CHAPTER 3

Sanitization and Visibility 1: Operating Systems

"Dumpster diving" is a time-honored technique for stealing confidential information and breaking into computer systems. The attacker simply waits until no one is looking then and rifles through the target's waste, seeking printouts, phone books, operations manuals, and any other kind of information that might be usable to accomplish his or her nefarious aims.

Stories of dumpster diving go back decades. Hackers in the Legion of Doom literally obtained telephone company manuals and passwords from dumpsters in the 1980s: with this information they penetrated computer systems and telephone networks.[SQ95] Dumpster diving has also been used by police to obtain information on suspects without the need to first obtain search warrants; the legality of this investigative technique was upheld by the Supreme Court in 1988.[US88]

This dissertation uses the term *sanitization* to refer to the intentional destruction of information on a computer system so that the information cannot be recovered by another party. The difficulty of removing data from media before the media is discarded or repurposed is an important part of the sanitization problem, but it is not the only part. There are many cases in which a user wishes to remove specific data from a computer that is in use without decomissioning the entire machine. For example, a person using a public computer at a library to access an Internet-banking site might reasonably wish to remove the information downloaded during the course of their web banking session. As we shall see in Chapter 4, today's web browser developers are aware of the need to provide tools for sanitizing information in such situations, but the tools that they provide are inadequate.

This chapter starts with an exploration of the sanitization problem. It discusses specific cases in which confidential information was compromised through sanitization failures, then presents the results of the *Remembrance of Data Passed* study to argue that these failures are widespread but hidden. The *Data Passed Traceback study* explores how the failures came about. The remaining two sections discuss responses by businesses and government, and finally presents solutions for resolving the data remanence problem.

3.1 Background

Ten used computers were purchased from a small-town computer store for $20 In August 1998 for the purpose of testing a telecommunications program under development. Most of the computers had been sitting on a shelf for more than a year and the store's owner didn't know if they even worked.

When the computers were turned on, it was discovered that the computer store had neglected to sanitize the hard drives prior to selling the machines. An examination of the information contained on the computers found the following:

- One of the larger machines, a 486-class system with a 40 gigabyte hard drive, had been a Novell file server used by a law firm. The computer still had confidential client material on it, including contracts, wills, and billing records.
- A second computer had been used by a community-based organization that delivered mental health services to residents under contract with a state agency. The computer included a FileMaker Pro database that had the names, addresses and diagnoses of several dozen individuals.
- A third machine apparently belonged to a writer who wrote for a national magazine and was working on a novel. This machine contained unpublished works, works-in-progress, and personal correspondence.
- A fourth machine had correspondence between a woman and her daughter in college. This computer also had a copy of Quicken, a personal finance management system from Intuit, which the woman apparently used to manage her finances.

All of this information was visible in plain view once the computers were turned on; no special disk recovery software was needed. A telephone call to the store's owner revealed that he knew the systems had confidential information on them and that he had meant to sanitize the machines before he sold them. The owner had simply neglected to do so.

At the request of the store's owner, the hard drives of the computers were sanitized using FreeBSD and the `dd` command.[1]

This experience was hardly unique. In recent years there have been repeated examples of such cases, including:

- In April 1997, a woman in Pahrump, NV, purchased a used IBM PC and discovered records from 2000 patients who had prescriptions filled at a Smitty's Supermarkets pharmacy in Tempe, AZ. [Mar97].
- In 2000, Sir Paul McCartney's banking details were discovered on a computer that had been discarded by the firm Morgan Grenfell Asset Management. The PC had been sold on the secondary market without being properly sanitized. [Ley04b]

[1] To sanitize an IDE hard drive with FreeBSD, the hard drive is jumpered to be a "master" and then connected to the computer's secondary IDE interface. The following command is then typed as root on the computer's console: `dd if=/dev/zero of=/dev/da3 bs=65536`. The procedure writes ASCII NUL characters over every block on the disk, making recovery of the original data impossible using techniques available in the open literature.[GS02a]

- In August 2001, more than 100 computers from the consulting firm Viant containing confidential client data were sold at auction by Dovebid following the closure of Viant's San Francisco office. [Lym01]
- In Spring 2002, the Pennsylvania State Department of Labor and Industry sold computers containing "thousands of files of information about state employees." [Vil02]
- In August 2002, a Purdue student purchased used Macintosh computer at equipment exchange; the computer contained FileMaker database with names and demographic information of 100 applicants to Entomology Department. [bBL02]
- Also in August 2002,the United States Veterans Administration Medical Center in Indianapolis retired 139 computers. Some of these systems were donated to schools, others were sold on the open market, and at least three ended up in a thrift shop where they were purchased by a journalist. Examination of the computer hard drives revealed sensitive medical information, including the names of veterans with AIDS and mental health problems. Also found were 44 credit card numbers used by the Indianapolis facility. [Has02]
- In May 2003 a reporter for *PC World* purchased 10 used hard drives in Massachusetts and found sensitive business and personal data including credit card and social security numbers on all but one. A hard drive sold by a computer store had been used by an accountant and had four years' worth of client payroll and tax information; the accountant's nephew had upgraded the computer and never told his uncle what became of the disk. A second disk purchased at the Salvation Army Store in Cambridge had belonged to an attorney and contained bank account numbers, draft legal documents, and an America Online installation with a stored password. The firm's IT consultant had promised the attorney that the information on the drive would be destroyed, but it wasn't.[Spr03]
- In June 2004, the UK computer security firm Pointsec purchased 100 hard disks on eBay as part of a project on the "lifecycle of a lost laptop." Although all of the hard drives had "supposedly" been "wiped-clean" or "re-formatted," the company was able to recover data from approximately 70 of the drives. The company also purchased laptops at auction that had been lost at airport terminals in the Germany, Sweeden, the UK and the US; it verified that police did not sanitize the laptops prior to selling them. Reportedly the laptop recovered from Sweden "contained sensitive information from a large food manufacturer. The info recovered included four Microsoft Access databases containing company and customer-related information and 15 Microsoft PowerPoint presentations containing highly sensitive company information."[Ley04b, Tec04]

In addition to these cases, we have collected anecdotal information which we believe to be accurate, but which has not appeared in previously published accounts:

- The Federal Witness Protection Program reportedly sold at auction a computer containing the original identities and current aliases of several hundred protected witnesses. Reportedly this snafu happened sometime during the 1980s. We learned of this incident in 1989 while producing a video about computer security with *Commonwealth Films*, a training firm in Boston.
- Sometime during the spring of 2000, employees of a Boston-based manufacturer of electronic equipment sent a workstation RAID array back to the vendor for warranty repairs. The work-

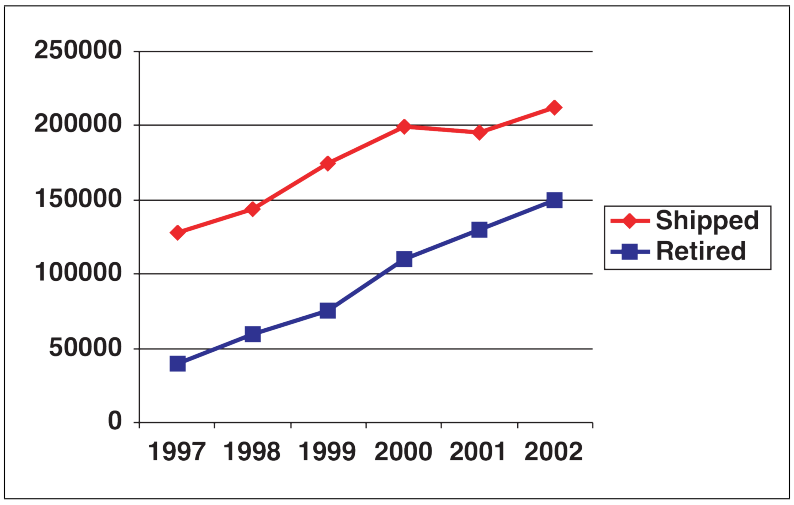

Figure 3-1: Hard drives shipped and retired between 1997 and 2002. Source: Dataquest.[Mon02].

station vendor sent the electronics firm a refurbished RAID array in return. Several months later, the engineers at the company received a call from a system administrator at a Massachusetts university: apparently the electronic firm's RAID array, repaired, had been sent to the university in exchange for one of the university's arrays (also, apparently, repaired under warranty). The workstation vendor had not made any attempt to sanitize or otherwise remove the information from the array before sending it to the university. We learned of this story from an employee at the electronics firm who received the phone call.

- After the publication of [GS02a], we received a telephone call from a woman who was president of a company that purchased computer equipment from the federal government at auction, refurbished the equipment, and sold it on the open market. She stated that she had frequently purchased lots at auction that contained classified materials—an apparent violation of federal law. Many times, she said, classification stickers were still on the computer systems that were packed into shipping containers and sold by weight to the highest bidder.

- In fall 2003, a student at the Harvard University Extension School purchased the hard disk from a cannibalized Macintosh computer at a Goodwill store in Massachusetts for $10. Upon copying the data off the disk the student discovered that it had been used at a small law firm and contained hundreds of client documents.

The story of the electronics corporation is particularly troubling: because the RAID array had malfunctioned, the company was not in a position of being able to sanitize the equipment before returning it to Sun. Instead, the firm's engineers had trusted the vendor, and this trust had apparently been misplaced.

According to the market research firm Dataquest [Mon02], nearly 150 million disk drives were retired in 2002—up from 130 million in 2001. Dataquest estimates that 7 disk drives will be retired for every 10 drives that ship in the year 2002; this is up from a 3-for-10 rate of retirement in 1997 (Figure 3-1).

Although many retired hard drives are in fact destroyed, the experience at the VA Hospital demonstrates that many drives that are "retired" by one organization can appear elsewhere. Indeed, the secondary market is rapidly growing as a supply source for even mainstream businesses, as evidenced by the cover story of the October 15th, 2002 issue of *CIO Magazine*, "Good Stuff Cheap: How to Use the Secondary Market to Your Enterprise's Advantage." [Ber02]

The anecdotes reported here are interesting both because of their similarity and because of their relative scarcity. Clearly, confidential information has been disclosed through computers sold on the secondary market more than a few times. Why, then, have there been so few reports of unintended disclosure?

[GS02a] proposes three possible answers to this question:

- Disclosure of so-called "Data Passed" information, while it occurs from time-to-time, is nevertheless exceedingly rare.
- Confidential information is disclosed so often on retired systems that such events are simply not newsworthy.
- Used equipment is awash with confidential information, but nobody is looking for it—or at least, few people who are looking for this data are publicizing the fact.

This chapter argue that the third hypothesis is correct; this conclusion is supported with data from the "Traceback study" presented in Section 3.4.

3.2 The Problem of Discarded Data

A fundamental goal of information security is to design systems that prevent the unauthorized disclosure of information that has been declared *confidential*. Traditionally this property was imprecisely referred to as *privacy*; in Section 2.3.1 we adapted the term *disclosure control*.

There are many ways for computer systems to provide disclosure control. One of the oldest and most common is physical isolation or physical access control. Confidential data can be kept on computers that are only accessible from authorized locations, and conventional security mechanisms, such as doors, locks and keys, are used to secure these locations. Even today, many personal computers use physical isolation as their primary means of disclosure control. On many small computers, such as cell phones and PDAs, physical access control is the *only* means of disclosure control that is employed.

Computer systems that can be used by more than one person typically rely on authentication and access control lists to provide disclosure control. Much of information security research over the past thirty years has centered upon improving techniques for authenticating users and then assuring that those users do not overstep their predetermined privileges.

Cryptography is another tool that can be used to provide disclosure control. Data can be encrypted as it is sent from one system and decrypted at its destination—for example, by using the SSL encryption protocol. Information that is stored on a computer's disk can be encrypted so that it will be inaccessible to processes or individuals who do not possess the appropriate key. Cryptographic

file systems [PGP98, Bla93, Mic02, Com05b, LK01] ask for a password or key on startup, after which they automatically encrypt data as it is written to the disk and decrypt the data as it is read; if a disk is stolen the data will be inaccessible to the thief. A surprising amount of work has been done on both academic and commercial file systems, and such file systems are widely available today—they are built into Windows XP and MacOS 10.3, for example. Nevertheless, it is widely believed that these tools are rarely used by the general public.

In the absence of cryptographic protections, confidential information on a disk can be readily disclosed if the disk is retired in an improper manner. The National Computer Security Center notes that this so-called *data remanence problem* has been recognized since the 1960s. [Gol91]

3.2.1 Historical basis for the data remanence problem

Although it is a common belief that operating system developers did not deploy a sanitizing file deletion in the 1970s and 1980s so that accidentally deleted files could be recovered using special tools, there are no references to support this claim. Indeed, had recoverability of accidentally deleted data been a goal, companies like Microsoft and Apple would surely have distributed such tools themselves—either as part of their operating system offerings or as after-market additions. (Some versions of DOS were distributed with an "UNFORMAT" command-line utility that could, in fact, unformat a disk, but this command appears to have been added relatively late in DOS history to exploit the format command's longstanding lack of sanitization; it is doubtful that the command was made explicitly non-sanitizing so that the UNFORMAT command could be written.)

Instead, it seems that the lack of a sanitizing delete-file is an accident resulting from the way that file systems evolved. Historically, multi-user computer systems did not need a sanitizing delete. Although today's computer users could benefit from the technology, this requirement was never made part of any formal file system specification—specifications that were largely written *after* hierarchical file systems were first developed, rather than before. Instead, today's storage systems are usually judged by other criteria, such as speed, reliability, availability, and compatibility.

The developers of the Compatible Time Sharing System (CTSS) at MIT in the 1960s did not consider the problem of sanitizing disk blocks after deleting files because the computer system frequently ran with disks that were full or nearly full: blocks that were freed were quickly overwritten with new data.[Sal04] The CTSS disk drives were rented from IBM and were never offered for sale on the secondary market. Indeed, the real data security problem was not that data on a disk returned for service might be accidentally sent to another IBM customer—the real problem was trying to keep data on the disks in the first place, as the drives were having head crashes every few days! In any event, while there was a general belief that CTSS should prevent one user from crashing a program being run by another user, overall the system did not have strong internal disclosure controls: the developers did not believe that CTSS was secure enough to store sensitive information.

Internal security between users was a design goal of the Multics operating system, but once again the designers never considered disk sanitization to be a priority. Multics did sanitize disk segments, the Multics equivalent of files, but it did so when the segments were allocated to a new process, not when they were released. Furthermore, early Multics systems were perennially short of disk space: once a disk block was freed, it would be quickly allocated to another process and, as a result, it would be quickly be sanitized.

It is widely acknowledged that Unix was developed in a research environment in which security was not a priority. For example, early Unix had no protections against denial of service attacks from authorized users. According to Ritchie, "In cases where denial of service attacks did occur, it was either by accident or relatively easy to figure out who was responsible. The individual could be disciplined outside the operating system by other means."[GS91, p.329]

When Unix first transitioned into the commercial world, it existed on systems that were run by trained system operators who would have been aware of the sanitization issue. Although Unix provided no specific tools for sanitizing disks, the `dd` command could be used for this purpose.

In the world of PC operating systems, an overwriting delete would have caused significant performance degradation on any operating system that did not have asynchronous access to the disk so that the sanitizing writes could have been performed concurrently with other tasks. Windows did not have such capabilities until 32-bit clean disk I/O drivers were available under Windows 95 and NT. Apple did not have such capabilities until it migrated to MacOS X.

3.2.2 The stability of hard disk data

In comparison with other mass storage media, hard disks pose special and significant problems in assuring long-term data confidentiality. One reason is that physical and electronic standards for other mass storage devices have evolved rapidly and in an incompatible fashion over the years, while the IDE/ATA and SCSI interfaces have maintained both forwards and backwards compatibility. Hard drives more than 10 years old can be easily read with modern computer hardware simply by plugging them in because these disks are both electrically and logically compatible. This high level of compatibility is one of the key factors sustaining both the formal and informal secondary markets for used hard drives.

Other kinds of storage media, including magnetic tapes, optical disks, and flash memory, have not shown such long-term stability. In these media there is considerably more diversity and more change: older media typically cannot be used with current readers due to physical changes. For example, a DAT IV tape drive cannot read a DAT I tape; a 3.5" disk drive cannot read an 8" floppy.

A second factor contributing to the data remanence problem in hard drives is the long-term stability of file system structures. Today's Windows, Macintosh and Unix Operating systems can transparently use the FAT12, FAT16 and FAT32 file systems developed by Microsoft in the 1980s and 1990s. Thus, not only are 10-year-old hard drives mechanically and electrically compatible with today's computers, but the data that they contain is readily accessible without special-purpose tools. This is not true with old tapes, which are typically written using proprietary backup software that may further employ proprietary compression and/or encryption algorithms.

3.2.3 Destroying information today

US DoD standard 5220.22-M[DoD95] specifies federally approved standards for sanitizing magnetic media that contain information that is sensitive but not classified:

- Physically destroy the drive, rendering the drive unusable.
- Degauss the drive, so that the magnetic domains are randomized—invariably rendering the disk drive unusable in the process (Figure 3-2).

Figure 3-2: A model HD-1TB disk and tape degausser manufactured by Data Security, Inc. One of the disadvantages of using this machine is that there is no way to visually inspect a drive and determine if it has actually been sanitized or not. Drives cannot be reused after they have been degaussed because the drive's operating system has been wiped and sensitive components on the drive's circuit board have been destroyed. Photo courtesy Data Security Inc.

- Overwrite the data on the drive so that the data cannot be recovered.

The National Security Agency/Central Security Service *Device Declassification Manual* specifies procedures for "clearing, sanitization, [and] declassification" of information stored on information storage devices ranging from Unclassified to Top Secret Codeword, including compartmented, sensitive, and limited-distribution material. According to that manual, overwriting can only be used for *clearing* a device; cleared devices cannot be declassified, but can be re-used within a secure environment. *Sanitization*, required for disk declassifying, can only be accomplished through the use of degaussing or incineration.[NC05]

Techniques for information destruction in an unclassified environment are complicated by social norms. Clearly, the most straightforward way to ensure the protection of information that a drive contains is to physically destroy the drive: this is the only technique that can be verified with casual inspection (see Figure 3-3 and Figure 3-4). But many people feel moral indignation when IT equipment is destroyed instead of being redirected towards schools, community organizations, religious groups, or lesser-developed nations that could benefit. As a result of such moral indignation, there now exist a plethora of organizations that help people find new homes for old computers[FTC05]—including some that ship these computers to rural villages in India.[Ass05]

3.2.4 The sanitization usability problem

The sanitization usability problem is one that pervades today's computer systems: when the user chooses a "delete" operation—for example, when deleting a file with Windows Explorer—often the information is not actually erased from the computer's recording media. Instead, the storage that is

Figure 3-3: **Drive Slagging** Following the publication of [GS02a], Dave Bullock, John Norman and "CHS" performed this demonstration of melting a hard disk with gas-fired furnace that they had built in a back yard. The authors concluded: "Drive slagging is a fool-proof method to prevent data recovery." *Photos used with permission.*

associated with the information is marked "free" or "available for use" and the specified information is rendered invisible and inaccessible from the user interface.

As a result of this efficient but non-intuitive behavior, today's computers frequently contain sensitive information that cannot be recovered using the tools that the computer itself provides. Frequently this is information that the computer's owner specified should be deleted, but which was not actually erased. This information can be recovered at a later point in time by an attacker who obtains physical access to the disk or has the ability to run a program on the computer which can access the raw device.

As a result of this sanitization usability problem, computer users have no readily apparent way other than physical destruction to determine if disposing of a computer system will jeopardize the security of information that was once stored on that system but was subsequently "deleted."

Disks, hidden data, and file systems

Broadly speaking, modern disk drives have the ability to store two kinds of information. The majority of information stored by the device is *directly addressable user data*—these are the actual blocks that are written by the computer's operating system onto the drive's media in response to WRITE commands and read back in response to READ commands. The second kind of information is *hidden* data that is used for the proper operation of the disk drive itself. This information includes the

Figure 3-4: A hard drive that was punched with a new machine that is being developed by Charles Smith of Greenville SC as a result of having read [GS02a]. Although this approach is easy to audit, it is probably not sufficient for classified material. *Photo used with permission.*

disk's firmware and spare blocks that the drive will use when blocks containing directly addressable user data begin to fail.

When a drive is sold by a manufacturer all the blocks that will be used to hold directly addressable user data are, by convention, filled with the ASCII NUL character—that is, the blocks are zeroed. (Many of the hidden blocks are not zeroed, but they cannot be accessed by the computer's operating system: for most practical purposes, these blocks do not exist.) Before the disk can be used, it must be initialized for used with a particular file system.

A *file system* is the piece of a computer's operating system that controls the allocation of disk blocks to individual files. Popular file systems include FAT[2] (used by Windows 3.1, Windows 95, and

[2]FAT stands for File Allocation Table, a linked list of disk clusters that the DOS operating system used to manage space on a random access device. The number 16 or 32 refers to whether the FAT uses sector numbers that are 16 bits or 32 bits in length. See [Mic00] for more details.

Windows 98), the NTFS[3] (used by Windows NT, 2000 and XP), FFS[4] (used by BSD Unix), and EXT2FS (used by Linux). The following discussion is for the FAT file system, but it applies with only minor changes to all modern file systems.

FORMAT doesn't wipe clean

Microsoft operating systems use a command called `FORMAT.EXE` to establish a new file system on recording media. When a disk is formatted with the FAT file system, the `FORMAT.EXE` scans the entire disk, reading every block to make sure that the block is functioning. `FORMAT.EXE` next writes the operating system's *boot blocks*, the disk's root directory, and finally a *file allocation table* that is used to distinguish blocks that are in use by the file system from those that are not. This process typically takes between 10 and 20 minutes, owing to the time required to read every block on the drive. Modern versions of `FORMAT.EXE` also have the ability to perform a "quick format" which omits the media scan (Figure 3-6). In this case, the entire disk can be formatted in just a few seconds. Quick format appears to be the default when formatting removable USB drives.

And what if there was confidential information on the disk when it was formatted? Once the root directory is written, any information that was previously on the disk is rendered inaccessible. Most of the data is still present but it cannot be retrieved using the Windows file system because the files and directories of the disk cannot be reached by starting at the disk's now empty root directory.

The failure of `FORMAT.EXE` to zero or otherwise initialize a hard drive has an interesting history. The first version of DOS, MSDOS 1.0, only worked with floppy disks. At the time floppies were sold without any track or sector information on their magnetic surface and they needed to be "formatted" before they could be used. In the process of formatting the disk any bad blocks were detected and noted in the disk's FAT so that they would not be used to store data. If a floppy disk containing data was formatted, the information that it contained would necessarily be overwritten when the new track and information was written. Thus the initial meaning of "format" to PC users in 1981 was a process that initializes a piece of magnetic media, making it usable, and destroying any data that the media might contain in the process.

DOS 2.0 was the first version of DOS to directly support hard disk drives. With this version of the operating system, the behavior of `FORMAT.EXE` was subtly changed when a hard disk was being initialized. Because hard drives were sold pre-formatted, it was only necessary for the FORMAT command to literally write a set of properly formatted data structures onto the disk's logical blocks so that the disk could be used with the operating system. Because the disks of the time were not extraordinarily reliable and lacked internal bad-block management, `FORMAT.EXE` continued to scan the entire disk for bad blocks—a process that might take between 10 and 30 minutes. Thus, the FORMAT command gave the impression that it was overwriting the entire disk because it took a long time and because the resulting disk appeared to contain no data. But no such overwriting took place! Thus, not only did the modified `FORMAT.EXE` turn visible data into invisible data, it did so in a manner that was *misleading*. Equally misleading was the warning that the command displayed which gave the impression that all of the data was in fact being destroyed. These misleading operations have been faithfully replicated in each version of `FORMAT.EXE` and are present in the contemporary versions (Figure 3-5).

[3] NTFS stands for New Technology File System. This is a journaling file system developed by Microsoft in the 1990s.

[4] FFS is the Fast File System, developed by the University of California at Berkeley in the 1980s.

```
C:\>format d: /fs:fat32

WARNING, ALL DATA ON NON-REMOVABLE DISK
DRIVE D: WILL BE LOST!
Proceed with Format (Y/N)?y
Verifying 8056M
Initializing the File Allocation Table (FAT)...
Volume label (11 characters, ENTER for none)? test
Format complete.

    8,233,244 KB total disk space.
    8,233,244 KB are available.

        4,096 bytes in each allocation unit.
    2,058,310 allocation units available on disk.

           32 bits in each FAT entry.

Volume Serial Number is 7C74-AB16

C:\>
```

Figure 3-5: Format of an 8 gigabyte hard drive using the Windows XP `format` command. After the computer prints "Verifying 8056M" the computers spends 5 minutes reading every block of the hard drive. After the computer prints "Initializing the File Allocation Table" the computer spends 15 seconds writing out a new FAT and performing a few write tests throughout the disk to ensure drive integrity. Notice that the command asserts that "ALL DATA ON NON-REMOVABLE DISK DRIVE D: WILL BE LOST." In fact, the data is only "lost" to users who do not have copies of "unformat" utilities.

One possible explanation for the revised behavior of `FORMAT.EXE` in DOS 2.0's was that the non-overwriting format was a usability optimization: overwriting each block of the hard disk would have made the already time-consuming FORMAT operating take twice as long, because every block would have first had to have been written, then read. Besides, disk drives at the time came with a separate "disk utilities" floppy which could perform an operation called a "low level format" on the physical disk. The details of the "low level format" actually varied from manufacturer to manufacturer and from drive to drive, which would have made it difficult for the operating system to perform such an operation. Mueller's 1991 book *Que's Guide to Data Recovery* discusses the difference between the low-level format performed by these utilities and `FORMAT.EXE`'s so-called "high-level format." Mueller notes: "You can recover data—unformat—from a high-level format."[ME91, p.99] But despite the fact that such information was available to the technical community, it does not seem to have been readily disseminated among the general population of computer users.

Another possible explanation for the behavior of DOS 2.0 `FORMAT.EXE` is that it was industry practice at the time for programs that initialized file systems to not overwrite all directly addressable user data blocks. The Unix `newfs` command writes an inode table, a root directory, and a collection of "superblocks," but it does not sanitize the disk—behavior that remains present to this day. [RT78, MJLF84] Likewise, the Linux `mkfs` command which creates Linux ext2fs and ext3fs file systems does not overwrite the entire disk. All of these commands write metadata and a clean root directory to the disk, but they do not perform a systematic overwriting of all of the disk's remaining blocks.

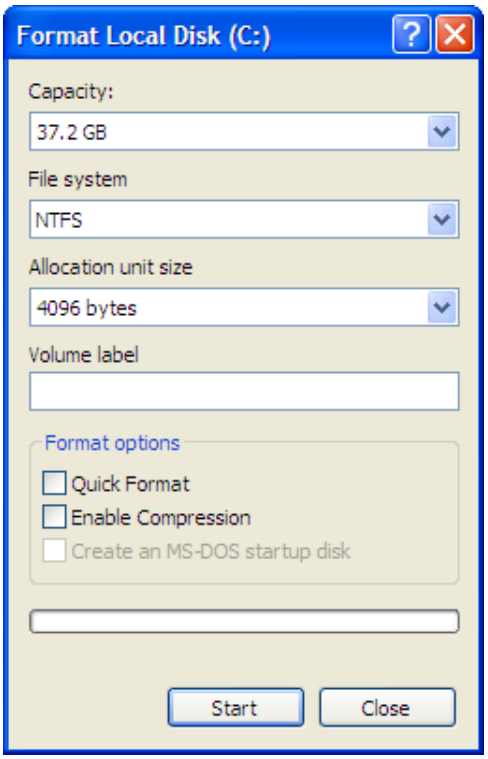

Figure 3-6: The Windows XP format panel has a "Quick Format" option which causes the program to just write a new file allocation table and a root directory on the media being formatted. If "Quick Format" is not selected the program takes considerably longer to format the disk because it is reading every block on the media to create a bad block table. It would be a simple matter for Microsoft to modify the format command so that every block on the disk is zeroed if "Quick Format" is unchecked.

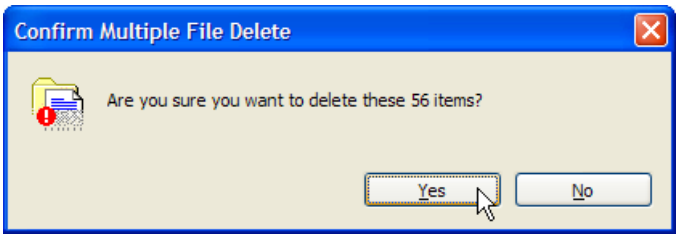

Figure 3-7: Microsoft Windows XP allows the user to confirm the deletion of files in the Recycle Bin; confirming the operation simply unlinks the files from the Recycle Bin directory; it does not actually remove the contents of the files from the computer's hard disk.

It is incredibly misleading for an operating system to give the impression that all of the information has been removed from a disk, when in fact the information has merely been made inaccessible to users who have not obtained special data recovery tools. Such a situation is an invitation for mishap: given a freshly formatted hard disk, there is no way for a user to audit the disk and determine if it is in fact clean, or if it has a treasure-trove of hidden, confidential information. Observations of this behavior and an analysis of resulting problems that it has caused are responsible for the USER AUDIT and COMPLETE DELETE design patterns discussed in Chapter 10.

Delete Doesn't Erase Information

Just as today's FORMAT command doesn't actually format disks, the commands for deleting individual files provided by today's computers do not actually perform that function, either. User-level commands such as `DEL`, `ERASE` and `rm` are implemented with calls to the Win32 `DeleteFile()`

or the POSIX `unlink()` system calls. But the `DeleteFile()` and `unlink()` system calls don't actually delete information—instead, they literally remove the link between the file's name in the containing directory and the disk blocks that the file occupies. Under the FAT and NTFS file systems, which support but a single link between directories and file contents, this results in the file's blocks being immediately returned to the free list. On Unix the `unlink()` system call decrements the file's link count; if the link count drops to 0, the blocks are returned to the free list.[RT78] But the blocks are not actually overwritten until they are needed again. System simulations done by Chow *et al.* indicate that such data may never be overwritten on a typical computer.[CPG+04]

Once again, the usability problem is that the operating system gives the user the *appearance* that the data has been removed from the computer, when it fact the data has merely been made *inaccessible by ordinary means*.

The usability problem for end-users is compounded by the fact that there is no mention of this behavior in either the end-user or developer documentation that is provided by either Microsoft or the Unix operating system. Developer documentation might be a particularly effective technique to get this information into the community of computer users, as developers are probably more likely to read documentation than users, and they are better poised to create work-arounds. But the Microsoft Developer Network documentation for `DeleteFile()` merely states that the function "deletes an existing file."[Net05b] The Unix documentation for the `unlink()` system call notes "If that decrement reduces the link count of the file to zero, and no process has the file open, then all resources associated with the file are reclaimed."[BSD93] In both cases, developers would be well-served by even a sentence mentioning the data remanence problem.

As with FORMAT, there was no conspiracy to keep secret the fact that DELETE doesn't actually erase the data that the user has targeted for destruction—a 1987 advertisement for the Mace Utilities appearing in *The New York Times* noted that the $59.95 program's functions included the ability to "Unformat, Undelete, Diagnose & Remedy."[Adv87, p.57] Users reading this advertisement in 1987 could have reasonably inferred that erased files could be recovered. But mention that files could be undeleted did not appear in a feature article in *The New York Times* until 1990, and then only in Peter Lewis' "Executive Computer" column on the 11^{th} page of the Business section. [Lew90]

Commands that claim to overwrite don't actually overwrite

Many modern applications support a so-called *document-based framework* in which opened documents can be saved under their original names ("Save...") or different file names ("Save As...") When a file is saved under the name of an existing file, the end result is that the existing file is deleted and the file being edited appears to have replaced it.

There are many ways to implement the "Save As..." command. One standard approach is to save the new file to the disk under a temporary name. The original file is then renamed to a second temporary name. The new file is then renamed to the name of the first file, and finally the first file is unlinked. This multi-step sequence ensure that the original file is not removed from the file system until the new file is safely in its place with the correct name. (The procedure is slightly more complicated when one or more backup files are kept.)

Despite the fact that most applications appear to follow the sequence outlined above, both the Macintosh and Windows operating systems have user interface elements that imply otherwise. The doc-

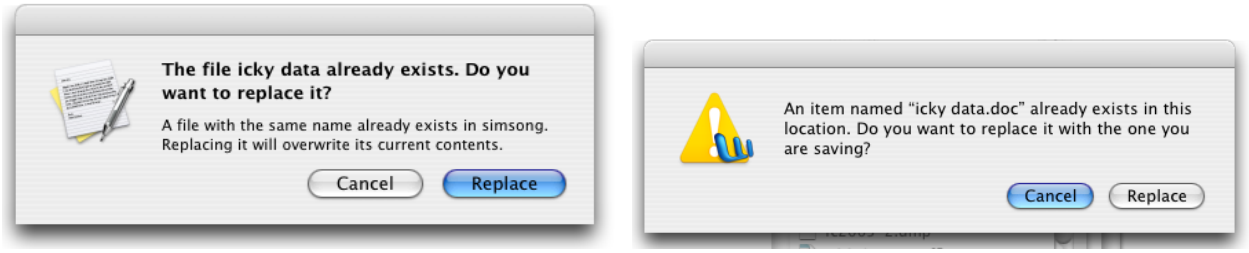

Figure 3-8: MacOS 10.3's Save As panel promises that saving a file with a name that is currently in use will result in the existing file being overwritten. In fact, the blocks of the second file will not be overwritten, but will be specifically preserved using the algorithm that the operating system implements. Interestingly, attempts to save a file named "icky data" over another file by the same name produced somewhat different responses in Apple's Text Edit (left) and Microsoft Word (right) applications.

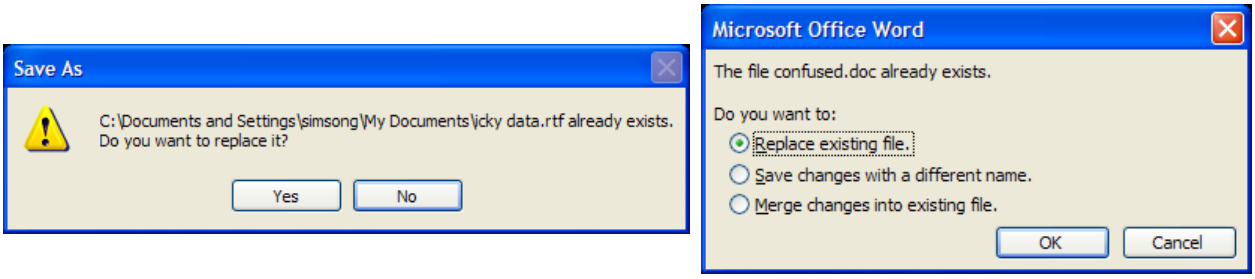

Figure 3-9: The Save As... panels in Windows Wordpad (left) and Word (right) also strongly imply that the data on the disk will be overwritten or merged into. In fact, the original documents are deleted but left on the disk, and a new file is created for the new document.

ument framework that is part of Apple's Cocoa environment specifically states that the "Save As..." command will "replace" the original file with the new file, and that "Replacing it will overwrite its current contents." (Figure 3-8, left) Microsoft Word on the Macintosh uses different terminology (Figure 3-8, right), but the implication is similar. The Wordpad program on Microsoft Windows makes a similar promise (Figure 3-9, left). Microsoft Word 2003 takes a different approach: the program offers to either replace the existing file or merge the changes from the current file into the existing file. It appears that the command is implemented by creating a new file and then deleting the old, so the original file's contents nevertheless remain accessible to those who are willing to perform a forensic analysis.

3.2.5 The overwriting question

In the previous section, the term "overwrite" was used without qualification. This section discusses what kind of overwriting might be sufficient for proper sanitization.

It has long been hypothesized that data stored on a magnetic media that is overwritten with a single pass of new information can be recovered by a determined and well-funded individual or organization. For example, it is commonly asserted that the National Security Agency has the capability to recover overwritten information.

The DoD sanitization standard specifies the following procedure for overwriting:

> "Overwrite all addressable locations with a character, its complement, then a random

> character and verify. THIS METHOD IS NOT APPROVED FOR SANITIZING MEDIA THAT CONTAINS TOP SECRET INFORMATION."[DoD95, Capitalization in original]

The verification step is important to ensure that the hard drive is actually writing the random data to the recording surface. This is to protect against hostile drives that claim to be sanitizing, but in fact are not.

Gutmann considers the question of recovering overwriting data at length:

> "In conventional terms, when a one is written to disk the media records a one, and when a zero is written the media records a zero. However the actual effect is closer to obtaining a 0.95 when a zero is overwritten with a one, and a 1.05 when a one is overwritten with a one. Normal disk circuitry is set up so that both these values are read as ones, but using specialized circuitry it is possible to work out what previous "layers" contained. The recovery of at least one or two layers of overwritten data isn't too hard to perform by reading the signal from the analog head electronics with a high-quality digital sampling oscilloscope, downloading the sampled waveform to a PC, and analyzing it in software to recover the previously recorded signal. What the software does is generate an "ideal" read signal and subtract it from what was actually read, leaving as the difference the remnant of the previous signal. Since the analog circuitry in a commercial hard drive is nowhere near the quality of the circuitry in the oscilloscope used to sample the signal, the ability exists to recover a lot of extra information which isn't exploited by the hard drive electronics (although with newer channel coding techniques such as PRML (explained further on) which require extensive amounts of signal processing, the use of simple tools such as an oscilloscope to directly recover the data is no longer possible)."[Gut96]

PRML stands for Partial-Response Maximum-Likelihood encoding, a technique that is similar to the encoding done by V.32 modems. According to Gutmann, this technique allowed the hard drive industry to increase drive capacities by 30–40%. The higher density is believed to make the recovery of overwritten data significantly harder. EPRML is Extended PRML, which included aerial density of recorded data between 20% and 70% above existing PRML.[Koz04]

Gutmann goes on to present a series of patterns which, when written to a magnetic drive, should dramatically decrease the chances that overwritten data could be recovered. Different patterns are presented for different recording technologies. Gutmann notes that it is theoretically harder to recover data as the density of magnetic media has increased and encoding has become more complicated. An epilogue added to online version of the 1996 paper concludes:

> "For any modern PRML/EPRML drive, a few passes of random scrubbing is the best you can do. As the paper says, "A good scrubbing with random data will do about as well as can be expected. This was true in 1996, and is still true now."[Gut96]

```
OVERWRITE-FILE(filename):
    len = LENGTH-OF-FILE(filename)
    OPEN filename FOR WRITING
    SEEK TO THE BEGINNING OF filename
    WRITE len RANDOM BYTES TO filename
    CLOSE filename
```

Figure 3-10: Pseudocode for overwriting the contents of a file.

Required support for overwriting individual files

Overwriting requires underlying software and hardware support to ensure that the intended data is actually overwritten. When trying to sanitize an entire drive, for example, it is important that the disk makes all of the directly addressable blocks available for writing. Modern ATA disk drives have the ability to create a password-protected "host protected area." If such an area is created, attempts to sanitize the disk drive using overwriting may miss key areas.

Sanitizing individual files through overwriting will only be successful if the underlying operating system is rather literal in its implementation of the `seek()` and `write()` system calls. Overwriting an individual file is commonly implemented with an algorithm similar to the one presented in Figure 3-10. The problem with this approach is that it implicitly assumes that when the contents of a file are overwritten the operating system overwrites the physical blocks that hold the file's contents. Although this is the case with most implementations of the FAT, FFS and EXT2FS file systems, it is not the case with other file systems. For example, file systems that provide the appearance of read-write access to write-once media can do so by writing data to new blocks, then rewriting metadata, and finally by rewriting a new root directory.[GL85, Gar91] Attempts to sanitize files on these systems through overwriting will fail.

Even file systems for rewritable media may not provide the necessary guarantees to reliably provide sanitization through overwriting. A journaling file system such as Microsoft's NTFS or Apple's HFS+ with journaling enabled may intentionally avoid overwriting old information with new information in order to provide reliability guarantees. A high-performance file system might further take advantage of a data write as a chance to unfragment a fragmented file, if consecutive blank blocks are available at the time of update. This argues that the functionality for sanitizing should be provided directly by the file system, and not through the manipulation of other system calls with the hope that sanitization will result as a side-effect.

3.3 Case Study: *Remembrance of Data Passed*

An important aspect of the sanitization problem that had not previously been subject to academic study is the prevailance of confidential information on hard drives that are sold by consolidators, resellers and other kinds of commercial scavengers on the secondary market.

3.3.1 Acquisition of hard drives and data

A total of 217 hard drives were purchased on the secondary market between January 1999 and April 2002. Primary sources for these drives were in-person visits to used computer stores, the MIT "Swapfest," and by winning bids on the eBay online auction web site. Efforts were made not to purchase more than 10 drives from a single reseller at any time. Most lots consisted of between 2 and 5 drives.

Upon receipt, each drive was given an accession number. This number was then used as the drive's identifier for all further work. An additional 20 drives (#99–#110, #112-#114, #116-#118 and #120–#121) were purchased on the secondary market by another researcher for an unrelated project. Those drives were imaged and then returned to that researcher.

A list of the hard drives involved in this study appears in Appendix A.

3.3.2 Drive imaging

Once a drive was cataloged, the next step was to *image* the drive. Imaging is a process that involves copying all of the drive's data into a single file, not surprisingly called an *image file*. Once the image file is created, all subsequent analysis can be done with the image file and the drive itself can be put in storage.

There are many advantages to working with image files instead of the actual disk drives:

- Modern disk drives are considerably faster than older drives. Once an image is made, it is dramatically faster to work with the image than to continually refer back to the original file.
- Modern disk drives are considerably more reliable than older drives. Many of the drives were in fact failing as they were imaged: in several cases the disk's internal mechanism was not entirely operational when the imaging operation was concluded.
- It is possible to have only a few ATA drives connected to a computer at once. On the other hand, it was possible to have *all* of the disk images resident on multiple computers at the same time.

Imaging was performed on a PC workstation running the FreeBSD operating system and using the Unix `dd` command to copy data from the raw ATA device (in the case of IDE/ATA disks) and from the raw SCSI partition (in the case of SCSI disks) into a single file. The `dd` options "noerror" and "sync" were specified; "noerror" tells the `dd` command to continue even if an error is detected. The "sync" option tells the command to keep the output file in sync with the input file by padding the output file with NULs when read errors are detected. The blocksize was set to 65536 bytes to speed transfer. The image was saved in a file called *driveid*.img. Figure 3-11 shows a typical `dd` command.

One way that the imaging process could be improved would be to modify the `dd` command so that sync error blocks, instead of being filled with NULs, would be filled with some specific and unusual pattern so that it would be possible for forensic analysis tools to tell the difference between blocks that had been read as all NULs and blocks that could not be read.

After the image was created, the FreeBSD `fdisk` command was used to create a human-readable

```
# dd if=/dev/ad2 of=/project/images/100.img conv=noerror,sync bs=65536
```

Figure 3-11: The "dd" command used to image drive #100.

```
# mdconfig -a -t vnode -f /big3/project/images/img/100.img -u 1
# mount -t msdos /dev/md1s1 /mnt
```

Figure 3-12: The FreeBSD commands to mount disk image #100 with the MSDOS file system.

```
# umount /mnt
# mdconfig -d -u 1
```

Figure 3-13: The FreeBSD commands to umount disk image #100.

printout of the disk's partition table. This image was saved in a file called *driveid*.fdisk. (This wasn't strictly necessary, as the fdisk command should have been able to use the raw disk image.)

At this point, the disk image was attached to the FreeBSD memory disk device and attempts were made to mount the image read-only using the FreeBSD FAT, NTFS, UFS, and Novell file system implementations (Figure 3-12). If the drive image could be successfully mounted, the files on the image were copied off using the Unix `tar` command. Finally, the disk file was unmounted (Figure 3-13).

Not surprisingly, a significant fraction of the drives were physically damaged, contained unreadable sectors, or were completely inoperable. These drives took substantially longer to image, as the drive electronics would repeatedly attempt to re-read the bad sectors and/or reset the drive's internal electronics. Where possible, partial drive images were collected.

In many cases disks were imaged but the filesystem could not be mounted. This may have been the result of a file system that was not supported by the FreeBSD operating system, a drive that was properly sanitized, or a drive that was physically damaged.

3.3.3 The Garfinkel/Shelat sanitization taxonomy

In order to facilitate the discussion of sanitization practices, Table 3-14 presents a *sanitization taxonomy*. This taxonomy can be used both to describe data found on recovered disk drives, and also to describe sanitization procedures. We have found this taxonomy extremely useful in describing matters relating to sanitization and forensic analysis.

3.3.4 Analysis of "data passed"

Analysis of the data imaged from the drives was performed using a variety of tools, including several written specifically for this project.

Level	Where Found	Description
Level 0	Regular Files	Information contained within the file system. Includes file names, file attributes, and file contents. The disk drive in the stolen laptop contains many Level 0 files.[a] No special tools are required to retrieve Level 0 data.
Level 1	Temporary Files	Temporary files, including print spooler files, browser cache files, files for "helper" applications, files in "recycle bins." Most users either expect that these will be automatically deleted in time or they are not even aware that these files exist. In fact, sometimes these files *are* automatically deleted over time. No special tools are required to retrieve Level 1 data, although special training is required so that the operator knows where to look. Level 1 files are a subset of Level 0 files. Experience has shown that it is useful to distinguish this subset, since many naive users will overlook Level 1 files when they are browsing a computer's hard drive to see if it contains sensitive information.
Level 2	Deleted Files	When a file is deleted from a file system, most operating systems do not overwrite the blocks on the hard disk on which the file is written. Instead, they simply remove the reference to the file from the containing directory. The file's blocks are then placed on the free list. These files can be recovered using traditional "undelete" tools such as Norton Utilities.
Level 3	Retained Data Blocks	Data that can be recovered from a disk but which does not obviously belong to a named file. Level 3 data includes information in slack space, backing store for virtual memory, and Level 2 data that has been partially overwritten so that an entire file cannot be recovered. One common source of Level 3 data is disks that have been formatted with Windows "Format" command or the Unix "newfs" command. Even though these commands give the impression that they overwrite the entire hard drive, in fact they do not, and the vast majority of the information on a formatted disk can be recovered with Level 3 tools. Level 3 data can be recovered using advanced data recovery tools that can "unformat" a disk drive, and using special-purpose forensics tools.
Level 4	Vendor-Hidden Data	This level consists of data blocks on the drive that can only be accessed using vendor-specific commands. This level includes the ATA "Host Protected Area" as well as the drive's controlling program and blocks used for bad-block management.
Level 5	Overwritten Data	Many individuals maintain that information can be recovered from a hard drive even after it is overwritten. Level 5 is reserved for such information.

[a] By definition, there has been no attempt to sanitize the information that is contained within Level 0 files. Level 0 also includes information that is written to the disk as part of any sanitization attempt. For example, if a copy of Windows 95 is installed on a hard drive in an attempt to sanitize the drive, then the files contained within the C:\WINDOWS directory would be considered Level 0 files.

Figure 3-14: A Sanitization Taxonomy, from [GS02a]

Block level analysis

For every drive image a program was run that computed the following information and stored the results in the database:

- Number of image blocks.
- Number of image blocks that were filled with NULs.
- The MD5 hash code of the image.

The count of blocks and zeroed blocks made it easy to find the disks that had been properly sanitized by zeroing all of the drive blocks. A list of these drives appears in Appendix A.

The MD5 hash code of the image was useful for integrity checking on the disk images from time-to-time. (Ghemawat *et al.* report that the error rates of consumer disk drives become significant when large amounts of information are copied, and recommend that applications or file systems perform

```
Invalid partition table
Error loading operating system
Missing operating system
MS-DOS_6   FAT16
Non-System disk or disk error
Replace and press any key when ready
```

Figure 3-15: Phrases that are commonly found in the boot blocks or partition table of disks formatted with the DOS or Windows operating systems.

their own integrity checks in these situations.[GGL03])

Disks that had been sanitized by writing random data on them from start to finish could be readily identified by the fact that these disks would contain no blocks consisting entirely of ASCII NUL characters. No such disks were found in the collection—that is, every disk was found to contain a significant number of completely blank blocks.

File level analysis: levels 0 and 1
Analysis of Level 0 and Level 1 files was performed exclusively using the information in the disk tar files (see Section 3.3.2).

For each of these files, an automated process unpacked the archive in a clean directory. Each of the files was then examined and the following information was stored in the database:

- File name and complete path name.
- File length
- File MD5 hash code
- File type (extension)
- Output of the Unix `file` command when run over the file.[5]

Information such as "file type" made it possible to rapidly find all of the Microsoft Word files in the collection, while the MD5 codes made it possible to rapidly distinguish the Word files that were on many disks (for example, template and tutorial files) from Word files that had been created by end users. This proved to be important in the Traceback study, discussed in Section 3.4.

In this study, the concept of a "unique file" proved to be useful. A unique file was defined to be a file whose MD5 was not seen in any other file or any other collection of MD5 codes that was obtained from any source. The hypothesis is that such unique files correspond to content that was created by the computer that used the hard drive in question, and is unlikely to have been part of a standard distribution of files from an external source—for example, a list of tutorial files that were delivered as part of a Microsoft Word installation.

[5]The Unix `file` command reports file type by an examination of the file's name and file contents. For certain kinds of files it can report additional information—for example, the width and height of various image formats.

File Type	# of unique files
GIF files	10,012
GIF in web browser cache	8,262
Dynamic Linked Library	7,751
Program Files (COM & EXE)	4,548
JPG files	2,958
JPG in web browser cache	2,345
Microsoft Word	783
Microsoft Excel Worksheets	184
Microsoft Outlook	69
Microsoft PowerPoint	30

Table 3.1: The number of unique Level 0 and Level 1 files found on the hard drives in the study. Although large numbers of unique images files were found, they were mostly confined to the web browser caches. The large number of program files found indicates that programs were generally not deleted before the drives were sold on the secondary market. But the fact that so few numbers of Microsoft Word, Excel, and Outlook files were found indicates that these files were intentionally deleted before the drives were sold.

Table 3.1 provides the number of unique files of various file types that were found on each drive, by document file type. We were surprised that so few drives appeared to contain unique files. When the study was started, it was expected that the majority of the drives obtained on the secondary market would be completely unsanitized. But this wasn't what we found. We found roughly 10,012 unique GIF files, but 8,262 were in web browser caches—making them actually hidden Level 1 files, files that were not obvious to most users. We found 7,751 unique DLL files and 4,548 unique COM and EXE files, indicating that a wide variety of programs had been installed on these systems. But when focusing on Microsoft Word files, we found only 783 unique files on all of the drives—and 484 of those were contained on just four drives of the sample.

The only reasonable explanation fitting this data is that many of the disks were manually purged before they were sold. That is, some person manually went in and deleted the Microsoft document files but left behind the program files. As we shall see shortly, the deleted document files were nevertheless recoverable, as they had been deleted with the Windows "DEL" command,

File Level Analysis: Levels 2 and 3

There are many programs on the market for doing analysis of Level 2 and 3 data. One such product category consists of data recovery programs that are sold to consumers and businesses to recover accidentally deleted information. A second category are the forensic analysis tools that are typically sold to law enforcement agencies for the purpose of performing detailed analyses of seized hard drives. Although many of these tools are beloved by their target audience, they were deemed inappropriate for the purpose of this project: all of these tools have an interaction model that assumes a practitioner has a lot of time to spend with a single hard drive image. What was needed for this project was a batch tool that could rapidly analyze and assess hundreds of disk images.

The program `fatdump` provides this functionality in the form of a *Forensic File System* (FFS). A forensic file system is a kind of semantic file system[GJSJ91] that is specially tailored to ease in the retrieval of forensic information.

For example, the FFS allows any block on the disk to be accessed as if it were a file by using the path name `/b:nnnn`, where `nnnn` is the number of the block to be accessed. The notation `/c:nnnn` allows the contents of cluster *nnnn* to be accessed in a similar manner. (Clusters are collections of blocks referenced by the FAT file system; the mapping of cluster numbers to block numbers requires first decoding information stored in the disk's BIOS Parameter Block (BPB).)

The FFS also makes it possible to interpret any block as if it were a directory. In traditional file systems it is only possible to resolve path names that start at the root directory. The forensic file system allows the use of the `/dir:nnnn` notation. If cluster #50000 is believed to contain a directory, it is possible to list the files referenced from that block by requesting a listing of the directory `/dir:50000`. Figure 3-16 illustrates this terminology. In this example, `fatdump` was used to list the contents of cluster #15 of image #113; the cluster probably was a directory that held an application. One of the directory entries, `?TEMP.000:del7111`, was deleted. Frequently entries that have been deleted nevertheless point to valid directory contents—which themselves have also been deleted. Figure 3-17 shows that the directory a cluster #411 of disk #113 was a directory that contained a significant part of data files used by the Microsoft Office Shortcut Bar.

`fatdump` allows the forensic notation to be used as both an input to commands and in directory lists. If a disconnected directory is listed by specifying its starting block number, for example, the file names that are displayed will themselves be reported with the disconnected form. The notation makes it easy to reference Level 2 or 3 data that is on the disk's surface. `fatdump` uses this feature extensively: the program has a "-lR" option which generates a recursive list of all directories on the disk. Unlike the standard Unix `ls -lR` command, the `fatdump` version of this command results in a scan of the entire disk image.

Finally, `fatdump` has the ability to tag each block of a hard drive image as "reachable" or "not reachable" from the image's root directory.

What the forensic analysis shows

Using `fatdump`, it is possible to answer both the question posed on page 105 and to relate this entire study to the topic of security and usability.

An analysis of the hard drives contained in the sample shows that the majority of operational drives contained data that had been deleted but that was nevertheless recoverable. The output of `fatdump` shown in Figure 3-18 is typical. This information, from image #182, shows that the disk contained resumes, letters to an admissions counselor, and other highly sensitive information.

However, when disk #182 is mounted on a Unix computer and examined using standard tools, none of this information is visible. Instead, all that is apparent are three files: `IO.SYS`, `MSDOS.SYS` and `COMMAND.COM`. This reason is that disk #182 was formatted before it was sold.

Disk #182 is a 2 gigabyte hard drive that contained approximately 1.8GB of information. None of this information would be seen by a person who had purchased the drive unless that person used a low-level disk editor or a forensic tool to recover "deleted" files. In this case, it appears that the third answer the question on page 105 is the correct answer. Disk #182 was awash with information, but the information was invisible.

```
cluster 15 looks like a directory...
 03/18/1999  00:00   /:dir15/URL.DLL
 09/01/1998  11:17   /:dir15/COMPOBJ.DLL
 08/24/1996  12:11   /:dir15/MSVCRT20.DLL
 08/24/1996  12:11   /:dir15/WIN32S16.DLL
 08/24/1996  12:11   /:dir15/TYPELIB.DLL
 08/24/1996  12:11   /:dir15/STDOLE.TLB
 08/24/1996  12:11   /:dir15/OLE2CONV.DLL
 08/24/1996  12:11   /:dir15/OLE2NLS.DLL
 08/24/1996  12:11   /:dir15/OLE2DISP.DLL
 09/01/1998  11:18   /:dir15/OLE2.DLL
 09/01/1998  11:17   /:dir15/STORAGE.DLL
 08/24/1996  12:11   /:dir15/?EMP.000:del7111
 08/24/1996  12:11   /:dir15/DDEML.DLL
 08/24/1996  12:11   /:dir15/OLESVR.DLL
 08/24/1996  12:11   /:dir15/OLECLI.DLL
...
```

Figure 3-16: The output of fatdump on #113 reveals that cluster 15 was probably a directory that was part of a Windows installation.

```
 04/04/2000  09:52   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Off7290s.tmp
 04/04/2000  09:52   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Off7290h.tmp
 04/10/2000  10:16   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042s.tmp
 04/10/2000  10:16   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042h.tmp
 05/12/2000  10:15   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/OffA042.tmp
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft Access.lnk
 04/03/2000  14:56   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft Excel.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/Microsoft FrontPage.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/New Appointment.lnk
 07/30/1999  15:08   /:dir441/Application Data/Microsoft/Office/Shortcut Bar/Office/New Contact.lnk
```

Figure 3-17: Cluster #441 of disk #113 appears to have been a deleted directory that contained an entry for a directory named “Application Data.” That directory appeared to contain data from a Microsoft Office installation. This figure shows how the forensic file system allows a Level 3 directory to be followed to a Level 2 directory hierarchy.

Using `fatdump`’s ability to tag blocks, it is evident that Disk #182 is representative of many hard drives that were acquired for this study. In figrefsanitize/all-drives, each drive that has been properly sanitized or that contains file system structures that can be interpreted by `fatdump` are represented by a vertical bar. The top of each bar (light green) represents blocks on the drive that were cleared or properly sanitized. The brown section in the middle represents data, like the data on Disk #182, that existed but could not be seen with the operating system’s tools. The bottom part of each bar (light gray) represents the data that was accessible from the file system.

The bars that are mostly white represent disks that were removed from service and sold without much attempt at all to delete confidential files. The bars that are entirely green are disks that were properly sanitized. But the bars that are mostly brown are disks that someone *tried* to sanitize—but the sanitization tools failed them. Enough of the pointers to the data was removed such that the data would not be visible on casual inspection. But the data was still there—and could compromise security or privacy if the disk were in the hands of a suitably skilled individual.

```
06/19/1999  01:36   /:dir210216/Four H Resume.doc
03/31/1999  12:41   /:dir210216/U.M. Markets & Society Advisor.doc
03/29/1999  18:14   /:dir210216/UM Activities & Academic Coordinators.doc
08/27/1999  16:39   /:dir222270/Resume-Deb.doc
03/31/1999  13:11   /:dir222270/Deb-Marymount Letter.doc
03/31/1999  16:56   /:dir222270/Links App. Ltr..doc
08/27/1999  16:36   /:dir222270/Resume=Marymount U..doc
03/31/1999  13:40   /:dir222270/NCR App. Ltr..doc
03/31/1999  13:42   /:dir222270/Admissions counselor, NCR.doc
08/27/1999  16:35   /:dir222270/Resume, Deb.doc
03/31/1999  17:14   /:dir222270/UMUC App. Ltr..doc
03/31/1999  21:51   /:dir222270/Ed. Coordinator Ltr..doc
03/31/1999  23:27   /:dir222270/American College Advisory Svc. Ltr..doc
04/01/1999  12:51   /:dir222270/Am. U. Admin. Dir..doc
04/01/1999  13:06   /:dir222270/Project Assoc.,School Health Policies.doc
04/05/1999  22:38   /:dir222270/IR Unknown Lab.doc
04/06/1999  23:53   /:dir222270/Admit Slip for Modernism.doc
04/07/1999  18:43   /:dir222270/Your Honor.doc
04/08/1999  13:44   /:dir222270/Air & Space Ltr..doc
04/09/1999  15:57   /:dir222270/AIU App. Ltr..doc
```

Figure 3-18: Deleted documents that could be recovered from Disk Image #182

```
% ls -l /mnt
drwxr-xr-x  1 root   wheel       0 Dec 31  1979 .
-r-xr-xr-x  0 root   wheel  222390 May 11  1998 IO.SYS
-r-xr-xr-x  0 root   wheel       9 May 11  1998 MSDOS.SYS
-rwxr-xr-x  0 root   wheel   93880 May 11  1998 COMMAND.COM
%
```

Automatic identification of disk owners through statistical means

There are a variety of statistical techniques that can be applied to the data in the disk image without regard to the structure of the disk's metadata. Such techniques are useful in cases where large portions of the disk are unreadable or where the disk's file system is not supported by available forensic tools.

Two very useful statistical techniques were created during the course of this research: a Credit Card Number Detector and an Email Histogram Tool.

The **Credit Card Number Detector**, developed with Abhi Shelat, scans the disk image for strings of ASCII digits that match the Credit Card Verification (CCV) algorithm.[Sti97] Because the CCV is a single digit checksum designed primarily to catch digit transpositions, 10% of all randomly chosen 15-digit numbers will pass the verification. This proved to be a problem: because many TIF images are coded as hexadecimal binary data, such blocks of data have many cases of 15-digit numeric strings that satisfy the CCV. Shelat was able to obtain significantly higher accuracy by coding into his detector a set of valid credit card number prefixes that he was able to find on the Web. The

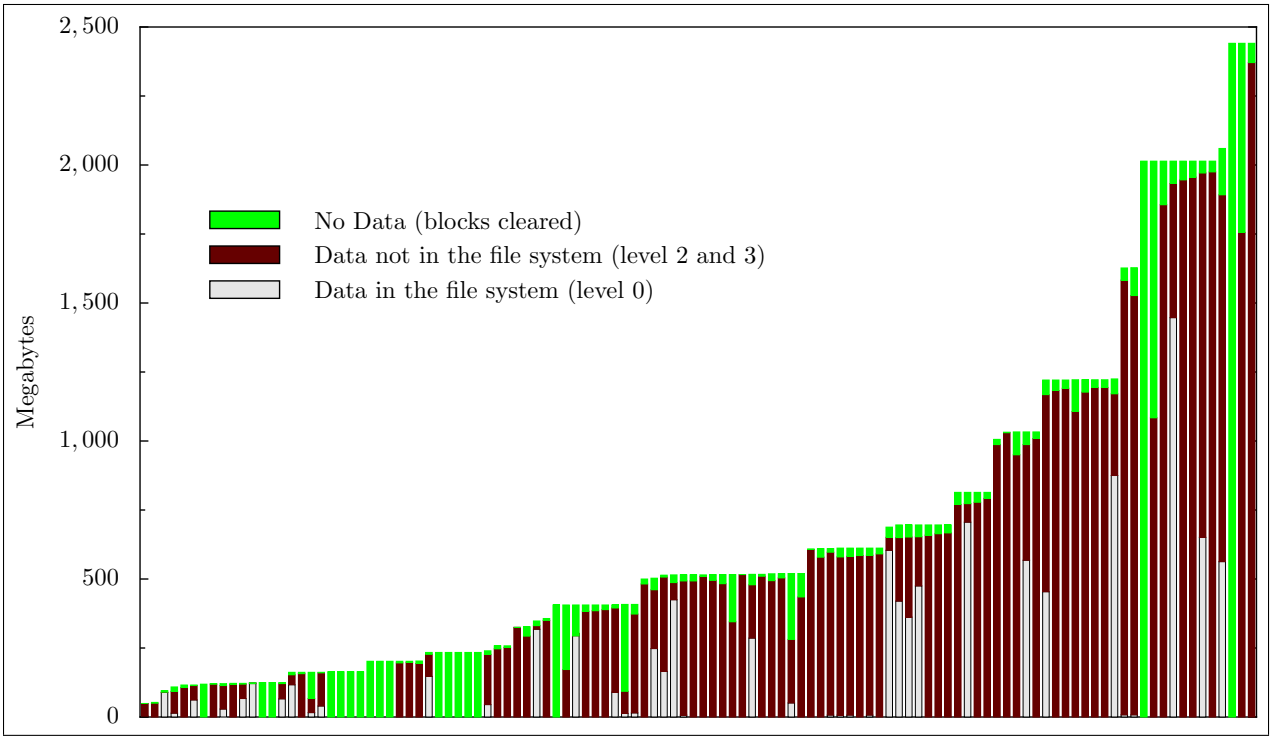

Figure 3-19: This graph depicts all of the remnant data found on the 114 hard drives that contained mountable FAT32 filesystems. Each bar has three parts: the bottom (light grey) part indicates the number of megabytes on the drive that could be reached starting at the root of the FAT32 file system—that is, the Level 0 data. The top of each bar (light green) indicates the number of megabytes of sectors that were all NULs that were found on the drive. The middle section (dark red) indicates the data that was on the drive but not reachable from the root directory—that is, the Level 2 and Level 3 data. As can be seen, while there are some drives that were completely erased, the majority of the drives contained large quantities of deleted-but-recoverable data, and some drives contained significant quantity of data that had not even been deleted!

resulting program can reliably find disk images that contain credit card numbers with only the occasional false positive.

The list of hard drives that contained large numbers of credit card numbers proved to be incredibly useful when conducting the Traceback study. Several organizations that probably would never have returned telephone calls became positively receptive and eager to help in the study when they were told that their old hard drives had been recovered and that the disks contained hundreds or thousands of customer credit card numbers.

The **Email Histogram Tool**, developed with Ben Gelb, scans the disk image for strings that appear to be email addresses. Each email address is then tabulated, and the top most common email addresses are displayed for the operator. We hypothesized that the most common email address on a hard drive would correspond to the individual who was the primary user of the computer from which the hard drive was removed. This is because this individual's email address would be present in messages sent both *to* the individual and those email messages sent *by* the individual.

Being able to rapidly identify the hard drive's primary user was also incredibly useful in the Trace-

back study. With this information we could quickly distinguish personal documents pertaining directly to that individual from the multitude of other information present in the disk image.

3.4 The Traceback Study

Several possible explanations for the large number of drives found to contain passed data were proposed in [GS02a]:

1. **Lack of knowledge.** The person disposing of the device simply fails to consider the problem.
2. **Lack of concern.** The person knows about the problem, but just doesn't care.
3. **Lack of concern for the data.** The person is aware of the problem, but is not concerned by the possibility that the data might be revealed.
4. **Failure to properly estimate the risk.** Although aware that data might be recovered, the person thinks that it is very unlikely that their particular data will be recovered.
5. **Despair.** The person disposing of the device is aware of the problem, but thinks that it cannot be solved.
6. **Lack of tools.** The person is aware of the problem, but doesn't have tools that will properly sanitize the device.
7. **Lack of training or incompetence.** The person takes measures to sanitize the device, but those measures are ineffectual.
8. **Tool error.** The tool does not behave as advertised. As most sanitization tools have not been evaluated and certified, this is actually a significant risk.
9. **Hardware failure.** The computer in which the hard drive resides may be broken, making it impossible to sanitize the hard drive without removing it from the computer and installing it in another one—a time-consuming process. (Hardware failure was apparently the case in the case of the Massachusetts electronics corporation discussed on page 103.)

Among non-expert users—especially those using the DOS or Windows operating system—it was hypothesized that "a lack of training" was "the primary factor responsible for poor sanitization practices."[GS02a]

There was, of course, only one way to actually test this belief: by contacting the individuals whose data we had recovered and asking them to reconstruct for us what had happened. Permission to contact these individuals was obtained in April 2003 from the MIT Committee on the Use of Humans as Experimental Subjects; work on this project began shortly thereafter.

3.4.1 Traceback results

Between April 2003 and April 2005 a total of 15 interviews were conducted with individuals and corporations located throughout the United States, corresponding to the data recovered on 19 drives. Substantial attempts were made to contact the owners of another 13 drives; in three of those cases, individuals contacted at the organizations were unresponsive to phone calls and follow-up attempts.

Because the drives that were traced were chosen based on their ability to be traced, rather than being randomly chosen, no statistical conclusions can be drawn from this sampling. Nevertheless, the individual cases are qualitatively revealing.

The hypothesis that poor training was responsible for the recovered information was not confirmed by the Traceback study. Although a lack of training did play some role in the poor sanitization practices, organization failure and misplaced trust were far more common causes in the cases that were investigated.

Trust Failures (8)
Eight drives from a total of four resellers were not sanitized because of a trust failure—the individual or organization that owned the drive had entrusted it to another party, and that second party had sold it without first sanitizing its contents:

ID	Trust Failure Notes
#54	This disk contained the *List Will and Testament* and detailed financial information of a 50-year old woman in Kirkland, WA. The computer had been taken to the "PC Recycle" store in Bellevue by the woman's son, who is employed as a researcher by one of the nation's national labs. The son paid PC Recycle either $5 or $10 to recycle the hard drive. PC Recycle "recycled" the hard drive by putting it on a table and selling it to another customer for $5.
#73	This disk belonged to a community college in Washington State. Information found includes student grades, final exams, email, and other information. The school did not have a procedure in place for wiping information from computers before they were disposed, but has one now.
#74	Another disk from the Washington State community college.
#75	Another disk from the Washington State community college.
#77	Another disk from the Washington State community college.
#128	This disk was in the computer used by the administrator of a church in South Dakota. That administrator left and the new administrator did not know the circumstances by which the disk was conveyed to the reseller, a firm called "PC Junkyard." The current administrator said that the previous administrator "was kind of crazy" and that the previous administrator probably sold the equipment. Other drives purchased from PC Junkyard were properly sanitized, but this one was not.
#193	This disk belonged to an automobile dealership in Maryland. The disk contained internal dealership documents, address books, email, and other materials. The individual who supplied computers to the dealership apparently took the dealership's old computers as part of a trade in; the machines were stripped and the parts sold on eBay without being sanitized.
#205	This disk was from the home computer of a Maryland family whose father is the owner of the automobile dealership that previously owned drive #193. This disk contained email and a mortgage application containing detailed personal financial information. This drive was also part of a computer that was traded-in to a trusted computer seller, who sold the family a new computer.

The case of drives #73 through #78 are particularly interesting. These drives were purchased as a lot of six from an individual in Washington State. While drives #73, #74, #75 and #77 contained federally protected confidential information that had not even been deleted with the Windows DEL command, the other two drives in the lot had little or no confidential information at all. Drive #76 had been formatted but an analysis with FATDUMP found no Microsoft Word or Excel files that could have contained confidential information. (We did find four copies of the LOVE-LETTER-FOR-YOU worm, however.) Drive #78 found 39 links to Word files in the `/Windows/Recent` directory, but the files themselves were not on the drive—instead, they appeared to be on a file server. Thus, it is entirely possible that the person who was disposing of the drives thought that *none* of them had confidential information on them, and that drives #73, #74, #75 and #77 contained the information because of an unanticipated violation of school policy, rather than a failure to sanitize.

In interviews with the former owner of drive #193 and #205, the individual expressed profound frustration that his computer consultant had removed the drives from the computers and there had been no way to audit whether or not the information had been deleted. The owner had trusted his consultant, and that trust was betrayed.

Tool failures or lack of training (3)
In three cases the organization attempted to sanitize the disks itself but the tool that was used—the DOS or Windows FORMAT command—did not actually overwrite the blocks of the disk in question. This can be thought of as either a "tool failure" or a "lack of training:"

ID	Tool Failure Notes
#7	This disk belonged to the California office of a major electronics manufacturer. The disk contained internal documents and source code. The system was declared obsolete by the manufacturer, inadequately sanitized with the FORMAT command, and sold to a surplus vendor.
#21	From a computer that did credit-card processing for a supermarket belonging to a major supermarket chain. The disk included 3,722 credit card numbers and supermarket bank information. At the time the disk was retired the supermarket chain was using Norton Disk Wipe to sanitize old hard drives, but the company believes that the tool was not used consistently in every case. In 2000 the company formalized its disk sanitization procedures as a result of HIPPA and VISA CISP regulations (see Section 2.6.5).
#134	This disk was the primary drive of an ATM machine that belonged to a major Chicago area bank. The bank was aware of the sanitization problem and had hired an outside firm to upgrade its ATM's: the contract specified that the removed disks needed to be sanitized, but failed to specify how. The contractor had hired a subcontractor to perform the actual drive removal and sanitization; the subcontract had specified that the drives should be sanitized, but failed to mention how. Although both the bank and the contractor were aware that the DOS FORMAT command did not properly sanitize hard drives, the subcontractor was not aware of this fact. As a result of being contacted for this survey, the financial institution revised its privacy policies and contracts: all contracts now specify not only *that* removed disks should be sanitized, but they also specify *how* the disks should be sanitized.

Lack of Concern (2)
In two cases the organizations that owned the disks simply did not care if the information on them was sanitized or not. In each case the company was going bankrupt or having significant layoffs; in interviews after-the-fact, the individuals responsible for disposing of the property reported that they simply didn't care whether or not the data the computers contained confidential data:

ID	Lack of Concern Notes
#15	This disk belonged to an Internet software developer. The disk contained a database of 1240 sales contacts and other corporate information. The company was going bankrupt and the computers were sold to a used computer vendor that would pick up the equipment and pay a minimal fee.
#44	This disk was in a computer used by a "publishing specialist" at a computer magazine in the San Francisco Bay Area. Although the company had an informal policy of sanitizing disk drives when employees left the company and computers were repurposed within the organization, in the summer of 2000 the company experienced a two-thirds reduction in force. At that time the company decided to sell as many of its computers as possible in order to recoup some of its investment; sanitizing the computers was not a priority.

Unknown Reasons (7)
In seven cases the original owner was determined but the reason for the sanitization failure could not be discovered:

ID	Unknown Failure Notes
#6	This disk was used by a biotech startup in the San Francisco Bay Area. The drive contained proprietary research documents for an HIV test that was under development. The systems were sold to equipment consolidators when the company was shut down. It is unknown if the company wished to have the contents of the disks sanitized or not.
#11	This disk belonged to the Greensboro, NC, office of a major electronics manufacturer. It contained internal documents. The company did not respond to repeated contact attempts.
#42	This disk was in a computer used by the assistant principal of a San Francisco Bay Area primary school. The computer contained grades and disciplinary letters sent home to parents by assistant principal. The school's staff said that they did not know how the computer had left the school.
#70	This disk had medical information which indicated that it was used by a mail order pharmacy. No attempt was made to determine if the disk contained patient records. The pharmacy claimed that it understood the seriousness of the situation, but subsequently did not return phone calls.
#94	This disk was in a computer used by consultants of a regional telephone company. A document found on the computer stated "This PC is infected with a virus. Call helpdesk at #XXX-7838." When contacted, the company said that too much time had elapsed to determine why the disk was not properly sanitized prior to its being sold as surplus.

ID	Unknown Failure Notes (cont.)
#96	This disk came from the computer used by the vice president of a Minnesota-based food company. Information on the disk included corporate records and details of an employee's "loan repayment plan" schedule. The organization did not respond to repeated contact attempts.
#214	This disk belonged to the Corporation Commission of a US state. The disk contained credit card numbers and other information associated with the filing of various state forms, as well as internal correspondence belonging to the state office. The state's IT division requested that a copy of the disk's image be uploaded to an FTP server. After the upload, no further communication was received from the IT division.

3.4.2 Traceback conclusions

The picture of American businesses and non-profit organizations that emerged from the Traceback study is a frightening picture indeed. It is a picture of organizations that are fundamentally not in control of their information technology. It is a picture of people who can do email and run a few applications but just didn't think about the implications of their actions.

There may be a sampling bias in this study: the Traceback study could only trace data to organizations that, by definition, had leaked their own personal or confidential information. But this isn't a comforting thought, when one considers that disks were traced to a major Chicago bank, to a major grocery store chain, to the headquarters of a public school system, and to a number of small businesses that held confidential customer information.

If anything, the Traceback study confirms the importance of security measures that are either automatic or else extraordinarily easy-to-use: the computer users encountered in this study simply can't handle anything else.

The Traceback study was significantly harder to perform than anticipated. The reason was not the difficulty of identifying the data subjects—the reason was the difficulty of identifying the person within the target organization who either had both knowledge of what had happened and was interested in participating in the study.

Finding a responsible individual for what can only be described as a significant security or privacy violation shouldn't have been a problem, but it was. Under Canadian and European data privacy laws, organizations must identify a specific individual who is responsible for communicating with the public on issues of data privacy. But US law has no such requirement. Once the organization name was determined, it should have been possible to go to that organization's web site, click on a "privacy" link, and immediately have contact information for the responsible individual. But many of the organizations that were contacted for the Traceback study didn't have privacy policy on their web sites. Instead, they had generic "contact" links on their web sites that invited visitors to send email to their web master or their press offices—mail that generally did not engender a response. Instead, contacts were made with responsible parties by repeated calling the organization's switchboard and asking to speak with the organization's network security group. This is not a strategy that should be recommended to the general public.

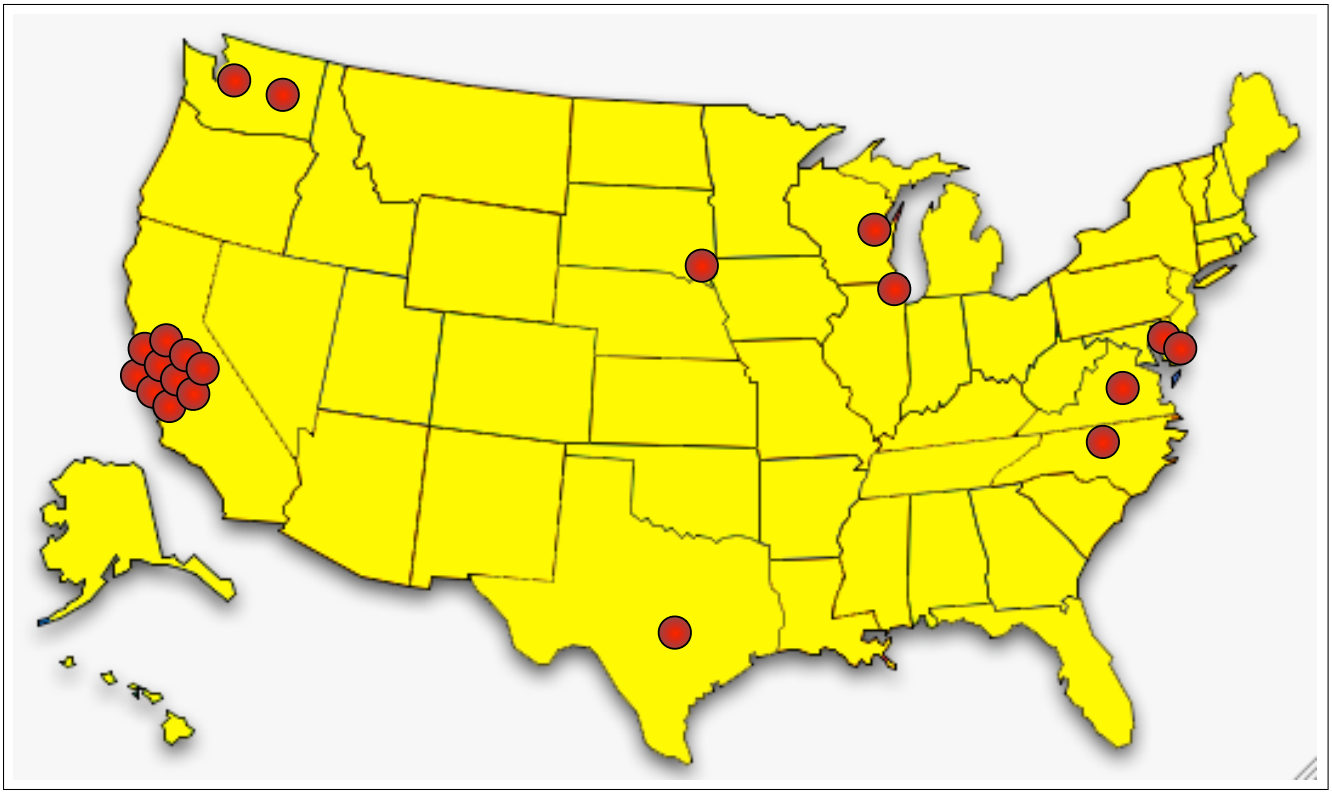

Figure 3-20: A map of the United States showing the locations of the organizations that were responsible for the data on the drives that were successfully traced.

3.5 Future Work: Cross-Drive Forensics

This section has only scratched the surface of a new branch of forensic analysis that we have chosen to call *cross-drive forensics*.

Computer forensics is a fast-emerging field that consists of many sub-specialties. Major areas of research and practice include *disk forensics*, discussed here; *network forensics*, which involves the collection and analysis of information traveling over a network; and *document forensics*, which involves the analysis of printed and digital documents to learn hidden details of their authorship, creation, or modification.

A growing number of tools have been created to aid the forensic examiner. But today's tools tend to assist in the clerical tasks and visualization associated with forensic analysis: they do not automate the forensic process. Today's interactive forensic tools were created in an era when forensics investigations were relatively rare and disk drives were generally small: a practitioner might have had 20 hours to spend analyzing a 10 gigabyte disk drive. The tools serve this purpose well and have provided a treasure-trove of information.

Given the success of these early investigations, many more disk drives are now routinely captured as part of police work or intelligence operations. As a result, there is a growing backlog of hard disks awaiting forensic analysis. There are stories of "rooms full of hard drives" that have been

captured in the course of drug and organized crime investigations and during the course of US military operations in Iraq and Afghanistan. We simply do not have enough analysts to analyze these drives.

Complicating matters is the fact that drive capacities are growing geometrically. Whereas programs like Encase[Kei03] and FTK[Acc05] were developed in an era of 2 and 4 gigabyte hard drives, today's drives range in size from 20 GB to 200 GB or more. Even a simple "string search" can take nearly an hour.

The current generation of forensic tools is simply not up to the task of analyzing massive quantities of information. What's more, their creators will find it difficult to modify them for today's forensic problems because the underlying approach that these tools take is incompatible with today's forensic problems. These tools use visualization to augment the intellect of the analyst.

The forensic techniques presented in this section do not follow the pattern of existing tools. Instead of allowing the detailed assessment of a single drive, they are designed for the rapid assessment of several hundred. This approach is likely to find increasing favor in the coming years, as both law enforcement investigations and US intelligence activities overseas have resulted in backlogs of drives that far exceed the capabilities of trained forensic investigators to analyze. At very least, some cross-drive approach must be used to determine which drives should be targeted for human analysis. At best, these new techniques can find patterns in the forest of drive data that are simply not visible when drives are examined one-at-a-time.

There are many ways that the techniques presented in this section could be readily expanded:

- The Unix `dd` command should be modified, as discussed above, so that read errors are copied over as specially tagged blocks, and not blocks of NULs. Further, when a 64k block cannot be read, an attempt should be made to read blocks of a smaller size.
- The Forensic File System should be finished and implemented as a user-level NFS or SMB redirector so that the full array of Unix tools can be used for forensic analysis.
- In addition to using hash codes to find identical files, we should explore using hash codes to find identical blocks. The theory here would be to effectively characterize Level 3 data. This approach could determine, for example, that a stretch of 50 blocks on the disk are actually two-thirds of a DLL that shipped with a copy of WordPerfect 4.2. Such information might be useful in its own right, or else might be used to eliminate these blocks from further analysis. Erik Nordlander at MIT is working on this technique as part of his masters' thesis.

3.6 Proposals for Addressing the Sanitization Problem

Although the need for proper sanitization of magnetic recording media has been long recognized as a serious issue for computer security practitioners, the problem has traditionally been addressed through the use of add-on software or physical destruction of the media itself. Only recently has the question of sanitization been addressed by computer operating system vendors themselves, and in the cases that we have considered, both Microsoft and Apple and have addressed it poorly.

What's needed, then, is some straightforward way to add a usable sanitizing delete-file function to

existing system.

Although a simple approach would be to resurrect Bauer and Priyantha's Linux implementation, such an approach would be incomplete. Bringing truth to the phrase "rm is forever" is not a particularly user friendly approach for moving forwards. Many usability experts have noted that humans make frequent mistakes; simply adding a warning box saying something to the effect that "rm deletes all of your files" is not a particularly strong barrier to improper use.

The remainder of this section considers two proposals for better addressing the problem of creating a safe sanitizing delete on modern desktop operating systems, and one proposal for solving the data remanence problem through a regulation on computer resellers.

3.6.1 Shredder and delayed irreversible actions

Norman suggested in 1983 that simple confirmation is an inappropriate response for actions that cannot be reversed: a more appropriate approach, he argued, is to accept the command but defer execution for a short period of time:

> "It is not sufficient to ask the user to confirm that a particular action sequence is wanted, because if confirmation is routinely asked for (and if the usual response is "yes"), the confirmation itself becomes an automatically invoked component of the command sequence. Thus, if the command is given in error, it is likely to have the confirmation invoked as part of the same error; in our experience, the confirmation is as apt to be in error as much as the original command. As Newman points out in his discussion of the paper by Schneider *et al.* , the normal response to requests for confirmation is something like this: "Yes, yes, yes, yes. Oh dear!" The point is that disastrous commands should be difficult to carry out; confirmations of the validity of the command may not offer sufficient difficulty to be a satisfactory safeguard.
>
> "Sometimes the command can act as if it were actually executed, when in fact, it has only been deferred. Consider the command to delete files from the system; the system could claim that it has removed the file, but has actually put it away on some temporary location so that it can be recovered later if its "deletion" was discovered to have been an error. (Real deletion can be done on an infrequent basis, say after a lapse of several hours or days.) In Interlisp[6] operations may be "undone," even operations such as writing on or destroying files."[Nor83]

Cooper makes a similar observation:

> "If I tell the computer to discard a file, I don't want it to come back to me and ask, 'Are you Sure?' Of course I'm sure, otherwise I wouldn't have asked. I want it to have the course of its convictions and go ahead and delete the file. "
>
> "On the other hand, if the computer has any suspicion that I might be wrong (which, of course, is always), it should anticipate my changing my mind and be fully prepare to

[6] Teitelman, W. and Masinter, L. "The Interlisp programming environment." *Computer 14*, 4 (April 1981), 25–33.

> undelete the file. In either case, the product should have confidence in its own actions, and not weasel and whine passing the responsibility onto me."[Coo99, p.167]
>
> ...
>
> "A confirmation dialog box is a convenient solution for the programmer because it absolves him from the responsibility of being the agent of an inadvertent erasure. But that is a misunderstanding of the real issues. The erasure is the user's responsibility, and she has already given the command. Now is not the time for the program to waiver. It should go ahead and perform the requested task. The responsibility that is actually being dodged is the program's responsibility to be prepared to *undo* its actions, even though the user requested them.
>
> "Humans generally don't make decisions in the same way that computers do, and it is quote normal and typical for a person to change his mind or what to undo a decision made earlier. In the real world outside of computers, most actions can be deferred, changed, or reversed. There is no reason that this can't also be true for software-based products, except that the programmers who create them don't think that way. "[Coo99, p.68]

The insight of these suggestions is matched only by the shortsightedness of the industry in its failure to adopt them.

The Shredder
One design for such an implementation would be to build upon the Recycler metaphor. Instead of having a set of "Empty Trash" and "Secure Empty Trash" commands, the revised implementation would have a single command: "Shred Trash." Choosing this command would move the contents of the Trash to the Shredder, where the blocks corresponding to the documents would be automatically sanitized and the files unlinked according to a policy: either at a particular time of day, or after the documents had been in the Shredder after a specified period of time. A typical set of rules might be:

- Shred any file that has been in the Shredder for more than 30 days.
- Shred all files in the Shredder when the user clicks the "Shred all files now" button.
- Shred all files in the Shredder at 7am every day.
- If a file is selected and the user chooses the "Shred" command from the File menu, move the file to the Shredder and schedule it for shredding in 5 minutes.
- If a file that is in the Shredder is selected and the user chooses the "Shred Now" command from the File menu, shred the file immediately.

These rules give users a chance to change their minds, but nevertheless provide for the possibility of immediate shredding, should such actions be necessary. Figure 2-28 shows a hypothetical user interface to implement this rule set. A suggested implementation is diagramed in Figure 3-21.

The name "Shredder" is superior to "Secure Empty Trash" because most people know what a paper shredder does; most people do not know what it means to securely empty trash. Thus, less initial

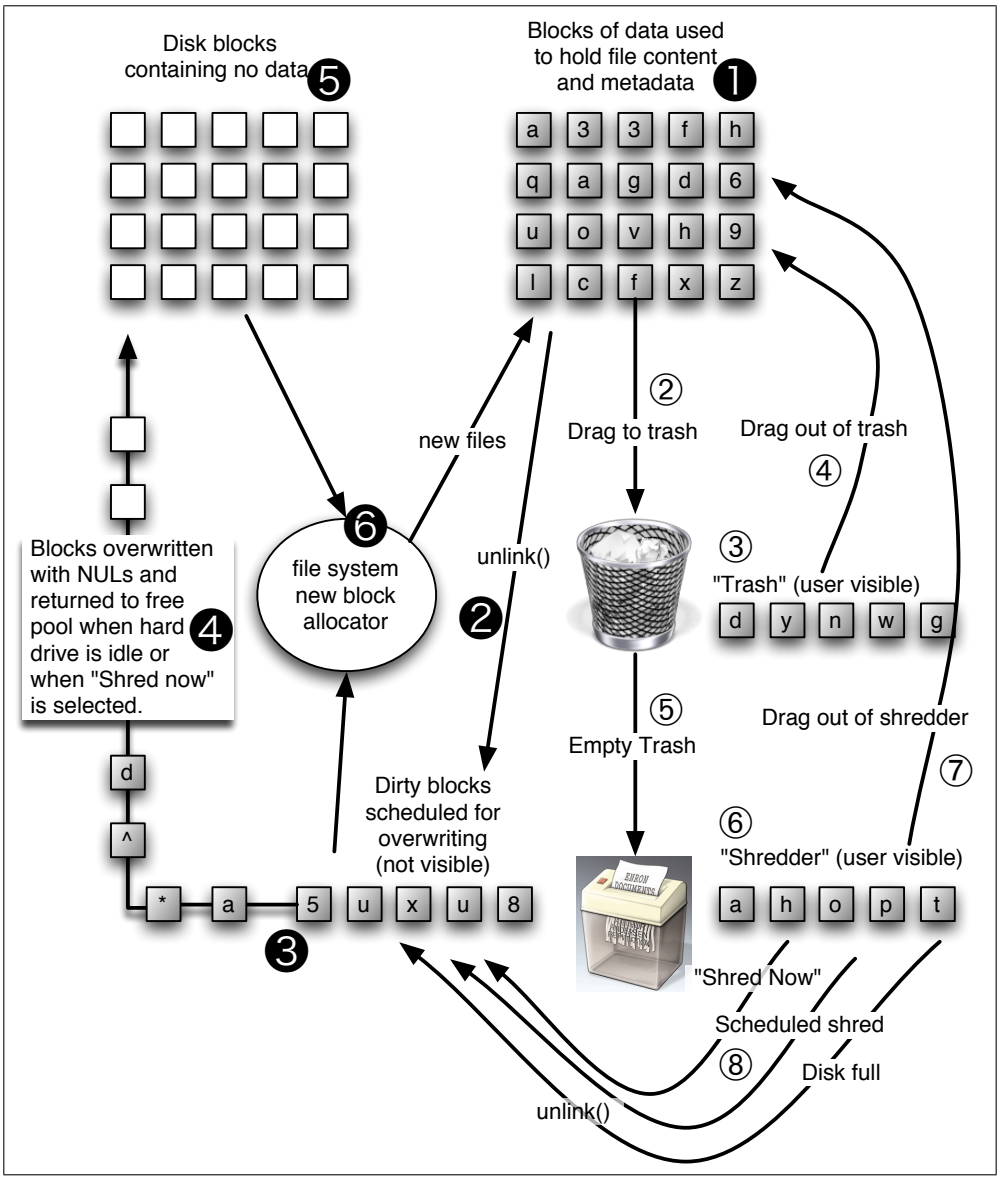

Figure 3-21: The proposed design for a unified trash and shredder system which incorporates the design patterns put forth in this section. ❶Blocks of data used to hold file content are visible and in the file system. ❷Files deleted with the `unlink()` system call are moved to ❸the list of dirty blocks that are scheduled for overwriting. ❹When the hard drive is otherwise idle, the system overwrites these blocks with NULs and returns them to ❺the pool of disk blocks containing no data. ❻When the file system block allocator needs a new block, it can draw first from the dirty blocks, automatically sanitizing them when they are used, and second from the reserve of sanitized blocks. Files can also be deleted if the user ②drags a file to the ③"Trash." To undelete a file that is accidentally dragged to the Trash, ④the user drags the file out. ⑤Select "Empty Trash" to move the files to the ⑥"Shredder." ⑦As with the Trash, files can be dragged out of the Shredder. ⑧Alternatively, the files will be unlinked and their blocks scheduled for overwriting if the user presses the "Shred Now" button, if there is a scheduled shred, or if the disk is full. In the case of "Shred Now," the overwriting operation takes place immediately, even if the disk is otherwise occupied.

user education would be required. And because shredding would be performed asynchronously, there would be no perceived penalty for using the feature. This is an application of the Whitten's *metaphor tailoring* approach, showing that the approach can be used for verbal metaphors in addition to visual ones.

If Shredder is implemented inside the file system, rather than within an application such as the MacOS Finder and the Windows Explorer, then it would be possible to tightly couple the deletion and sanitization. The `unlink()` system call could then be modified to put files into the Shredder and schedule them for sanitization; the file system's block allocator would be modified to obtain blocks from the Shredder that are scheduled for sanitization, since overwriting such blocks with new data would accomplish the same goal. Such modifications would have minimal impact on desktop operating systems, which spend a great deal of their time idle. Such policy could be disabled on operating systems on servers, if proper sanitization could be administratively ensure prior to disposal; alternatively, the policy could be selectively implemented on some directories but not others, as did Bauer and Priyantha. But it might not be necessary to do either, if the shredder only runs when the disk is not otherwise serving requests from the operating system. A schematic for such an integration between the file system, the Trash and the Shredder is presented in Figure 3-21

3.6.2 Data Hauler: a regulatory proposal for addressing the data passed problem

The last proposal in this chapter is for federal and state governments to pass legislation that would require hard drives to be properly sanitized before being resold on the secondary market. Sanitization can *easily* be accomplished as part of the testing procedure, as was the case for roughly a dozen of the 236 drives that were purchased for this study. Although it seems that a substantial number of drives are sold without testing, perhaps they shouldn't be. It's not obvious that the sale of untested used hard drives represents a substantial contribution to the nation's economy: although it would be silly to outlaw the sale of a few used hard drives, it is completely feasible to regulate organizations that sell more than a hundred per month.

Unfortunately, the FTC Rule implementing the Fair and Accurate Credit Transactions Act of 2003 (see Section 2.6.5 on page 92) completely exempts so-called "service providers" from compliance with the Rule if the service providers are not specifically told that computers being disposed of contain consumer reports. The Commission specifically deleted an example of a "garbage collector" from its Proposed Rule when it published its Final Rule. It appears that ignorance is indeed bliss: any scavenger or dealer in used computer systems that does not look for consumer reports on its systems and is not notified the reports exist is not be responsible for destroying those reports before selling the systems. This is a significant loophole that could easily be addressed.

Microsoft has an incentive to create a sanitization process that removes all user data and applications but leaves the operating system intact. Such a process would help users to manage licenses for software applications and help end the illegal practice of selling used computers with all of their applications. Although some computers have in the past been sold with "system restore disks" that return the computer to the configuration that it was in on the day that it was sold, these disks present a significant security problem themselves: they return the computer to its unpatched configuration. Experience has shown that such a system will be compromised within a few minutes of

being placed on the Internet.[Pro04]

Federal regulations cover the management of hazardous waste in the United States. Organizations that generate as little as 100 kilograms of hazardous waste per month are covered by the EPA regulations and must employ a federally licensed Hazardous Waste Hauler.[Age05] In California, haulers must also be registered with the State of California and have a a *Certificate of Compliance* by the California Highway Patrol. The California regulations are quite stringent, requiring that any organization generating more than 100 kilograms of hazardous waste in any calendar month "must ship the waste off-site within 90 days after the first drop of waste enters the storage container."[Bus05] Such regulations help protect communities and workers by preventing the accumulation of hazardous waste in facilities that are not suitable for long-term storage. Shipments must be tracked using the EPA's Hazardous Waste Manifest System, which can help inform first responders as to the nature of a hazardous waste shipment in the event of an accident.[Age05]

In many ways remnant data is the hazardous waste of the information age and needs to be treated as such. The fact that some of the hard drives containing personal information belonged to companies that had gone bankrupt is a very close analog to the so-called Superfund and "brown field" sites of the 1980s, where companies had failed, leaving contaminated land and ground water. Regulatory responses are appropriate. Currently there is no one who is responsible for sanitizing the collected personal information on the hard drives of a corporation when that corporation fails. It makes sense to place this burden on those who benefit from trafficking in the corporation's electronic equipment.

3.7 Patterns for User Visibility and Sanitization

Based on the careful consideration of the information presented in this chapter, this thesis presents five patterns for aligning usability and security in the realm of data sanitization are proposed herein. The patterns are introduced here and presented in detail in Chapter 10.

These patterns were chosen based on Alexander's pattern selection criteria. That criteria holds that patterns should be chosen based on their *moral value*.[Ale96] Speaking before the 1996 OOPSLA conference in San Jose, Alexander stated that patterns should be chosen which "actually make human life better as a result of their injection into the system."

Although at first glance this may appear to be a subjective criteria that is not easily measured or repeated, Alexander insisted that "there is striking agreement" between professionals when asked whether or not a specific pattern makes human life better.

Addressing the computer scientists at OOPSLA, Alexander said that in his field of architecture, the idea of making human life better actually means something. He wasn't sure if there was an analog in computer science—he said he didn't know if the scientists at OOPSLA were merely looking for technical performance that is good, or something that was profoundly good from a moral point of view.

In the case of giving people tools to sanitize their computers, there is a clear moral good that can be achieved. Each of these patterns are designed to make computers safer, more enjoyable, better,

and promote more secure operation. The question is simple: would you rather have a computer that incorporated these patterns, or one which did not?

Given that goal, the technical question is whether this is a minimal set of patterns, of if there is additional functionality that can only be captured through the introduction of additional patterns. It does not appear that this set of patterns can be reduced any further. On the other hand, additional requirements could certainly create the need for additional patterns.

- **EXPLICIT USER AUDIT** (page 324)
  This pattern holds simply that users should be able to see all of the information that they are responsible for in the system that they are using. The pattern refers to such information as "user-generated information." This is a broad term which includes information directly generated by the user, documents they type, and information that is collected about the user during the operation of the machine—for example, the information contained in logfiles and web browser caches.

  The EXPLICIT USER AUDIT pattern is a direct application for Fair Information Practice (see Section 2.6.1) to computer systems. It views the software that is running on the computer not only as a tool of the user, but also as an agent of the software's creator. That creator has a moral obligation to make sure that there are no secret databases—no information that could harm the user, but of which the user is unaware.

  In other words, computer systems should not lie to users. They should not give the user the impression that information is not present in the system, when it fact it is.

  There are two simple ways to implement this pattern: either the computer can never allow the user to delete any information at all, or else the computer must ensure that the specific memory used to store that information is sanitized when the user asks that the information be deleted.

- **EXPLICIT ITEM DELETE** (page 326)
  There are two paradigms for deleting information in a computer system: the information can be deleted item-by-item, or else a region of the computer (or the entire computer) can be wiped clean. EXPLICIT ITEM DELETE is the first data deleting pattern.

  This is the pattern implemented by the DOS `DEL` and `ERASE` commands, by the Unix `rm` command, and by the graphical interaction metaphor of dragging an item to the trash. This pattern holds that the tools for deleting information should be made available to the user where the information is displayed in the user interface.

  This pattern relies upon the COMPLETE DELETE pattern to actually remove the information.

- **RESET TO INSTALLATION** (page 326)
  The second way to delete information on a computer system is to reset the system to an installation state. This is analogous to the action of running the Windows `FORMAT` command or performing a "hard" reset on a PalmOS a computer.[pal05] It's a useful function that should be exposed to users whenever possible. (Norman writes how the navigation system in a rented car was not equipped with any simple way to erase all of the previous destinations.[Nor97] As a result, each renter tended to leave their destinations in the computer, where they could be easily reviewed by future renters.)

As we have learned in this chapter, many computer systems that do provide a RESET TO INSTALLATION feature do not implement that feature properly. In Section 3.3, we saw that the data on many disks that had been formatted could be trivially recovered. That is because the Windows FORMAT command does not implement the COMPLETE DELETE pattern, described next.

- **COMPLETE DELETE** (page 327)
  Providing deletion functionality is not enough. The system must also ensure that information is completely deleted so that it cannot be recovered. This is the idea behind the COMPLETE DELETE pattern.

  A simple way to implement COMPLETE DELETE on a computer system is to overwrite the data that is being deleted. Most computer systems, as we have discussed, do not do this. Instead they merely remove the link between the data and the memory containing the data, then mark the memory as "free" and available for re-use.

- **DELAYED UNRECOVERABLE ACTION** (page 328)
  As discussed in Section 3.6, if computer systems are going to have the ability to perform unrecoverable actions, one way to prevent these actions from being performed in error is to institute a delay between the time that the action is invoked and the time that the action is performed. This specific interaction pattern is referred to as the DELAYED UNRECOVERABLE ACTION. It can be implemented with a timer and a mechanism for aborting the requested action.

3.8 The Policy Implications of "Complete Delete"

The information presented in this chapter establishes the fact that there is a widespread problem of confidential information being left behind on discarded systems. Although some of this information is obsolete, much of it is not. During the time that this work has been performed, the problem of data remanence has become the subject of considerable debate. This chapter establishes that the data remanence problem on consumer computer systems it the result of historical accident, rather than the result of intentional design. Finally, the chapter proposes solutions to the problem.

In an eloquent article, The Honorable James M. Rosenbaum, chief district judge of the federal district of Minnesota, argues that the legal profession's current obsession with the ability to recover deleted information from computer systems is unhealthy to our system of law and, ultimately, our humanity. But he ultimately doesn't blame the lawyers—he blames computer systems: "The real flaw is that the computer lies when it says DELETE. This mechanical lie ought not to debase and degrade the humans who are, and ought to be, its master."[Ros00]

Rosenbaum argues that there should be some kind of "cyber statute of limitations" which would hold deleted information off limits in many cases:

> "I suggest that, barring a pattern of egregious behavior, or an objective record of systematic conduct—absent, if you will, a real 'second set of books'—that the courts recognize the existence of cyber trash. This is the stuff, which, in less electronic times, would have been wadded up and thrown into the wastebasket. This is what the DELETE button was meant for, and why pencils still have erasers."[Ros00]

The ability exists to correct this great technological lie. We don't need to create a new statute of limitations—all we need to do is to fix the `unlink()` and `DeleteFile()` system calls. But such a change would not merely protect businesses and individuals: it would also dramatically complicate the work of investigators trying to uncover wrongdoing. Oliver North's violations of federal law came to light because investigators were able to recover North's deleted PROFS messages. Likewise, much of the Enron bankruptcy case was unraveled through the use of deleted Lotus Notes messages.[DiS02] If complete delete is built into operating systems, similar evidence of wrongdoing in the future might be unavailable to investigators.

It is possible that more harm is being done by the failure of our operating systems to sanitize deleted files than good is resulting from the ability of forensic investigators to recover deleted information. It is also possible that criminals will increasingly use readily available programs to remove information of wrongdoing from their computer systems as knowledge of forensic capabilities spreads through the criminal world.

But even if criminals make more use of sanitization technology than upstanding citizens, this should not the ruler by which the desirability of the feature is measured. Ultimately, whether or not computers should create covert records of their users' activities is a question that should be the subject of public discussion. Judge Rosenbaum's article is a beginning of that discussion. More voices need to take part.

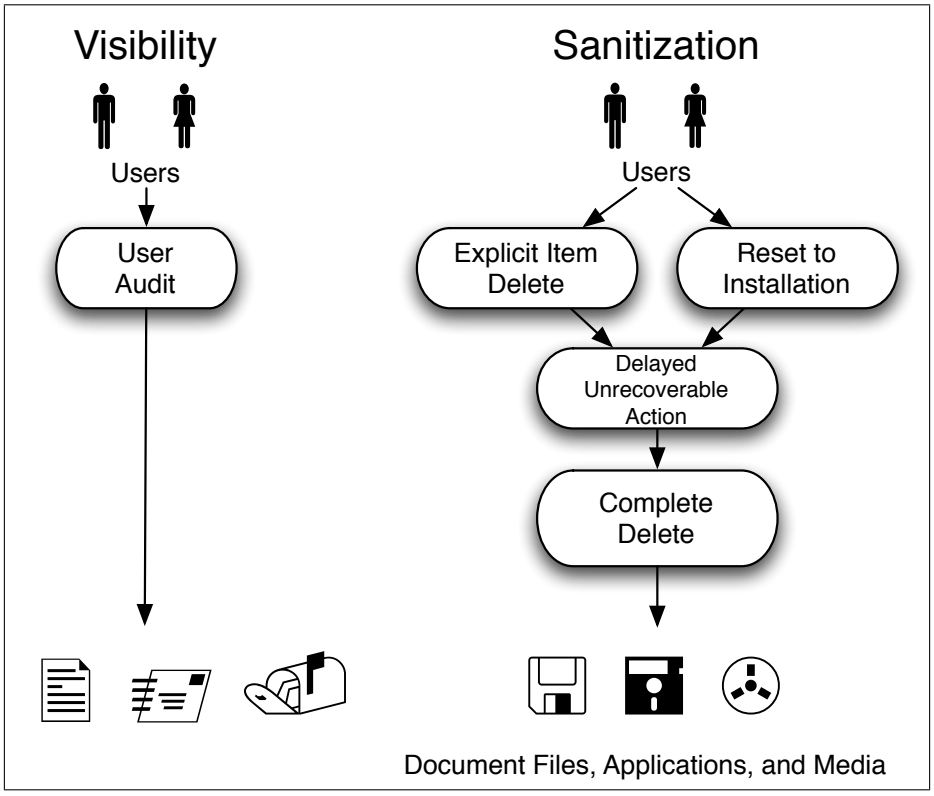

Figure 3-22: A graphical representation of the five patterns involved in visibility and sanitization, showing how they relate to each other and to the user