Forensic Discovery

Wietse Venema

wietse@porcupine.org

IBM T.J.Watson Research, USA

```
rm -rf / &
```

Dawn of the Internet age in Eindhoven, The Netherlands

- Eindhoven university was among the first universities in the Netherlands to get an internet connection.
- This attracted a number of users who had no official relationship with the university.
- Most unofficial users were careful not to draw attention to their activities.
- Unfortunately, there was one exception...

[a recently emptied file system]

Challenges

- *Problem*: empty disks don’t reveal how intrusions happen (or that is what I thought at the time).

  Solution: instrument the network software to log activity *before* disaster strikes.

- *Problem*: no source code and no expertise to update network software of SUN, Digital, Apollo, HP, and IBM systems across campus.

  Solution: write the smallest possible program to log the type and origin of network connections.

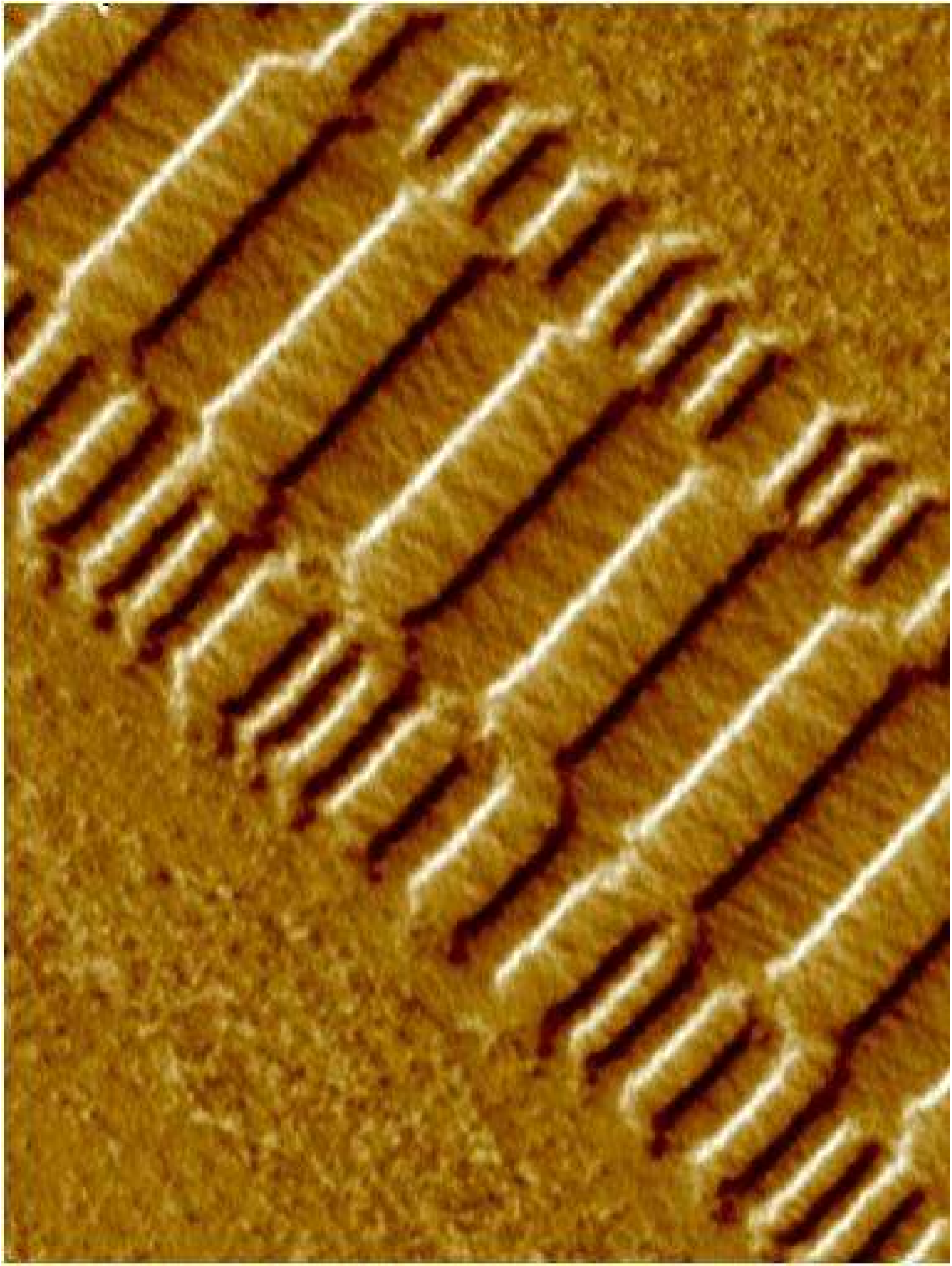
```
date   time     hostname service  content of logged message
May 21 14:06:53 tuegate: systatd: connect from monk.rutgers.edu
May 21 16:08:45 tuegate: systatd: connect from monk.rutgers.edu
May 21 16:13:58 trf:     systatd: connect from monk.rutgers.edu
May 21 18:38:17 tuegate: systatd: connect from ap1.eeb.ele.tue.nl

May 21 23:41:12 tuegate: systatd: connect from mcl2.utcs.utoronto.ca
May 21 23:48:14 tuegate: systatd: connect from monk.rutgers.edu
May 22 01:08:28 tuegate: systatd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:14:46 tuewsd:  fingerd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:15:32 tuewso:  fingerd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:55:46 tuegate: systatd: connect from monk.rutgers.edu
May 22 01:58:33 tuegate: systatd: connect from monk.rutgers.edu
May 22 02:00:14 tuewsd:  fingerd: connect from monk.rutgers.edu
May 22 02:14:51 tuegate: systatd: connect from RICHARKF-TCACCIS.ARMY.MIL
May 22 02:19:45 tuewsd:  fingerd: connect from RICHARKF-TCACCIS.ARMY.MIL
May 22 02:20:24 tuewso:  fingerd: connect from RICHARKF-TCACCIS.ARMY.MIL

May 22 14:43:29 tuegate: systatd: connect from monk.rutgers.edu
May 22 15:08:30 tuegate: systatd: connect from monk.rutgers.edu
May 22 15:09:19 tuewse:  fingerd: connect from monk.rutgers.edu
May 22 15:14:27 tuegate: telnetd: connect from cumbic.bmb.columbia.edu
May 22 15:23:06 tuegate: systatd: connect from cumbic.bmb.columbia.edu
May 22 15:23:56 tuewse:  fingerd: connect from cumbic.bmb.columbia.edu
```

Overview

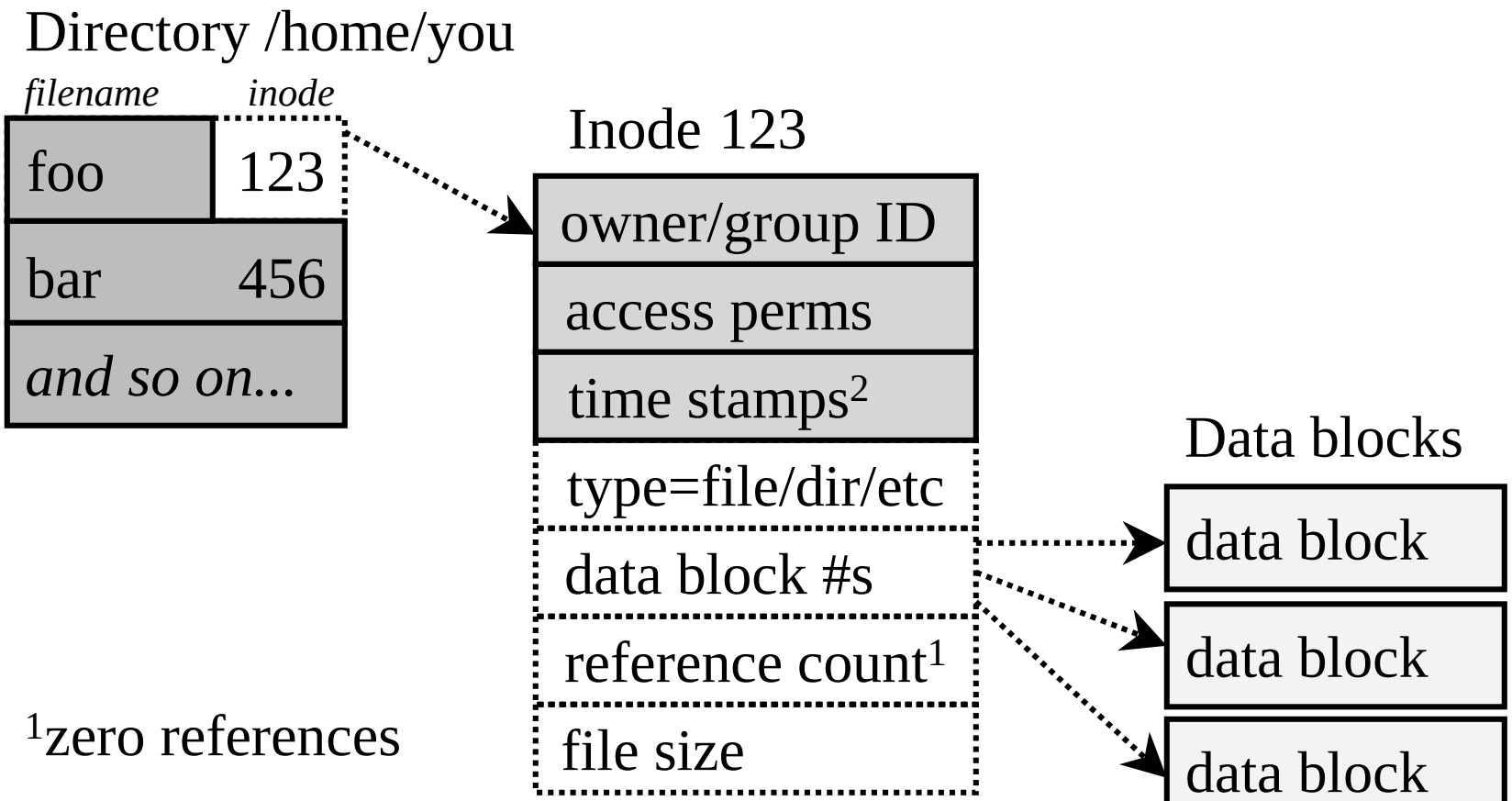
- *Introduction*: why I didn't write Coroner's Toolkit utilities many years earlier.
- *Recent work*: volatility and persistence from file systems to main memory.
- *Outlook*: subversion from user-land to hardware.

File System Persistence

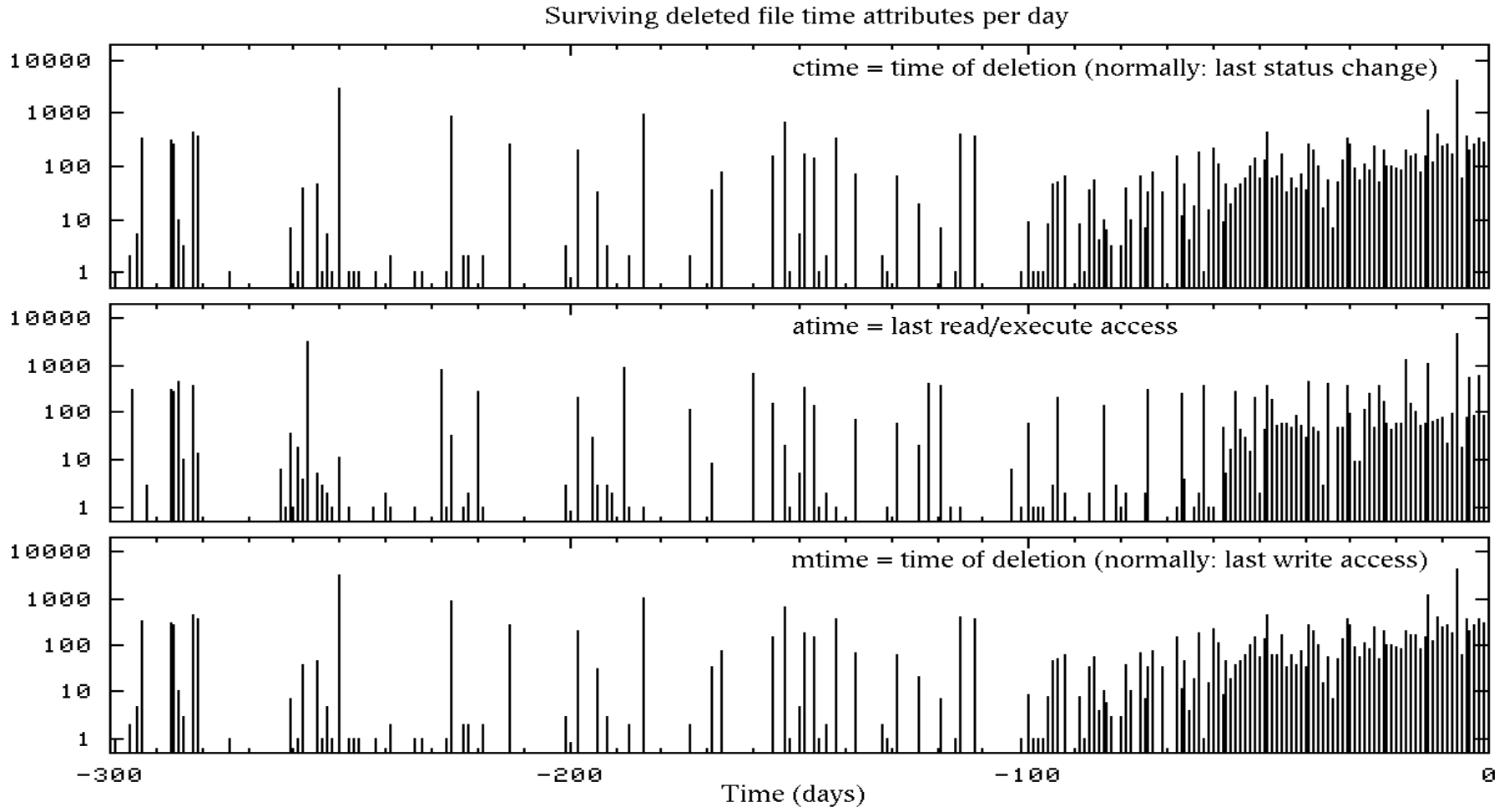
Deleted file info may be more persistent than existing files



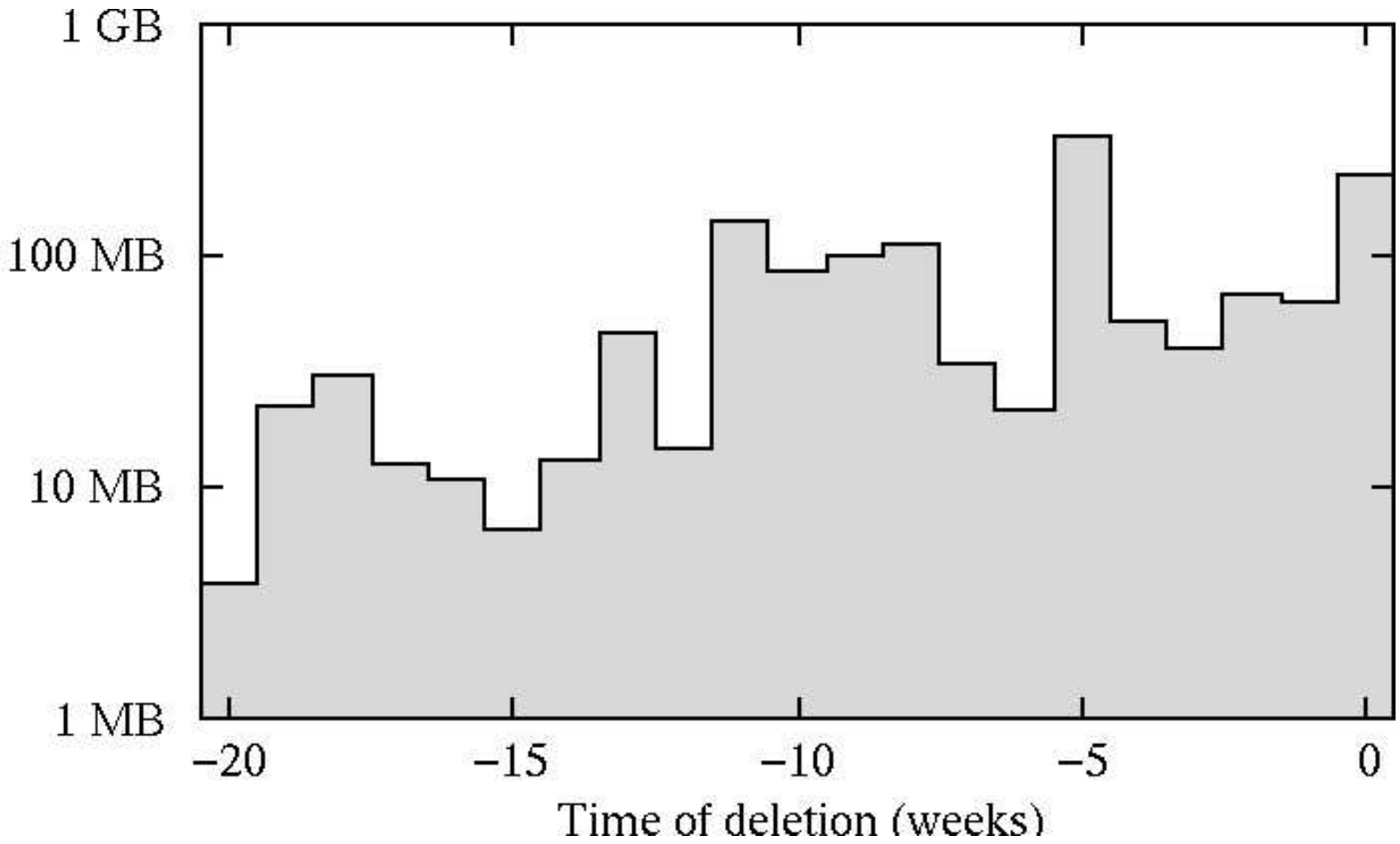
Deleting a file destroys structure not content



Persistence of deleted file time attributes - dedicated UNIX server



Persistence of deleted file content - same dedicated UNIX server



Summary: persistence of deleted file content

Machine	File system	Half-life
spike.porcupine.org[1]	entire disk	35 days
flying.fish.com[2]	/	17 days
flying.fish.com[2]	/usr	19 days
www.porcupine.org[1]	entire disk	12 days

[1]FreeBSD [2]Linux

Main Memory Persistence

Recovering Windows/XP files
without knowing the key

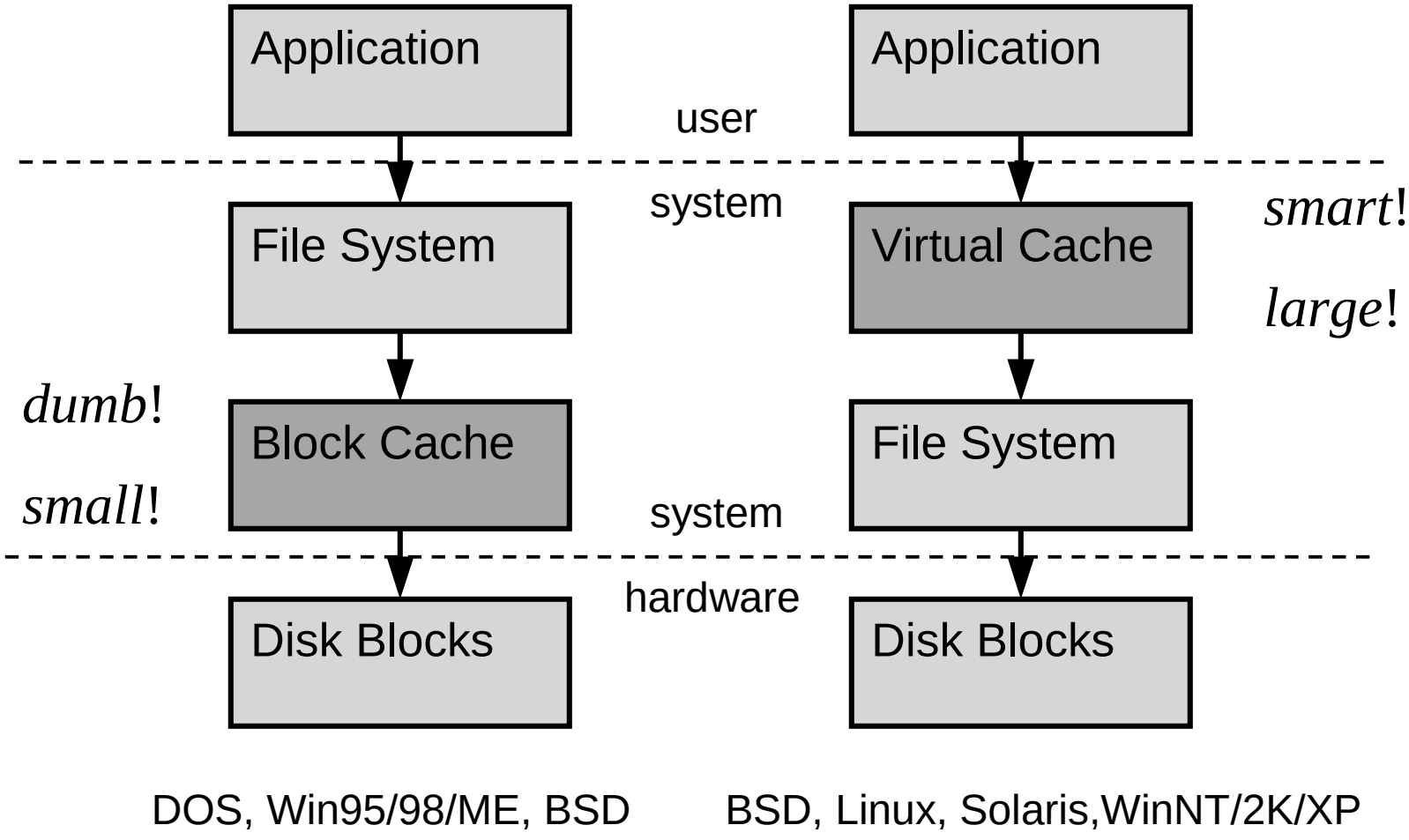
Information in main memory

- Running processes[1].
- Terminated processes[1].
- Kernel memory.
- Recently active files/directories (file cache).
- Deleted files (from process or from cache).
- All have different persistence properties.

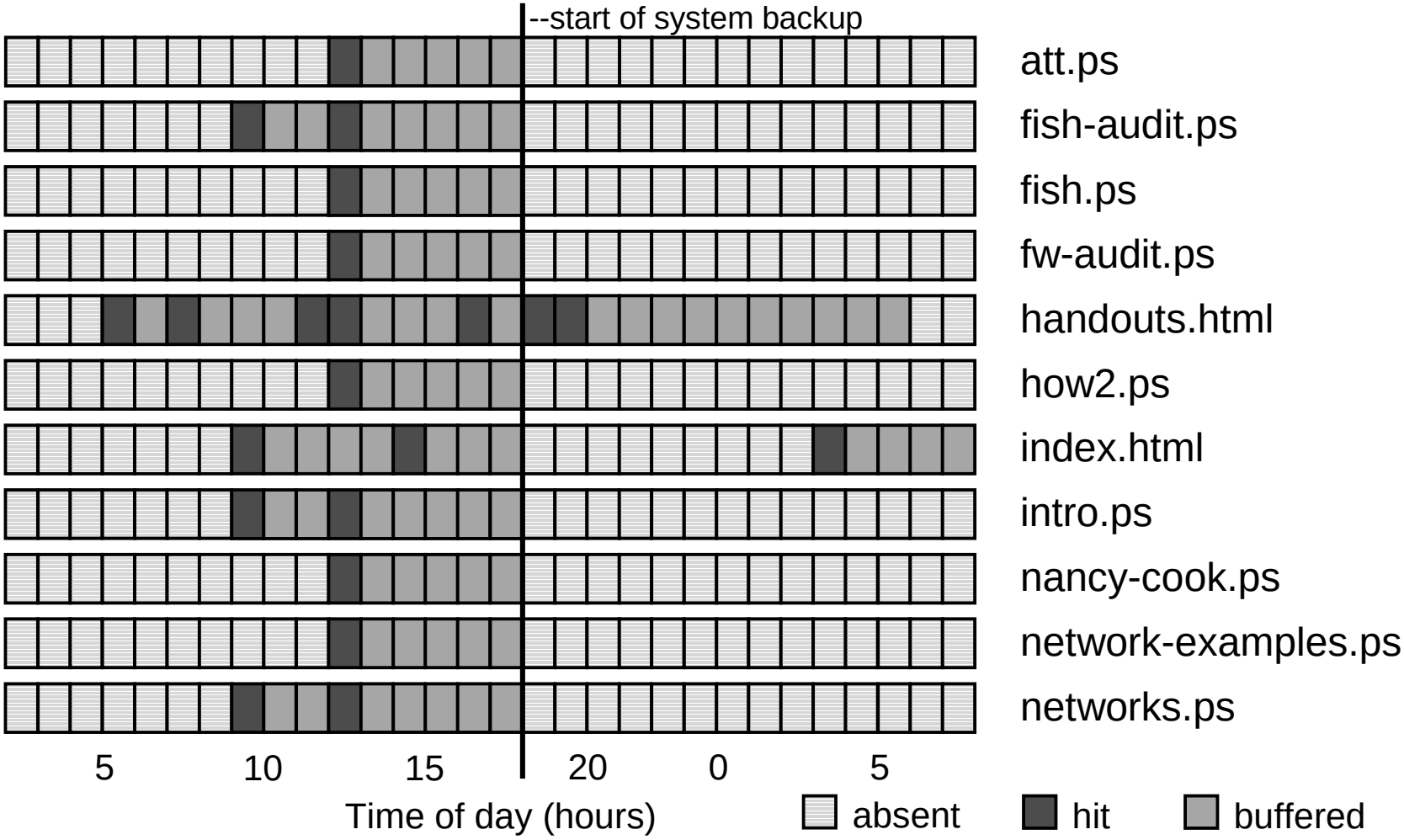
[1]Some information may be found in swap files.

Block cache versus virtual cache

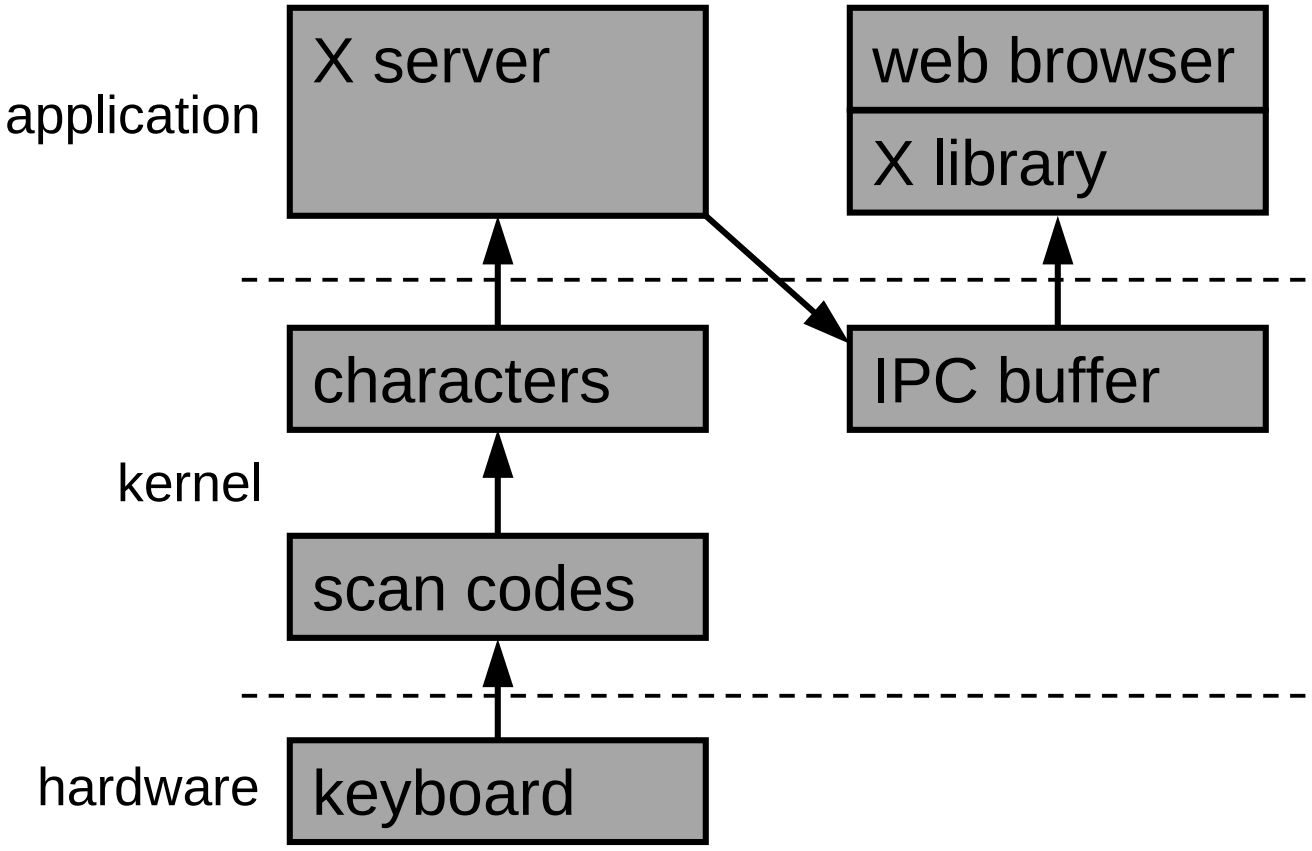
(owned by system, not by applications)



File caching in main memory
(low-traffic web pages, FreeBSD)

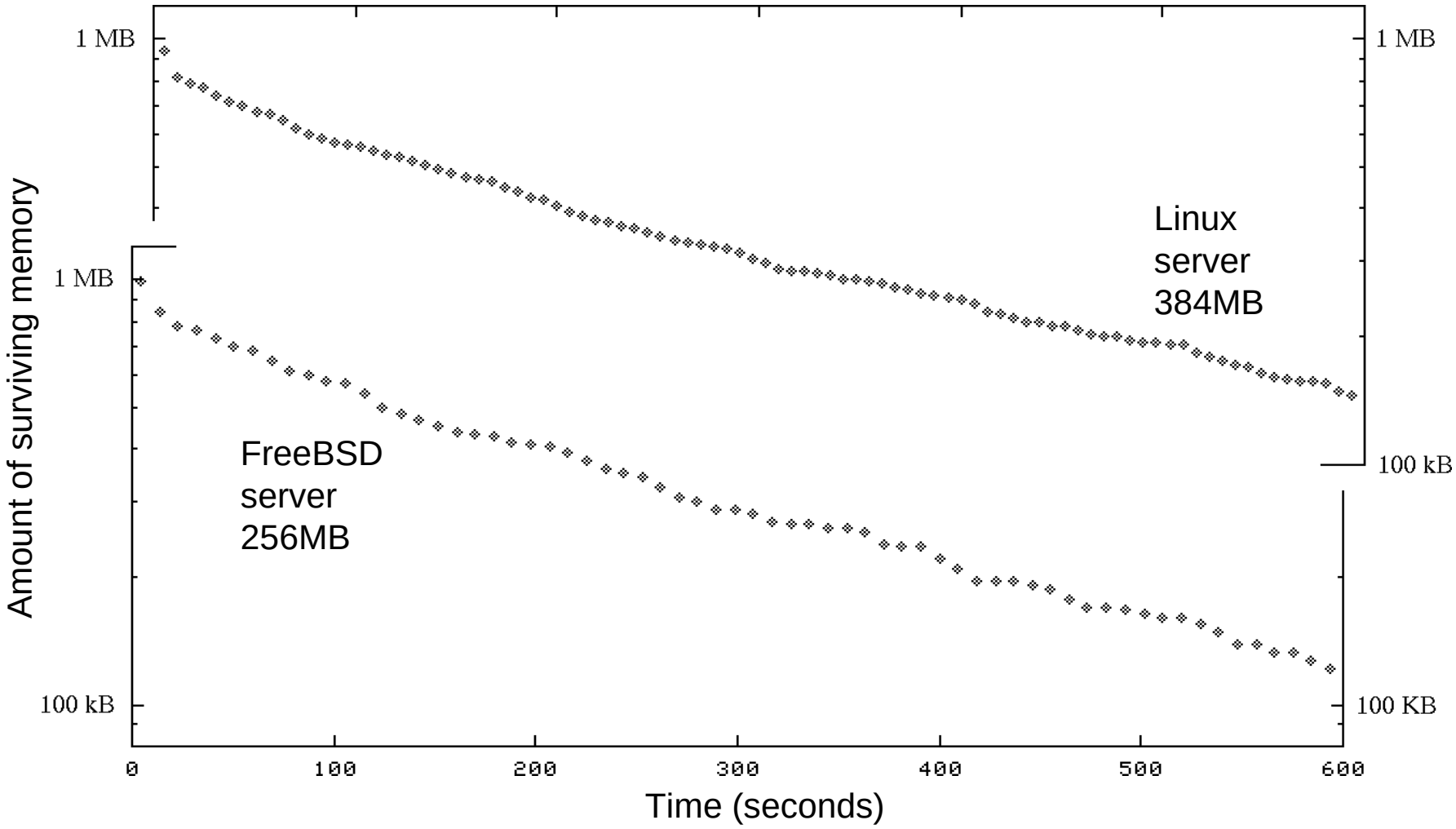


Trail of secrets across memory
(after Chow *et al*.)



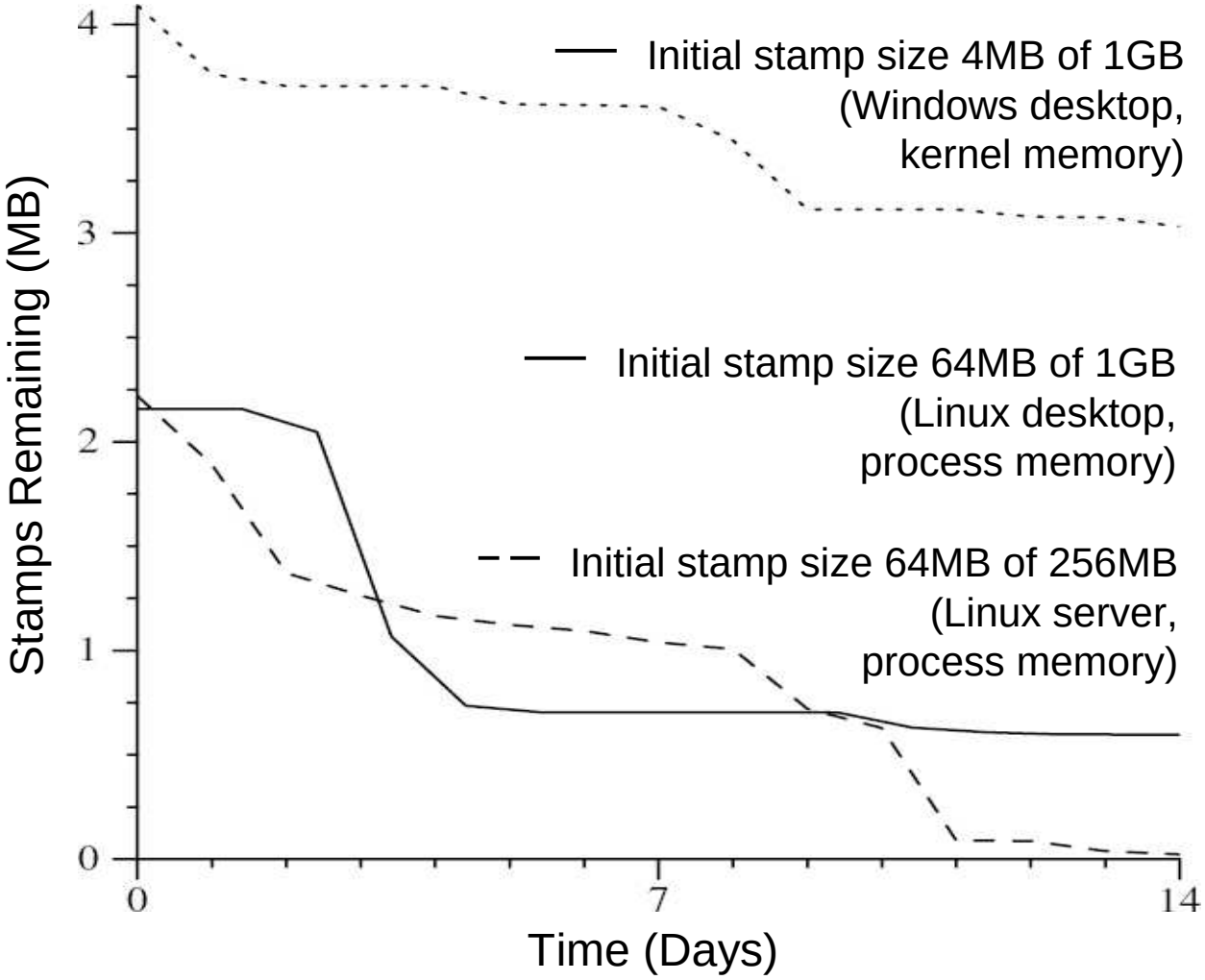
Short-term memory persistence

after process termination (1MB stamp)



Long-term memory persistence

(Chow *et al*., USENIX Security 2005)



Recovering Windows/2K/XP encrypted files without key

- EFS[1] provides encryption *by file* or *by directory*. Encryption is enabled via an Explorer property dialog box or via the equivalent system calls.
- With encryption by directory, files are encrypted *before* they are written to disk.
- Is unencrypted content of EFS files cached in main memory?
- If yes, for how long?

[1]EFS=Encrypting File System

Experiment: create encrypted file

- Create “encrypted” directory c:\temp\encrypted.
- Download 350kB text file via FTP, with content:

```
00001 this is the plain text
00002 this is the plain text
...
11935 this is the plain text
11936 this is the plain text
```

- Scanning the disk from outside (VMware rocks!) confirms that no plaintext is written to disk.

Experiment: search memory dump

- Log off from the Windows/XP console and press Ctrl/ScrollLock twice for memory dump[1].
- Analyze result with standard UNIX tools:

```
%strings memory.dmp | grep 'this is the
  plain text'
03824   this is the plain text
03825   this is the plain text
. . .etcetera. . .
```

- 99.6% of the plain text was found undamaged.

[1]Microsoft KB 254649: Windows 2000 memory dump options.

Recovering Windows/XP encrypted files without key

- *Good*: EFS encryption provides privacy by encrypting file content before it is written to disk.
- *Bad*: unencrypted content stays cached in main memory even after the user has logged off.
- Similar experiments are needed for other (UNIX) encrypting file systems. Most are expected to have similar plaintext caching behavior.

Trends in Subversion

Hardware is getting softer as complexity increases

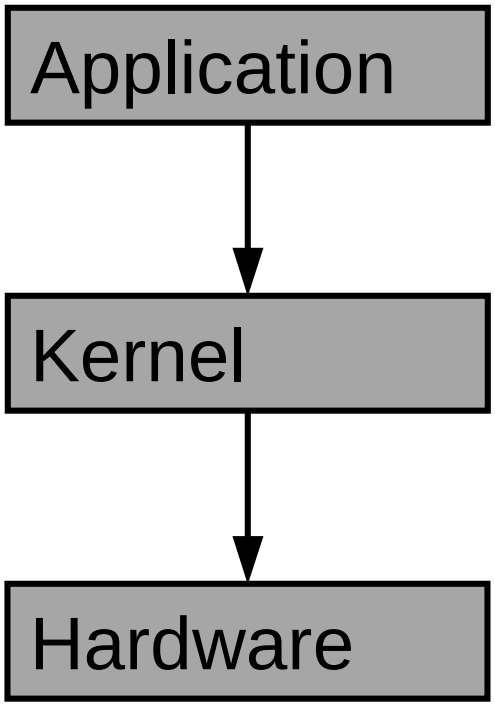
Root kits gen#1 - command level

- Malware (example: ethernet password sniffer).
- Backdoor (example: modified login program).
- Patched *command/library files* to hide malware and backdoor processes/files/connections.
- Sometimes: logfile editors, file checksum fixers.
- Easy to find via inconsistencies (*echo* **.*** <=> *ls*).
- *Easy to find in post-mortem disk images.*

Root kits gen#2 - kernel level

- Malware (distributed denial of service, spam relay, or other remote control).
- Backdoor (example: modified system call or network handling code).
- Patched *running kernel* to hide malware and backdoor processes, files, or connections.
- May show up via inconsistencies (*ps* <=> */proc*).
- *May not show up in post-mortem disk images.*

Progression of subversion by extrapolation



First generation

Second generation

The future is here?
(focus on the machine itself
instead of evil plug-in hardware)

Hardware is not what it used to be

- Nowadays, almost every electronic device has firmware that can be updated.
- Popularity ranking according to Google (8/2005):

+dvd +firmware	1.2M hits
+satellite +firmware	1.0M
+disk +firmware	***930k***
+phone +firmware	910k

- Not all hits are “officially supported”.

Reflashing for fun and profit

(lock-in vs. unlocking the true potential)

It’s all about business models.

- Time to market: ship it now, fix it later.
- Watch satellite etc. TV without paying.
- Re-enable wireless telephone features.
- Disable DVD player region locks.
- Upgrade camera to more expensive model.

Note, these are all special-purpose devices.

What about computer systems?

- Pentium CPU instruction set updates need to be “signed” and don’t survive power cycle.
- Limited number of types of system BIOSes, or embedded processors/OSes (as used in disks).
- Enough variation to make worm-like exploitation error-prone (potential for creating door stops).
- However, this won’t stop motivated individuals from updating firmware in specific machines.
- *Would not show up in disk images.*

Conclusion

- Main memory is a great source of forensic information. With infection of running processes or running kernels, main memory even becomes a primary source.

- Hardware is becoming softer all the time, as systems become more and more complex. When do we stop trusting a device to give us all the information that is stored on it?

Pointers

- Dan Farmer, Wietse Venema: “Forensic Discovery”, Addison-Wesley, Dec. 2004. http://www.porcupine.org/forensics/ http://www.fish2.com/forensics/
- Brian Carrier: Sleuthkit and related software. http://www.sleuthkit.org/
- Jim Chow *et al.*: “Shredding Your Garbage”, USENIX Security 2005; “Understanding Data Lifetime”, USENIX Security 2004.