Invisible to Visible

“I wouldn’t have written the TCP Wrappers….”

- Wietse

Recovery of Lost or Hidden Data

- Why?
- Hiding among the visible
- Speed
- Types of data
- Traditional methods used to dis/uncover
- Traditional analysis
- Validity of data & methods

Why?

- Disaster/Accident recovery
- Password activity
- Recovering logs
- Exploit code, malware
- Evidence of Activity
- Spying

Hiding Among the Visible

- Trivial to hide things
- Very difficult to find
- Cryptography/stenography
- Imbedded data in valid files

Speed… not.

- Slow
- Mountains of data
- There is a reason for files and structure…
- Slow
- Difficult to automate analysis
- Collection slow, analysis slower

Types of Data

- Unallocated files
- The used filesystem
- Memory
- Swap

Methods

- **`dd(1)`**
- **`cat(1)`**
- **`strings(1)`**
- unrm
- icat
- ils

Analysis

- **`grep(1)`**
- **`less(1)`**
- perl/shell scripts
- Lazarus

Lazarus Functionality

- Data reconstruction based on content
- Brings structure to unstructured data
- Pattern recognition
- Data broken into files
- Hypertext user interface
- Enables browsing of data

(Quick Demo of Laz)

[a quick overview of what it looks like and how the basic function works, not much else]

"I admire the progress that you have made on what I view as a nearly intractable problem!"
-- Kirk McKusick

- Not Norton Utilities!
- How good is it?
- Why does it work?
- Logfiles
- Usage
- Time required

Just How Good is this Thing?

- It depends…
- Small files are easy
- Large files are not, unless it has a regular format or type that is easy to recognize
- Size of file vs. size of filesystem
- It should get **ALL** data

How does it work?

- 1024 is a magic number
- Unix filesystems suggest locality
- Unix FS try to create contiguous blocks
- Data isn’t going anywhere
- Simple recognition techniques work!

Algorithm

- Examine first 100 bytes of a block of data
- **If** (text), try regular expressions
- **If** (binary), use **`file(1)`**
- **If** (last block analyzed was same as current), append
- **Else** create new file

- Post processing

(Demo - show sniffer log stuff)

[explain how working on program and simply looking at data found something very interesting]

Example - Finding Exploit Code

- A combination of:
  - MACtime (see handout)
  - unrm
  - lazarus
  - grep program files for source code

Post Processing

- Standard Unix tools
- Reconstructing mail files
- Text-based log files
- Correlator (binary, repeating logs, etc.)
- File sanity checking

Validity of Results

- Pictures are easy
- Poor reliability
- Easy to miscategorize or miss data
- Analysis fraught with peril
- Conclusions based on this are probably the most dangerous of all