVA</td>
<td>000D AccessCheckByTypeResultList</td>
</tr>
<tr>
<td>00037F54</td>
<td>00063284</td>
<td>Function RVA</td>
<td>000E AccessCheckByTypeResultListAndAuditAlarmA</td>
</tr>
<tr>
<td>00037F58</td>
<td>0006337D</td>
<td>Function RVA</td>
<td>000F AccessCheckByTypeResultListAndAuditAlarmByHandleA</td>
</tr>
<tr>
<td>00037F5C</td>
<td>00062FEC</td>
<td>Function RVA</td>
<td>0010 AccessCheckByTypeResultListAndAuditAlarmByHandleW</td>
</tr>
<tr>
<td>00037F60</td>
<td>00062F3A</td>
<td>Function RVA</td>
<td>0011 AccessCheckByTypeResultListAndAuditAlarmW</td>
</tr>
<tr>
<td>00037F64</td>
<td>0002D7E0</td>
<td>Function RVA</td>
<td>0012 AddAccessAllowedAce</td>
</tr>
<tr>
<td>00037F68</td>
<td>0001B8B9</td>
<td>Function RVA</td>
<td>0013 AddAccessAllowedAceEx</td>
</tr>
<tr>
<td>00037F6C</td>
<td>00063635</td>
<td>Function RVA</td>
<td>0014 AddAccessAllowedObjectAce</td>
</tr>
<tr>
<td>00037F70</td>
<td>00009888</td>
<td>Function RVA</td>
<td>0015 AddAccessDeniedAce</td>
</tr>
<tr>
<td>00037F74</td>
<td>0006359F</td>
<td>Function RVA</td>
<td>0016 AddAccessDeniedAceEx</td>
</tr>
<tr>
<td>00037F78</td>
<td>00063683</td>
<td>Function RVA</td>
<td>0017 AddAccessDeniedObjectAce</td>
</tr>
<tr>
<td>00037F7C</td>
<td>00013C00</td>
<td>Function RVA</td>
<td>0018 AddAce</td>
</tr>
<tr>
<td>00037F80</td>
<td>00001B89</td>
<td>Function RVA</td>
<td>0019 AddAuditAccessAce</td>
</tr>
</tbody>
</table>
Example: Keylogger

- Imports User32.dll and uses the function `SetWindowsHookEx` which is a popular way keyloggers receive keyboard inputs.
- It exports `LowLevelKeyboardProc` and `LowLevelMouseProc` to send the data elsewhere.
- It uses `RegisterHotKey` to define a special keystroke like Ctrl+Shift+P to harvest the collected data.
Ex: A Packed Program

- Very few functions
- All you see is the unpacker

Table 2-3. DLLs and Functions Imported from PackedProgram.exe

<table>
<thead>
<tr>
<th>Kernel32.dll</th>
<th>User32.dll</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetModuleHandleA</td>
<td>MessageBoxA</td>
</tr>
<tr>
<td>LoadLibraryA</td>
<td></td>
</tr>
<tr>
<td>GetProcAddress</td>
<td></td>
</tr>
<tr>
<td>ExitProcess</td>
<td></td>
</tr>
<tr>
<td>VirtualAlloc</td>
<td></td>
</tr>
<tr>
<td>VirtualFree</td>
<td></td>
</tr>
</tbody>
</table>
The PE File Headers and Sections
Important PE Sections

- `.text` -- instructions for the CPU to execute
- `.rdata` -- imports & exports
- `.data` - global data
- `.rsrcre` - strings, icons, images, menus
PEView (Link Ch 2e)
Time Date Stamp

• Shows when this executable was compiled
• Older programs are more likely to be known to antivirus software
• But sometimes the date is wrong
  – All Delphi programs show June 19, 1992
  – Date can also be faked
• Virtual Size - RAM
• Size of Raw Data - DISK
• For `.text` section, normally equal, or nearly equal
• Packed executables show Virtual Size much larger than Size of Raw Data for `.text` section
Not Packed
<table>
<thead>
<tr>
<th>Name</th>
<th>Virtual size</th>
<th>Size of raw data</th>
</tr>
</thead>
<tbody>
<tr>
<td>.text</td>
<td>A000</td>
<td>0000</td>
</tr>
<tr>
<td>.data</td>
<td>3000</td>
<td>0000</td>
</tr>
<tr>
<td>.rdata</td>
<td>4000</td>
<td>0000</td>
</tr>
<tr>
<td>.rsrcc</td>
<td>19000</td>
<td>3400</td>
</tr>
<tr>
<td>Dijfpds</td>
<td>20000</td>
<td>0000</td>
</tr>
<tr>
<td>.sdfuok</td>
<td>34000</td>
<td>3313F</td>
</tr>
<tr>
<td>Kijijl</td>
<td>1000</td>
<td>0200</td>
</tr>
</tbody>
</table>
Resource Hacker

• Lets you browse the .rsrc section
• Strings, icons, and menus
• Link Ch 2f
Resource Hacker