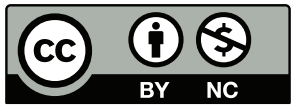
After an Attack



Sometimes the Bad Guys Win

- What do you do if a machine is compromised?
- How do you assess the damage?
- How do you recover?
- What else should you do?

How Can Machines be Compromised?

- 0-day attacks?
- Carelessness?
- Insider attacks?
- In some sense, it doesn't matter; you still have to recover

Damage Assessment — Why?

- What has to be thrown out?
- What can be saved?
- How did the bad guys get in?

A General Rule

- It is frequently impossible to cleanse an infected system
- Hiding back doors is relatively easy
- The usual advice: reformat your disks and reinstall

Hidden Back Doors

- Cron jobs
- Standard services — with a twist
- Programs buried in someone's `.profile`
- Jobs started via **`at`** or **`batch`**
- Trojan horses in commands likely to be executed by **`root`**
- More ways? Of course

Self-Repairing Malware

- Malware contains two or more programs with back doors
- Each also checks for the other's presence
- If one has been deleted — perhaps by an administrator cleaning up — the other repairs it
- Solution: clean up system while running from a CD or other known-good medium

Bots and Boats

- Many infections these days turn machines into “bots”
- Modern bot software can upgrade itself
- Bots can download different payloads — spam engines, DDoS engines, keystroke loggers, etc.
- They’re sometimes known as “boats”, because they can carry anything

Protecting Malware

- Bots often install rootkits, to prevent detection
- The “botherd” — the person who has “p0wned” your machine doesn’t want to lose control of it
- Some bots close other security holes, to prevent bot-jacking
- Other bots, though, block Windows Update and (especially) anti-virus updates
- Goal: to prevent A/V software from detecting its presence

The Network

- The attacker has access to all network-accessible resources of the compromised machine
- Attack shared file systems
- (Many worms spread that way)
- The compromised machine is on the inside of the firewall
- The compromised machine is on your LAN

Trust Patterns

- Your users will (probably) trust an insider more than an outsider
- The attacker can gain knowledge to use in a subsequent attack
- Were any password files compromised?

Backups Are Your Friend

- Back up your system frequently
- Make sure you have a 0-day backup, from before the system went live
- Recover your data — but not your programs — from the backups

(Is That Enough?)

- Suppose there's some non-sensitive application with a buffer overflow
- It reads one of your "secure" data files
- The attacker puts the buffer overflow into the data file, triggering a new penetration when the data files are restored after reinstalling the code
- Oops. . .

Restore and Compare

- If you have good backups, you could restore to another machine and compare files
- Very time-consuming
- Besides, some files change

Tripwire

- Create a cryptographic checksum of each file
- To detect changes, recalculate the checksums and compare against the stored copy
- Easier said than done. . .

Changes

- Can you trust your master list of checksums?
- Can you trust the software that's calculating the new checksums?
- An attack: detect when Tripwire is running and give a different answer

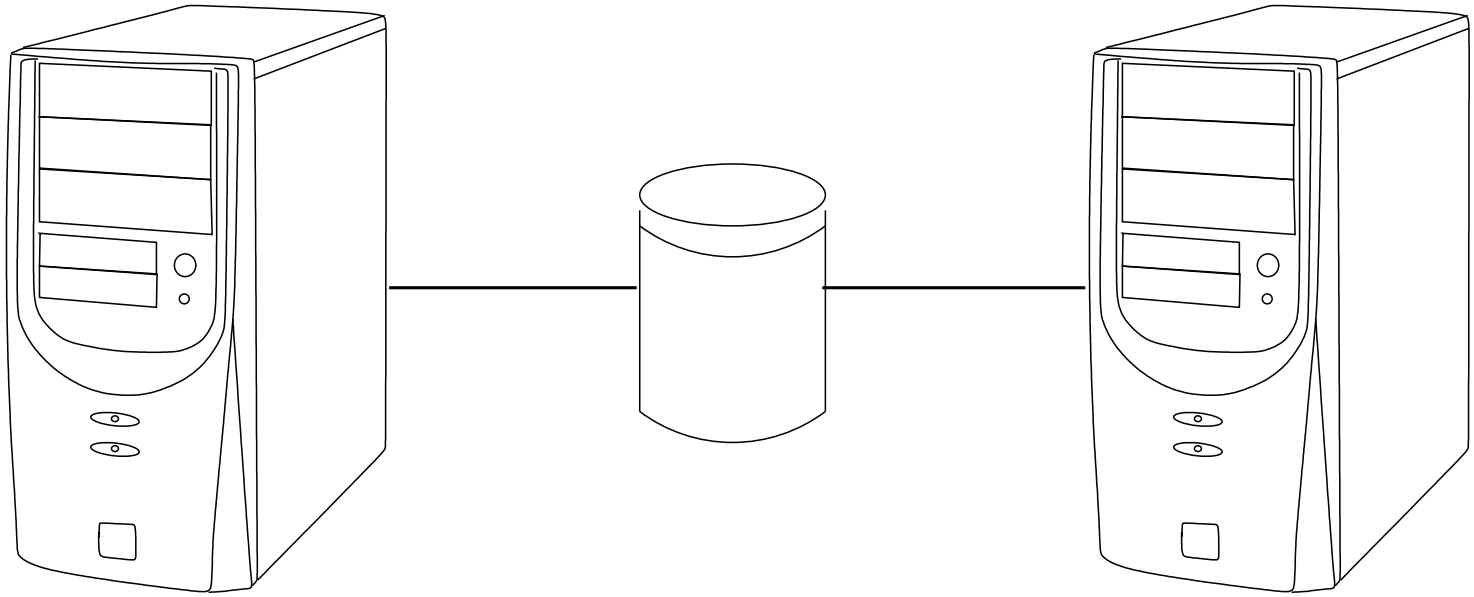
A Real Example

- A (Linux) loadable kernel module intercepted file system operations
- If pid 1 tried to open **`/sbin/init`**, it got the Trojan horse version
- If any other process did the open, it got the real version
- Tripwire wouldn't detect the substitution!

Safely Using Tripwire

- Store the checksum file on (physically safe) media
- Use another machine to read the disk you're checking
- Don't trust any software on the (possibly) compromised machine

Dual-Ported Disk



Analyzing a Hacked System

- Suppose you want to analyze a compromised system
- What if the bad guy tried to hide?
- How do you proceed?

Work with a Copy

- *Never* try to work with a live disk
- You don't want to destroy metadata
- Be careful of the malware!
- Make a copy — preferably an image copy; failing that, use dump/restore
- Don't use anything that will change file access times

Live CDs

- If you don't have a spare machine (with compatible hardware), trying booting a "live" CD
- A live CD is a a bootable, runnable system
- Example: Knoppix; Ubuntu installer

Mounting the Image

- Always mount it read-only, with the “noexec” and “nodev” options
- Most newer systems allow you to mount a file as a block device (**`vnd`** on BSD; **`lofiadm`** on Solaris; loopback device on Linux; .dmg files on Mac OS; etc.)

Things to Look For

- What files were changed recently?
- Note: look at **`ctime`**, not just **`mtime`** (why?)
- Or run Tripwire against either the 0-day Tripwire dump or a known-good installation disk

Log Files

- Check your logs for suspicious entries from compromised machines
- ☞ Netflow data is especially important here
- Outbound connections from known-infected machines can indicate attempts to spread the problem
- Earlier, inbound connections *to* the infected machine can show how the problem started — and identify other infected machines
- (If the infection came from outside, do you notify the site? How?)

Funky Filenames

- Files and directories can be hidden by using strange file names
- Examples: “...” (3 dots), “bin ” (trailing blank), **`/usr/lbb`** (instead of **`/usr/lib`**), C:\WINDOWS\system32\Com\Inf\[4 BLANK SPACES].exe
- Names resembling real filenames: C:\WINDOWS\Windows Explorer.exe

Finding Deleted Files

- Deleting a file doesn't delete the data
- Instead, it changes some metadata — the filename on FAT filesystems; the i-node number and i-list entry on traditional BSD filesystems
- The blocks are returned to the freelist — but they may not be reallocated immediately
- Clever tools can recover deleted files

Digression: Serious Threats

- Even overwriting a block doesn't delete physical traces of the data
- There are (classified?) techniques to recover data
- At a minimum, disks need to be overwritten three times — and sometimes, you just destroy the disk *thoroughly*

Rebuilding Deleted Files

- Suppose there are no clues in directories or the i-list
- Sometimes, it's possible to do magic with the freelist
- Files aren't random. . .

File Types

- Different file types have different byte distributions
- Example: C has lots of { and }; text has distinctive capitalization patterns, etc.
- Sort blocks by (probable) type

Contact Probabilities

- Look for matches between the end of one block and the start of the next
- Look for syntactically correct statements
- Log files have timestamps!

Are Deleted Files *Better* for Forensics?

- A normal file can be overwritten easily
- A deleted file can't be touched
- Block allocation policies are invisible to the application
- Some claim that deleted files are *more* likely to be intact

Looking at Memory

- If the system is still up, dump main memory (**`/dev/kmem`**)
- Can often find plaintext of the malware
- Encrypting file systems write ciphertext to disk — but where's the plaintext? Often, in RAM
- The decryption key is in RAM, too, if the file system is mounted

Digression: Doing Crypto

- Always zero out plaintext as soon as possible
- That's even more true for keys
- Especially do this before program exit, when pages are handed back to the OS
- Also, lock the pages into memory, to make sure there's no copy in a swap file

Criminal Prosecution

- Suppose you want to prosecute the bad guys
- Should you do these forensics?
- No!

Evidence Procedures

- Evidence must be handled *very* carefully
- Must avoid defense charges of tampering, forgery, misinterpretation (to say nothing of legal issues such as proper warrants)
- Parties with more interest in a case can be portrayed as biased

Techniques

- Chain of custody
- Rigorous marking, labeling, logging, etc.
- Careful records of all analysis
- Not a job for amateurs

Conclusions

- A lot can be learned from compromised systems
- A really thorough analysis is difficult, and probably more time-consuming than reinstallation
- For special situations, get expert help