Recovering previous versions of Microsoft Word documents

Murilo Tito Pereira and Alexandre Cardoso de Barros
Brazilian Federal Police
{murilo.mtp , alexandre.acb}@dpf.gov.br

Abstract— The Microsoft Word is one of the most utilized editors in the world. Most of the time, the analysis of a Microsoft Word document is just simple as verifying if its content interests to the investigation, and, sometimes, extract some metadata. We show in this paper, that Microsoft Word can, under some circumstances, store older versions of a document. We also present a technique to extract these versions, where our program prove to be efficient to recover up to 12 older versions of a document.

***Index Terms*—Microsoft Word previous versions**

I. Introduction

The Microsoft Word is today one of the most utilized editors in the world. Due to this, the great majority of the medias seized in police procedures or in private investigations includes several files created with this program. Usually, the analysis of these documents is just verifying the relevance of the content to the case in subject and reporting it or not. Other information, called metadata, can also be extracted, such as author, company, title, date and hour of creation, modification, impression, etc.

However, we observed that, in some cases, the size in bytes of a Microsoft Word binary file increases, even if part of the text is deleted in several sessions of work. Opening the file with a hexadecimal editor, we verified that the deleted text still continues stored, in spite of not being more used by the editor.

The objective of this work was to study the binary structure of the files of Microsoft Word documents and to develop a tool to recover the texts already deleted, but still presents inside of the binary, what we called previous versions of the current document.

II. Structure of a Microsoft Word Document

We found published the binary formats of the documents of Microsoft Word 6.0 and of Microsoft Word 97 (version 8) in the site www.wotsit.org, however we did not find formats of more recent versions. The information presented in this work base on the binary format of the document of Microsoft Word 97 (version 8) [2], but tests and other references [3] show that the versions, until the Word 2003, possess similar structure. Microsoft announced that next version of Microsoft Word (Word 2007) will use XML as file format, altering the extension of .DOC to .DOCX, leaving the option to also utilize the format of the Word 97-2003 [3,4].

The information presented in this section are quite summarized and simplified to facilitate the understanding. More details are available in [2].

The documents created with Microsoft Word are binary files that follow the Microsoft OLE 2.0 (Object Linking and Embedding). This means that the Word uses the functions and procedures defined by standard OLE, which are available as a programming interface (API), to create and maintain a document. Thus, to access specific information inside a binary Word file, we should open it and to examine it through existent functions in the library of functions of API OLE.

The Word binary file is composed of several streams, and the ones that interest us are: main stream, summary information stream and table stream. The main stream consists of the Word file header (FIB - File Information Block), the text, and the formatting information. The FIB gives the beginning offset and lengths of the document's text stream and subsidiary data structures within the file. The summary information stream stores basic information, like the author name and company. The table stream containing several data structures that describes a document. In some cases, it is made a backup, and exists two streams, called 0Table and 1Table. The data structure that most interests us is the piece table, because it describes the logical sequence of the characters in a document. We called physical position of a character its position in the binary file and logical position its position in the document that the file represents, that is its position as visualized by the user in the moment of the edition with the Word.

III. Analysis of a Microsoft Word document

The analysis of documents produced with Microsoft Word or with other text editor is just, in most of the cases, verifying if its content links with the investigation in process. In some circumstances, it is also interesting to observe other information (metadata) embedded in the summary of the document, like, for example, author, company and date and hour of creation, modification and impression of the document. However, there are cases that the simple knowledge of the current content of the document and of the metadata attachment is not enough to form proofs. An example of that would be a case for which is important to know if the document suffered previous editions and which were the content of these previous versions. Let us imagine the situation where the analyzed document presents the content below:

" To Mr. John.
Please, make a deposit of US$ 100.000,00 in the account number 1234-5, Bahamas Bank, on the 10/10/2000.
Mr. Smith. "

Probably, a document of this type was sent by fax to Mr. John, whenever Mr. Smith needed of a deposit. Mr. Smith's normal procedure, whenever he needed a new deposit, it was open the document above and to alter the data that interest, as date, value, and account, and to send a fax to Mr. John with the new solicitation. As the document was altered and possibly safe, a new document won't be created. In a simple analysis, only the last deposit solicitation would be visualized. Probably, however, several solicitations based on this document may have happened before. Our work intends to present a solution for this demand of deeper Microsoft Word documents analysis.

IV. Proposed Solution

Motivated by the problem presented in the previous section, we studied the Word files based on empiric tests and in [2]. As our work involved a lot of Word non documented actions, they can vary from version to version.

The first verification during the work was that when the program option "Allow fast saves" is enabled and a certain document suffered several editions, the respective texts of previous editions will still exist in the binary of the file. That is easy to verify, just editing a document, and saving it some times. After that, if we open the document in a hex editor, the document may contain text that was previously deleted. Microsoft has already noted this behavior [6]. The forensic expert, however, doesn't know where those texts were located. It is important to stand out that the indexed search tools usually work on the text that those files represents, not indexing words that were deleted, but that can be inside of the binary of the file. Obviously a search in the whole content of the file would locate the deleted words.

The following step was to discover if, in some way, the piece table (table that describes the logical sequence of the characters in a document) registered these alterations. Even so, we verified that the piece table is static, storing only one version of the document. We also verified that Word uses two table streams, 0Table and 1Table, alternately to save the alterations in a complex document. This way, there would be at least two piece tables, one for the current document and another for the previous version, being possible, then, to recover the last version of the document.

In our tests, we observed that the physical location of the piece table inside of the file moves when it is created again, and the new one does not wipe the old table. With that, we started to seek for the previous piece tables and we verified that they continued intact inside of the file. That scenery is similar to what happens in FAT/NTFS file systems when a file is deleted: the file is marked as deleted, but its content and the reference to the file continue in the disk. The act of recovering the previous piece tables allows easily to extract the previous versions of the document since the piece table describes the logical sequence of the characters in a document and the characters in a complex document are not erased.

The structure of the piece table consists of two vectors of n positions, where n is the amount of disjoint blocks of text, stored sequentially. The first vector, called CP (Character Position), defines the partitioning of the document in non-continuous parts. The second vector, called PCD (Piece Descriptors), registers the physical position of each part of the text indexed by the CP vector. PCD also keeps the references for the formatting information of the text.

An indicator of one byte, represented by the number two, and the size in bytes of the piece table are stored before the vectors. An example of the vectors of the piece table is shown bellow, not considering the formatting information of PCD:

Index	1	2	3	4	5	6	7
CP Vector	0	5	16	24	30	50	60
PCD Vector	900	950	870	920	1000	1200	1100

This example shows that the document is divided in 7 text blocks. The first block, that stores the characters from 0 to 4, is located starting at the physical position 900. The second block, that stores the characters from 5 to 15, is located starting at the physical position 950, and so forth.

We developed an algorithm to locate possible piece tables inside of the table streams. The algorithm scans the whole

space of the table streams, seeking for a data structure similar to a piece table. Finding such structure, the text referenced is extracted as a previous version of the document. Step by step, we have the following:

1. Seek for the indicator 2;
2. Read 4 bytes, corresponding to the size in bytes of the piece table. This value should be smaller than the size of the file so that it is valid.
3. With the size, the amount of positions of the vector is calculated, that we called n, knowing that each position of CP occupies 4 bytes and each of PCD 8 bytes. The result for n should be integer so that it is valid.
4. Read n positions of 4 bytes and to verify if it is in ascending order.
5. Case all the previous steps succeed we considered that we found a piece table, and the text regarding it is extracted.

This algorithm was implemented in C and the corresponding program gets to recover with success texts of previous editions, since the document is complex and elaborated in the Word versions 97 to 2003. Microsoft says that occurs up to 15 fast save actions before a document is reconstructed [5], so it is, in thesis, possible to recover up to 15 versions. In tests, we got to recover up to 12 texts, being one of them the current version, because our program doesn't make distinction. It is not possible to determine a chronological order for the versions neither a pattern of behavior, because it is a not documented action of Word. In our tests it was not found any piece table by mistake, showing that the restrictions imposed above are enough to find the correct piece tables, without incurring in false positive. The largest limitation of the program is not recovery formatting information and objects (illustrations, graphs, videos, etc), that was left for a future work.

Besides the text, the program also extracts the list of the users' names that saved the document and the respective save directories. This function is documented in [2].

The tool is available for police institutions, by contacting the authors.

V. Conclusion

In this work we presented a technique that can be used by Forensic Computer experts to recover previous versions of Microsoft Word documents. These informations exist inside of the document binary file, even so in a way not documented and not accessible to the user. It is important to stand out that the indexation tools usually work on the updated text only, and they won't index words of previous versions.

We showed in which situations the Word store the previous versions, how the developed algorithm works, the results and the limitations of the developed tool. We left as future work the recovery of formatting information and objects (illustrations, graphs, videos, etc) of the document.

References

[1] *Microsoft Word 6.0 Binary File Format*, available at http://www.wotsit.org/download.asp?f=wword60.
[2] *Microsoft Word 97 Binary File Format*, available at http://www.wotsit.org/download.asp?f=word8.
[3] *Walkthrough: Word 2007 XML Format*, available at http://msdn2.microsoft.com/en-us/library/ms771890.aspx.
[4] *What's New for Developers in Word 2007*, available at http://msdn2.microsoft.com/en-us/library/ms406055.aspx.
[5] *Frequently Asked Questions About "Allow Fast Saves"*, available at http://support.microsoft.com/kb/291181.
[6] *Word Document That Is Opened in Text Editor Displays Deleted Text,* available at http://support.microsoft.com/kb/287081/EN-US.