Towards Carving PDF Files in the Main Memory

Ziad A. Al-Sharif[1] Dana N. Odeh[2] Mohammad I. Al-Saleh [2]
Software Engineering Department[1], Computer Science Department[2]
Jordan University of Science and Technology
Irbid, 22110, P.O. Box 3030, Jordan
zasharif@just.edu.jo, danaoudeh@yahoo.com, misaleh@just.edu.jo

ABSTRACT

Digital forensics concerns about extracting and analyzing the contents of digital devices. It is used to locate digital evidences in order to support legal actions against criminals in the court of law. This paper utilizes file carving techniques to extract digital evidences in the RAM about opened PDF files. This paper observes that PDF files consist of objects. Each object is marked with special indicators that mark its start and end. These indicators are used to locate and extract the objects of a PDF file from the RAM. We design several experiments and show that carving PDF files from the RAM is possible even after closing the PDF file viewer.

KEYWORDS

Digital Forensics, File Carving, Dumped Memory, PDF Objects, RAM.

1 INTRODUCTION

Digital forensics is a relatively new discipline for investigating crimes committed with the help of digital devices. It investigates the contents of electronic devices such as computers and smart phones. The investigation process includes seizing devices, data acquisitions, evidence extraction, and drawing conclusions. This can lead to identify criminals and discover their illegal activities. In order to be admissible in the court of law, digital evidences must be accurate. Inspecting files is inevitable in digital forensics because the Operating System stores most digital contents as files. Even though a file can be normally reconstructed with the help of the file system meta data, such meta data is not always available. However, file carving techniques are the solution to such a problem, where a file can be reconstructed again based on its internal structure.

File carving is very important in the digital investigation because it provides the ability to retrieve evidences that maybe useful to identify criminals and recognize their activities. File carving is the process of retrieving files even after it has been deleted from the device and their meta data are also lost after overwriting the data or having a media damage. Basically, file carving involves searching bulk data for blocks of certain structures and then correlating the found blocks in order to reconstruct the original file.

Until recently, investigators had only been interested in extracting evidences from permanent storage devices such as hard disks (HD) and solid state drives (SSD). However, main memory (RAM) forensics has been proven powerful in investigation. All operations, such as reading or writing to files, need to go through the RAM temporarily. This gives an idea how important it is to inspect the RAM. Particularity, File carving in the RAM is very useful given that how scattered memory contents are and that meta data does not necessarily reside there. Dumping memory during the investigation process might prove helpful in evidence extraction and file carving. A memory dump is an exact physical copy of the RAM. This paper seeks to extract PDF files from the RAM solely based on the PDF file structure. PDF files, in special, can be used on almost all platforms ranging from mobile phones and tablets to conventional desktops and laptops; regardless of

their operating systems and hardware specifics.

2 BACKGROUND

Various types of evidences can be obtained from the ephemeral information that is found in the RAM of a running machine. RAM forensics depends on the analysis of physical memory dumps. It is mainly used in the investigation of computers to obtain and locate evidences about live activities that otherwise cannot be obtained. These evidences may include (but are not limited to) network connections [7], recent web browsing, running processes, and opened files.

Memory forensics starts by obtaining a physical memory dump. Then the dump is examined by searching its contents using different techniques and tools. Simple techniques just tries string matching. More advanced techniques consider the RAM structure and how data is stored in it. This section introduces some useful tools that can be used in memory forensics and PDF manipulations.

2.1 Volatility

Volatility is an open-source portable tool that can be used to analyze various memory dumps. The first version was released in 2007. The most recent release is 2.4 (as of this writing). Volatility supports the analysis of different architectures and operating systems such as: Linux, Windows, Mac, and Android. It is written in Python programming language. Also, it can be used on different memory formats such as: physical memory, crash dumps, raw dump, and VirtualBox core dumps. It can be used to extract vital information such as processes, modules, network connections, files, and rootkits.[5].

2.2 PDFMiner

PDFMiner is a tool used to extract and analyze text data found in PDF files. In particular, it allows users to obtain the exact location of a particular text, font, or line in a page. Furthermore, it can be used to convert PDF files to other text formats such as HTML [2]. This tool is written in Python and it was introduced for the first time in 2004.

2.3 pyPDF & PyPDF2

pyPDF is a Python library dedicated for PDF files. It can be used to extract information out of a PDF file. It also provides features such as splitting, merging, cropping, encrypting, and decrypting PDF documents. For example, it can be used to split and/or merge a PDF file page by page. However, it does not work on file streams, but it entirely works on StringIO objects, allowing PDF files to be modified in memory. So, it is useful for websites that modify or manage PDF files. This tool is released under the modified BSD license [3]. PyPDF2 is a fork for the original pyPDF project [4]. It is a pure PDF toolkit that adds new classes and support for Python 3.

3 FILE CARVING

File carving is a technique used to recover or reconstruct files (or file fragments) based on their structure and/or contents. File carving does not depend on any information provided by the operating system or its file system [21]. Instead, it helps retrieving deleted or damaged files or file fragments from digital devices by utilizing the structure of the target file itself. For example, carving specific types of file can be done through identifying headers and/or footers (trailer) and extracting (carving out) blocks between these two specific boundaries. See Figure 1.

4 PDF FILE STRUCTURE

PDF is a portable document structure that can be used to present documents that include different kinds of data such as text, multimedia elements, images, etc. A PDF file consists of four major parts: header, body, xref table, and finally the trailer. Figure 2 presents the basic structure of a PDF file. Carving PDF files benefits from its internal structure. Thus, we need to understand

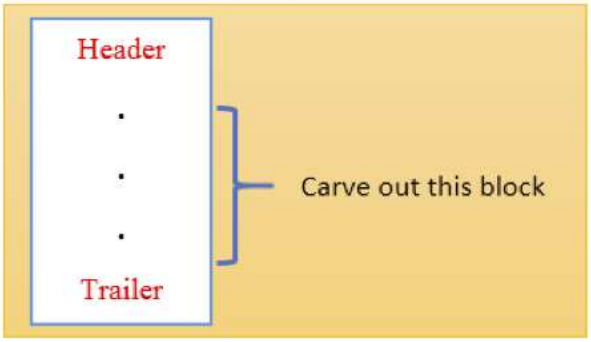

Figure 1. An Example of File Carving.

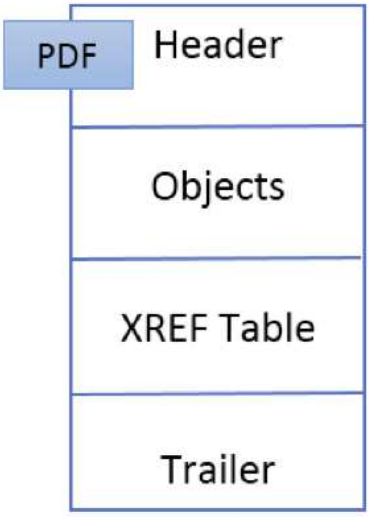

Figure 2. PDF File Structure.

the structure of a PDF file in order to be able to carve it.

4.1 Header

It is the first part of a PDF file (see Figure 2). It is one line that identifies the version number of the PDF specifications used in the document. The UNIX `xxd` command (as shown in Figure 3) can be used to read the header.

4.2 Body

Body is the second part of a PDF file structure. It contains a set of objects. Each object contains the actual data stored in that file. Figure 4 shows an example of object stream of a PDF file that was taken from [1].

```
# xxd temp.pdf | head –n 1

0000000: 2550 4446 2d31 2e33 0a25 c4e5
f2e5 eba7 %PDF-1.3%.....
```

Figure 3. An Example of Extracting a PDF Header Using the UNIX xxd Command.

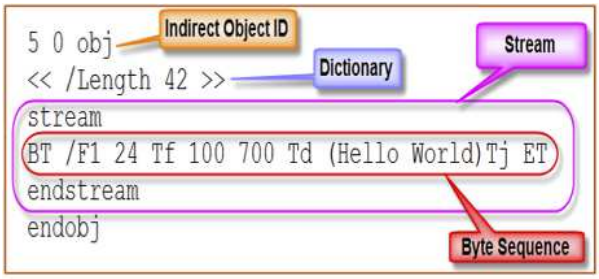

Figure 4. An Example of Object Stream of PDF File, Taken From [1]

4.3 XREF Table

The xref table stands for cross reference table (XREF Table). This part contains a reference for each object in the PDF document. It identifies the total number of objects in the table, the starting offset of each object, and the length (in bytes) for each object. This table is intended to allow random access to objects in the PDF file. Thus, it is not required to read the complete PDF document in order to locate a particular object. Each object is represented by one entry in the cross reference table. Figure 5 shows an example of an xref table that was taken from [6].

In the example shown in Figure 5, there are four subsections (note the four lines that only contain two numbers). The first number in those lines corresponds to the object number, while the second line states the number of objects in the current subsection. Each object is represented by one entry, which is 20 bytes long; including the CRLF. The first 10 bytes represent the object's offset (from the start of the PDF document to the beginning of that object). What follows is a space separator with another number specifying the object's identification code. After that there is another space separator followed by a letter 'f' or

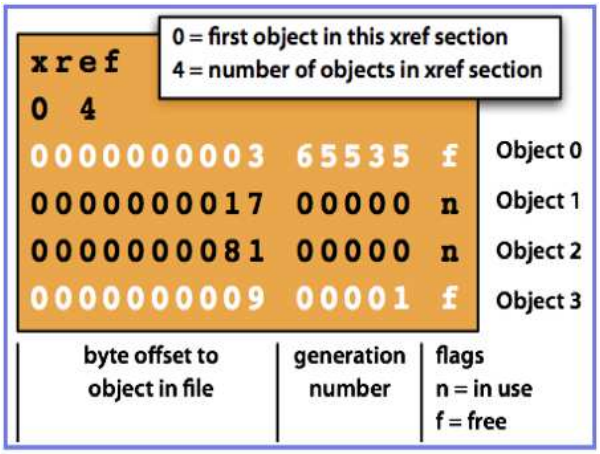

Figure 5. An Example of Cross Reference Table (XREF Table), Taken From [6].

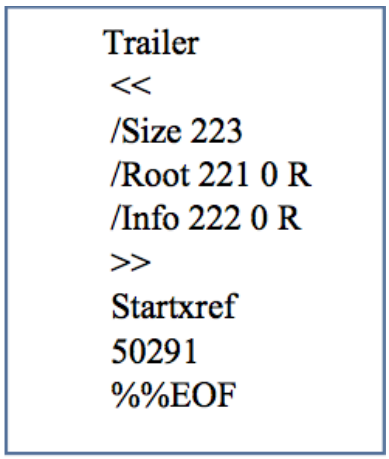

Figure 6. An Example of a PDF Trailer.

'n' to indicate whether the object is free or in use.

4.4 Trailer

The PDF trailer includes information used to assist a PDF reader application to understand the internal structure of the current PDF document. Figure 6 shows an example of a PDF trailer. It contains references to document's xref table and to special objects contained in the trailer dictionary. This part ends with "%%EOF", indicating the end of the file. All PDF readers start reading the PDF file from its trailer.

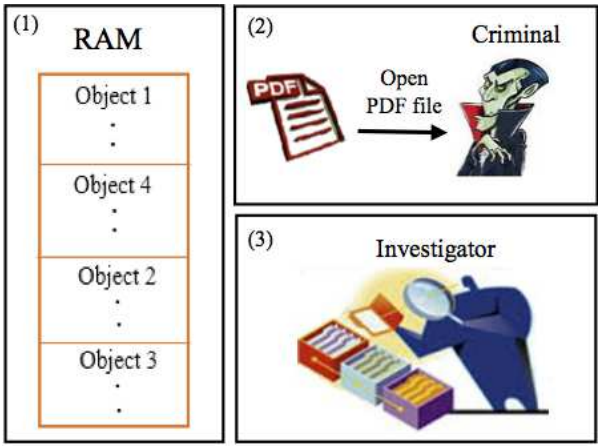

Figure 7. Investigation Model; The Criminal Opens a PDF File and the Investigator Searches to Find it. Find PDF Objects in the RAM Memory (A Memory Dump).

5 INVESTIGATION MODEL

In the investigation model (See Figure 7), a criminal opens a PDF file that contains a crucial information to the investigator. Furthermore, the criminal might not have sufficient permission to open the file. The PDF file might be on the machine's secondary storage or it might be downloaded from the Internet using one of the web browsers or ftp servers.

The investigator wants to reveal whether the target file was opened or not. Using file craving on PDF files, this can be reached by searching the contents of the RAM for objects that belongs to the original file. Extracting, analyzing, and reassembling the original file from the extracted data follow.

6 EXPERIMENTS AND RESULTS

A RAM dump is an exact bit-by-bit copy of the physical memory that is extracted into a file. Because the RAM was dumped while or after the PDF file is opened, this RAM dump might include a set of objects (sequence of bytes) related to the opened PDF file. We look for these objects in the RAM dump and compare them with objects extracted from the original PDF file.

We built a regular expression to locate all objects of the PDF file in the dumped file and

Table 1. PDF files used in our experiments.

PDF File	Size/Bytes	No.Pages	No.Objects
f1	228296	9	163
f2	267772	10	447
f3	381882	17	478
f4	477534	11	908
f5	604906	9	413
f6	681252	13	758
f7	745022	9	587
f8	843490	8	426
f9	1365581	9	520
f10	1574807	9	2240

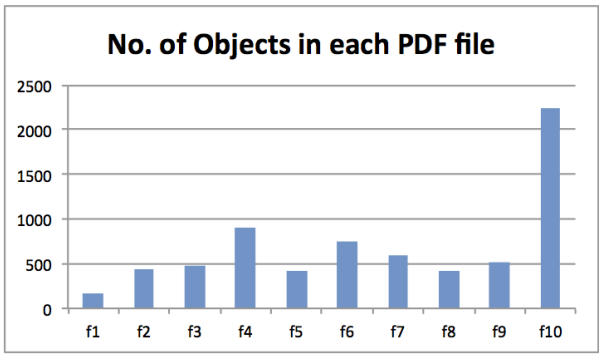

Figure 8. The Number of Objects in a PDF File Has No Relationship with its Size.

store them in a list. The list of carved objects can be used to reconstruct the original PDF file.

We tested 10 different PDF files of different sizes and different number of objects (See Table 1). Files in this table are listed in an ascending order based on their size (in bytes), which is presented in the second column. The third column presents the number of pages in each PDF file, whereas the last column presents the number of PDF objects in each PDF file. This table shows that there is no direct correlation between the size of the PDF file and the number of objects it has (See Figure 8).

We designed two different experiments:

- *Experiment 1 (Local)*: here the PDF files (shown in Table 1) are stored on the local machine's hard disk and opened using the `Okular` PDF viewer that comes by default with openSUSE KDE based GUI.

- *Experiment 2 (Internet)*: here the PDF files are downloaded from a server over the network using the Firefox web browser. Then the downloaded file is directly opened with the browser itself.

In both experiments (1 & 2), the same set of PDF files (shown in Table 1) are opened and then the RAM memory of the Virtual Machine (VM) is dumped in four different scenarios (steps):

- *Step 1*: The RAM is dumped while the PDF file is opened.

- *Step 2*: The RAM is dumped right after the PDF is closed (by closing the PDF viewer or the web browser).

- *Step 3*: Then a small `jpg` image is selected from the Dolphin file manager on openSUSE. This image is viewed using the default image viewer named `Gwenview`. The RAM is dumped while the image is being viewed. This step is intended to investigate the impact of a small image on the number of PDF objects that can be located and recovered from the RAM dump.

- *Step 4*: Without closing the image viewer in step 3 above, we started the Word Processor from the `LibreOffice`. Then the RAM is dumped for the fourth time (after closing the PDF viewer and starting an image viewer and a word processor). This step is intended to investigate the impact of a relatively larger process on the RAM (larger footprint in the RAM) and its effects on the number of PDF objects that may remain after closing the file and can be located and recovered from the RAM dump.

6.1 Experimental Setup

This paper experiments with two different experimental setups. In our first experimental setup (Setup 1), we used a VM that runs the openSUSE Linux version 13.1. In this setup, the VM has 512 MB of RAM memory. See Figure 9. In our second experimental setup (Setup 2), we increased the RAM size from 512 to 1024 MB of RAM memory.

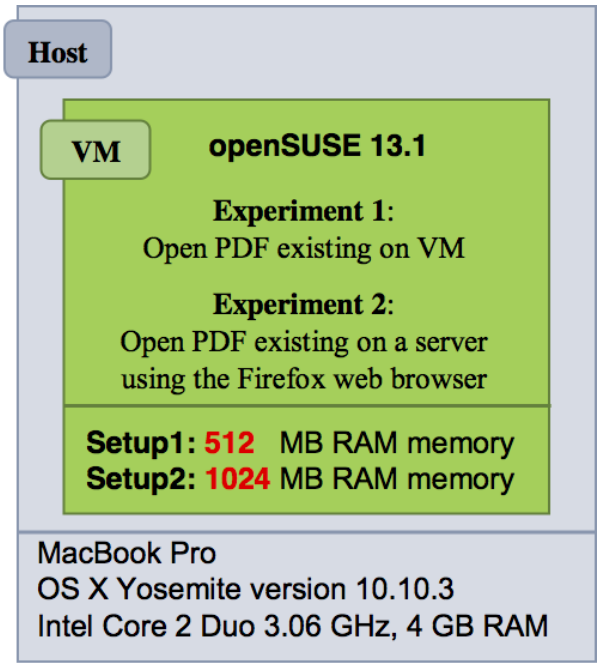

Figure 9. Basic Experimental Setup: One VM that Runs openSUSE 13.1. *Setup 1*: has 512 MB of RAM. *Setup 2*: has 1024 MB of RAM. On each setup we have two different experiments. *Experiment 1*: opens a PDF file from the local storage on the VM itself. *Experiment 2*: opens a PDF file from a remote server using the Firefox web browser.

6.2 Results

Tables 2 and 3 show the results for Experiments 1 & 2 on Setups 1 & 2 respectively. The first row in both tables establishes the ground truth to which we will compare our findings. Each row shows the results after each step explained in the above section. The tables show that there is better chances to extract more objects in a large RAM than in a small one. Furthermore, more objects can be extracted in case a PDF file is viewed over the Internet than that in case of opening the file locally (compare the results for Experiments 1 & 2). Also, the tables show that we always have a better chance to extract objects while the file is open (compare the first row to the second row in both tables). Furthermore, just closing files decreases the number of extracted objects (compare row 2 to row 3 in both tables). RAM contents depend heavily on what is being run in the target machine. Rows 4 and 5 in the tables show that conducting more activities after closing the files might dramatically decrease the number of objects even though this effect is less obvious in large RAM than in that of a small one.

Table 2. Results on Setup 1.

File/Memory Dump	Experiment1	Experiment2
Avg. # Obj. in PDFs	694	694
Avg. # Obj. in Step 1	462.6	1732.4
Avg. # Obj. in Step 2	250.3	0.1
Avg. # Obj. in Step 3	145.4	0.1
Avg. # Obj. in Step 4	10.8	0.1

Table 3. Results on Setup 2.

File/Memory Dump	Experiment1	Experiment2
Avg. # Obj. in PDFs	694	694
Avg. # Obj. in Step 1	655.3	3169.5
Avg. # Obj. in Step 2	655.3	1614.4
Avg. # Obj. in Step 3	655.3	1313.6
Avg. # Obj. in Step 4	655.3	1251.9

7 DISCUSSION AND FUTURE WORK

Processes and files in RAM memory are divided into pages. The challenge is elevated when the size of a PDF object is larger than the page size. If the page size is not enough for an object, the system will divide it into more than one page. Pages of the same file are not necessarily to be consecutive. Therefore, this escalates the difficulties to search for such objects and may reduce the number of correctly identified and located objects in the memory dump.

This paper experiments with 10 different PDF files. In each experiment, only one PDF file is considered. This means that the located PDF objects belongs to one and only one PDF file. Though, our future work aims to improve our techniques to search for objects belonging to multiple PDF files (and not limited to a single file).

8 RELATED WORK

Technology has become an essential part of people's daily life. Consequently, digital forensics

is an important discipline to counter violations. Investigators look for evidences at various locations, such as hard disks [19, 14, 17], solid state drives [18], main memories [16, 23, 22, 27], networks [24, 9, 20], and mobile devices [8, 26, 25, 15].

The RAM contains a wealth of information that is of investigators' interest. Researchers show that TCP memory artifacts can still be found in memory even after a long time [7]. Sensitive information, such as encryption keys and passwords, could be extracted from the RAM [13, 11, 10]. File carving[12] is well-known in the literature. It solely relies on the structure of the files to reconstruct them without having any meta data.

This paper focuses on carving PDF files from the RAM based on their structure. It extracts live objects of a PDF file from the RAM in order to extract the original file.

9 CONCLUSION

Digital investigation has evolved over years to accommodate the developing technologies. Even though most evidences can be reliably extracted from permanent storage devices, the RAM memory still contains exceptionally valuable information that sometimes cannot exist elsewhere. In some cases, an investigator needs to prove not only that a criminal has a file, but also she opens it. Such proof can be sought in the RAM. This paper presented a systematic process that can be used to carve such evidence by extracting PDF files from the RAM memory. Our technique of file carving has successfully utilized the structure of PDF files.

REFERENCES

[1] Object stream of PDF file, http://blog.didierstevens.com/2008/05/19/pdf-stream-objects/.

[2] PDFMiner, http://www.unixuser.org.

[3] pyPdf, http://pybrary.net/pyPdf/.

[4] PyPDF2, https://pythonhosted.org/PyPDF2/.

[5] The Volatility Foundation, http://www.volatilityfoundation.org.

[6] XREF table, http://labs.appligent.com.

[7] Mohammed I. Al-Saleh and Ziad A. Al-Sharif. Utilizing data lifetime of tcp buffers in digital forensics: Empirical study. *Digital Investigation*, 9(2):119 – 124, 2012.

[8] Mohammed I Al-Saleh and Yahya A Forihat. Skype forensics in android devices. *International Journal of Computer Applications*, 78, 2013.

[9] Robert Beverly, Simson Garfinkel, and Greg Cardwell. Forensic carving of network packets and associated data structures. *Digital Investigation*, 8(Supplement 1):S78 – S89, 2011. The Proceedings of the Eleventh Annual DFRWS Conference, 11th Annual Digital Forensics Research Conference.

[10] Pete Broadwell, Matt Harren, and Naveen Sastry. Scrash: a system for generating secure crash information. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, SSYM'03, pages 19–19, Berkeley, CA, USA, 2003. USENIX Association.

[11] Jim Chow, Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum. Shredding your garbage: reducing data lifetime through secure deallocation. In *Proceedings of the 14th conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 22–22, Berkeley, CA, USA, 2005. USENIX Association.

[12] Michael Cohen, Simson Garfinkel, and Bradley Schatz. Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. *Digit. Investig.*, 6:S57–S68, September 2009.

[13] Dawson Engler, David Yu Chen, Seth Hallem, Andy Chou, and Benjamin Chelf. Bugs as deviant behavior: a general approach to inferring errors in systems code. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 57–72, New York, NY, USA, 2001. ACM.

[14] Peter Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*, SSYM'96, pages 8–8, Berkeley, CA, USA, 1996. USENIX Association.

[15] Mohammad Iftekhar Husain, Ramalingam Sridhar, Sanjay Goel, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson. *iForensics: Forensic Analysis of Instant Messaging on Smart Phones*, volume 31, pages 9–18. Springer Berlin Heidelberg, 2010.

[16] Hajime Inoue, Frank Adelstein, and Robert A Joyce. Visualization in testing a volatile memory forensic tool. *Digital Investigation*, 8(Supplement):S42–S51, 2011.

[17] Matthew Kiley, Shira Dankner, Marcus Rogers, Indrajit Ray, and Sujeet Shenoi. *Forensic Analysis of Volatile Instant Messaging*, volume 285, pages 129–138. Springer Boston, 2008.

[18] Christopher King and Timothy Vidas. Empirical analysis of solid state disk data retention when used with contemporary operating systems. *Digital Investigation*, 8, Supplement(0):S111 – S117, 2011. The Proceedings of the Eleventh Annual DFRWS Conference. 11th Annual Digital Forensics Research Conference.

[19] James L. Lyle. Nist cftt: Testing disk imaging tools. *IJDE*, 1(4), 2003.

[20] Emmanuel S. Pilli, R.C. Joshi, and Rajdeep Niyogi. Article: A generic framework for network forensics. *International Journal of Computer Applications*, 1(11):1–6, February 2010. Published By Foundation of Computer Science.

[21] Rainer Poisel, Simon Tjoa, and Paul Tavolato. Advanced file carving approaches for multimedia files. *JoWUA*, 2(4):42–58, 2011.

[22] Andreas Schuster. The impact of microsoft windows pool allocation strategies on memory forensics. *Digital Investigation*, 5, Supplement(0):S58 – S64, 2008. The Proceedings of the Eighth Annual DFRWS Conference.

[23] Jason Solomon, Ewa Huebner, Derek Bem, and Magdalena Sze?ynska. User data persistence in physical memory. *Digital Investigation*, 4(2):68 – 72, 2007.

[24] Peter Sommer. Intrusion detection systems as evidence. *Computer Networks*, 31(23?24):2477 – 2487, 1999.

[25] Vrizlynn L.L. Thing, Kian-Yong Ng, and Ee-Chien Chang. Live memory forensics of mobile phones. *Digital Investigation*, 7, Supplement(0):S74 – S82, 2010. The Proceedings of the Tenth Annual DFRWS Conference.

[26] Nicolas Christin Timothy Vidas, Chengye Zhang. Toward a general collection methodology for android devices. *Digital Investigation*, 8(Supplement):S14–S24, 2011.

[27] Aaron Walters and Nick L Petroni. Volatools : Integrating volatile memory forensics into the digital investigation process. *Digital Investigation*, pages 1–18, 2007.