

# Analyzing Malicious Code

Nicolas Brulez  
Ryan Russell

Disassembly with a time constraint  
Recon 2005

# Major Analysis Methods

- Sacrificial Lamb
- Resource Monitoring
  - Filemon, Regmon, Ethereal
- Disassembly
  - Including light debugging
- BinDiff

# Focus on Disassembly

- Gives the most complete picture
- Allows reuse of code fragments
- Enables modification for research
  
- Unfortunately, can also be the slowest method

# Purpose of Disassembly

- Understanding!
- You are trying to get the binary into your head
- Most of the time you are NOT trying to modify the binary, find a vulnerability, or fix and improve it

# Contrast With

- Cracking
- Vulnerability research
- Debugging
- Borrowing algorithms and methods

# Why MC Analysis is easy

(Easier than porting Linux to a closed device)

- We don't need to patch the binary
- We already know some of what it does
- We can make big sweeping assumptions
- We can skip big sections of code

# Bottom-up Analysis

- Yes, we use IDA Pro
- Identify sections that give you the biggest impact
- Try to start with the most commonly-used pieces
- Stick to the program flow you need to know about

# Priority

- Library Functions
- Imports
- Function Prototypes
- Entry Points
- Interesting calls
  - LoadLibrary, network, rand, registry, file, CreateThread
- Structure cleanup (SEH, fragments, etc...)

# Unpacked Demo

Hotworld

Special thanks to Zone Labs for  
assistance with this trojan

# My Conventions

- Bottom-up, identify RETNs first
- Mark loops
- Name vars
- Naming convention
  - TO meaning or FROM meaning
- Copious comments, manually trace register values

# Barriers

- Packing/crypting
- Higher-level languages/Object Orientation
- P-Code
- Self-modification

# Packed Demo - Intro

Michael Jackson trojan

Unpacking - Nico

# Unpacking PE Files

- Unpacking knowledge is very handy for a Reverse Engineer.
- A lot of files are packed nowadays.
- Especially malware.
- There are a LOT of different PE packers and PE protectors.

# Is my file Packed/Protected?

- Is the last section executable ?
- Is the first section writeable ?
- Is the first section's rawsize null ?
- Is the Entry Point starting in the last section ?

# Is my file Packed/Protected?

- Check the section names
- Check the Import Table : Very few imported functions ?
- Check the strings : no strings at all ?
- Is the Raw Size way smaller than the Virtual Size? Compressed!

# Is my file Packed/Protected?

[ PE Editor ] - c:\documents and settings\nico\bureau\binaries\day2\unpacking fs

Basic PE Header Information

EntryPoint: 00005000      Subsystem: 0002 ...

OK  
Save

Im [ Section Table ]

Name	VOffset	VSize	ROffset	RSize	Flags
	00001000	00003000	00000000	00000000	C00000E0
	00004000	00001000	00000400	000000BA	C00000E0
	00005000	00001000	00000200	00000200	C00000E0

S  
Fi  
M

# Is my file Packed/Protected?

The image shows two windows from the PE Editor. The left window is the 'Section Table' dialog, which displays a table of sections. The right window is the 'Section Flags' dialog, which shows a list of flags for the selected section.

**Section Table**

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00003000	00000200	00000600	E0000060
.rsrc	00004000	000108CC	00000800	00001200	E0000020

**Section Flags**

Set Flags

- Shareable in memory
- Executable as code
- Readable
- Writeable
- Contains extended relocations
- Discardable as needed
- Can't be cached
- Not pageable
- Contains COMDAT data
- Contains comments or other infos
- Won't become part of the image
- Contains executable code
- Contains initialized data
- Contains uninitialized data
- Shouldn't be padded to next boundary

Alignment: default Bytes

Current Value: E0000020

Buttons: OK, Cancel

# Basic Unpacking Methods

## **.Find the Original Entry Point**

- Trace slowly until you jump to the real program code.
- Use Static Disassembly to find the jump to original entry point.
- Smart use of hardware breakpoints.
- Breakpoints on API Functions.

## **.Dump the Process to disk**

# Basic Unpacking Methods

- **Reconstruct the Import Table**
  - Trace the packer and find where the IAT handling is, so you can grab information about the import table and reconstruct it manually, eventually.
  - You can just use Import Reconstructor to reconstruct the import table and get ride of the boring work.

# Unpacking

## Demo

Questions?

# Thank you!

Ryan Russell – [ryan@thievco.com](mailto:ryan@thievco.com)

Nicolas Brulez - [nico@recon.cx](mailto:nico@recon.cx)