Global Information Assurance Certification Paper

Unix Secure File Deletions

By Ken Hatfield
June 2001
GSEC version 1.2d

Introduction

In the war of data protection, one aspect that can be overlooked is the potential for information compromise from ineffective file deletions. Due to the mechanics of writing data to magnetic media, it is possible to retrieve data that the user (and the Operating System) believes to have been removed. This paper will show how this type of data compromise can occur and review capabilities within the Unix operating system and other available Unix tools to mitigate this potential for data loss.

The Threat

Except in the most simplistic of deletions, physical access to the disk media is necessary. Though this may seem like a barrier, this physical access can happen through an insider threat, theft, or even in very innocuous ways. For example, when a computer disk has a failure, often it sent out to a repair shop. Frequently, the disk is not functional and the vendor simply replaces the defective hard drive. And in most cases, the customer does not even think to ask for the old drive back. The data on that drive could be compromised. Also, as technology improves ever so faster, computer systems are refreshed for newer systems in shorter and shorter time frames. The "old" computer system can end up in the hands of nearly anyone. Attackers can determine how these computer systems are recycled to the public and place themselves in the recycle chain. Also, the device does not even have to leave a company to create a data threat. I work at a national laboratory, where sensitive and classified information exists. The need to know regarding this data is stringent. Transferring a device to be used by someone outside of that need to know without proper disk sanitation, could lead to a severe data compromise.

The Attack

There are some open source utilities available that will attempt to recover deleted Unix files. The utilities are "unrm" and "lazarus". These tools can be found in The Coroner's Toolkit or TCT. The unrm utility analyzes a Unix file system reporting on all the used but unallocated space. The lazarus utility then attempts to reconstruct the data. It is by no means perfect and requires some knowledge of the type of data you are looking for. It also needs a considerable amount of disk space to process. This type of attack is successful in instances where Unix files were removed via rm or even a new partition created over an old one using mkfs or newfs.

To understand the other types of attack, a brief discussion of disk technology is necessary. In other words, to see how data can be recovered from magnetic media, it is

of value to understand how data is written to magnetic media. As most already know, data for computers is composed of zeros and ones. So as data is written and rewritten to disk media, one would assume that the media always records a zero or one. In actuality, the recording mechanism cannot write on exactly the same position. The resulting effect is then best described as obtaining a 0.95 when a one overwrites a zero, and a 1.05 when a one overwrites a one. The standard circuitry of a disk will read both of these values as one. However, there now is a remnant of the previous data still residing on the magnetic media. Are there ways to access these data remnants?

Enter Magnetic Force Microscopy (MFM) and Scanning Tunneling Microscopy (STM). MFM images magnetized patterns. MFM is derived from Scanning Probe Technology (SPM). The technique uses a magnetic tip that is placed close to the surface where it interacts with the emanations from the surface, creating an image. It can produce results in a very short amount of time and accomplishes it goal with very small samples. STM, a more recent technique, uses different metal plating on the tip and a very low voltage potential to tunnel electrons across the gap of the probe tip and the magnetic surface. It images the media in the same way as MFM.

So remnants of the data continue to reside on the media, even after overwriting, creating in effect, layers of data on the media. Each layer has its unique pattern that can be determined. As successive overwrites of the media occur, it does become increasingly difficult to obtain the lower layers of the data remnants.

There are several thousand of these types of devices in the field today. Granted, many (and hopefully most) of the MFM or STM devices are used legitimately by either disk manufacturers or honest companies that help users recover the data from their own disk drives. However, these techniques can be used by the less ethical once physical access to the magnetic media is obtained. It is also worth noting that a very workable MFM device can be built using a PC and a controller very inexpensively. For more information on these techniques, see Peter Gutman's article.

With this overview of the attacker's tools, let's now see what (if any) steps can be taken to deter this attack.

Defense: Inherent OS tools

rm

As with all security, defenses are built in layers and dictated by the perceived impact caused by a compromise. The rm command in Unix is the most simplistic and unsecured method of deleting a file. When a file is deleted using rm, all that happens is that the inodes for the file are freed to be used by file system. The data is in no ways deleted. To recover this data, does not actually require physical access to the media. The utility tools, unrm and lazarus as previously mentioned, will often enable recovery of data.

mkfs, newfs

The commands such as mkfs/newfs are used to create a fresh Unix file system. These commands can be equated somewhat to the FORMAT command in the windows environment. These commands do not actually delete any data; they define for the operating system the structure of the file system so data reads and writes can occur. The unrm and lazarus tools could recover the data. And certainly a disadvantage to these commands is that all the files will be "deleted" as this operates on an entire partition. In the end, it no more secure than the rm command.

Format commands

Each flavor of Unix has more low-level commands that will actually destroy the data on a disk. For example, Solaris has the format utility and HP has the mediainit utility. These commands after execution, have redefined the entire disk, checking and marking bad sectors. These commands will make it impossible to for the unrm/lazarus utilities but not for MFM and STM technologies. Also, again there is the disadvantage of destroying all the data on a disk not just a file. But if you can ensure that the disk cannot physically be compromised and wiping out an entire disk is tolerable, this may be as far as you need to protect.

Defense: open source and purchased products

In describing the attack, an approach was mentioned that would offset the attack. As data is written and rewritten to magnetic media, it becomes increasingly difficult to access the lower layers. Some developers have attempted to take advantage of this. The following tools use that as their main strategy in securing the file deletions.

wipe

This utility has more than one developing author. The version wipe 0.16 written by Berke Durak had the most cross-platform capability. Wipe will delete a single file, multiple files, and recursively traverse directories. During the overwriting, wipe uses 34 patterns of which 8 are random patterns. You need to define the number of overwrite passes with the range being from 1 – 32. Certainly, the more the better; however it is also the case that more the slower. Included is a list of the options available with wipe:

```
   Usage: wipe [options] files...
Options: -f Force, ie. don't ask for confirmation
         -c Do chmod on write-protected files
         -r Recurse into directories
         -q Quick wipe, less secure, 4 random passes by default
         -Q <number>: set number of passes for quick wipe
         -a Abort on error
         -i Informational (verbose) mode
         -s Silent mode
         -R Set random device OR random seed command
```

-S (r|c|p) Random seed method
r Read from random device (strong)
c Read from output of random seed command
p Use pid (), clock () etc. (weakest)
-M (l|r) Set PRNG algorythm
l Use libc ()'s random ()library call
-a Use arcfour encryption algorythm
-v Show version information
-k Keep files, i.e. do not remove() them after overwriting
-F Do not attempt to wipe filenames
-T <tries> Set maximum number of tries for free
filename search; default is 10
-P <passes> Set number of passes for filename wiping.
Default is 1.
-h Display this help
-Z Do not wipe file size
-l <length> Set wipe length to <length> bytes, where <length> is
an integer followed by K (Kilo:1024), M (Mega:K^2) or
G (Giga:K^3)
-o <offset> Set wipe offset to <offset>, where <offset> has the
same format as <length>
-e Use exact file size: do not round up file size to wipe
possible remaining junk on the last block
-b <buffer-size-lg2> Set the size of the individual i/o buffers
by specifying its logarithm in base 2. up to 30 of these
buffers might be allocated

Another version of wipe developed by Thomas Vier Jr. seems to have the decent ongoing development. However, this utility works well on the Linux platform only. The software can be found at http://wipe.sourceforge.net.

secure delete

This utility has been developed by van Houser. Secure delete is actually a set of four utilities that will securely delete files (srm), securely overwrite unused disk space (sfill), a clean the swap file system (sswap), and secure removal of data from memory (smem). As with wipe, it is originally developed on Linux. I was able to compile this on my Solaris machine. This utility when run in its most secure mode will do the following:

1. The overwriting procedure is done 38 times. The disk cache is flushed after each pass.
2. The procedure performs a truncate of the file. An attacker won't know which disk blocks belong to the file.
3. The file is renamed so an attacker can't obtain any information from the filename as to the contents of the file.
4. The file is then deleted (unlink).

The options for secure delete are as follows:

```
srm   [-d] [-f] [-l] [-l] [-v] file [file] [another file] [etc.]
sfill  [-f] [-l] [-l] [-v] target-directory
sswap  [-f] [-l] [-l] [-v] /dev/of_swap_filesystem
smem   [-f] [-l] [-l] [-v]

-d   don't delete the dot special files "." and ".." on the
commandline (only srm)
 -f   fast writes without O_SYNC and sync() between writes. Much faster
but less secure.
 -l   lessens the security. Only one random plus one pass with 0xff are
      written.
 -l   a seconds time as parameter switches into the insecurest mode,
      it overwrites the file only once with 0xff.
 -v   turn verbose mode on.
 file   file to delete. Wildcards are of course allowed.
For unix: you need write permissions. For msdos: It may be hidden,
system, readonly etc. we don't care.
target-directory   target is a directory in the filesystem to write to.
swap_filesystem   your swap filesystem. Unmount it first!! only tested on linux

Options may be applied like "-lv", "-s -v" or a mix.
```

UniShred Pro

This is a commercial product sold by Los Altos Technologies that has some interesting features and certainly some limitations. It operates on the same principles as wipe and secure delete, i.e. overwriting data numerous times, then removing the data. UniShred supports most of the main flavors of Unix (Solaris, HP-UX, AIX, IRIX). UniShred overwrites not only all addressable areas of the disk, but also renders the grown defect areas of the disk secure as well. The other utilities fail at this as they can only access areas that the disk can access. UniShred has been recognized by many US government agencies such as the Dept of Navy, Dept of Defense, and Dept. of Energy as meeting their stringent regulations for disk cleansing and sanitizing. The product supports all the predefined overwrite patterns specified by these agencies and also allows the user to specify their patterns. There is a good reporting system that allows the user to create a hard copy of the actions taken by UniShred for audit purposes. According to the literature, UniShred can also sanitize a single disk system in a limited fashion. On the downside, UniShred only support SCSI disks. Also, this product cannot delete single files. The best it can do is partitions.

Degauss and destroy

To truly protect the data that you have erased on a disk, the only solution is to render the disk totally unusable. Degaussing is a technique that accomplishes this. A device that degausses is an external de-magnetizer that reduces the magnetic flux on recorded media rendering the magnetic media unreadable. Degaussing is effective on the native media

except with hard drives. The housing on the hard drive itself creates a shielding which can prevent the degaussing of the media. To truly degauss a hard drive would require the hard drive to be disassembled or have a degausser that is so strong it could destroy diskettes, tapes, or other media within several yards. Another measure that would certainly assure the inability to recover the data is to remove and destroy some or all platters from the hard drive assembly.

Conclusion

Deletion of data from magnetic media is not as straightforward as one might believe. The data owner needs to realize that media no longer used can still end up in the hands of someone who can compromise their data. As with other security measures, there are layers of protection that a data owner can institute to ensure that deleted data has been securely removed. The level will depend on the sensitivity of the data and/or the paranoia of the data owner. The levels can range from simply using the Unix operating system commands to physically destroying the magnetic media.

References

Peter Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory", July 22-25, 1996, Department of Computer Science University of Auckland, San Jose, California, http://www.cs.auckland.ac.nz/~pgut001/pubs/secure_del.html

Dan Farmer & Wietse Venema, “The Coroner’s Toolkit”, http://www.fish.com/tct

Berke Durak, “wipe 0.16”, http://gsu.linux.org.tr/wipe

Van Houser, “Secure Delete”, http://freshmeat.net/projects/securedelete/download/secure_delete-2.2.tar.gz

Los Altos Technologies, “UniShred Pro”, http://www.lat.com/usp_main.htm

Samuel Toler, “Do You Know What's Left On Your Disk? “Data Remanence" “, October 2000, http://www.sans.org/infosecFAQ/covertchannels/remanence.htm