

Persistence of data on storage media

Jamie Riden 2007-06-26

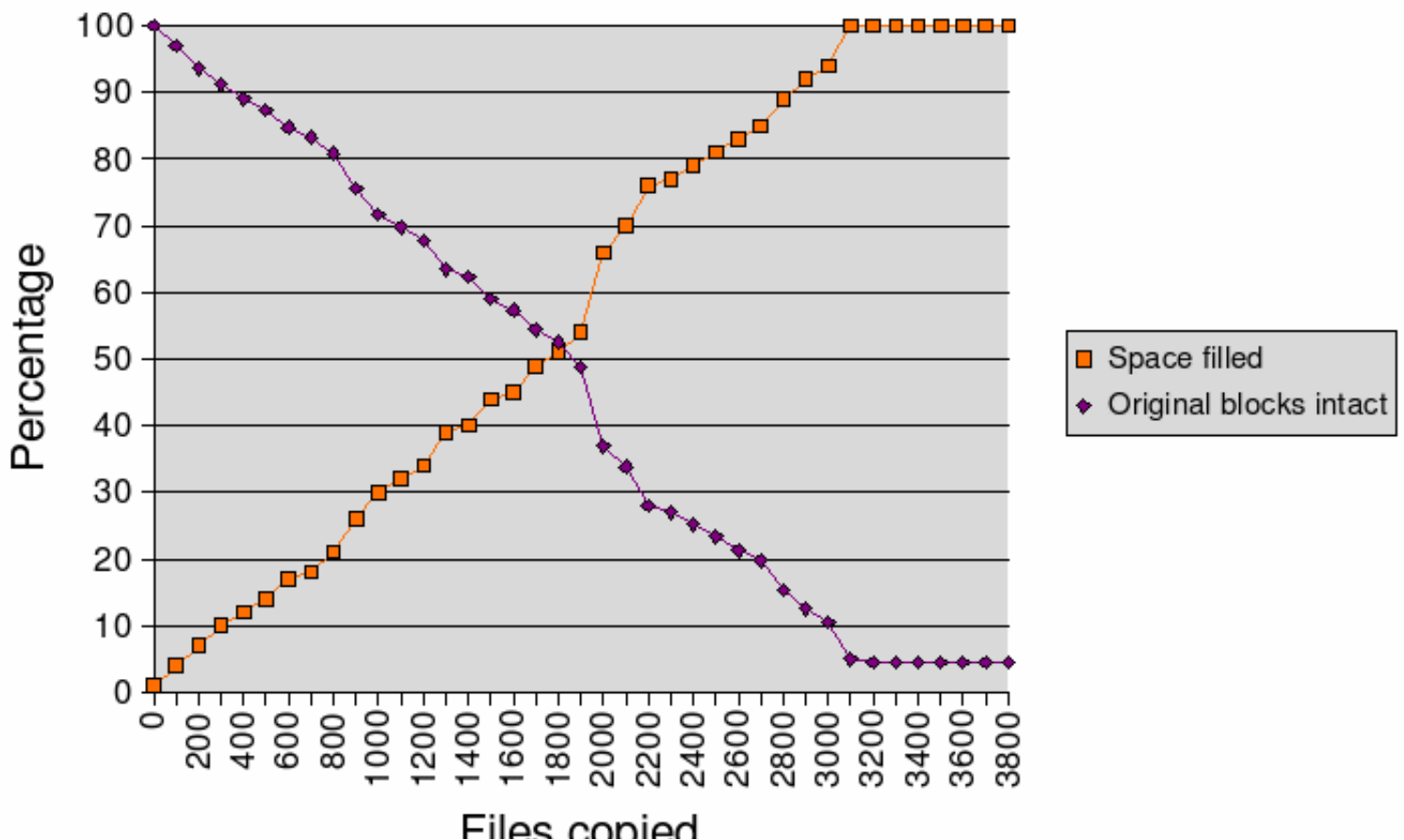
The problem of sensitive data being leaked through the re-use of storage media is by now well-documented. This is unfortunately a reasonably common occurrence, as shown by various stories of sensitive media being lost or sold ([1], [2], [3]). However the problem isn't just limited to those files which are left intact when the media is disposed of. To quote Wikipedia: "Slack space or file slack is the area between the end of a file and the end of the last cluster or sector used by that file. This area is simply wasted storage potential, so file systems that use smaller clusters utilize the disk space more effectively." [4]. You will notice this if you have lots of files which are very small; a correspondingly large amount of space on your disk will be wasted. However a greater problem is that some of your data which you thought had been overwritten, is still available to any casual snoopers who come into possession of your storage media. This includes attackers who manage to obtain root access to one of your servers, even if they are not physically present.

How bad is the problem?

In a simple filesystem, with 1Kb blocks, if a file of 973 bytes is overwritten by a file of 744 bytes, there will be 229 bytes of the original file which will not be overwritten. Most files are not this short, but the same effect will happen with the last block. Depending on what your data is, this may or may not be a problem; a comma separated file comprising names, addresses and other confidential data may leak too much data for comfort in this situation.

We analyzed a particular model to discover how long it takes blocks in a filesystem to be recycled. The experimental method is to write a string into each 512-byte block in a 64Mb VFAT filesystem - this was chosen because it corresponds to a USB pen drive I possess. While most pen drives are now bigger than 64Mb, the increase in size may well make things worse, as there is less chance of overwriting a particular block. Then files were copied randomly from the filesystem of a Linux installation to the USB pen drive.

Decay of filesystem blocks over time



Files copied

Figure 1. Decay of file system blocks as files are copied to the partition. No files are deleted.

In Figure 1, we created a file containing a particular string in each 512-byte block. The VFAT filesystem was then created within this file, and files were repeatedly copied from the hard disk onto the pen drive until it was full. You can see that by the time 3,200 files have been copied the majority of the original blocks have been overwritten. The remaining ones, around 4-5%, are assumed to be in the slack space on the file system. Obviously file systems which tend to have smaller blocks will have less slack space and therefore will leak less data in this manner.

Decay of filesystem blocks over time

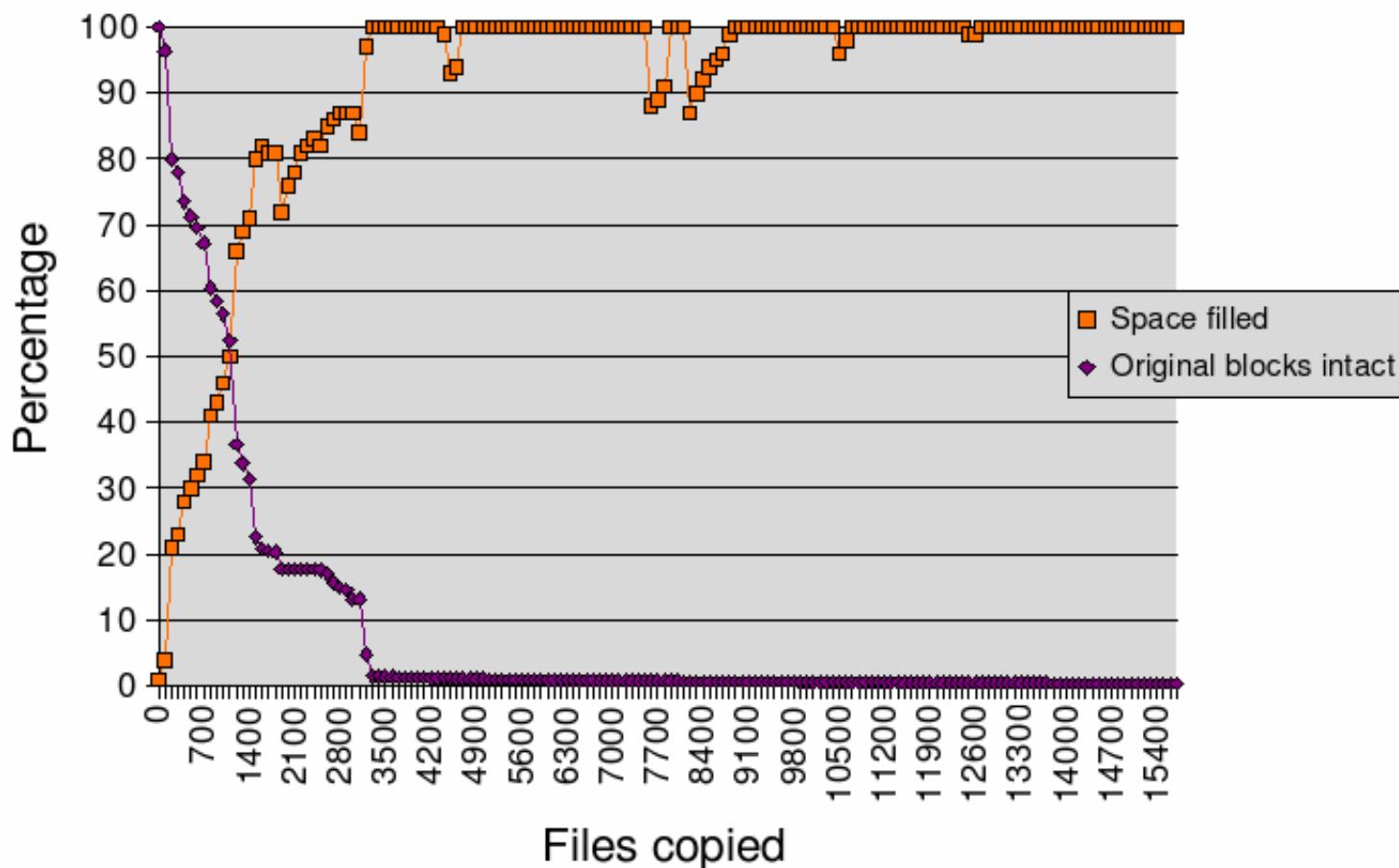


Figure 2. Decay of file system blocks over time. Files are being deleted and new files copied. In Figure 2, we have altered the simulation so that it deletes files randomly while continuing to copy new files to the file system, so that it doesn't remain full and hence static as in the previous example. Although the number of original blocks left intact appears to drop off to near zero, we can see more detail in Figure 3 below:

Decay of filesystem blocks over time

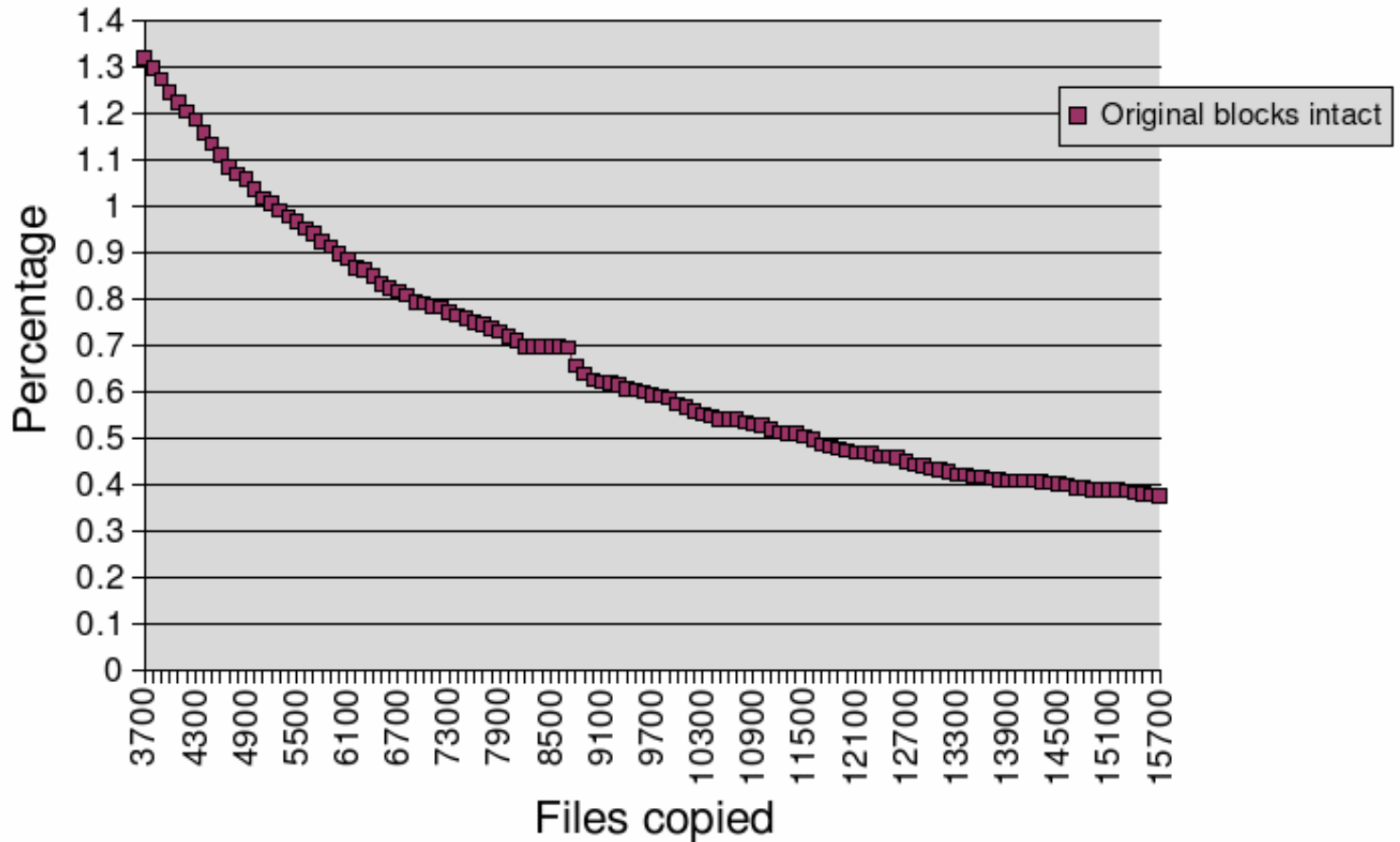


Figure 3. Decay of file system blocks over time. Files are being deleted and new files copied. Figure 3 is an enlargement of part of the graph in Figure 2. This shows that between 3,700 and 15,700 files copied the number of blocks intact is only decaying gradually. At the end of the simulation there are still 0.4% of the original blocks left, which could be a considerable amount of data on a larger disk. (0.4% of a 2Gb pen drive is 8Mb.)

Decay of filesystem blocks over time

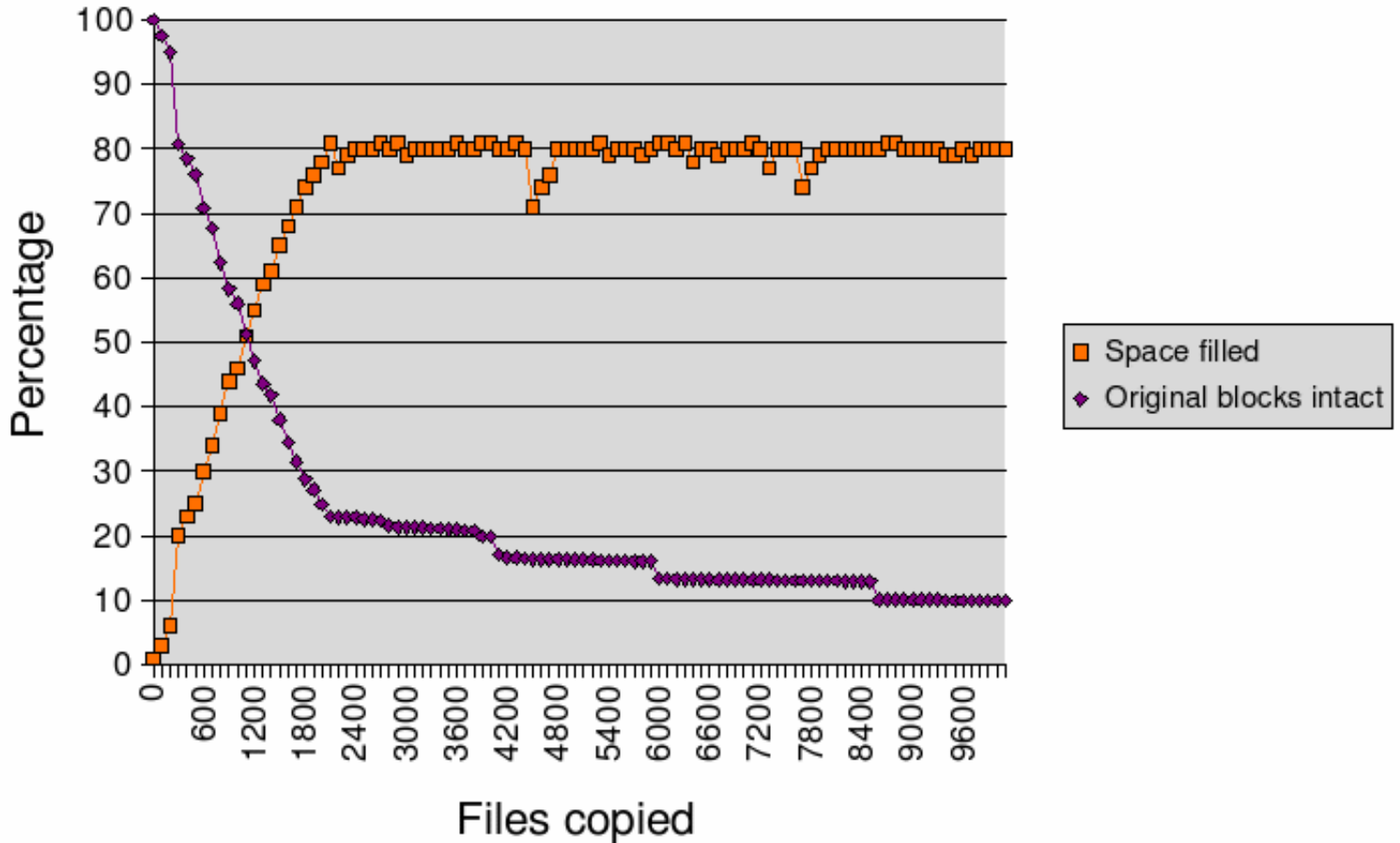


Figure 4. Decay of file system blocks over time, files are deleted when free space is less than 20%.

In Figure 4, every time the free space drops below 20% we delete files until it is above 20% again, and then proceed with the simulation. Although this is not completely realistic, it is a reasonable model for normal usage of a pen drive. You can see a fast initial decay of the number of original blocks, which slows considerably as it nears the 20% mark. It then declines more gradually, with occasional downward steps. Worryingly, we can expect normal hard disk or pen drive usage to be more like Figure 4 - copying and deleting files while maintaining a certain amount of headroom.

Countermeasures

The best thing to do is use disk or file encryption software. For example, using [Truecrypt](#), you can allocate the space on your pen drive to a truecrypt encrypted volume. It can then be mounted on each computer you need to use it on, and decrypted on the fly. There is still the potential for some data to be stored in unencrypted form on the computer, such as in swap space (the page file), or in caches but the problem is greatly reduced. If you don't want to use truecrypt, there are alternatives, but please choose something that uses peer-reviewed algorithms such as AES for encryption, HMAC/SHA-1 for message integrity, etc. and not Acme Proprietary WonderCrypt-128(TM). Remember also that a secure encryption algorithm is useless if you use it with an insecure key.

If you can't use disk or file encryption software, or your drives are always within a secure facility, you still need to take some action before disposal, or re-assignment. In this case it is probably best to securely erase every hard disk, whether it strictly requires it or not. For individual files, there are tools such as [SDelete](#) for Windows or [wipe](#) for Linux. These will overwrite a file with zeros, or ones or bit patterns to ensure that the data is certainly not going to be readable by a casual attacker. Wipe will also securely erase whole partitions,

for example:

```
wipe -k /dev/hda1
```

will securely erase the first partition of your first IDE disk. This latter method is probably the best if you need to deal with journal-ing filesystems such as ReiserFS and ext3 as you cannot guarantee that information about the file is not held within the journal. To erase a whole disk, there are various bootable floppy or CD images available, such as [Darik's Boot and Nuke](#). These will obviously erase Windows and DOS partitions equally well as UNIX.

Of course the hard disk may have remapped some bad sectors and there may be bits of data available in these. An attacker who can perform a surface scan of the disk may be able to recover some confidential information. This is a good argument for either using disk encryption from the start, or destroying the data by other means; however disk encryption is generally easier than degaussing or heating your media to 1500 Kelvin.

For more information on securely erasing media, see Peter Gutmann's fascinating paper [Secure Deletion of Data from Magnetic and Solid-State Memory](#). Farmer and Venema's excellent book [Forensic Discovery](#) also deals with the topic of persistence of data on disk and in memory, as well as many other important matters.

Summary

A relatively large amount of data can persist in slack space for long periods. Whether this is an issue for you will depend on the sensitivity of your data - for example, the text of your upcoming financial results is very important to keep confidential, where as an audit of your buildings waste water systems is less likely to cause problems were it to be made public. Another factor is locality; if your data consists of records of customers, then a single record could quite easily be recovered from each block of slack space. Compare this to the secret source code for your new application - unless your competitor recovers a substantial amount of contiguous source code, they are probably better off writing their own version. Finally, your industry may also be subject to general or specific data protection and privacy laws and guidelines. All of these factors need to be taken into consideration when doing a risk assessment on the possibility of leaking confidential or secret data.

About the author

[James Riden](#) is currently writing embedded software in Bristol, UK. He is a member of the [UK Honeynet Project](#) and holds a CISSP.

Reprints or translations

Reprint or translation requests require [prior approval](#) from SecurityFocus.

© 2007 SecurityFocus

Comments?

Public comments for Infocus technical articles, as shown below, require technical merit to be published. General comments, article suggestions and feedback are encouraged but should be sent to the [editorial team](#) instead.

[Privacy Statement](#)

Copyright 2006, SecurityFocus