Practical Malware Analysis

Kris Kendall and Chad McMillan
Outline

- Why Analyze Malware?
- Creating a Safe Analytical Environment
- Static Analysis Techniques
- Dynamic Analysis Techniques
- Packing
- Finding Malware
What is Malware?

Generally
- Any code that “performs evil”

Today
- Executable content with unknown functionality that is resident on a system of investigative interest
  - Viruses
  - Worms
  - Intrusion Tools
  - Spyware
  - Rootkits
Analyzing Malware

Why Analyze Malware?

• To assess damage
• To discover indicators of compromise
• To determine sophistication level of an intruder
• To identify a vulnerability
• To catch the “bad guy”®
• To answer questions…
Why Analyze Malware?

Business Questions

1. What is the purpose of the malware?
2. How did it get here?
3. Who is targeting us and how good are they?
4. How can I get rid of it?
5. What did they steal?
Why Analyze Malware?

Business Questions

6. How long has it been here?
7. Does it spread on its own?
8. How can I find it on other machines?
9. How do I prevent this from happening in the future?
## Why Analyze Malware?

### Technical Questions

1. Network Indicators?
2. Host-based Indicators?
3. Persistence Mechanism?
4. Date of Compilation?
5. Date of Installation?
Why Analyze Malware?

Technical Questions

6. What language was it written in?
7. Is it packed?
8. Was it designed to thwart analysis?
9. Does it have any rootkit functionality?
Creating a Safe Analytical Environment
Creating a Safe Environment

- Do Not Run Malware on Your Computer!
- Old And Busted
  - Shove several PCs in a room on an isolated network, create disk images, re-image a target machine to return to pristine state
- The (not so) New Hotness
  - Use virtualization to make things fast and safe
  - VMware (Workstation, Server [free])
  - Parallels (cheap)
  - Microsoft Virtual PC (free)
  - Xen (free)
VMWare Snapshot Manager
Virtualization is not the Only Option

- Use Truman (by Joe Stewart @ Lurhq) to automatically re-image physical machines
- Use a hard drive write cache card
  - CoreRestore from CoreProtect
  - Designate a portion of the hard drive as protected, all writes to the protected portion get redirected to another part of the disk
  - Reboot to restore the drive to the previous state
Reduce Risk using Platform Diversity

- If possible, perform static analysis in a different OS than your malware targets
  - Avoid the oh-$@!7 double-click
  - IDA Pro for OS X is coming soon
Creating a Safe Environment

- It is easier to perform analysis if you allow the malware to “call home”…

- However:
  - The attacker might change his behavior
  - By allowing malware to connect to a controlling server, you may be entering a real-time battle with an actual human for control of your analysis (virtual) machine
  - Your IP might become the target for additional attacks (consider using TOR)
  - You may end up attacking other people
Creating a Safe Environment

- Therefore, we usually do not allow malware to touch the real network
  - Use the host-only networking feature of your virtualization platform
  - Establish real services (DNS, Web, etc) on your host OS or other virtual machines
  - Use netcat to create listening ports and interact with text-based client
  - Build custom controlling servers as required (usually in a high-level scripting language)
Virtualization Considerations

- Using a Virtual Machine helps, but…
- Set up the “victim” with no network or host-only networking
- Your virtualization software is not perfect
- Malicious code can detect that it is running in a virtual machine
- A 0-day worm that can exploit a listening service on your host OS will escape the sandbox
  - Even if you are using host-only networking!
Performing Malware Analysis on Windows
Static vs. Dynamic Analysis

- **Static Analysis**
  - Code is Not Executed
  - Autopsy or Dissection of “Dead” Code

- **Dynamic Analysis**
  - Observing and Controlling Running (“live”) Code
  - Ant Farm

- The Fastest Path to the Best Answers Will Usually Involve a Combination of Both.
Static Analysis “the dissection”
Static Analysis

Static Analysis is Safer

• Since we aren’t actually running malicious code, we don’t have to worry (as much) about creating a safe environment
File Fingerprinting

- As a first step, fingerprint the files you are examining so you will know if they change during analysis
- Use md5deep, md5sum, Hex Workshop, etc

```
krk@ws ~> md5sum hello* > md5sum_hello_files.txt
krk@ws ~> cat md5sum_hello_files.txt
  611957bd6a2ad9642027904a65f3638e  hello
  7ab03b44ac6a20b0fa0cc80b636b0f51  hello.c
  bef5bfe7ddf597c8ea86eeecb2cbf52a3  hello_debug
  38e85544dd4349c523430923eafc86ac  hello_static
```

- When you have completed your analysis, or at various points along the way, you should go back and check the md5sums to ensure the values have not changed!

```
krk@ws ~> md5sum -c md5sum_hello_files.txt
```
Virus Scan

- Always scan new malware with an up to date virus scanner.
- Someone else may have already discovered and documented the program you are investigating.
- If the code is not sensitive, consider submitting to http://www.virustotal.com
PEiD

- PEiD is a free program that will tell you details about Windows executable files
- Identifies signatures associated with over 600 different “packers” and compilers
“Caprica6”

- Mandiant tool that identifies packed code (amongst other things)
- Covered in detail later in this talk
Strings

- Sometimes things are easy
- First look at the obvious – strings

$ strings unknown2.exe
...
<host> <port>
-install <host> <port>
-remove
EC.1
EC.2
cmd.exe
connect thread started!
...

- Strings, Bintext, Hex Workshop, IDA Pro
- Be aware of Unicode
Strings

C:\analysis> strings

Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals - www.sysinternals.com
usage: strings [-s] [-n length] [-a] [-u] [-q] <file or directory>
-s    Recurse subdirectories
-n    Minimum string length (default is 3)
-a    Ascii-only search (Unicode and Ascii is default)
-u    Unicode-only search (Unicode and Ascii is default)
-q    Quiet (no banner)
Strings

- Be careful about drawing conclusions
- There is nothing stopping the attacker from planting strings meant to deceive the analyst
- However, strings are a good first step and can sometimes even provide attribution

```
rem barok -loveletter(vbe) <i hate go to school>
rem by: spyder / ispyder@mail.com / 
       @GRAMMERSoft Group / Manila, Philippines
On Error Resume Next
dim fso,dirsyste,dirwin,dirtemp,eq,ctr,file,
    vbscopy,dow eq="" ctr=0
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullName,1)
```
Conducting Web Research

- Look at unique strings, email addresses, network info
  - But! the intruder/author could be watching for you.
- Search the web
  - Be careful … Google cache != Anonymous
  - You might find other victims, or complete analysis
  - Don’t forget newsgroups
- It helps if you know Chinese (or Russian, or Spanish)
No Strings Attached

- Point-and-click “packers” make it easy for intruders to obfuscate the contents of binary tools

- More on packers later…

- We can still gather useful information by examining the layout of the executable file
Inside the PE Format

- Executable File Formats
  - Windows: PE (Portable Executable)
    - www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx
  - Linux: ELF (Executable and Linking Format)
    - www.skyfree.org/linux/references/ELF_Format.pdf

- Useful Information
  - Imports
  - Exports
  - Metadata
  - Resources
Inside the PE

- **Tools**
  - PEview – Wayne Radburn
    http://www.magma.ca/~wjr/
  - Depends – Steve Miller
    http://www.dependencywalker.com
  - PEBrowse Professional – Russ Osterlund
    http://www.smidgeonsoft.com
  - Objdump – Cygwin
    http://www.cygwin.com
  - IDA Pro – DataRescue
    http://www.datarescue.be
  - Resource Hacker – Angus Johnson
    http://www.angusj.com/resourcehacker/
<table>
<thead>
<tr>
<th>pFile</th>
<th>Data</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000084</td>
<td>014C</td>
<td>Machine</td>
<td>IMAGE_FILE_MACHINE_386</td>
</tr>
<tr>
<td>00000086</td>
<td>0005</td>
<td>Number of Sections</td>
<td></td>
</tr>
<tr>
<td>00000088</td>
<td>4405203C</td>
<td>Time Date Stamp</td>
<td>2005/07/25 Tue 13:44:28 UTC</td>
</tr>
<tr>
<td>0000383C</td>
<td>00000000</td>
<td>Pointer to Symbol Table</td>
<td></td>
</tr>
<tr>
<td>00000090</td>
<td>00000000</td>
<td>Number of Symbols</td>
<td></td>
</tr>
<tr>
<td>00000094</td>
<td>0000</td>
<td>Size of Optional Header</td>
<td></td>
</tr>
<tr>
<td>00000096</td>
<td>000F</td>
<td>Characteristics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td>IMAGE_FILE_RELOCS_STRIPPED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0002</td>
<td>IMAGE_FILE_EXECUTABLE_IMAGE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0004</td>
<td>IMAGE_FILE_LINE_NUMS_STRIPPED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0008</td>
<td>IMAGE_FILE_LOCAL_Syms_STRIPPED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0100</td>
<td>IMAGE_FILE_32BIT_MACHINE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0200</td>
<td>IMAGE_FILE_DEBUG_STRIPPED</td>
<td></td>
</tr>
</tbody>
</table>
Resource Hacker

```c
1 VERSIONINFO
FILEVERSION 6,6,2600,1569
PRODUCTVERSION 6,6,2600,1569
FILEOS 0x40004
FILETYPE 0x2
}
BLOCK "StringFileInfo"
{
  BLOCK "040904b0"
  {
    BLOCK "040904b0"
    {
      VALUE "ProductName", "Microsoft (R) Windows (R) Operating System"
      VALUE "CompanyName", "Microsoft Corporation"
      VALUE "FileDescription", "Microsoft Service Provider"
      VALUE "FileVersion", "6.6.2600.1569"
      VALUE "LegalCopyright", "(C) Microsoft Corporation. All rights reserved."
    }
  }
  BLOCK "VarFileInfo"
  {
    VALUE "Translation", 0x0409 0x04B0
  }
```
Disassembly

- Automated disassemblers can take machine code and “reverse” it to a slightly higher-level
- Many tools can disassemble x86 code
  - Objdump, Python w/ libdisassemble, IDA Pro
- But, IDA Pro is what everyone uses
- Manual examination of disassembly is somewhat painstaking, slow, and can be hard
  - Keep your goals in mind and don’t get bogged down
push ecx
    ; 0
call ds:connect
test eax, eax
jz short loc_401604

Sleep and loop back

loc_401604:
    ; size_t
push    44h
push    0
    ; int
lea     eax, [ebp+StartupInfo]
push    eax
    ; void *
call    _moomoot
add     ebp, 80h
mov [ebp+StartupInfo.cb], 44h
mov [ebp+StartupInfo.dwFlags], 101h
mov [ebp+StartupInfo.hShowWindow], 0
mov ecx, [ebp+s]
mov [ebp+StartupInfo.hStdError], ecx
mov edx, [ebp+o]
mov [ebp+StartupInfo.hStdOutput], edx
mov eax, [ebp+s]
mov [ebp+StartupInfo.hStdInput], eax
lea     ecx, [ebp+ProcessInformation]
push    ecx
    ; lpProcessInformation
lea     edx, [ebp+StartupInfo]
push    edx
    ; lpStartupInfo
push    0
    ; lpCurrentDirectory
push    0
    ; lpEnvironment
push    0
    ; dwCreationFlags
push    1
    ; bInheritHandles
push    0
    ; lpThreadAttributes
push    0
    ; lpProcessAttributes
push    offset CommandLine
    ; "cmd.exe"
push    0
    ; lpApplicationName
call    ds:CreateProcessA
test    eax, eax
jnz     short loc_401678
call    ds:GetLastError
Performing Dynamic Analysis
Static Analysis will reveal some immediate information

Exhaustive static analysis could theoretically answer any question, but it is slow and hard

Usually you care more about “what” malware is doing than “how” it is being accomplished

Dynamic analysis is conducted by observing and manipulating malware as it runs
Safe Environment

- Our nice, safe analytical environment wasn’t that important during static analysis

- As soon as you run an unknown piece of code on your system, nothing that’s writable can be trusted

- In general we will need to run the program many times. Snapshots make life easier
System Monitoring

- What we are after
  - Registry Activity
  - File Activity
  - Process Activity
  - Network Traffic

- The tools
  - SysInternals Process Monitor
  - Wireshark
  - + a whole bunch of other stuff
Process Monitor

- Process Monitor is a SysInternals tool that records information about File System, Registry, and Process/Thread activity.
- If you liked Filemon and Regmon—you’ll really like Process Monitor.
- Changes from Filemon/Regmon:
  - Procmon will record everything, user can change display filter at will.
  - Procmon tracks process activity.
Process Monitor

- The key to effective use of Process Monitor for malware analysis is filter configuration.
Wireshark

- Wireshark is a protocol analyzer that captures and decodes network traffic
- Wireshark is not aware of what process generates traffic
- As with Process Monitor, the key is using filters to focus on what is relevant
Dynamic Analysis Example

- Use Process Monitor and Wireshark to quickly reveal the behavior of a malicious program
<table>
<thead>
<tr>
<th>Other Tools</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Port Explorer</td>
<td></td>
</tr>
<tr>
<td>• <a href="http://www.diamondcs.com.au/portexplorer/">http://www.diamondcs.com.au/portexplorer/</a></td>
<td>Monitors network traffic at the connection level</td>
</tr>
<tr>
<td>Malcode Analysts Pack</td>
<td></td>
</tr>
<tr>
<td>• <a href="http://labs.idefense.com/labs-software.php?show=8">http://labs.idefense.com/labs-software.php?show=8</a></td>
<td>fakeDNS</td>
</tr>
<tr>
<td>Paros, Fiddler</td>
<td></td>
</tr>
<tr>
<td>• Web proxies that can capture and modify traffic</td>
<td></td>
</tr>
<tr>
<td>Norman Sandbox</td>
<td></td>
</tr>
</tbody>
</table>
Beyond System Monitoring

- In some cases, simple Static Analysis and System Monitoring will answer your questions

- Beyond this point, you need a debugger
  - Windbg (Microsoft)
  - Ollydbg (Oleh Yuschuk)
  - Ida Pro (Datarescue)

- Or a scriptable debugger
  - Paimei - http://pedram.redhive.com/PaiMei/docs/
  - Vtrace - http://www.kenshoto.com/vtrace/
Armored Malware
Armor Features

- Encryption
- Compression
- Obfuscation
- Anti-Patching
  - CRC Checking
- Anti-Tracing
  - SoftICE, ICEDump Detection Code.
  - Crashes OS if they are Found in Memory
- Anti-Unpacking

- Anti-Vmware
- Polymorphic/Self-Mutating
- Restrictive Dates
- Password Protected
- Configuration Files
Viruses can Circumvent AV?

We're offering anti-detection service for any type of windows modules. There are many ways how to make your module undetected hence you can see below quite complicated price table with examples. To order this service write a mail with full description of what you need to holy_father@phreaker.net. Feel free to write a mail if you're not sure how much would your order cost or if you have special demands (e.g. bypassing any detector that is not in list).

<table>
<thead>
<tr>
<th>feature</th>
<th>Morphine</th>
<th>Hacker defender</th>
<th>Hacker defender driver</th>
<th>Other (no driver or libraries)</th>
<th>Libraries</th>
<th>Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic fee</td>
<td>€ 30.00</td>
<td>€ 20.00</td>
<td></td>
<td>€ 15.00</td>
<td>€ 15.00</td>
<td>€ 15.00</td>
</tr>
<tr>
<td>morphined¹</td>
<td>x</td>
<td></td>
<td></td>
<td>+ € 02.50</td>
<td>+ € 02.50</td>
<td></td>
</tr>
<tr>
<td>morphined - unique²</td>
<td>x</td>
<td></td>
<td></td>
<td>+ € 02.50</td>
<td>+ € 20.00</td>
<td>x</td>
</tr>
<tr>
<td>per AV³</td>
<td>+ € 10.00</td>
<td>+ € 05.00</td>
<td>+ € 05.00</td>
<td>+ € 08.00</td>
<td>+ € 09.00</td>
<td>+ € 10.00</td>
</tr>
<tr>
<td>all AV³</td>
<td>x</td>
<td></td>
<td>+ € 25.00</td>
<td>+ € 30.00</td>
<td>+ € 35.00</td>
<td>+ € 40.00</td>
</tr>
<tr>
<td>unique version⁴</td>
<td>+ € 20.00</td>
<td>+ € 25.00</td>
<td>+ € 20.00</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>source code</td>
<td>+ € 20.00</td>
<td>+ € 30.00</td>
<td>+ € 15.00</td>
<td>− € 10.00</td>
<td>− € 10.00</td>
<td>− € 10.00</td>
</tr>
<tr>
<td>no driver</td>
<td>x</td>
<td>+ € 10.00⁵</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>special</td>
<td>x</td>
<td>special⁷</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Packers

- **Origins**
  - Compression
    - Bandwidth reduction
    - Save space

- **Current use**
  - Bypass anti-virus signatures
  - Prevent reverse engineering
### Packers

- **UPack** by [Dwing](#). 08.IV.2005.
- **Mew** by [Northfox](#). 22.IX.2004.
- **UPX** by [Laszlo & Markus](#). 03.VII.2004.
- **Packman** by [bubba](#). 27.II.2005.
- **CExe** by [Scott](#). 20.III.2003.
- **PE Diminisher** by [tERAPHY](#). 11.IX.1999.
- **PESpin** by [cyberbob](#). 09.III.2005.
- **NSPack** by [North star Tech](#). 05.VI.2005.
- **eXPressor** by [CGSoftLabs](#). 28.III.2005.
- **Thinstall** by [Jonathan Clark](#). 29.III.2005.
- **PECompact** by [DevelTek](#). 06.IV.2005.
- **AS-Pack** (shareware) by [Solodovnikov Alexey](#). 07.I.2002.
- **NeoLite** (shareware) by [NeoWorx Inc](#). 04.IV.1999.
- **ARM Protector** by [SMoKE](#). 22.IX.2004.
Side effects of Packing

- No strings (legitimate)
- “Few” imports
  - Kernel32.dll
    - LoadLibrary
    - GetProcAddress
    - VirtualAlloc
    - VirtualFree
- High entropy sections
  - Marked as code / executable
  - Large difference in Virtual size of section vs. real size
- Fewer Sections
# Side Effects of Packing - Imports

## Unpacked

<table>
<thead>
<tr>
<th>RVA</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>01007A4Ch</td>
<td>cmdlg32.dll</td>
</tr>
<tr>
<td>010074Ah</td>
<td>SHELL32.dll</td>
</tr>
<tr>
<td>01007B3Ah</td>
<td>WINSPOOL.DRV</td>
</tr>
<tr>
<td>01007B5Ah</td>
<td>COMCTL32.dll</td>
</tr>
<tr>
<td>01007C76h</td>
<td>msvcrtd.dll</td>
</tr>
<tr>
<td>01007D08h</td>
<td>ADVAPI32.dll</td>
</tr>
<tr>
<td>01008ECh</td>
<td>KERNEL32.dll</td>
</tr>
<tr>
<td>0100825Eh</td>
<td>GDI32.dll</td>
</tr>
<tr>
<td>0100873Ch</td>
<td>USER32.dll</td>
</tr>
</tbody>
</table>

## Packed

<table>
<thead>
<tr>
<th>RVA</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101AE3Ch</td>
<td>kernel32.dll</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RVA</th>
<th>Hint</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101AE00h</td>
<td>0000h</td>
<td>LoadLibraryA</td>
</tr>
<tr>
<td>0101AE04h</td>
<td>0000h</td>
<td>GetProcAddress</td>
</tr>
<tr>
<td>0101AE08h</td>
<td>0000h</td>
<td>VirtualAlloc</td>
</tr>
<tr>
<td>0101AE0Ch</td>
<td>0000h</td>
<td>VirtualFree</td>
</tr>
</tbody>
</table>
Side Effects of Packing – Section Size and Entropy

Unpacked: Entropy (st dev): 0.7653

<table>
<thead>
<tr>
<th>Name</th>
<th>Virtual Size</th>
<th>Virtual Address</th>
<th>Size of Raw Data</th>
<th>Pointer to Raw Data</th>
<th>Characteristics</th>
<th>Pointing Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>00000274h</td>
<td>01001000h</td>
<td>000003800h</td>
<td>000000400h</td>
<td>E0000000h</td>
<td>Import Table, Debug Data, Load Config</td>
</tr>
<tr>
<td>data</td>
<td>0000012A8h</td>
<td>01008000h</td>
<td>000003800h</td>
<td>000007C00h</td>
<td>C00000040h</td>
<td>Resource Table</td>
</tr>
<tr>
<td>rsrc</td>
<td>000003858h</td>
<td>01008000h</td>
<td>0000038400h</td>
<td>00008400h</td>
<td>400000400h</td>
<td>Resource Table</td>
</tr>
</tbody>
</table>

Packed: Entropy (st dev): 1.0666

<table>
<thead>
<tr>
<th>Name</th>
<th>Virtual Size</th>
<th>Virtual Address</th>
<th>Size of Raw Data</th>
<th>Pointer to Raw Data</th>
<th>Characteristics</th>
<th>Pointing Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>00010000h</td>
<td>01001000h</td>
<td>000004000h</td>
<td>000000400h</td>
<td>E000000060h</td>
<td></td>
</tr>
<tr>
<td>rsrc</td>
<td>0000000000h</td>
<td>01014000h</td>
<td>000007C00h</td>
<td>00000400h</td>
<td>E000000020h</td>
<td>Import Table, Resource Table</td>
</tr>
</tbody>
</table>
Strings on Packed Binary

C:\analysis>strings sak.exe
Strings v2.1
Copyright (C) 1999-2003 Mark Russinovich
Systems Internals - www.sysinternals.com
!Windows Program
$PE
@.data
.idata
$s!
;Ot
(!B
KERNEL32.dll
LoadLibraryA
GetProcAddress
DM.D
&DS
d'D
~E-
So, Packing == Bad?

- No … there are legitimately packed apps
  - Google Desktop Search – “Troubleshoot Network.exe” : PECompact V 2
  - Adobe Acrobat 7.0 – AdobeUpdateManager.exe : PECompact V 2
- So, how do you tell the difference?
Knowing the difference

- Collect as much static info about a PE as feasible
- Correlate the extracted information
- Score / rate what you know
PE Header Anomalies
- Incorrect image size
- Unaligned sections
- Non-ASCII section names
- Overlapping headers

Entry point signature detection
- Quick scan using xor from the entry point

Full section roaming signature detection (using modified Boyer-Moore search)
- Signatures are developed subjectively (by reverse engineering) to lower FP’s and obtain signatures for “code” that would be difficult to modify (and are necessary for unpacking)
“Caprica Six” (cont’)

- Comparison of EP and roaming signature
  - Used to determine trying to hide packer use
    - Adding a new section with “fake code”
    - Instruction swaps at the entry
- Full section entropy calculation (sliding window check with standard deviation – patent application submitted)
  - Comparison against section characteristics (code, executable … not resource)
- PE Import extraction
- Digital Signature checking (code signing with X509 Cert)
- Additional info (section names, section sizes, etc).
- Scoring
PE Anomalies – Overlapping Header

- DOS Header is actually overlapped by NT Headers
- Loader allows for this (fields in header are ignored)
- Module and functions are scattered!
- What compiler does THIS???
Roaming Signature – Multiple detections

- Signature detected in 2 sections
  - .data
  - .rsrc
- Why? Embedded executable ALSO packed!!

```xml
<Section>
  <Name>.rsrc</Name>
  <Type>Resource</Type>
  <LengthInBytes>81920</LengthInBytes>
  <DetectedCharacteristics>Read</DetectedCharacteristics>
  <Entropy>
    <AverageValue>1.1230640</AverageValue>
  </Entropy>
  <DetectedSignatureKeys>
    <string>Aspack v 1.02X - 1.08X</string>
  </DetectedSignatureKeys>
</Section>

<Section>
  <Name>.data</Name>
  <Type>None</Type>
  <LengthInBytes>4096</LengthInBytes>
  <DetectedCharacteristics>Read Write</DetectedCharacteristics>
  <Entropy>
    <AverageValue>0.3580073</AverageValue>
  </Entropy>
  <DetectedSignatureKeys>
    <string>Aspack v 1.02X - 1.08X</string>
  </DetectedSignatureKeys>
</Section>
```
Unpacking

- Ollydbg → OllyScript → OllyDump
- Ollydbg → bp in Library → OllyDump
- UnFSG, upx, etc
- PEiD
- ProcDump
- OEPFinder
- etc…
Other Unpackers

- Ollydbg with the Ollydump plugin and a variety of OllyScripts *
- IDAPro with the “Universal Unpacker Plugin”.
- DeYoda by C-ripper. 18.II.2001.
- DePE-PACK by Unknown One. 03.V.2002.
- StealthKiller by Snow Panther. 04.IX.2002.
Questions?

kris.kendall@mandiant.com
chad.mcmillan@mandiant.com