Annex:
Deletion methods

Introduction

Manufacturers of systems or components focus on preventing data loss as a consequence of human errors, electrostatic charge, cosmic radiation or component failure. When developing a ballot printer, the very intention is that no permanent storage is desirable, just temporary storage and total control over whether the data is accessible. Technologies that have been designed in such a way that they actually restrict storage or accessibility in a verified manner is only used in a number of niche markets. As a result of this, the development and availability of this type of technology is extremely limited. A relatively well-known example is the banking sector, which is the driving force behind the development of smart cards and HSMs, components that guarantee the safe and verified storage and use of cryptographic keys.

If a software or hardware component in standard computer architecture 'deletes' data, generally it is only the reference to that data that is deleted from a central index. This means that the data itself is still there, but is no longer recorded as stored and the space that contains the data is available for storing new data. Until data is overwritten by other data nothing has actually been done to delete the data. The reason for this is that genuine deletion is usually not necessary, and deletion from the central index is much quicker. Even reformatting a disk, for instance, provides few guarantees. The fact that data is not really deleted in many cases is also the basis for much successful forensic investigation into 'lost' data.

This annex first gives an idea of the data that could remain after deletion. Then it looks at the methods for genuinely deleting data.

Where could data be left behind

As stated in the introduction, data can still be present in its original location after deletion because it is only the reference to the data that is deleted. This applies to, for instance, data that is processed in the internal memory and data that is stored in a file. In addition, standard software and hardware that is used for processing data can cause copies of processed data or parts of it to remain in other locations. This occurs during processing of data and can be caused by the operating system, database management system, standard software modules, a printer or a screen. Examples of locations where processed data could remain are:

- Log files, where the processing of data is logged. Log files are used at a later stage to determine that the software has functioned correctly or to find the cause during problem resolution.
- Transaction files, where processed data is stored temporarily until data processing has been fully completed or to allow the data processing to be rolled back. Transaction files are used to make data processing robust if anomalous situations arise, such as loss of power or when errors occur during data processing; then the result of the ongoing but not yet completed data processing is rolled back based on the data in the transaction file.
- Memory overflow files (known as a page file), where data from the internal volatile memory is stored temporarily if there is insufficient internal memory for this. In situations where the operating system crashes a complete copy of the contents of the volatile internal memory could end up in a file, also known as a core dump.
- Cache, parts of the internal volatile (non-permanent) memory, that is used to store read-in data or store data that has been written to permanent memory, so that it is quickly available if needed once again.

- Index files, that are created to be able to quickly search for data. Parts of the processed data could end up in these files. A variant of this is statistic data that is kept for reporting search actions or to speed up search actions.
- Print files, containing data that is sent to the printer.
- Backup files, where a copy of the data is stored to be able to restore the data to a previous state in the event of disaster involving the equipment or in case of errors in data processing.
- In peripherals that are used, for instance in the buffer of a printer, a hard drive cache, the internal memory of a screen controller or the internal memory that is shared by the screen controller and the processor in the processing unit.
- In a database management system, such as a SQL database. Here too, when data is deleted it could be that just the access to the data is removed from indexes but the actual data could still be reconstructed. In addition, a database management system will store the data or parts of the data in multiple locations for processing and accessing the data. The reason for this is, for instance, to be able to roll-back transactions (transaction log), search for data (indexes) and for sorting data. In this way, even after the data itself has been deleted from the database the data derived from it could still exist elsewhere in the database.

A complicating factor is that for software tools, standard software, the operating system and equipment is not or limited documented where processed data can be stored. Where there is a lack of information about this, analysis of the source code or tests will have to be used to determine where data could remain. When doing so, it is not only the normal course of events that must be examined, but also unusual (error) situations, such as data processing being aborted where data could for instance remain behind in an error log, transaction log or memory dump.

Deletion methods

The best method for not having to delete data is not storing it. However, when the vote choices are being processed by a ballot printer, the vote choice will have to be stored in the volatile internal memory at least. Once the vote choice has been printed, the vote choice has to be deleted from the internal memory. Merely releasing the memory is not sufficient because then the choice is still there, therefore the vote choice in the internal memory will have to be overwritten.

In addition, it will be necessary to avoid data that does not have to be stored ending up in other locations and then still being stored there. This can be done by implementing the following measures, for instance:

- Turning off logging where not required, and where logging is used by not including any data from which the vote choice can be derived in the log.
- Changing settings in the operating system so that no traces are created of data that has to be deleted later, such as disabling memory overflow, disabling core dumps, disabling data caching, disabling data indexing, printing directly instead of through a print spooler and disabling backups. Such settings should be disabled with discretion because it could have consequences for, among other things, processing speed and robustness.
- Using peripherals (such as printers and a screen) that do not temporarily store the data that has to be or has been processed in their own memory but processes data directly.

Where deleted data being left behind or residual traces of it cannot be prevented and this data can be retrieved using tools, the data to be deleted will have to be cleaned-up. The following measures could be implemented to do this, for instance:

- Using standard software that deletes data and removes traces of it – by overwriting it for example – at the time that the data is deleted.

- Using an internal command in a database management system, storage medium or device to clean-up processed data and traces of it.
- Processing ‘dummy’ data to purge data from a transaction log, cache or processing memory.
- Temporarily switching off components for a sufficient length of time to make stored data disappear.

The effectiveness of a deletion method that is to be used will have to be tested exhaustively. This is because there could be unusual (error) situations in which not all data or traces of it are cleaned up. When testing the effectiveness of a deletion method, forensic methods will have to be employed to check if deleted data still exists, and if so where.