On the Investigation of Application Specific Data within Digital Forensics

Harald Baier and Achim Brand

Center for Advanced Security Research Darmstadt (CASED),
Hochschule Darmstadt, Darmstadt, Germany
e-mail: harald.baier@cased.de, achim.brand@stud.h-da.de

Abstract

Microsoft Word and Skype are widespread applications in our daily IT life. Up to now, if a computer forensic examination is required, the majority of forensic investigators tends to use commercial software to analyse this application-specific data. However, commercial software is rather expensive and typically closed-source. This paper aims at exploring if an application-specific forensic investigation is feasible by using free available software and whether its findings then still meet the investigators' demands. We contribute to this question by developing a guideline for the forensic investigation of Microsoft Word binary files (aka .doc files) and Skype chat log files. Solely free of charge available tools are proposed for use. In addition, we develop a Python-based, platform independent tool to enable a more in-depth-analysis of .doc-metadata. This tool does not rely on any third-party application libraries (e.g. Microsoft APIs (Application Programming Interfaces)). Furthermore we optimise an existing tool for analysing Skype's .dat files by reverse-engineering the file's structure. Finally, we present a questionnaire completed by 4 experienced practitioners. In spite of the small number of participants their answers underline that our approach meets their needs.

Keywords

Forensic Investigation, Application Forensics, Microsoft Word, Skype, Binary Files, Guideline, Open-Source Software, Anti-Forensics

1. Introduction

Application-specific forensics is a rather new branch in the computer forensic community, but it is very crucial. This is obviously due to the fact that the very basis of all our forensic investigation activities is for the accurate extraction of information from computer-based systems, such that it may be presented as acceptable evidence in a trial (Sammes and Jenkinson, 2007), (Geschonneck, 2011). Therefore, we need to extract all possible information from an application-specific file: the actual content of a file and all its relevant metadata. Thus, we can explore relationships between common actions and associated application metadata (Casey, 2010), (Marshall, 2008). These results can be the cutting edge to solve a forensic case.

Office applications (e.g. Microsoft word), browsers (e.g. Internet Explorer, Firefox), mail clients (e.g. Microsoft Outlook, Thunderbird), or Instant Messaging (e.g. Skype) are of central importance in our daily use of computers. Apart from the

standard usage of these applications, some people use them for illegal practices. If this is the case and a trial is supposed to take place, a computer forensic investigator must examine the data which was created by the use of these applications. The majority of forensic practitioners tends to use commercial software which can possibly be rather costly (e.g. standard tools like EnCase, FTK or X-Ways). Additionally commercial software typically is closed-source and hence cannot be inspected (e.g. if we can trust its functionality).

Thus, the question is raised whether it is also possible to conduct such an investigation solely by using free available software and whether its findings then still meet the investigators' demands. As a first contribution we therefore develop a questionnaire, which we sent to established German forensic practitioners (including law enforcement people). Unfortunately only 4 of them answered our questions (though the results are not universally valid, they give us a hint that we are on the right track with our approach). The main statement of their answers is that they have to investigate Microsoft Word and Skype files on a regular basis, they use a variety of rather expensive tools for those purposes, and they are not able to extract all intended information by their current tools.

Additionally this paper contributes to application-specific forensics by developing a guideline for the forensic investigation of Microsoft Word binary files (aka .doc files) and Skype chat log files. Solely free of charge available tools are used in the guideline. In case of Microsoft Word we develop a Python-based, platform independent tool to enable a more in-depth-analysis of .doc-metadata. We call this tool `wordmetadata.py`. Our tool does not rely on any third-party application libraries (e.g. Microsoft APIs). Its source code is open and can thus be trusted. Our tool is superior to previous tools not only due to its simplicity and independence, but also due to its functionality: it is designed to read more metadata from a .doc file than other tools do. We will show that in contrast to current software our tool is able to detect some anti-forensic measures of .doc files.

Finally, reverse engineering of Skype's .dat files is conducted and the file's structure is described. Moreover, the results of our reverse engineering processing are used to improve an existing analysis tool for Skype, the Skype Chatsync Reader. All our tools are available via `www.dasec.h-da.de`.

The rest of this paper is organised as follows: we first present our questionnaire and the related main results in Section 2. Then we describe our guideline to investigate Microsoft word .doc files and our tool `wordmetadata.py` in Section 3. Next in Section 4 we show how to investigate Skype's .dat files. We close our paper with our conclusions in Section 5.

2. Questionnaire to Forensic Practitioners

In order to learn about the professional investigators' needs for an accurate examination of Microsoft Word and Skype files, we set up a questionnaire, which we sent to experienced forensic practitioners (including law enforcement people) in Germany. They were enquired about their daily work including questions on whether

they regularly have to investigate Microsoft Word and Skype files, which tools they usually use for this purpose, which information they are mostly looking for and if they had already had a forensic case where revealed information of these applications helped to solve the case. The answers were given anonymously.

The number of experienced IT forensic practitioners is, however, small. From the 11 contacted persons 4 filled out the questionnaire. Although this is only a small amount of answers, the informative value is nevertheless high due to the huge experience of the investigators in question. In order to stress the qualitative rather than quantitative property of our results, we do not use a percentage presentation of the results.

Some general results of our survey are listed in Table 1. The investigators had to investigate Microsoft Word and Skype files, and in fact, they have to do it on a regular basis. They use a variety of rather expensive tools to fulfil their task. Some forensic cases were mentioned in which information originating from a Word/Skype file helped to solve the case. Finally, as not all relevant information may be extracted, the interviewees appreciate a new application-specific tool.

Do you have to investigate Microsoft Word doc- and Skype-files on a regular basis?	Very often: 1 of 4, Often: 1 of 4, Regularly: 2 of 4
Which tools do you use for such an application-specific forensic investigation?	EnCase, FTK, Ways Application-specific like MS Word, OpenOffice, Skype
Are you able to extract all relevant information using these tools?	Yes: 2 of 4 No: 2 of 4

Table 1: General aspects of our questionnaire

Regarding Microsoft Word some results of our survey are given in Table 2. The relevant information for the forensic investigation is said to be the content, the title, the author, the comments, the last author, the creation date, the modification date, the last print date, the editing time, the used template, the reviewers, and contained VBA (Visual Basic for Applications) macros. Further interesting information includes the name and the model of the printer used, if the file was printed at all.

Some of the survey's interviewees were not able to extract all relevant information using their tools. As a consequence they appreciate a new tool. Missing points compared to existing applications are

- the capability of showing the byte-offset where the inspected information is stored within the doc-file,
- name and model of the utilised printer, and
- the GUID of the computer, where Microsoft Word was used.

Which data structures are of interest when investigating a Microsoft Word doc-file?	Content: 4 of 4 Title: 2 of 4 Author: 4 of 4 Comments: 3 of 4 Last editor: 2 of 4 Creation date: 4 of 4 Last changed: 4 of 4 Last printed: 3 of 4
Are you able to extract all relevant information using your tools?	Yes: 2 of 4 No: 2 of 4
Do you appreciate a new tool for investigating Microsoft Word doc-files, especially if additionally the byte-offset of the inspected data structure within the file is listed?	Yes: 3 of 4 No: 1 of 4

Table 2: Microsoft Word doc-file specific answers of the questionnaire

Finally, Table 3 lists some Skype specific aspects of our survey. The relevant information for the forensic investigation is said to be chat's counterpart, date and time of sent messages, the messages' content, the call's counterpart, the date, time, and duration of a call, the file name of a sent file, the file size of this file, the date and time of the transfer, and its duration. Besides, it is also interesting to learn whether the call was incoming or outgoing. In contrast to Microsoft Word doc-files the majority of the participants were not able to retrieve all relevant information by using their tools (only 1 of 4 was able to do so).

Which data structures are of interest when investigating a Skype file?	Chat communication parties: 4 of 4 Chat time stamps: 4 of 4 Chat content: 4 of 4 Call communication parties: 4 of 4 Call time stamps: 4 of 4 Call incoming / outgoing: 4 of 4 File transfer: Name and size: 4 of 4
Do you investigate only the binary dat-files, the database db-files or both?	Only dat-files: 0 of 4 Only db-files: 0 of 4 Both file types: 4 of 4
Are you able to extract all relevant information using your tools?	Yes: 1 of 4 No: 3 of 4

Table 3: Skype specific part of the questionnaire

3. Investigation of Microsoft Word doc-Files

For the investigation of Microsoft Word files it is essential to get, besides the actual content of the file, all the contained metadata, e.g. authors, subject, title, keywords, creation date/time, last saved date/time, last author, last printed date/time, printer name, reviewers and company or organisation name; only to name a few. MS Word stores some of them without any user's influence. In order to be able to conduct a profound forensic investigation of Word doc-files, it is necessary to know which metadata is stored within those files. As a matter of fact, each piece of these metadata could help to solve a forensic case in some way.

When starting our work we were surprised about missing published information on that topic. This paper only addresses Microsoft Word binary files, i.e. .doc-files. According to common market share overviews of Office software (Hümmer, 2011) MS Word has a market share of about 80 to 85% worldwide. Though introduced with Microsoft Office 2003 the XML-based file structure becomes the default format recently. Thus the binary Microsoft Word doc-format is currently the most important office format from a forensic point of view.

The results of our questionnaire of Section 2 show, that most forensic applications do not show all the possible information such as the reviewers, the printer name or the offset to the stored information. Furthermore, the majority of professional tools are closed source and there is no possibility to check if the tool works correctly besides some black box tests. Therefore, we developed a new Python-based open source tool `wordmetadata.py` to overcome these shortcomings.

In order to understand our tool we first give some short insights to .doc-files in Section 3.1. Then in Section 3.2 we present our guideline to investigate Microsoft Word binary files. For each investigation step we recommend a tool, which may be used free of charge. Finally, we discuss in detail our tool `wordmetadata.py` in Section 3.3.

3.1. Foundations of the Microsoft Word doc-File Structure

Word binary files are using OLE (Object Linking and Embedding) structured storage to manage the structure of the file format (Microsoft Corporation OLE Web Site, 2012). These files are also called *compound files*. The reason for using compound files is that traditional file systems encounter challenges when they attempt to efficiently store multiple kinds of objects in one document. Compound files provide a solution by implementing a simplified file system within a file.

Structured storage defines how to treat a single file as a hierarchical collection of two types of objects, *storage objects* and *stream objects*, which behave as directories and files, respectively. This reduces the overhead and performance penalties associated with storing separate objects in a flat file and also solves performance problems by eliminating the need to entirely rewrite a file when someone changes its content. If there has been a change, new data will be written in the next free space available in the file, and the storage object will update an internal structure which maintains the

locations of its storage and stream objects. In addition, structured storage enables end users to treat compound files as if they were a single file rather than several objects. Hence, these files can be copied, backed up, and e-mailed like any other ordinary single file.

A Word binary file uses several structures to organise the file (see e.g. Microsoft OLE 2012). Figure 1 shows the fundamental OLE structures and the byte offsets into the file as absolute offsets from the beginning of the file. The design resembles a classical FAT file system, where the DIFAT (double-indirect file allocation table) is used to find the FAT sectors in the compound files, the FAT is used to find the object chain (like a FAT in the FAT file system), and the mini FAT is used for streams (which are not relevant in our scope). Again like in a FAT file system, directory entries are used to address objects in the doc-file.

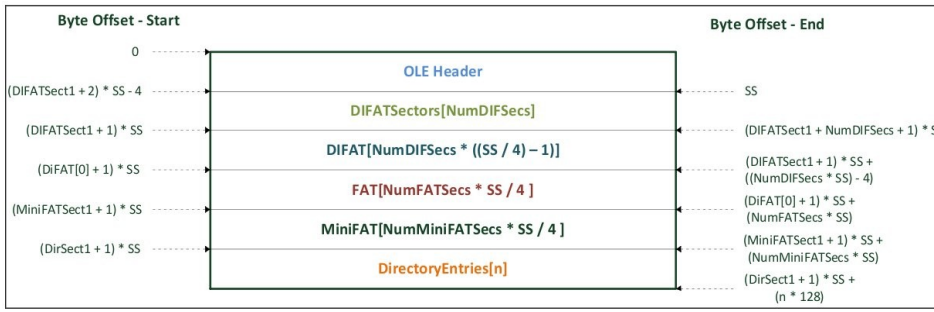

Figure 1: OLE objects in a Microsoft Word doc-file

3.2. Our Guideline to Investigate Microsoft Word doc-Files

In this section we present our guideline to investigate Microsoft Word doc-files. Our aim is to get hold of the content and the (forensic relevant) metadata of the files, just by using *free available software*, preferably open source software. The reason for this paradigm is reliability and cost-effectiveness of the tools. An additional rich source of open source tools is the web site (Open Source Forensics Web Site, 2012).

Currently we are not aware of any published guideline to investigate Microsoft doc-files and an enumeration of adequate freely available tools. Neither the well-known literature mentioned in this paper nor the up-to-date doc-section of the Forensics Wiki (Garfinkel, 2012) yields support for that. Our aim is to fill this gap.

Figure 2 shows a flow chart on which an investigator can rely on if he has to investigate .doc files. For each step we propose concrete tool(s) to perform this step. The reasoning about our tool choice and a short overview of its capacity is given in the Appendix. However, the overall design of the process model is straightforward.

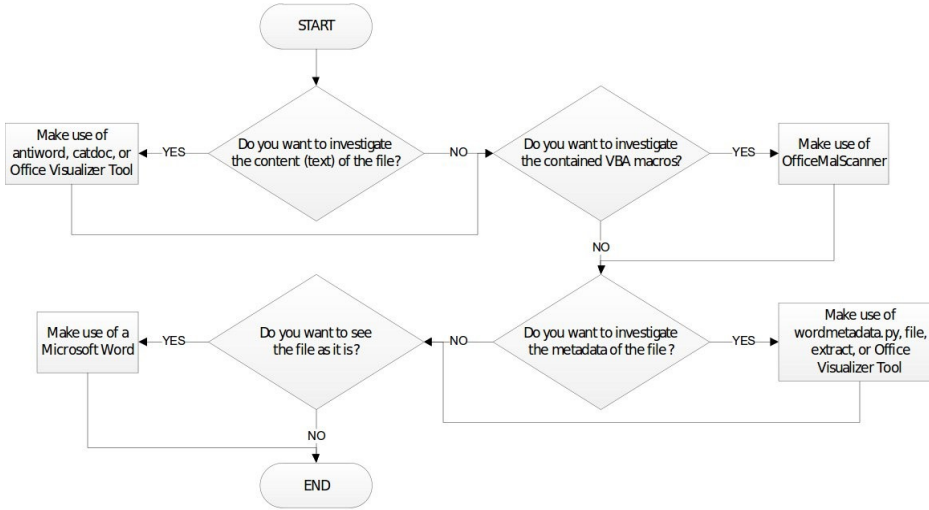

Figure 2: Guideline to investigate Microsoft Word doc-files

3.3. Our Tool wordmetadata.py

The answers of forensic practitioners presented in Section 2 reveal shortcomings of currently available software to analyse .doc files. We therefore developed a Python tool named `wordmetadata.py`. It is available via `www.dasec.h-da.de`. The aim of this tool is to gain as much forensic relevant metadata of a Word .doc file as possible and to be platform independent (i.e. to be usable on any common Operating System). Furthermore, this tool is based only on information provided by the MSDN (Microsoft Developer Network) and therefore it does not rely on any APIs. So, the correctness of the results is transparent and an investigator can rely on the tool.

With respect to the practitioners' needs our tool performs the following tasks on Word .doc files (a sample run of `wordmetadata.py` on a test file `hello.doc` is given in the Appendix):

- Show file system information: file system file size, creation date and time, last modified date and time, and last access date of the file.
- Show internal file size: the file size as given by the internal FAT.
- Show relevant metadata from the “DopBase”: creation date and time, last save date and time, revision number, editing time, word count, character count, and paragraph count. This is also the information which is often read by other tools to show the document’s metadata.
- Interpret and show all Summary Information Property Set (SIPS) metadata.

- Show byte offset to each value (absolute from the beginning of the file).
- Show number of reviewers and their names.
- Show last save date and time read from the File Information Base (FIB).
- Show printer information where the file was printed.
- Sanity check of file sizes (i.e. comparison of file system based file size to internal FAT stored file size) to recognize anti-forensics.

As the above-noted list and the sample output for `hello.doc` in the Appendix show, some redundancy regarding certain metadata can be noticed, e.g. the creation time is stored in the DopBase and in the SIPS. While analysing the doc-file structure it became apparent that all tested tools read the metadata solely from one source. In contrast, our tool reads metadata from several locations and there is one particular reason for doing so: to recognise anti-foreniscs.

As a matter of fact it could happen that a person tries to obfuscate some chargeable content or action by manually editing the file using a hex editor or an obfuscation tool, e.g. he tries to change the file's creation date and time. Due to the redundancy of this information (DopBase and SIPS), the person probably only changes one location and leaves the further locations unchanged. This obfuscation will be discovered using our `wordmetadata.py` tool. In addition, it might very well happening that he changes the date and time to an invalid value: the value of the seconds for SIPS times must always be ":00" (cf. green rectangle in the output in the Appendix) since Word stores time information in the SIPS only accurate to the minute level. Therefore, the time information in the SIPS plays a vital role and to conform to the specification the `wordmetadata.py` tool calculates these times exact to the second. Thus, if a time stamp within the SIPS contains any other value than ":00" for the seconds, a closer look for a potential anti-forensics manipulation is worth doing.

Showing the offset for each data structure of a doc-file is another advantage of our tool: this feature considerably helps an investigator or any other person comprehending the file structure of .doc files. First, the tool can be used to find the relevant data structure at the corresponding offset. Then, a hex editor/viewer can be used to review and evaluate the findings. Besides proving the correctness of investigation results this proceeding has a positive side-effect on the learning process of forensic relevant data structures. Besides, the findings of the survey in Section 2 demonstrated that three out of four investigators welcome such a feature.

The following example concerning the reviewer's information within a doc-file illustrates how this can be done:

```
sherlock@ubuntu:~$ xxd -s 1938441 -l 40 hello.doc
 01d9409: 0700 5500 6e00 6b00 6e00 6f00 7700 6e00  ..U.n.k.n.o.w.n.
 01d9419: 0300 6500 7600 6500 0300 6200 6f00 6200  ..e.v.e...b.o.b.
 01d9429: ffff 0300 0800 0000                      ........
```

From our sample output in the Appendix we know that the offset to the data structure containing the reviewers is 1,938,441 bytes (see section `Miscellaneous`

`Metadata` of the output). Now `xxd` can be used to seek to this offset and to show the actual value of the data structure. The reviewers are "Unknown", "eve", and "bob". This is correct as Word stores the user "Unknown" as a first reviewer by default into each file as soon as the "Track Changes" feature has been enabled. Thus, the user "Unknown" is no reviewer and therefore there are only two reviewers named "eve" and "bob" (as our tool claims).

A further benefit of using our tool is to perform a sanity check of the file size. When comparing the actual file size (file system file size) to the file size regarding the internal FAT, another way of doing anti-forensics can be identified. Hence, it is possible to append something (e.g. a picture) to an already existing .doc file. A "standard" check using the tools described in this paper would not reveal that there has been another file appended to the .doc file. This fact would not even be discovered by opening the .doc file using Word. As the following listing shows, a file called `picture.jpg` is appended to the test file `hello.doc`. A run of the `wordmetadata.py` on this modified `hello.doc` then reveals after the sanity check that there is an inconsistency corncerning the file.

```
sherlock@ubuntu:~$ cat picture.jpg >> hello.doc
sherlock@ubuntu:~$ python wordmetadata.py hello.doc
--------------------------------------------------
File system information about hello.doc:
--------------------------------------------------
Size:                 3900549 (bytes)
[REMOVED]
File Size regarding FAT:      1972736                 (bytes)
############################################################
# Caution: File size sanity check failed.                  #
# Actual file size is larger than file size regarding FAT. #
# Something could be hidden within the file.               #
############################################################
```

Furthermore our tool shows information about the printer which was used to print the file, provided that Word (or another Office application) stored this information (typically only older versions of Word behave so). The following listing shows a run of the tool on a file containing some printer information (our sample file `hello.doc` in the Appendix does not contain printer information as declared at the end of the corresponding output). Again we address a dedicated request of the practitioners as discussed in Section 2.

```
sherlock@ubuntu:~$ python wordmetadata.py foo-printed.doc
[REMOVED]
--------------------------------------------------
Printer Information:
Offset to Printer Information:       77812
--------------------------------------------------
Name:          \\srvfps\HP LaserJet 2420 Sekretariat
Port:          Ne05:
Driver:        HP LaserJet 2420 PCL 5e
Product Name:  HP LaserJet 2420 PCL 5e
--------------------------------------------------
End of Printer Information.
```

4. Investigation of Skype Log-Files

Although there are some publications available with respect to Skype network communication (Biondi and Desclaux, 2006), (Baset and Schulzrinne, 2006) for the forensic investigation of Skype log files it is essential to know all used file structures and their content. The latter, however, is not investigated in detail. Information which is of particular interest regarding Skype log files are the following: chat's counterpart, date and time of sent messages, the messages' content, the call's counterpart, the date, time, and duration of a call, the file name of a sent file, the file size of this file, the date and time of the transfer, and its duration. Besides, it is also interesting to learn whether the call was incoming or outgoing. This information is stored in different files, as the following section will describe.

We contribute to a dead-analysis of Skype log files by extending a common existing tool in Section 4.2. Before, we shortly explain in Section 4.1 the locations, where Skype saves forensic relevant persistant data.

4.1. Structure of Skype Log File Folders

The Skype log files are stored in the following folder (henceforth referred to as "log folder") which is dependent on the operating system:

- Windows 7: C:\Users\<osUser>\AppData\Roaming\Skype\<skypeUser>\
- Linux: /home/<osUser>/.Skype/<skypeUser>/

whereby <osUser> is the user name of the operating system user and <skypeUser> is the user name of the Skype user.

As to Windows versions of Skype, all conversations are stored in both, an SQLite3 database "…\main.db" and several binary .dat files within the "…\chatsync" folder. Due to the fact that there is no official documentation for Skype or for the structure of its log files, it remains unclear why Skype stores redundant information using the logs in two different formats.

The Linux versions of Skype, too, store all conversations in two various ways, in several binary .dat files within the "…\chatsync" folder and in several .dbb files within the log folder ("…\"). However, in contrast to Windows versions there is no "…\main.db".

The `main.db` is an SQLite3 file, which can be opened with any SQLite3 client to extract the relevant information. It comprises 19 tables. The most interesting ones are the tables *Messages* (all chat conversations are stored), *CallMembers* (all members of a call), *Calls* (Information about calls), *Contacts* (all Skype contacts), and *Transfers* (all file transfers). Details about an investigation of the `main.db` are given in (Brand, 2011).

For both versions (Windows and Linux) of Skype, there is another interesting file, the "…\config.xml" file, which contains the configuration of Skype and some further information. This file can be opened with any text editor or Internet browser and thus reveals its content. The "config.xml" file contains the configuration of Skype as well as other interesting information. The bulk of the file cannot be interpreted due to the lack of information about the structure. But there are two interesting things stored within this file (see sample `config.xml` in the Appendix): a UNIX timestamp showing the last time when Skype was used by the corresponding user is stored between the start tag "<LastUsed>" and the end tag "</LastUsed>" (which is in this example "20.07.2011 10:32:06 UTC"). In addition, all contacts (Skype user names) of the corresponding user (in this case Sherlock) are stored between the start tag "<u>" and end tag "</u>" (in this example only the Skype user "doktor_watson_001", which means he is Sherlock's only contact).

4.2. Reverse Engineering of Binary .dat File Structure

Due to the necessity of understanding the file structure of the binary .dat files and of providing its description, a programme *Skype Chatsync Reader* (SCR) was selected (Skype Chatsync Reader Web Site 1, 2012), (Skype Chatsync Reader Web Site 2, 2010). The Skype Chatsync Reader parses the log files and extracts the contained conversations. Based on the source code of SCR we were able to do a reverse engineering of the file structure and to describe it. We sketch our results in what follows and point to (Brand, 2011) for details.

The "Skype Chatsync Reader" is intended to work with files of Skype version 4.2.0.169. Therefore, this exact version of Skype is needed to perform an initial reverse engineering. As a next step, a test bed with three test users (Sherlock, Watson, and Alice) was created. This test bed contains, in addition to the (up to the present) newest versions for Windows and Linux, also the version of Skype that is supported by SCR. The reverse engineering of SCR's source code combined with a further reverse engineering of the .dat file structure has shown that SCR does not work properly. Therefore, we fixed the bugs in SCR, released an improved version of the SCR, and verification shows that our improved version works now without any failures (for details we refer to (Brand, 2011)).

The combinations of both reverse engineerings lead into a description of this structure. We released a table providing a complete overview of the identified data structures, which deal with the binary file structure of Skype .dat files. A mapping of this table to a sample hexdump is discussed in (Brand, 2011). Our improved SCR is available via `www.dasec.h-da.de`.

5. Conclusions

We have shown that currently the majority of forensic practitioners tends to use commercial software, but that they are not able to extract all intended information by their current tools. We therefore developed a guideline for the forensic investigation

of Microsoft Word binary files (aka .doc files) and Skype chat log files using solely free of charge available tools. In case of Microsoft Word we provide a Python-based, platform independent tool to enable a more in-depth-analysis of .doc-metadata. Our tool is superior to previous tools due to its simplicity, independence, and functionality. Finally, reverse engineering of Skype's .dat files lead to an improved version of the Skype Chatsync Reader. All our tools are available via `www.dasec.h-da.de`.

6. References

Brand, A (2011), “On the Investigation of Application Specific Data within Digital Forensics”, *Master's thesis*, Hochschule Darmstadt, available via www.dasec.h-da.de

Biondi, P. and Desclaux, F (2006), “Silver Needle in the Skype”, *BlackHat Europe*

Baset, S.A. and Schulzrinne, H.G. (2006), “An Analysis of the Skype Peer-to-Peer Internet Telephony”, *INFOCOM 2006,* 25th IEEE International Conference on Computer Communications, pp1-11

Casey E. (2010), “Handbook of Digital Forensics and Investigation”, *Academic Press*

Hümmer T. (2010), “International OpenOffice market shares - Portal – Tutorials, Tipps und Tricks für Webmaster auf Webmasterpro.de”, www.webmasterpro.de/portal/news/2010/02/05/international (Accessed 13 February 2012)

Garfinkel, S. (2012), "Forensics Wiki, a Creative Commons-licensed wiki devoted to information about digital forensics", www.forensicswiki.org, (Accessed 28 February 2012)

Geschonneck, A (2011), "Computer-Forensik: Computerstraftaten erkennen, ermitteln, aufklären", *dpunkt-Verlag*

Marshall, A (2008), "Digital Forensics – Digital Evidence in Crime Investigation", *Wiley-Blackwell*

Microsoft Corporation OLE Web Site (2012), "Compound File Binary File Format", msdn.microsoft.com/en-us/library/dd942027%28v=prot.13%29.aspx (Accessed 4 February 2012)

Open Source Forensics Web Site (2012), “Open Source Digital Forensics”, www2.opensourceforensics.org/tools/application (Accessed 27 February 2012)

Sammes, T. and Jenkinson, B. (2007), “Forensic Computing – A Practitioner's Guide”*, Springer-Verlag*

Skype Chatsync Reader Web Site 1 (2012), “Read Skype Data: Chatsync and SQLite”, itsecuritylab.eu/index.php/2010/07/07/read-skype-data-chatsync-and-sqlite/ (Accessed: 21 February 2012)

Skype Chatsync Reader Web Site 2 (2010), “Skype.dat reader is updated”, itsecuritylab.eu/index.php/tag/read-skype-chatsync-files/ (Accessed: 21 February 2012)

Appendix

Tools Used in the doc-Analysis-Guideline

Tool	Reason
antiword *(Open Source)*	Antiword is a free software reader for Linux that makes it possible for binary Microsoft Word documents to be read and to be converted into plain text, PostScript, and also into PDF (Portable Document Format) files. Besides, it recognizes pictures within the files and identifies them as "`[pic]`". This also holds for other embedded objects, e.g. an .mp3 or .wav file. Additionally, antiword is ported to several platforms, including Windows and DOS (Disk Operating System). A major asset of antiword is that it can display the content regardless of the used font, font color, effect, style, or any other formatting. This means that content which is not visible by default, such as hidden text or text with white font colour on white background, are being displayed just like everything else.
catdoc *(Open Source)*	Catdoc is quite similar to antiword. Catdoc, too, produces the text of the Word file as plain text. But there is no possibility to export the output to PostScript or PDF and it does not identify contained objects such as pictures or .mp3 files the way antiword does. However, there is an option "`-b`" to process also broken Word files and maybe this helps to read a broken file. Catdoc can be considered to be a second-verification tool in addition to antiword.
Office Visualizer Tool *(Free Available)*	The "Office Visualizer Tool" is a parser for Microsoft Office OLE structured files, i.e. Excel, PowerPoint, and Word binary files. It definitely offers a good possibility to have an organized view of a binary file. The tool is divided into two panes: The left pane (called "Raw File Contents") shows the raw content of the file (in hexadecimal values). The right pane (called "Parsing Results") shows the results from the parsing, i.e. the name of the current data structure, its value, byte offset within the file, size, and type. A click on a certain data structure within the "Parsing Results" indicates the corresponding raw data in the "Raw File Contents" pane and vice versa. This simplifies the discovery of a certain value a lot and helps an investigator to detect the wanted information rather quickly. Due to the high complexity of OLE structured files, the tool cannot interpret every data structure in a sufficient way as an investigator would need it. Thus, this must be done manually, which is far too inconvenient for

	practical usage in a forensic scope. On the other hand, the tool may just be the perfect choice for some other scenarios, e.g. if an investigator needs an offset to a certain value or if he needs to learn how the OLE file structure works in general.
OfficeMalScanner (Free Available)	With regard to an investigation of Word files it can also be interesting to find out which VBA macros are stored within these files. A highly comfortable way to extract the source code of contained macros provides the "OfficeMalScanner" tool for Windows. OfficeMalScanner is a Microsoft Office forensic tool that finds malicious shellcode within (legacy binary and new XML) Office files. Additionally, it saves identified VBA macro code to disk. Thus, an investigator can use this tool for both purposes: Checking Office files for malicious shellcode and extracting contained VBA macros without the risk of infecting one's own system, which would happen if the file was opened with Office. Furthermore, it prevents the performing of unwanted calls which could come from the VBA macro. Furthermore, this tool was developed by Frank Boldewin who is well-known in the field of digital forensics.
wordmetadata.py (Open Source)	See section "Our Tool wordmetadata.py".
file (Open Source)	The Linux/UNIX command "file" is commonly used to determine the file type of a certain file, e.g. if the file is an .mp3, .jpg or .png file. For .doc files it gives a short overview of the file's metadata. This command comes preinstalled with almost any version of Linux/UNIX and so it can be used for a very first inspection of the file's metadata.
extract (Open Source)	The "extract" tool was developed to read metadata from certain file types. The website of "extract" says it is able to read metadata from any files, which sounds a bit overbearing. But for the scope of this paper it works pretty well: It gives a short overview of the investigated file's metadata. Additionally, it shows even more information of older versions of .doc files: It is able to extract the revision log of a .doc file, in case there is one. This could be the cutting edge to solve a forensic case. Extract is a good choice if an investigator needs only some basic information about the file's metadata or, for older files, to read the revision log.
Microsoft Word	If the tools mentioned above do not evoke the wanted results, one of the last possibilities is to use Microsoft Word itself to open the document (read-only). The investigator will then

	see the document from the same point of view as the author or the suspicious person did. However, he has to take into account that he should be well-informed about all the features offered in Word (e.g. hiding pictures and text). Otherwise he might easily be deceived. Additionally, there might also be some VBA macros or other malicious code be contained in the file which could change the file or even infect the investigator's computer. Thus, using Word for the investigation of .doc files should never be an investigator's first choice.

A Sample Output of our doc-Metadata-Tool `wordmetadata.py`

```
sherlock@ubuntu:~$ python wordmetadata.py hello.doc

--------------------------------------------------
File system information about hello.doc:
--------------------------------------------------
Size:                  1972736 (bytes)
Creation Date:         2011-08-13 07:31:18 (UTC)
Last Modified Date:    2011-06-28 14:40:45 (UTC)
Last Access Date:      2011-08-13 07:31:18 (UTC)
--------------------------------------------------
End of file system information.
--------------------------------------------------
Header Signature:      0xD0CF11E0A1B11AE1     (OLE compound file header)
File Size regarding FAT:      1972736          (bytes)
Offset to 1Table:             1932800
File Identification:          0xA5EC  (Word Binary File)
--------------------------------------------------
DopBase Metadata:
--------------------------------------------------
Name:          Creation Date
Value:         2011-06-28 16:21 (UTC)
Offset:        1938923
--------------------------------------------------
Name:          Last Save Date
Value:         2011-06-28 16:40 (UTC)
Offset:        1938927
--------------------------------------------------
Name:          Last Printed Date
Value:         Never
Offset:        1938931
--------------------------------------------------
Name:          Revision Number
Value:         6
Offset:        1938935
--------------------------------------------------
Name:          Editing Time
Value:         0
Offset:        1938937
--------------------------------------------------
Name:          Word Count
Value:         9
Offset:        1938941
```

```
--------------------------------------------------
Name:          Character Count
Value:         63
Offset:        1938945
--------------------------------------------------
Name:          Page Count
Value:         1
Offset:        1938949
--------------------------------------------------
Name:          Paragraph Count
Value:         1
Offset:        1938951
--------------------------------------------------
End of DopBase Metadata.
--------------------------------------------------
--------------------------------------------------
Summary Information Property Set Metadata:
SIPS Size: 428 (bytes)
Number of Properties:  16
--------------------------------------------------
Name:                          CodePage
Value:                         000004E4
Offset to TPV packet:          1940152
--------------------------------------------------
Name:                          Title
Value:                         Hello World
Offset to TPV packet:          1940160
--------------------------------------------------
Name:                          Subject
Value:                         Application Layer Forensics
Offset to TPV packet:          1940180
--------------------------------------------------
Name:                          Author
Value:                         sherlock;watson
Offset to TPV packet:          1940216
--------------------------------------------------
Name:                          Keywords
Value:
Offset to TPV packet:          1940240
--------------------------------------------------
Name:                          Comments
Value:                         This files serves as investigation test
file.
Offset to TPV packet:          1940252
--------------------------------------------------
Name:                          Template
Value:                         Normal.dotm
Offset to TPV packet:          1940308
--------------------------------------------------
Name:                          Last Author
Value:                         alice
Offset to TPV packet:          1940328
--------------------------------------------------
Name:                          Revision Number
Value:                         6 (regarding to CodePage)
Offset to TPV packet:          1940344
--------------------------------------------------
Name:                          Application Name
Value:                         Microsoft Office Word
Offset to TPV packet:          1940356
--------------------------------------------------
```

```
Name:                           Creation Date
Value:                          2011-06-28 14:21:00 (UTC)
Offset to TPV packet:           1940388
-----------------------------------------------------
Name:                           Last Save Date
Value:                          2011-06-28 14:40:00 (UTC)
Offset to TPV packet:           1940400
-----------------------------------------------------
Name:                           Page Count
Value:                          1
Offset to TPV packet:           1940412
-----------------------------------------------------
Name:                           Word Count
Value:                          9
Offset to TPV packet:           1940420
-----------------------------------------------------
Name:                           Character Count
Value:                          63
Offset to TPV packet:           1940428
-----------------------------------------------------
Name:                           Document Security
Value:                          0
Offset to TPV packet:           1940436
-----------------------------------------------------
End of Summary Information Property Set Metadata.
-----------------------------------------------------
-----------------------------------------------------
Miscellaneous Metadata.
-----------------------------------------------------
Number of Reviewers:            2
Offset to Reviewers:            1938441
Reviewers:                      eve;bob
Last Save Date (from FIB):      2011-06-28 14:40:45 (UTC)
No Printer Information found.

--- end of script ---
```

A Sample Skype file config.xml

```
<?xml version="1.0"?>
<config version="1.0" serial="18" timestamp="1311157960.8">
  <Lib>
    <Account>
      <LastUsed>1311157926</LastUsed>
    </Account>
    <CentralStorage>
      <SyncSet>
...       <u>
          <doktor_watson_001>7bff23a8:2</doktor_watson_001>
        </u>
      </SyncSet>
    </CentralStorage>
</config>
```