

Linux Data Hiding And Recovery

Anton Chuvakin

Just when you thought your data was removed forever, Anton Chuvakin shows us how to recover data and even how data can surreptitiously be hidden within space on the filesystem.

It is common knowledge that what is deleted from the computer can sometimes be brought back. Recent analysis of security implications of "alternative datastreams" on Windows NT by Kurt Seifried has shown that Windows NTFS filesystem allows data hiding in "alternative datastreams" connected to files. These datastreams are not destroyed by many file wiping utilities that promise irrecoverable removal of information. Wiping the file means "securely" deleting it from disk (unlike the usual removal of file entries from directories), so that file restoration becomes extremely expensive or impossible.

Some overview of what remains on disk after file deletion, how it can be discovered and how such discovery can be prevented are provided in Secure Deletion of Data from Magnetic and Solid-State Memory by Peter Gutmann. The author recommends overwriting files multiple times with special patterns. Against casual adversaries, simply overwriting the file with zeros once will help.

Linux has no alternative data streams, but files removed using `/bin/rm` still remain on the disk. Most Linux systems uses the ext2 filesystem (or its journaling version, ext3 by Red Hat). A casual look at the design of the ext2 filesystem shows several places where data can be hidden.

Let us start with the classic method to hide material on UNIX filesystems (not even ext2 specific). Run a process that keeps the file open and then remove the file. The file contents are still on disk and the space will not be reclaimed by other programs. It is worthwhile to note that if an executable erases itself, its contents can be retrieved from `/proc` memory image: command `"cp /proc/$PID/exe /tmp/file"` creates a copy of a file in `/tmp`.

If the file is removed by `/bin/rm`, its content still remains on disk, unless overwritten by other files. Several Linux unerase utilities including `e2undel` recover attempt automated recovery of files. They are based on Linux Ext2fs Undeletion mini-HOWTO that provides a nice guide to file recovery from Linux partitions. Recovery can also be performed manually using `debugfs` Linux utility (as described in the above HOWTO).

Overall, if recovery is attempted shortly after file removal and the partition is promptly unmounted, chances of complete recovery are high. If the system was heavily used, the probability of successful data undeletion significantly decreases. However, if we are to look at the problem from the forensics point of view, the chances of recovering *something* (such as a small part of the illegal image for the prosecution) is still very high. It was reported that sometimes parts of files from several years ago are found by forensic examiners.

Thus files can be hidden in free space. If many copies of the same file are saved and then erased, the chance of getting the contents back becomes higher using the above recovery methods. However, due to the intricacies of ext2 filesystem, the process can only be reliably automated for small files.

A more detailed look at ext2 internals reveals the existence of slack space. Filesystem uses addressable parts of disk called blocks, that have the same size. Ext2 filesystems typically use 1,2 or 4 KB blocks. If a file is smaller than the block size, the remaining space is wasted. It is called slack space. This problem long plagued early Windows 9x users with FAT16 filesystems, which had to use block sizes of up to 32K, thus wasting a huge amount of space if storing small files.

On a 4GB Linux partition, the block size is typically 4K (chosen automatically when the `mke2fs` utility is run to create a filesystem). Thus one can reliably hide up to 4KB of data per file if using a small file. The data will be invulnerable to disk usage, invisible from the filesystem, and, which is more exciting

Linux Data Hiding And Recovery

Anton Chuvakin

for some people, undetectable by file integrity checkers using file checksumming algorithms and MAC times. Ext2 floppy (with a block size of 1KB) allows hiding data as well, albeit in smaller chunks.

The obscure tool bmap exists to jam data in slack space, take it out and also wipe the slack space, if needed. Some of the examples follow:

```
# echo "evil data is here" | bmap --mode putslack /etc/passwd
```

puts the data in slack space produced by /etc/passwd file

```
# bmap --mode slack /etc/passwd
getting from block 887048
file size was: 9428
slack size: 2860
block size: 4096
evil data is here
shows the data:
```

```
# bmap --mode wipeslack /etc/passwd
cleans the slack space.
```

Hiding data in slack space can be used to store secrets, plant evidence (forensics software will find it, but the suspect probably will not) and maybe hide tools from integrity checkers (if automated splitting of the larger file into slack-sized chunks is implemented).

Now lets turn to discovering what is out there on the vast expanses of the disk drive? If looking for strings of text, a simple "strings /dev/hdaX | grep 'string we need'" will confirm the presence of string on the partition (the process *will* take along time). Using a hex editor on the raw partition can sometimes shed some light on the disk contents, but the process is extremely messy. Thus further analysis puts us firmly in the field of computer forensics. The best tool for snooping around on the disk is The Coroner's Toolkit by Dan Farmer and Wietse Venema and the tctutils set of utilities for it. The software provides functionality for gathering forensics data, recovering files, analyzing drive contents for changes (using file timestamps), locating content on disks (using inode and block numbers) and for other fun forensics tasks. Detailed description of the toolkit goes well beyond the scope of this paper.

Now lets briefly review how to prevent the adversaries from finding private data. Several Linux file cleansing utilities exist. All but one can only be used to wipe files, rather than empty disk space. GNU shred (by Colin Plumb), srm (by Todd Burgess), wipe (by Tom Vier) and srm from thc kit (by THC group, see <http://packetstorm.linuxsecurity.com/groups/thc/>). Some use the multiple random passes as recommended in the above paper and some simply overwrite the file with zeros once. Some does not work under certain circumstances or for specific filesystems. As reported in shred man page "shred relies on a very important assumption: that the filesystem overwrites data in place". If this condition is not met, no secure deletion will be performed (with no error message!).

To eliminate the traces of old removed files, one might want to wipe the empty space. The simple method is to use a standard Linux "dd" utility. To wipe the empty space on /home partition use:

```
dd if=/dev/zero of=/home/bigfile
sync
rm /home/bigfile
sync
```

Linux Data Hiding And Recovery

Anton Chuvakin

The commands will zero out the empty space on the partition. Doing the same for /tmp partition might cause some applications to crash, thus one must be cautious. Utility "sfill" from the THC secure_delete package can overwrite empty space using more stringent algorithm.

It should be noted that swap space can also contain pieces of private data and should be wiped as well if additional security is desired. Another program (sswap) from THC toolkit can be utilized for the purpose.

The important fact to note is that when empty space is wiped, slack space for all files remains intact. If file is wiped (at least using current version of GNU shred), the associated slack space is NOT wiped with it!

The article briefly touches upon hiding, finding and destroying data on Linux filesystems. It should become clear that the area of computer forensics, aimed at recovering the evidence from captured disk drives, has many challenges, requiring knowledge of hardware, operating systems and application software.