Research Project 2
DFRWS Challenge 2010 - Mobile forensics

Joeri Blokhuis
joeri.blokhuis => os3.nl

Axel Puppe
axel.puppe => os3.nl

November 25, 2010

UNIVERSITY OF AMSTERDAM

System and Network Engineering
2009-2010

Contents

1 Introduction

1.1 DFRWS 2010 Forensics Challenge

Since 2001, the Digital Forensic Research Workgroup (DFRWS) has held annual workshops for researchers, forensic examiners, and analysts[1]. As of 2005 they have added a forensic challenge with the goal to push the boundaries of (digital) forensic tools and research. So the challenge is a lot more than just a puzzle for fun or a way to hone your skills, it is an incentive to advance the forensic field.

Taken from the DFRWS website, the challenge has been defined in the following way[2]:

> After an extensive undercover operation, a known arms dealer named Monsieur Victor, commonly known as "The General", was lured out of hiding and apprehended in the Netherlands. He had expected a meeting to finalize a large sale of weapons, including tanks, missiles, attack helicopters, and assault rifles. Instead he met with police. When he realized the situations, he threw a mobile device in a nearby canal. The device was later retrieved by scuba divers, and was found to be a Sony Ericsson K800i Cybershot.
>
> Mr. Victor has been connected with several front companies, and a fleet of military cargo planes used to deliver weaponry. He has openly mocked arms embargoes, and is suspected of selling arms to both sides in military conflicts around the globe. As a result of his brazen behaviour of delivering large shipments to high risk regions all over the world, he developed a reputation as the "UPS of arms dealers". In some situations, he traded weapons in exchange for oil, copper, cobalt, uranium 294, 298, 380, thorium, titanium and other materials he could resell. However, in the past, investigators could not find sufficient evidence to link him to any arms deals. Despite regular surveillance by authorities, and efforts to monitoring of his communications, he managed to slip through the net of several sting operations. One of his methods of operation is to use stolen and/or throw away cell phones, making it more difficult for investigators to track and monitor his activities.
>
> In the current operation, undercover investigators arranged an arms deal through one of Mr. Victor's front companies, Smurf Celtic. To convince him that the deal was real, an initial down payment was made by electronic funds transfer to Smelt Bank in France into an account owned by another front company named RipTide Security. His meeting in Amsterdam was to finalize the full payment to a bank account in Dubai.
>
> After Mr. Victor's Sony Ericsson K800i Cybershot was retrieved, it was provided to the Netherlands Forensic Institute for processing. The Memory toolkit was used to acquire the contents of NAND and NOR Flash memory from the device.
>
> You have been asked to recover any evidence that can connect Monsieur Victor to the sale of arms through Smurf Celtic, and the receipt of payment to RipTide Security. In addition, you have been asked to recover any leads that might connect him to other individuals, companies, or bank accounts that are involved in Mr. Victor's international arms business.

Along with this scenario, the following questions were asked:

- Evidence connecting Monsieur Victor to the sale of arms through Smurf Celtic.
- Evidence of the receipt of payment to RipTide Security.
- Recovery of any leads that might connect Monsieur Victor to other individuals, companies, or bank accounts that are involved in his international arms business.

1.2 Overview of challenge data

The challenge has provided two different types of flash memory for the investigation. Table 1 summarizes the data and the calculated hashes(MD5 and SHA1). The hashes matched with the hashes provided by the challenge[1].

SonyEricsson_K800i_NAND_NAND512R3A.bin	MD5: ec7b80c62cc2a377c9b981f07d20e9a8
filesize: 66MB	SHA1: 27692ab4e963451bf3ae2e242f20c81f3bc76db9
SonyEricsson_K800i_Norflash_PF38F5060M0Y0BE.dmp	MD5: c01d11f1652401d0a20bdc2c97251081
filesize: 64MB	SHA1: e3e1159681cf65ea9a06d5a9a29ef2bcb22f523c

Table 1: challenge data

When referring to the memory dump `SE_NAND` and `SE_NOR` will be used respectively. The terms NAND and NOR will be used when it applies to the generic structure of either one.

As described in section 1.1 the memory was retrieved from a Sony Ericsson K800i Cybershot. To determine its capabilities a list was created according to the specifications of the mobile phone[16]. Tables 2, 3, 4 and 5 outline these capabilities. On the left side of the table the items are shown and the characteristics of it are shown on the right side. The table itself indicates the category.

Platform	Java J2ME, Java 3D
Storage	64 MB Internal Flash memory
	Expansion external memory card
Bluetooth	wireless communication
Infrared	wireless communication

Table 2: Technical specifications

[1]Challenge content:`http://www.dfrws.org/2010/challenge/dfrws2010-challenge.zip`

Call log	missed/received/dialed calls
Data formats	Contacts: vCard 2.1
	Calendar: vCalendar 1.0(vEvent)
	Tasks: vCalendar 1.0(vTodo)
	Bookmarks: vBookmark 1.0
	Notes: text/plain
SMS	storage on the SIM card and phone.
	Reply via SMS, MMS, voice message, email
EMS	support for storage on SIM card.
	Text messaging with support for pictures and sounds,
MMS	Messaging of pictures and videos.
	Bound to an Internet profile.
	Compose email(To, Cc, Bcc), MSISDN[a],
	has a field for: subject and delivery time.
Address book	storing contacts

Table 3: Phone data

[a]Mobile Station International Integrated Services Digital Network: telephone number in a GSM or UMTS mobile network

Audio	Various formats:
	mp3, mid, wav, m4a, ra, emy, imy, wma, etc.
Video	Back side recorder
	Front side calls
	Various formats:
	mp4, 3gp, rv, wmv
Pictures	3.2MP Digital Still Camera
Documents	

Table 4: File data

HTML browser	Browser cache size (300kB)
	history
	vBookmark 1.0
Email	sent/received/drafts/settings
RSS Reader	(news) feeds
Instant messaging	Wireless Village/My Friends
Security	SSLv3/TLSv1,X.509 certificate support

Table 5: Internet data

1.3 Background

Flash memory

Both memory files NAND and Nor flash have its individual use. The latter is often used in embedded systems to store and execute firmware, while NAND is mostly used in mobile media storage such as USB disks, digital cameras and mobile phones[17].

The specification of the NAND memory is publicly available and explains its internal structure[4]. SE_NAND is divided into pages, each page consists of 512 bytes + 16 bytes of spare area[2], which result in a total page size of 528 bytes. A group of pages will form a block. In SE_NAND a block consists out of 32 pages and will therefore have a size of 32*528 bytes. Figure 1 shows the memory structure of SE_NAND. From this point the term sectors will be used when referring to a page.

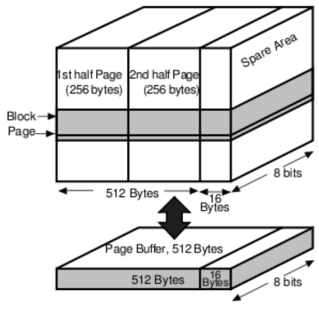


Figure 1: NAND memory organization

The specifications of NOR memory shows a different structure[5].

The memory is divided into eight 64-Mbit partitions. Each partition is divided into thirty-two 256-kByte blocks. Each block of 256-kByte is then divided into 1-kByte regions. And finally each region is divided into thirty-two 32-Byte segments. No spare area bytes are used in SE_NOR memory.

The fact that SE_NAND memory is 2MB larger in size is due to the use of spare area, as it matches exactly the difference with SE_NOR.

1.3.1 Data acquisition

A memory toolkit(hardware) was used by the Netherlands Forensic Institute to acquire data from both memory files. When reading memory physically with a hardware toolkit the structure of the memory dump differs from how a memory dump is normally presented to an Operating System. Hence the difference between the physical and logical level should be explained. The physical representation of the memory dump contains a structure of a lower file system which resides on the flash chip itself. The lower file system is created by the manufacturer of the flash chip and is often proprietary. The logical representation however, is able to present itself as a storage device to an Operating System. This is called the higher level file system, examples of a higher file system are FAT, EXT3, NTFS, etc. Breeuwsma et al[17] and Luck et al[18] explain what steps and or method is needed to rebuild a higher level file system from a physical NAND memory dump.

1.4 Scope

Our goal was more than just answering the questions of the challenge, therefore two tools were developed to help the (mobile phone) forensics field. One tool can carve (damaged) XML out of any data, including a damaged memory dump. The other tool scans for FAT file and directory metadata, this can be done on a physical or logical level to make it work on any FAT image or memory dump.

[2]contains information about the page or block(e.g.: bad block)

In order to repair or recreate the higher level file system all one needs to know is how to recreate the right order of physical blocks. Breeuwsma et al. listed three main questions on restoring a file system based on a physical memory dump.

- What is the granularity of the flash file system?
- Where is the meta data stored?
- How can the meta data be interpreted?

The answers to these question can sometimes be found by the manufacturer(datasheets) or in literature. When the answers are not found in either one of these ways, reverse engineering of the flash chip is considered as the final option. On reverse engineering of flash devices, Breeuwsma et al. stated the following: 'In this case a reference stick of same make and model is nearly indispensable'.

During the initial research no answers were found in literature or in the datasheets from the manufacturer. Therefore reverse engineering was our last option. However it also creates several road blocks that prevented us from pursuing this direction: First, Breeuwsma et al. stated that a reference device is very helpful in this process. In this case no reference device(memory dump) is provided. Even if it is managed to get a reference device we would still not have the equipment to dump any memory from a flash chip. Second, we noticed during the initial research what appeared to be damage on file(s) and the file system.

Also considering the phone was retrieved from the water, rebuilding the higher file system might not even be possible due to (extensive) damage. Therefore reverse engineering the firmware of the phone was left outside of the scope, as it requires an extensive amount of work, due to limited time/expertise and the possible gain is unsure.

1.5 Outline

The report is structured as follows. In chapter 2 the used research methods are described. This includes research into forensic analysis tools and the development of own tools. The subsequent chapter describes the results as part of the investigation which will be concluded with a time line for a clear overview on the found evidence. Finally a conclusion is given in chapter 4, followed by some recommendations for future research in chapter 5.

2 Methodology

This section describes the methods used during this research. To get a better understanding of the subject, a similar phone was used to analyse certain aspects. Next the analysis tools will be discussed. It will include common forensic tools as well as Linux commands. This section will be concluded with the subsection 'developed tools' which describes two developed tools and a script: Xarver, FAT Walker and spare area remover.

2.1 Sony Ericsson K800i

A live phone will be used to acquire additional knowledge about the phone its working and capabilities. The Sony Ericsson K800i allows the phone to be connected to a computer via a USB-cable. By connecting the phone to a computer it was found that the mounted memory is using FAT file system of type 16. The file system was found by creating an image with 'dd' and showing all mounted drives with 'mount'.

The following output is taken from 'mount' command which shows the type of file system and parameters used.

```
/dev/sdh1 on /media/PHONE type vfat (rw,nosuid,nodev,uhelper=devkit,uid=1000,
gid=1000,shortname=mixed,dmask=0077,utf8=1,flush)
```

The Linux command 'file'section 2.2 was used to determine the type of FAT(12, 16 or 32). The output below show the results of 'file'.

```
vodafone-k800i.dd: x86 boot sector, code offset 0x0, OEM-ID "MSWIN4.1",
sectors/cluster 8, FAT  1, root entries 512, Media descriptor 0xf8,
sectors/FAT 88, hidden sectors 7, sectors 145057 (volumes > 32 MB) ,
serial number 0x4c23ad22, label: "PHONE      ", FAT (16 bit)
```

By using a live phone an indication was given that FAT16 file system was used for user storage. However, the mounted partition size reported 70MB as the total amount of space. This is quite remarkable considering the fact that the total amount of SE_NAND memory is only 66MB(64MB user data) and SE_NOR only 64MB. An explanation could be given due to the phone its behaviour. When connecting the phone to a computer, it asks whether it should be used in storage mode or ordinary phone mode. When storage is selected the phone reboots and looses all its phone capabilities. Therefore it could be considered that parts of both memory files are mounted by the phone as a single storage device.

To determine the file system for SE_NOR memory and to verify SE_NAND memory, a small script was written. Section 2.3.3 will explain the script and outcome in depth.

2.2 Analysis tools

This section discusses a variety of forensic and Linux Operating System tools that were used during this research. Not all tools were as valuable as opposed to others, nevertheless these tools will be mentioned briefly.

Strings

The Linux 'strings' command line tool looks for at least 4 printable characters in a row, printing them to the screen. In combination with 'grep' (regular expression pattern matching) it can be a powerful yet quick search tool to look for specific words in any file.

File

‘File’ is a UNIX command line tool to attempt to classify or identify files. It consists of performing three tests, in the following order: file system tests, magic number tests, and language tests. As soon as a test succeeds it will output the file type.

- **File system:** This test checks if it is a special file system file.
- **Magic number:** This test looks for specific and well-known fixed formats, like a standard header or footer.
- **Language:** This test assumes it is a text file and tries to identify which encoding was used.

If the above tests fail it will output “data”. This tool can be quite valuable when working with unknown data or files as it can push you into the right direction for understanding the data within the file.

Hexworkshop

Hex Workshop v6[7] will be used to analyse files on a byte level and conduct keyword searches in ASCII and Unicode strings.

Scalpel

Scalpel is a linear carving program to recover files based on their headers, footers, and internal data structures. Scalpel is based on foremost[6] but has been completely rewritten to enhance performance and decrease the extensive memory usage. It can recover files from a disk image or raw block device and its effectiveness is shown when files are stored contiguously. Three extensions were written to scalpel to extract data and will be discussed below.

Phone data formats Details of the mobile phone in section 1.2 showed that data formats were used to store calendar events, to-do items, contact cards and bookmarks. All these data formats are standards, specified by an RFC. iCalendar is specified in rfc5545[8], vCard in rfc2425[10] and vBookmark is defined in the Infrared Mobile Communications (IrMC) specification[11]. Because of the type of information stored in these data formats it is trivial to research how these data formats are stored as they might contain evidence.

The specifications of the different data formats describes how the objects(vCard, vBookmark and vCalendar) must be created. The characteristics of creating an object is generic and is done as follows:

```
BEGIN:OBJECT
nested (multiple) objects such as vEvent, vTODO.
END:OBJECT
```

When OBJECT is replaced by VCARD, VBKM or VCALENDAR a data object is created. Inside the object it is possible to construct other objects such as vEvent and vTODO. Because the data formats are using a standardized header and footer for its objects, a small extension was written to scalpel to extract any possible useful information.

The following lines of code were added to the scalpel configuration file. Four columns need to be specified and there is an optional fifth for a footer. The first column defines the extension name, followed by case sensitive(yes/no). The third column sets the amount of bytes. Scalpel will carve until the given amount of bytes has been reached or stop when the footer has been found before reaching the maximum. The last obligatory column is the header column were a string(BEGIN:OBJECT) is translated into a hexadecimal value. The optional column must also be translated in a hexadecimal format and should be used when a footer is present.

```
vbmk  y    1000     \x42\x45\x47\x49\x4e\x3a\x56\x42\x4b\x4d      \x45\x4e\x44\x3a\x56\x42\x4b\x4d
vcal  y    1000     \x42\x45\x47\x49\x4e\x3a\x56\x43\x41\x4c\x45\x4e\x44\x41\x52(header)
    \x45\x4e\x44\x3a\x56\x43\x41\x4c\x45\x4e\x44\x41\x52(footer)
vcard y    1000     \x42\x45\x47\x49\x4e\x3a\x56\x43\x41\x52\x44  \x45\x4e\x44\x3a\x56\x43\x41\x52\x44
```

Address book entries During initial research on both memory dumps, data was found with characteristics of an address book. Further analysis of this data showed repetitive header and footer for each entry. Therefore the header and footer were added to scalpel.

```
adr       y        300        \x00\x02\x00        \x05\x00\x01\x00\x60
```

Browser history Running keyword searches on the memory dump for 'http' returned numerous URLs. Looking closer to these URLs it was found that for each URL a header and footer was placed around it. This extension to scalpel will extract the title of the page and URL itself.

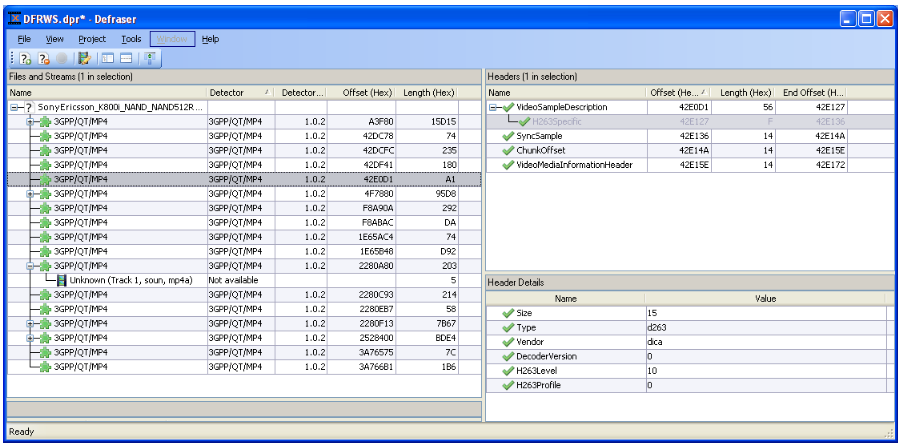
```
his        y        500        \x00\x00\x00\x03\x00\x01\x00        \x69\x39
```

Defraser

Defraser is an open source tool developed by the NFI to carve media (mostly video) files[12].

Most video files have more (internal) headers and footers, this allows for more detailed file carver than just simply looking for the start and end of the file. Not only can it find these detailed headers and display the full information within it, you can also move those blocks of data around in attempt to repair the file(s). The repaired media file can be sent from within the Defraser to any installed media player to directly see the result. However, we would like to point out that the defraser is only a 'dumb' tool and cannot automatically repair video files, thus, its effectiveness is directly related to the person using it.

Once a data file has be loaded, the entire file is scanned for the specific headers and displayed in the following way:



Files and Streams (1 in selection)

Name	Detector	Detector...	Offset (Hex)	Length (Hex)
SonyEricsson_K800i_NAND_NAND512R...				
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	A3F80	15D15
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	42DC78	74
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	42DCFC	235
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	42DF41	180
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	42E0D1	A1
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	4F7880	95D8
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	F8A90A	292
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	F8ABAC	DA
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	1E65AC4	74
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	1E65B48	D92
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	2280A80	203
Unknown (Track 1, soun, mp4a)	Not available			5
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	2280C93	214
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	2280EB7	58
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	2280F13	7B67
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	2528400	BDE4
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	3A76575	7C
3GPP/QT/MP4	3GPP/QT/MP4	1.0.2	3A766B1	1B6

Headers (1 in selection)

Name	Offset (He...	Length (Hex)	End Offset (H...
✓ VideoSampleDescription	42E0D1	56	42E127
✓ H263Specific	42E127	F	42E136
✓ SyncSample	42E136	14	42E14A
✓ ChunkOffset	42E14A	14	42E15E
✓ VideoMediaInformationHeader	42E15E	14	42E172

Header Details

Name	Value
✓ Size	15
✓ Type	d263
✓ Vendor	dica
✓ DecoderVersion	0
✓ H263Level	10
✓ H263Profile	0

Figure 2: Defraser

On the left the main data chunks are displayed, clicking them gives the corresponding data on the right. This data can be expanded to view more detailed information. These chunks of data can be sent to a 'workpad', the workpad allows you to shift the data around to put the data in the correct order. From within the workpad you can save the file or send the reconstructed data to a media player to view the result.

Google Goggles

Just like all people, forensic investigators have limited knowledge and might find new and unknown things to them. Search engines are getting better every day and can help investigators to find out more about a specific topic or subject. This can give a clearer understanding of found evidence, resulting in a more elaborate report with detailed findings. However, while search engines are ever improving, they are all still text-based. Finding out what a specific object or location is from a picture can still be tricky, as the visual data has to be interpreted and translated to text by the investigator.

Google Goggles is an application for the (mobile) operation system Android, this application can take a photo and use that photo as a search query[13]. Google will scan the uploaded photo and try to identify what is in it and can then use that information to search for the object or similar images.

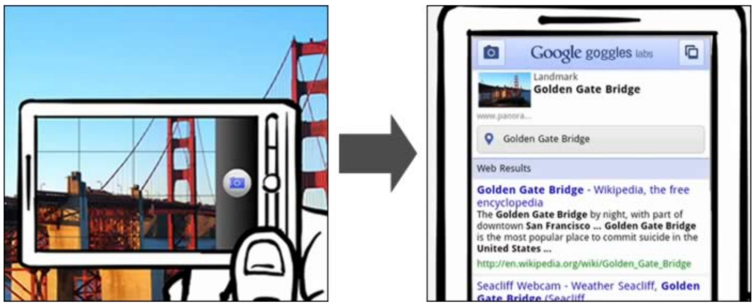

Google goggles labs
Landmark
Golden Gate Bridge
Golden Gate Bridge
Web Results
Golden Gate Bridge - Wikipedia, the free encyclopedia
The Golden Gate Bridge by night, with part of downtown San Francisco ... Golden Gate Bridge is the most popular place to commit suicide in the United States ...
http://en.wikipedia.org/wiki/Golden_Gate_Bridge
Seacliff Webcam - Weather Seacliff, Golden Gate Bridge (Seacliff

Figure 3: Google Goggles in action

In essence, it scans the photo, identifies it, and can then translate it to a normal text-based query. Currently this application is only available on Android powered smart phones, but perhaps the application will come to the desktop as well. While the application is rather simple, the idea behind is quite powerful. It can give an edge to investigators when it comes to identifying visual data and fill in missing knowledge.

Common forensic analysis tools

A variety of forensic analysis tools(commercial/non-commercial) were tested to provide or extract any information from the memory dumps. However, these tools did not retrieve any (new) beneficial information, but will be mentioned briefly. An explanation could be given

Magicrescue	showed similar results as scalpel, however it is capable to
(opensource file carver)	extract some XML based on the XML version tag.
Autopsy/Sleuthkit	Unable to load both memory dumps as
(opensource forensic fs analysis)	no file system was detected.
Encase v6.2.1	Detected the file system, extracted some files, a picture
(commercial forensics)	and html files.
FTK v1.8	Extracted some files, pictures but showed no
(commercial forensics)	different results than the available open source tools
Pyflag	Could not detect file system,
(opensource)	program terminated.
Paraben Cell Ceisure	It is not possible to load a memory file from disk.
(commercial: phone forensics)	Only a live cell phone connected to the computer can used.

Table 6: Tested commercial/non-commercial tools

due to loading the physical image into an analysis tool instead of an expected image build at a logical level. Breeuwsma et al explain this issue. They state that an image at a logical level must be rebuild from a physical image before any use of common analysis tools. But when rebuilding an image is not possible, a different method must be chosen.

2.3 Developed tools

2.3.1 Xarver

In this section we introduce and explain our own developed Carver. We developed this carver purely for this challenge, however, due to the broad usage of XML we feel that the tool will be very useful in many other cases or situations as well. A few examples of XML usage: contacts on SIM cards[9], data of webpages, Open Office XML (Word documents and related) and development of mobile applications(e.g.: Android).

To ensure others can benefit from this tool we developed it in Java to make it multi-platform. The project including the source code can be found at `http://Xarver.sourceforge.net`.

Introduction After our initial research on the memory dumps, we came across a great deal of XML data. While 'strings' or a hexeditor does show the raw XML data, it cannot build a full XML tree with all the nested nodes and values. Well-known carvers like Magic Rescue or scalpel can only find well-formed XML files if they contain an XML declaration at the start of the file. The declarations often looks like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

A lot of valuable information was stored in XML files without such declarations, so that was missing from the output. Because all of this, we opted to develop our own carver completely geared towards XML to tackle this problem.

Another reason to develop this tool was to tackle the problem of damaged data and XML. In this scenario the memory was extracted from a damaged phone, because of the extensive damage a versatile carver is required with the capabilities to read any data and deal with possible damage.

Inner workings The XML standard has been fully documented and is open to the public to ensure everyone can read and correctly implement it[14]. Our initial idea was to use the standard as a strict guideline to stick to. However, we quickly realized this would not work on damage data and XML. Corrupted XML would not pass the strict rules, but can still contain human-readable and possibly valuable information. To ensure that the most data would be found our XML carver is a rather liberate carver. It simply looks for tags and matching data, discarding any non-human-readable data to get rid of the majority of false-positives.

Finding the correct balance between accepting or discarding possible damaged XML was not easy. If you're too liberal the false-positives rise quickly resulting in a low signal-to-noise ratio. If you're too strict you will only find the undamaged XML, and miss out on valuable data.

Reading the initial data is done byte by byte, this allows Xarver to read and carve any data file you put into it. This way Xarver can carve XML tags out of highly damaged memory dumps (like from a phone retrieved from the canals). However, Xarver is not capable of retrieving actual lost data, if the XML data is gone then Xarver cannot help you. The tags are found by looking for the signature inequality signs < and >, removing any non-printable characters between them and checking the adjustable maximum length.

Once the data has been read byte by byte and all the possible tags are identified, the following steps take place to start building the actual XML trees:

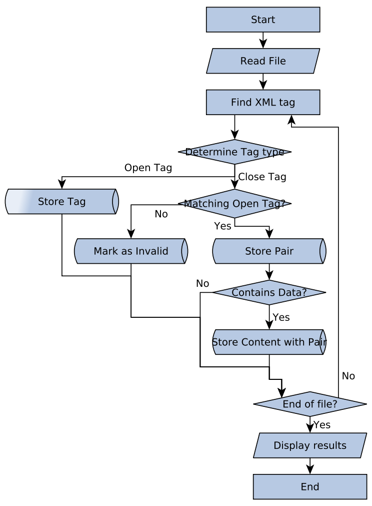
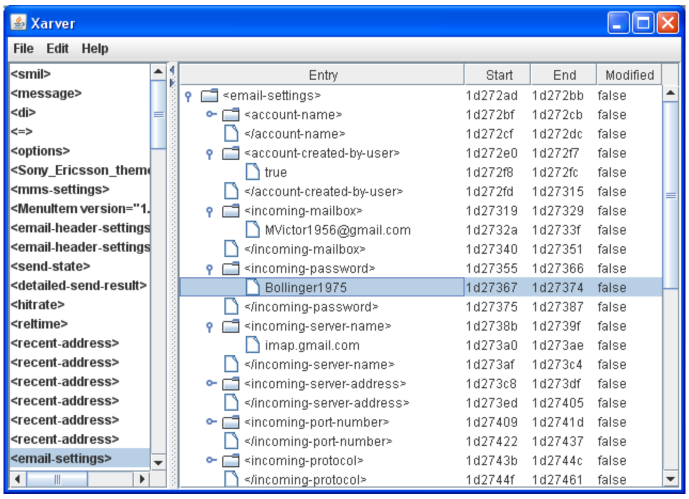


Figure 4: Flowchart of Xarver

Matching an open and close tag is done in a rather liberal way to deal with possible damages. Both tags are stripped of their brackets and the smallest tag (in size) is compared to the bigger one, if the characters appear in the same order plus an adjustable maximum faulty characters then it is considered a match. XML data found within a tree, but without a matching tag is not discarded but placed in the tree according to its retrieved location. This way even damaged data is visible to the user and open for interpretation.

At the end of the full examination, the data is presented to the user as depicted in Figure 5.



```
<smil>
<message>
<di>
<=>
<options>
<Sony_Ericsson_theme
<mms-settings>
<MenuItem version="1.
<email-header-settings
<email-header-settings
<send-state>
<detailed-send-result>
<hitrate>
<reltime>
<recent-address>
<recent-address>
<recent-address>
<recent-address>
<recent-address>
<recent-address>
<email-settings>
```

Entry	Start	End	Modified
<email-settings>	1d272ad	1d272bb	false
<account-name>	1d272bf	1d272cb	false
</account-name>	1d272cf	1d272dc	false
<account-created-by-user>	1d272e0	1d272f7	false
true	1d272f8	1d272fc	false
</account-created-by-user>	1d272fd	1d27315	false
<incoming-mailbox>	1d27319	1d27329	false
MVictor1956@gmail.com	1d2732a	1d2733f	false
</incoming-mailbox>	1d27340	1d27351	false
<incoming-password>	1d27355	1d27366	false
Bollinger1975	1d27367	1d27374	false
</incoming-password>	1d27375	1d27387	false
<incoming-server-name>	1d2738b	1d2739f	false
imap.gmail.com	1d273a0	1d273ae	false
</incoming-server-name>	1d273af	1d273c4	false
<incoming-server-address>	1d273c8	1d273df	false
</incoming-server-address>	1d273ed	1d27405	false
<incoming-port-number>	1d27409	1d2741d	false
</incoming-port-number>	1d27422	1d27437	false
<incoming-protocol>	1d2743b	1d2744c	false
</incoming-protocol>	1d2744f	1d27461	false

Figure 5: Screenshot of Xarver

On the left the XML 'roots' are displayed, upon selection a root the right pane is filled with the matching XML data. This data can be collapsed to see the full XML tree along with the start and end offsets. The offsets are important as it allows others to independently verify the results with the tool(s) of their choice (like a hexeditor). The last column shows if the data was stripped of any non-printable characters, this can be a good indication if the data is damaged as such characters should not be present in valid XML.

Throughout the development of this tool, and at the completion of it, the validity of the results were verified and checked by comparing the output to the data we saw with a hexeditor. This way we ensured that Xarver gives correct results.

2.3.2 FAT Walker

In section 2.3.3 a developed script was used to determine the file system of both memory dumps. For both memory files the File Allocation Table(FAT) was determined. According to FAT[15] valuable file information such as date, time and filename are stored inside a Directory Table Entry. Therefore a tool has been developed to extract Directory Table information from the FAT file system. As mentioned, the acquisition of the memory dumps is created at a physical level instead of at a logical level. Therefore most forensic analysis tools(sleuthkit, pyflag) will not be able to perform (any) analysis on the files. These tools are more efficient when data is presented at a logical level.

Introduction FAT Walker is a tool written in JAVA with platform independence in mind. It extracts information from a FAT file system such as filenames, MAC times[3] and the location of file entries regardless of the presence of a logical FAT structure (e.g.: boot sector or FAT table). This tool can extract information from a physical or logical memory dump. This allows an investigator to determine user behaviour of the system such as what files were created, accessed, modified or deleted before acquisition. It will give an investigator a quick overview if any relevant information might reside on the memory dump based on a file its characteristics before rebuilding any higher file system at a logical level. Optionally an absolute path towards the root can be build from any file.

FAT Walker will also be available at sourceforge and can be accessed via the following url:`http://fat-walker.sourceforge.net`.

Inner workings FAT Walker extracts all directory information available in a FAT file system and also records the files/directories that belong to the same directory. This is determined when multiple files or directories are found as the entries of a Table Directory are stored sequentially. This can be of use when building an absolute path towards the root.

The specifications of FAT[15] discusses how a Directory Table and its entries are build. A single entry consists out of 32 bytes, also called a 8.3 filename. Usually a directory table can be found on sector boundaries, therefore a sector can be skipped when the first entry is not valid. A more extensive approach is introduced when no sectors are skipped and every 32 bytes are being checked for a valid filename entry. This might be useful when an image is damaged. The extensive search method is presented as a flow diagram in Figure 6.

[3]Modified, last Access and Creation date/time

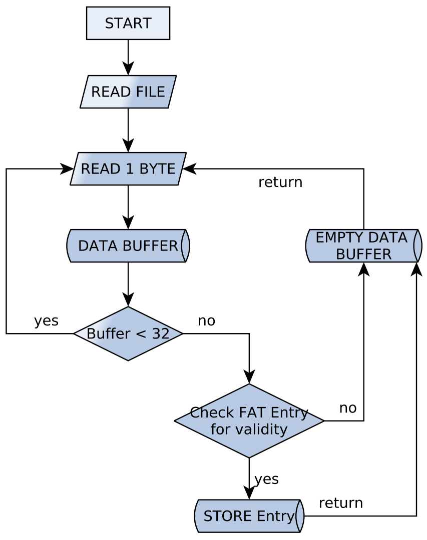


Figure 6: Flowchart of FAT Walker

When a file like a disk image, memory dump or an any data file is presented to the program it will read each byte of the file individually. It will read up to 32 bytes(size of filename entry) each time and will then check whether the buffered data contains a valid entry. When a filename entry is found it is stored on the system. When multiple entries are found in sequence a unique identifier is assigned to those entries. If the directory entry is not a root directory[4], entries will start with a '.'[5] followed by the '..'[6] entry. This provide the following information: the current (or working) directory contains a number(Start of File Cluster) which is also held by the actual filename entry, stored in another table(which is actually a pointer to the start of that table). The information stored in a directory table is shown in Table 7.

Description	**Example**	**Length**
Filename	string	8 bytes
Extension	string	3 bytes
Attribute	hexa value	1 byte
Reserved	0	1 byte
cTime Resolution	0-199	1 byte
Create Time	hour:min:sec	2 bytes
Create Date	year:month:day	2 bytes
Access Date	year:month:day	2 bytes
EA-Index	0	2 bytes
Last modified time	hour:min:sec	2 bytes
Last modified date	year:month:day	2 bytes
Start of File cluster	numeric value	2 bytes
File size(bytes)	numeric value	4 bytes

Table 7: Directory Entry Storage Organization[15]

[4]is the first or top-most directory in a hierarchy
[5]refers to the current directory
[6]refers to the parent directory

A filename and file extension can provide useful information, together with the corresponding MAC times an information base can be build to query any of the last modified/access/created times. Optionally this can be filtered for specific file extensions such as JPEG, 3GP, etc.

To determine an absolute path from a given directory or file it is important that the current and parent directory are available. As already mentioned a filename entry contains the start of cluster number which can also be found in the different directory table which is the "current directory" filename entry. By following the Start of Cluster numbers of each filename entry to the 'current directory', an absolute path can be build by chaining the filename entries to each other until no more 'current directory' entries are found. When no more 'current directory' entries are found a possible root directory is reached. When this is verified by the researcher the absolute path of a file has been found.

Figure 7 shows a screenshot of FAT walker containing the attributes of a directory table entry. It allows one to select any of the visible fields to perform queries on.

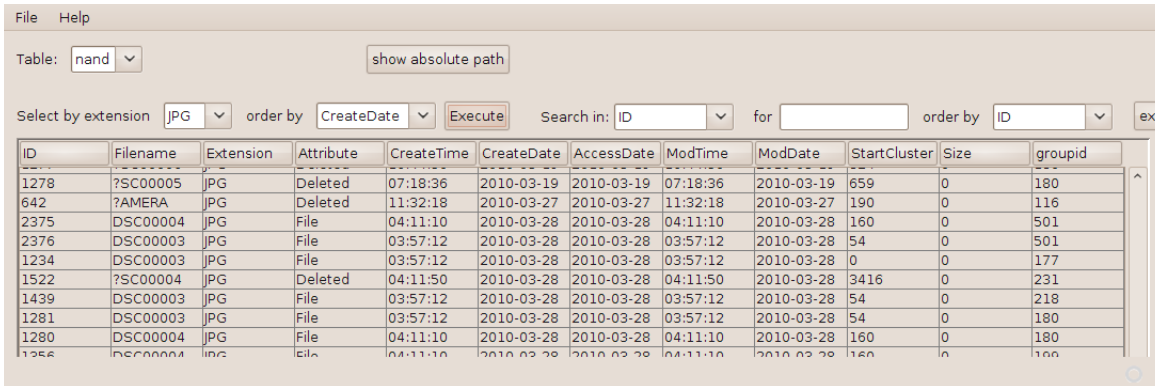


Table: nand

Select by extension: JPG order by: CreateDate

Search in: ID for: order by: ID

ID	Filename	Extension	Attribute	CreateTime	CreateDate	AccessDate	ModTime	ModDate	StartCluster	Size	groupid
1278	?SC00005	JPG	Deleted	07:18:36	2010-03-19	2010-03-19	07:18:36	2010-03-19	659	0	180
642	?AMERA	JPG	Deleted	11:32:18	2010-03-27	2010-03-27	11:32:18	2010-03-27	190	0	116
2375	DSC00004	JPG	File	04:11:10	2010-03-28	2010-03-28	04:11:10	2010-03-28	160	0	501
2376	DSC00003	JPG	File	03:57:12	2010-03-28	2010-03-28	03:57:12	2010-03-28	54	0	501
1234	DSC00003	JPG	File	03:57:12	2010-03-28	2010-03-28	03:57:12	2010-03-28	0	0	177
1522	?SC00004	JPG	Deleted	04:11:50	2010-03-28	2010-03-28	04:11:50	2010-03-28	3416	0	231
1439	DSC00003	JPG	File	03:57:12	2010-03-28	2010-03-28	03:57:12	2010-03-28	54	0	218
1281	DSC00003	JPG	File	03:57:12	2010-03-28	2010-03-28	03:57:12	2010-03-28	54	0	180
1280	DSC00004	JPG	File	04:11:10	2010-03-28	2010-03-28	04:11:10	2010-03-28	160	0	180

Figure 7: Screenshot of FAT Walker

FAT Walker is no fool proof method to extract all directory entries when a file system is damaged. For FAT Walker to determine if it has found an entry it must suffice the condition of a FAT entry. When these conditions of an entry are not met due to damage, the entry will be discarded. Throughout the development of FAT Walker the validity has been tested and checked against several non-damaged logical FAT images.

2.3.3 Scripts

File System Finder File System Finder is a script(included as Appendix B) which ties together varies UNIX command line tools ('dd' and 'file') in an attempt to detect possible file systems/partitions from a single file (like a memory dump). Besides selecting the input file, you also need to select the sector size of the flash memory(often 512). The script will run 'file' on every block of data and if it is identified as something which contains the word 'sector' it will output that result plus the offset. This small yet effective tool allows the user to learn more about an image or memory dump of a system, helping him in attempts to restore the data to its original format.

While the use of a live phone already showed that a FAT file system was used for the SE_NAND memory, it still had to be determined for SE_NOR. To be 100% sure about SE_NAND it had to be tested at a low level as well.

The results from the script showed that multiple FAT boot records were found in SE_NAND as well in SE_NAND. As reported by Breeuwsma et al. it is common that multiple version of a sector reside in flash memory this is due to behaviour of the flash chip. When multiple versions are found the version at the higher memory address is most likely to contain the latest data [17].

File system	**Offset**
FAT (16 bit)	0x1E03E04
FAT (12 bit)	0x1E1AE00

Table 8: File systems found on NOR

File system	**Offset**
FAT (16 bit)	0x14AE70
FAT (12 bit)	0x202D7DA
FAT (12 bit)	0x20BCD70
FAT (16 bit)	0x2308A70

Table 9: File systems found on NAND

On both memory dumps two FAT file systems are found: FAT12 and FAT16. It is suspected that the phone uses two different partitions. A partition which is accessible to the user and a partition for storing phone content. Practical experience with the phone does conclude that some content which is found in SE_NAND is not visible when the phone is connected to the computer as a storage device.

Spare Area Remover Spare Area Remover is a script that can remove header and/or footer data from every sector, outputting in essence a clean and raw copy of the full data. If restoration of the file system is not possible, the spare area is best removed as it might interfere or confuse carvers. The size and presence of headers or footers can often be discovered in the technical specifications of the memory or by analysing the memory by hand. The spare area data is outputted into a separate file, this way it can be analysed separately if required. The script is included as Appendix C.

3 Investigation

In section 1.1 four tables were created to outline the phones its capabilities which also provides as a search scope for the evidence. Therefore the following sections are created in a similar order. The technical specification will discuss the findings in the FAT file system. The following section discusses phone data. Section 3.3 shows which multimedia files were recovered. The subsequent section discusses Internet data and finally the last section gives an overview of the found evidence.

3.1 Technical specifications

FAT was determined on both memory(`SE_NOR` and `SE_NAND`) files, therefore FAT Walker was used on both dumps. The analysis of FAT Walker is described in section 3.1.1.

3.1.1 FAT entry analysis

To determine user behaviour of the mobile phone, both memory dumps were read and analysed using FAT Walker. It was noticed that in `SE_NOR` memory all filename entries had either one of the following MAC times:

```
Creation Time/Date: 23:56:08/2008-01-30 Access Date: 2008-01-30 Modified Time/Date: 23:56:08/2008-01-30
Creation Time/Date: 00:34:52/2008-01-31 Access Date: 2008-01-31 Modified Time/Date: 00:34:52/2008-01-31
```

This indicates that MAC times are not used in `SE_NOR` memory. By that being said, `SE_NAND` however showed more optimistic results. When sorting directory entries by date, a clear gap in time becomes visible.

```
<----omitted output---->
Creation Time/Date 00:34:52/2008-01-31 Access Date: 2008-01-31 Modified Time/Date: 00:34:52/2008-01-31
Creation Time/Date 12:42:00/2010-03-16 Access Date: 2010-03-16 Modified Time/Date: 12:42:00/2010-03-16
<----omitted output---->
```

Further analysis shows that access and modification times are not updated in `SE_NAND` and hold the same value as creation time. This can be explained due to extending the life time of memory when not using access and modification time. Flash memory can only be erased between 10^4 and 10^6 times before it will be impossible to write any more data to it [17].

To determine what files were created since 2010-03-16 all file extensions were filtered since that date. There were no other filters applied, meaning the results include deleted files as well. Figure 8 shows a chart were all files are uniquely sorted by extension.

The top most files created are binary files(BIN)[7], images(JPG), data files(DAT)[8] and XML. JPG is found near the top of the list, so this could indicate that pictures were taken or downloaded to the phone. Also the occurrence of XML files could indicate that relevant information such as messages can be found in . All the above information extracted with FAT Walker gives an initial overview of what data resides on the system and possible anomalies such as the not utilised access and modified times. Also the fact that no files on `SE_NOR` were created after 2008-31-1 could an investigator do decide to focus on the other memory dump which does show more recent activities. Thereby it must be stated that this should only be done with real care and full understanding of the inner workings of the mobile phone. Therefore it must be known what files were created by the system that could store possible evidence. For instance if the address book was created in 2008, possible contacts could be stored that were added beyond that date.

[7]used for a wide variety of content and can be associated with a great many different programs.

[8]can include text, graphic, or general binary data. No specific structure is specified.

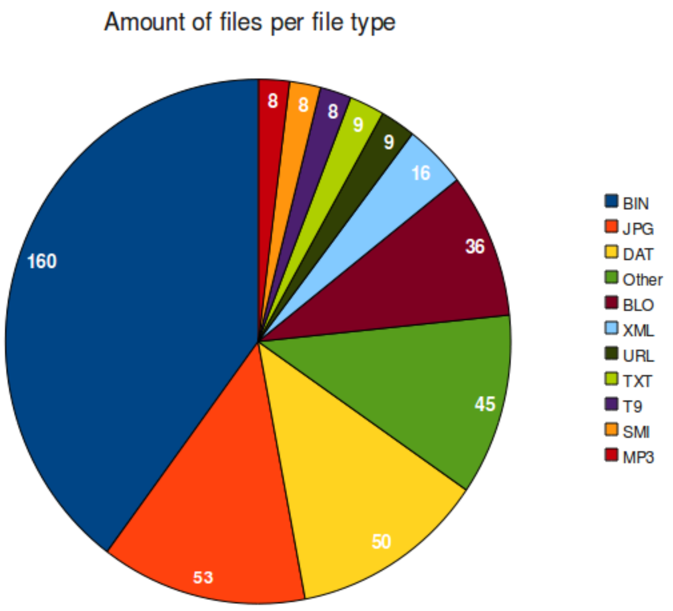


Figure 8: Files by extension(cdate > 2010-03-15)

3.2 Phone data

3.2.1 Time

Along the entire investigation numerous timestamps were found in multiple locations; plain text in XML, Unix timestamps in XML, metadata of the FAT filesystem. As the sources of all these timestamps are unknown it cannot be verified or linked to any external source. The times are most likely set by the system clock, but without an external reference or source it cannot be fully trusted as the clock might be set incorrectly. The time frame which covers the full investigation data includes the switch to DST[9]. This means that at March the 28th the clock was most likely moved forward one hour. However, without an external time source to verify this we cannot give any conclusive results. Hence, the timestamps need to be taken at face value.

We opted for the following time stamp notation `YYYY-MM`[10]`-DD HH:MM`[11]`:SS`.

[9]Daylight Saving Time
[10]Month
[11]Minutes

3.2.2 Logs

Xarver showed an XML root named 'RecentList', it contains a list of phone numbers and email addresses plus a timestamp in Unix time[12]. Correlating the timestamps with the timestamps from the MMSes and emails showed that all these timestamps are unique. Perhaps the phone numbers along with the timestamp indicated that a call was made at that point, however, this would not explain the email addresses in this list. This 'RecentList' is displayed in Table 10.

Timestamp	**Contact**
2010-03-16 13:51:41	DunkinBlue@hotmail.com
2010-03-18 21:51:24	MuhammedAamina@hotmail.com
2010-03-19 17:29:34	0620125081
2010-03-19 17:29:34	0632122345
2010-03-24 01:13:58	0643926087
2010-03-28 06:15:36	grassyjansen@yahoo.com
2010-03-29 07:12:43	Riptide101@rocketmail.com
2010-03-29 07:21:36	0615646978
2010-03-29 13:51:17	0632082356
2010-03-29 15:35:58	0650428000
2010-03-30 17:43:10	Smurf_Celtic@live.com
2010-03-31 08:37:50	0631356695

Table 10: Recent contact list found in NAND at offset 0x2D96AAD

3.2.3 MMS

MMS Stands for Multimedia Messaging Service and it extends the of SMS (Short Message Service). An MMS allows for the embedding of multimedia content into a message. As metadata of MMS is stored in XML format, Xarver was able to extract the metadata of three MMS messages. However, as only the metadata and not the actual content is stored in XML, the content is still missing. Scanning the memory dumps for the subjects did not give any results. The content of the MMS files are therefore not known.

Timestamp	**Subject**	**Number**	**Attachment**	**Offset**
2010-03-19 07:18:46	Look at this?	0632082356	DSC00001.JPG	1F9502D
2010-03-16 12:16:03	Mooie omgeving :-) .	0632082356	DSC00005.JPG	40B3F2D
2010-03-29 13:35:11	This?	0650428000	DSC00001.JPG	230192D

Table 11: Recovered MMS messages from the NAND memory

3.2.4 Address Book

Skimming through the `SE_NOR` memory with a hex editor revealed what appeared to be an address book. It could possible be the contact list stored on the phone, however, as all the

[12]Unix time is a system for describing points in time, defined as the number of seconds elapsed since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds.

entries are missing phone numbers this is either unlikely or strange. A direct memory dump of an address book entry looks like the following:

```
03fc1ea0  00 52 00 69 00 70 00 54  00 69 00 64 00 65 00 20 |.R.i.p.T.i.d.e.|
03fc1eb0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1ec0  00 53 00 65 00 63 00 75  00 72 00 69 00 74 00 79 |.S.e.c.u.r.i.t.y|
03fc1ed0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1ee0  00 35 00 02 20 48 00 65  00 6b 00 77 00 65 00 67 |.5...H.e.k.w.e.g|
03fc1ef0  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1f00  00 20 00 35 00 7c 00 53  00 63 00 68 00 65 00 76 |. .5.|.S.c.h.e.v|
03fc1f10  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1f20  00 65 00 6e 00 69 00 6e  00 67 00 65 00 6e 00 7c |.e.n.i.n.g.e.n.||
03fc1f30  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1f40  00 5a 00 75 00 69 00 64  00 2d 00 48 00 6f 00 6c |.Z.u.i.d.-.H.o.l|
03fc1f50  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1f60  00 6c 00 61 00 6e 00 64  00 7c 00 32 00 34 00 39 |.l.a.n.d.|.2.4.9|
03fc1f70  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1f80  00 35 00 50 00 47 00 7c  00 4e 00 65 00 74 00 68 |.5.P.G.|.N.e.t.h|
03fc1f90  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff |................|
03fc1fa0  00 65 00 72 00 6c 00 61  00 6e 00 64 00 73 00 04 |.e.r.l.a.n.d.s..|
```

As the above dump shows there are numerous problems with carving this data out due to the format:

- Data interleaved with 16 times `0xFF`
- UTF-16LE (The extra `0x00` after each character)
- Other non-'address book' data characters around it

The 16 bytes `0xFF` might break up a footer or header. The UTF16-LE encoding confuses 'strings' and possible other string/text-based tools. And the other "random" data around has to be filtered as well to give a nice clean address book in the end. Due to all these difficulties, more was required than just Scalpel. The initial carving is still done with the use of Scalpel as explained in section 2.2. However, that gave 250 results, most of them were false positives and the true positives still were rather 'messy'. In order to filter these results further, we wrote a Linux script D. The script filters out all the non-printable characters and replaces the '|' with new lines to make it more readable.

After the script runs on the Scalpel output files, it returned 17 entries. Some entries were still double due to a missing first name, after manually filtering those the address book contained 9 entries. A single entry looks like the following:

RipTide Security
Hekweg 5
Scheveningen
Zuid-Holland
2495PG
Netherlands

Table 12: Single entry from address book

The full list of entries can be found in Appendix F.

3.2.5 Calender

Recovering any calender entries such as todo, event (birthday, meeting, etc) was done using the written extension to scalpel according to the specifications described in section 2.2. However no data was recovered on either of the memory dumps. This could either indicate that no calendar events are stored or that events are damaged is such a way that no header could be found.

3.3 Multimedia/File data

This section includes all relevant parts found for file data.

3.3.1 Pictures

Pictures were recovered on both memory images. On NOR memory, JPG images were found that belong to the firmware of the mobile phone and are therefore omitted. Because of SE_NAND memory makes use of spare bytes after each sector, carving for files such as JPEG can be obstructed by spare area bytes which generate 'garbage' bytes after carving more than 512 bytes from a sector. Therefore the script 'sparearea remover' was used to build a new copy of the SE_NAND memory while leaving every data sector intact. Two outputs are generated, one containing the data sectors and one containing all spare area bytes. Therefore a new MD5 hash was calculated on both files.

```
2ca5f79cd20b0661894e9d63902df3f1  new_nand.bin  64MB
6375bd8d4a676b3c9e78faecc64b6f0f  spare-area.bin 2MB
```

The results from carving on SE_NAND with or without spare area bytes are significant. Carving with spare area bytes resulted in 6 files that could not be viewed. Recovering JPG files from SE_NAND without spare area bytes resulted in 9 files which were partially viewable. Table 13 show files recovered from SE_NAND without spare area bytes. This results in different offsets from the original NAND file.

Filename	MD5	Size	Offset
00001024.jpg	25950c65f0880e5c9f1f8601e0e169e2	7.5 KB	0x80000
00028880.jpg	42cfd9051a414a9f1ad6a6b57624e042	1126.4 KB	0xB2C000
00028896.jpg	18f1e2ddba4feaba9850beaea06b8949	28.5 KB	0xE1C000
00030896.jpg	3efef824fffaee950d813a741e4169fd	975.2 KB	0xF16000
00045752.jpg	48b0058f38cc98a93a8c253c917edffc	185.8 KB	0x1657000
00058088.jpg	cf512922a7eedcdb2126dd95934cb420	196.1 KB	0x1C5D000
00062752.jpg	e9783f05a6496e188041d7d681d9de45	680.1 KB	0x1EA4000
00066344.jpg	2f8a61e2c2a744495ce478a1d02cfba9	1536 KB	0x2065000
00090256.jpg	34f964aa42dfc739a058f5024d70c2bc	7.2 KB	0x2C12000
00125040.jpg	8e592120c814877efefdfbed6efca3a4	196.1 KB	0x3D0E000

Table 13: Pictures from NAND memory(without spare bytes)

On some files metadata was recovered along with the picture. The metadata indicated that the pictures were taken by the mobile phone. Table 14 shows three metadata fields that were found and have a significant value. The camera brand and model specify the type of

Camera Brand	**Camera Model**	**Date Taken**
Sony Ericsson	K800i	Year:Month:Day Hour(AM/PM):Minute:Second

Table 14: Metadata fields

phone which match the investigated phone. To determine if all pictures were found that were taken by the phone a search query was placed using the name of the camera model 'K800i'. One new picture was found using this method. Although the picture could not be recovered manually metadata indicated that the picture was created at '2010-03-31 08:36:14'. Table 15 shows the date of when pictures were taken.

Filename	**Date taken**
00028896.jpg	2010:03:31 08:36:01 AM
00045752.jpg	2010:03:28 05:57:12 AM
00058088.jpg	2010:03:31 08:36:25 AM
00125040.jpg	2010:03:28 06:11:09 AM

Table 15: Metadata stored in JPG files

Because the majority of the pictures only showed a part of the image, it was tried to extract any thumbnail data that might reside inside an image. For this an exif reader was used, which resulted in three full viewable thumbnail pictures. The pictures are included as Appendix E.

Figures 9 and 10 show two pictures of public places indicating that the user of the phone was at that place. Metadata found in both pictures even show at what date and time they were taken. Both pictures were taken on **2010-03-31**. Picture 00028896.jpg was taken at **08:36:01 AM** and 00058088.jpg **24 seconds later**. Figure 10 shows the name of the coffeeshop **'De dampkring'** which is located in Amsterdam. This fact states that the user of mobile phone was found to be in Amsterdam on March 31st 2010 at 08:36:01 AM in front of the Dampkring. The picture on the left could not be identified by Google Goggles to provide more information about the location of the building. It is expected to be in Amsterdam as well because of the gap of only 24 seconds between the pictures.

Figure 9: 00028896.jpg
Date: 2010-03-31 08:36:01 AM

Figure 10: 00058088.jpg
Date: 2010-03-31 08:36:25 AM

3.3.2 Videos

For carving out media files the Defraser is the best tool for the job, which is more elaborately explained in Section 2.2. The initial results with the defraser were rather discouraging; quite a few false positives and one blocky and short video of what appeared to be an aircraft taking off. As the spare area data was still present in the SE_NAND memory, it can easily interfere with carvers, especially when it comes to big files (like video) which crosses numerous blocks. To solve this problem, we used the spare area data script as described in Section 2.3.3. After running this script on the memory dump, the defraser was used again on the 'new' memory dump. This time at offset `0x40D5D8` and `0x4D1000` the same two video chunks as before were found, however, it was a lot clearer.

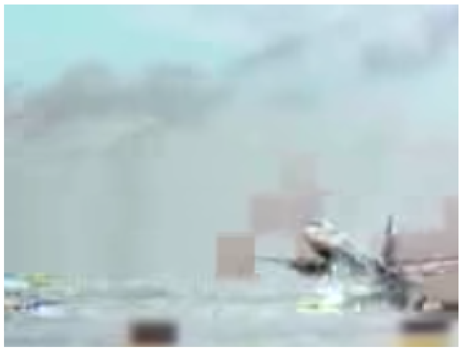


Figure 11: Thumbnail of carved video *without* spare area remover script used

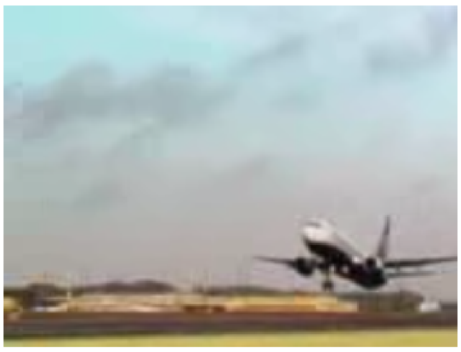


Figure 12: Thumbnail of carved video *with* spare area remover script used

The before and after screenshot clearly show a big improvement, and with that, shows the value of the spare area remover script. The meaning of this video or relation to anything else eludes us, as we cannot link this video to any form of contact or context.

3.4 Internet data

3.4.1 Browser data

History

The browser history was recovered from the `SE_NAND` by using a custom filter for Scalpel. 16 URLs were extracted and are shown in table 16.

Title	Url
Dubai Bank	`http://www.dubaibank.ae/`
Uranium 380 - Google Search	`http://www.google.com/m?q=Uranium+380 &btnG=Google+Search&hl=en&source=wax&ie=UTF-8&oe=UTF-8i`
Thorium - Google Search	`http://www.google.com/m?oe=UTF-8&source=wax&hl=en&q=Thorium`
Thorium	`http://www.google.com/gwt/x?oe=UTF-8&q=Thorium &hl=en&resnum=2&ei=W_eyS5CdOo_UjAfhh4ikAQ&sa=X &oi=blended&ct=res&cd=2&source=m&rd=1&u= http%3A%2F%2Fwww.world-nuclear.org%2Finfo%2Finf62.html`
ScienceDirect - Microchemical Journal : Differential spectrophotometric determination of micro amounts of metal ions in mixtures	`http://www.google.com/gwt/x?oe=UTF-8&q=Uranium+380 &hl=en&resnum=1&ei=q_ayS9COKuPNjAf5sLOkAQ &sa=X&oi=blended&ct=res&cd=1&source=m&rd=1 &u=http%3A%2F%2Flinkinghub.elsevier.com %2Fretrieve%2Fpii%2F0026265X73900829i9`
Uranium hexafluoride - Wikipedia, the free encyclopedia	`http://www.google.com/gwt/x?q=Uranium +298&hl=en&resnum=2&ei=UPSyS4CRJMmnjAeWspHpAg&sa=X &oi=blended&ct=res&cd=2&source=m&rd=1 &u=http%3A%2F%2Fen.wikipedia.org%2Fwiki%2FUranium_hexafluoride`
Google	`http://www.google.com/m?hl=en`
Google	`http://www.google.nl/`
Google	`http://www.google.com/`
Welcome to Navig8 mobile shop	`https://navig8uk.wisepilot.com/elite1/ mobileShop/webshop.jsp?step=purchaseInfo`
Welcome to Navig8 mobile shop	`https://navig8uk.wisepilot.com/elite1/ mobileShop/webshop.jsp?step=orderOverview`
Welcome to Navig8 mobile shop	`https://navig8uk.wisepilot.com/elite1/ mobileShop/eula.jsp?backStep=orderOverview`
Welcome to Navig8 mobile shop	`https://navig8uk.wisepilot.com/elite1/ mobileShop/webshop.jsp?userId=4155`
Get	`https://navig8uk.wisepilot.com/mobileShop/webshop.jsp?userId=4155`
null	`http://navig8uk.wisepilot.com/support/K800/en/Navig8.jad`
Software download	`http://navig8uk.wisepilot.com/elite1/ota/install.jsp?pm=SEk800`

Table 16: History found in NAND

The history shows a link or interest in 3 fields: Bank of Dubai, radioactive metals, and navigation software.

Bookmarks

Bookmarks were recovered from `SE_NAND` by using the extension to scalpel. Ten files were extracted and are shown in Table 17.

The bookmarks contained several URLs to `http://favorites.kpn.com/nl` and one to `http://seleyuan.com`. These URLs do not refer to any illicit content or websites.

3.4.2 Email

As metadata of email is stored in XML format, Xarver was able to extract the metadata of ten emails. However, as only the metadata and not the actual content is stored in XML,

filename	**Offset**	**Size**
00000006.vbmk	4afde46a60adcaf04dcc7decb21543e1	204 bytes
00000007.vbmk	fe79d81184ed6787419e6d5722845206	198 bytes
00000008.vbmk	90620970d89219736bd8136b784655d7	207 bytes
00000009.vbmk	4227845afc0f6f2f992726325ef152a6	202 bytes
00000010.vbmk	9bbc7a0e3b378f2570c4bb28319cb64e	204 bytes
00000011.vbmk	4a11dc6de995e96f43cac08f47cdd4c4	213 bytes
00000012.vbmk	9d7a53c75633227a61ff22ddefed85d8	201 bytes
00000013.vbmk	68d4a3a3990194ff29928ca85f8bf9de	201 bytes
00000014.vbmk	0fc53f3f531ad73dc9cbf2944a27101a	225 bytes
00000015.vbmk	b060d533d13dbb1bd52a2930c6e43b07	281 bytes

Table 17: vBookmarks found in NAND

the content is still missing. Searching the memorydumps for the subjects did result in a few hits. However it was found that the possible content of a message did not contain any header, footer or pointer to provide a fool proof link between the content and subject.

Timestamp	**Subject**	**From**	**To**	**Attachment**	**Likely content**	**Offset metadata**	**Offset content**
2010-03-01 04:32:11	Import your contacts and old email	mail-noreply@gmail.com	mvictor1956@gmail.com	No		0x3C1D5AD	
2010-03-01 04:32:11	Access Gmail on your mobile phone	mail-noreply@gmail.com	mvictor1956@gmail.com	No		0x3C217AD	
2010-03-01 04:32:11	Get started with Gmail	mail-noreply@gmail.com	mvictor1956@gmail.com	No		0x3C238AD	
2010-03-08 23:51:08	Look at this	dunkinblue@hotmail.com	stephanaaldenberg@gmail.com	No		0x215FEAD	
2010-03-09 00:17:31	*no subject*	muhammedaamina@hotmail.com	stephanaaldenberg@gmail.com	No		0x20AD82D	
2010-03-27 11:38:50	Contact	MVictor1956@gmail.com	Smurf_Celtic@live.com	No		0x29C50AD	
2010-03-28 04:12:18	Engine	MVictor1956@gmail.com	grassyjansen@yahoo.com	DSC00004.JPG	Dear mister Dutch, Is this what you mean? Kind regards, M V	0x1E4C0AD	0x1E3AB48
2010-03-29 04:39:30	Buy	MVictor1956@gmail.com	Smurf_Celtic@live.com	No	Hi, Make sure replacement engines are bought. I will arrange further air deployment for customer Best regards, M V	0x254E6AD	0x1E3ABD8
2010-03-30 15:41:01	Content	MVictor1956@gmail.com	Smurf_Celtic@live.com	No	Hi, Can you contact Iraquer? He wants to know when it arrives. M V	0x29FFD42	0x1E3AE40
2010-04-02 04:21:36	payment	riptide101@rocketmail.com	MVictor1956@gmail.com	No		0xE9522D	

Table 18: Recovered emails

Besides those emails, Xarver also found the email settings in the `SE_NAND` memory containing the username, password and connection settings.

Account name	Mv
Incoming mailbox	MVictor1956@gmail.com
Incoming password	Bollinger1975
Signature	— Sent using a Sony Ericsson mobile phone
Offset	0x3C3302D

Table 19: Recovered email settings

The signature found in email settings was used as a keyword to search for sent messages and possible content, but returned no results.

3.4.3 Instant messaging

Due to limited time instant messaging has not been fully analysed. Even so, no traces or indications of possible evidence was found during the initial research, further investigation is required to fully exclude this.

3.5 Linking the evidence

3.5.1 Analysis tools

The results of the investigation using the described methodology has resulted in various pieces of evidence like XML (MMS/Email) and pictures. As Email and MMS in some occasions contain pictures as attachments, the correct picture must be linked to the right message.

MMS In Table 11 three MMS messages are shown containing an attachment. Searching for the attachment DSC00001.JPG in FAT Walker results in two files, which are shown in Table 20.

Date/time	**Start of file(cluster)**
2010-03-16 12:16:02	444
2010-03-19 13:35:10	498

Table 20: Creation date/time pictures

Because they show a dissimilar 'start of file' values it can be concluded that both entries do not point to the same content. The pictures can already be further distinguished by date and time. Creation date of both files match the timestamps found in the MMS message. The creation time shows a deviation of 1 second with the timestamp of the message. Therefore it is highly likely that these pictures belong to those messages.

From both files the absolute path was calculated as well. This shows that the pictures are stored in a separate directory containing more files (XML and SMIL) regarding the MMS. This is shown in Table 21

StartOfFile(cluster)	Absolute path
444	/tpa/system/messaging/mms/vol_main/out/1/DSC00001.JPG
465	/tpa/system/messaging/mms/vol_main/out/2/DSC00005.JPG
498	/tpa/system/messaging/mms/vol_main/sent/3/DSC00001.JPG

Table 21: Absolute path

The different directories “sent” and “out” could indicate that the first two messages were not yet sent to its final destination. Correlating the creation date/time to any of the pictures found in section 3.3.1 showed no match.

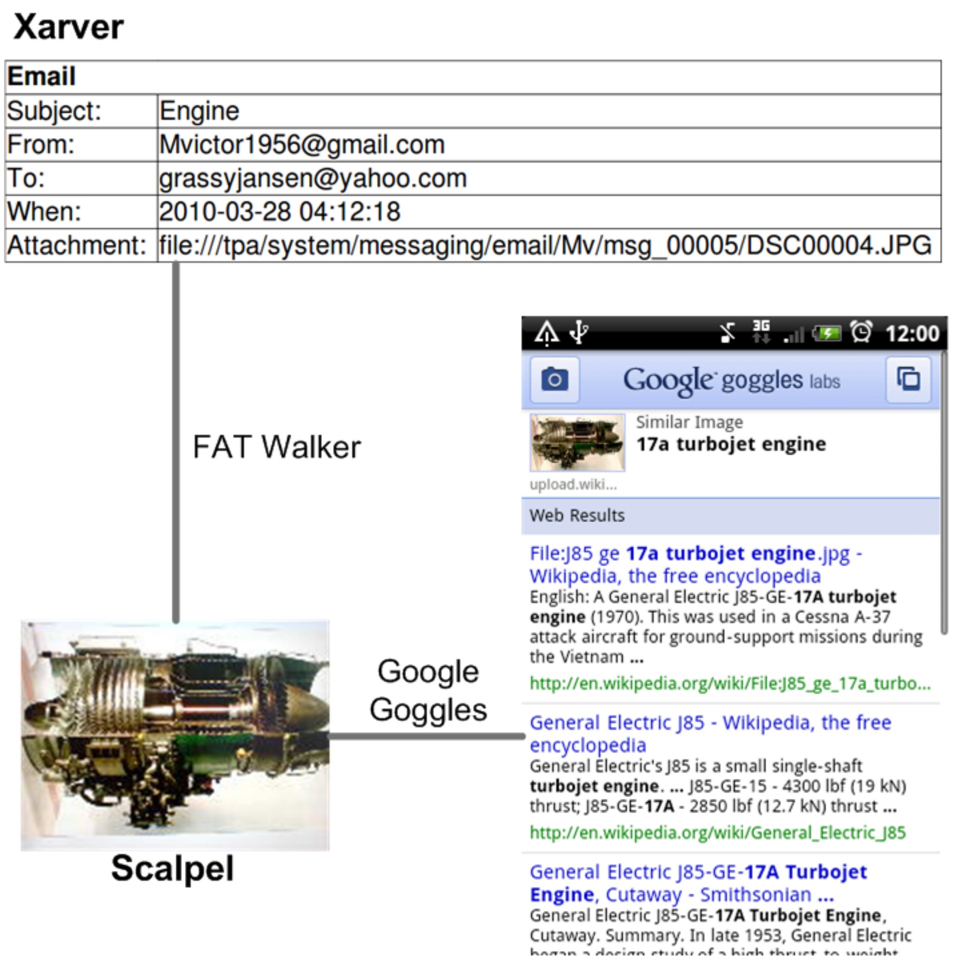
E-mail In section 3.4.2 one e-mail was found with the subject ‘Engine’ that contained an attachment ‘DSC00004.JPG’. FAT Walker shows that the file is created at ‘2010-03-28’ on ‘04:11:09 AM’ and deleted on ‘04:11:50 AM’. However multiple entries were found but there were only two distinct StartOfFile clusters. On for file creation(160) and one for deletion(3416). FAT Walker was then used to obtain the absolute path to determine the file its location. This is shown in Table 22.

StartOfFile(cluster)	Absolute path	Size(bytes)
160	/DCIM/100MSDCF/DSC00004.JPG	643117
3416	/tpa/system/messaging/email/Mv/msg_00005/DSC00004.JPG	49498

Table 22: Absolute path

The picture with cluster number 160 has a path which is also found when connecting the phone to the computer as a storage device. Its the default directory for pictures taken by the mobile phone. It allows the user when connected to the computer to approach these pictures. The picture with cluster number 3416 is located in the directory of a message. This can be explained when sending an email. When sending an email, attachments can be selected. It was found using the practical experience of the phone, that selecting a picture from the phone as attachment raised the following question: *Picture selected with a too high resolution, do you want to resize the picture?* When resizing the original is kept and a new picture is created. It is expected that this picture is stored in the email directory. Also the size is smaller which can be the result of resizing. And lastly the filename remains the same after selection and resizing. Therefore the attached picture is highly likely a resized version of the ‘original’ stored in /DCIM/100MSDCF/. The ‘original’ picture shows a deviation in time of two hours and 1 second from the metadata recovered. This can be explained due to what is described in section 3.2.1 concerning the system clock and DST. As FAT Walker also shows that no other pictures where taken that day the picture can be associated to filename DSC00004.JPG.

Figure 13 shows the relation between the pieces of evidence. Google Goggles was used to identify the picture and did so by providing the name of the object. The picture was identified as ‘17a turbojet engine’. The relation between subject and attachment are closely related and can therefore be expected that they belong together.



Xarver

Email	
Subject:	Engine
From:	Mvictor1956@gmail.com
To:	grassyjansen@yahoo.com
When:	2010-03-28 04:12:18
Attachment:	file:///tpa/system/messaging/email/Mv/msg_00005/DSC00004.JPG

12:00

Google goggles labs

Similar Image
17a turbojet engine
upload.wiki...

Web Results

File:J85 ge **17a turbojet engine**.jpg - Wikipedia, the free encyclopedia
English: A General Electric J85-GE-**17A turbojet engine** (1970). This was used in a Cessna A-37 attack aircraft for ground-support missions during the Vietnam ...
http://en.wikipedia.org/wiki/File:J85_ge_17a_turbo...

General Electric J85 - Wikipedia, the free encyclopedia
General Electric's J85 is a small single-shaft **turbojet engine**. ... J85-GE-15 - 4300 lbf (19 kN) thrust; J85-GE-**17A** - 2850 lbf (12.7 kN) thrust ...
http://en.wikipedia.org/wiki/General_Electric_J85

General Electric J85-GE-**17A Turbojet Engine**, Cutaway - Smithsonian ...
General Electric J85-GE-**17A Turbojet Engine**, Cutaway. Summary. In late 1953, General Electric

Figure 13: Link between email and picture

3.5.2 Contacts

All the contacts found in the Logs(section 3.2.2), MMS(section 3.2.3) and Email(section 3.4.2) can be combined into a single graph, giving a clear overview of the contacts of Monsieur Victor. The graph displayed in Figure 14 is partially directional; if the arrows are missing the direction of contact is unknown, otherwise the arrow points into the direction of a contact. Three clear groups can be distinguished: MSISDN, MVictor1956@gmail.com and stephanaaldenberg@gmail.com.

The MSIDSN part shows all the phone contacts, this encompasses MMSes and possible phone calls. The email address MVictor1956@gmail.com was used the most, showing quite a lot of activity from 2010-03-01 till 2010-04-02 (see Section 3.4.2). The email address stephanaaldenberg@gmail.com appears to be the odd one out, receiving two emails from different contacts, but with an unclear link to the phone or Monsieur Victor. There are three possible explanations for this email address; Monsieur Victor was a BCC[13] recipient of these emails, he used the email address stephanaaldenberg@gmail.com along with MVictor1956@gmail.com or there are certain emails missing, erased or damaged.

[13]Blind Carbon Copy: sending an email to multiple recipients which conceals individual email addresses

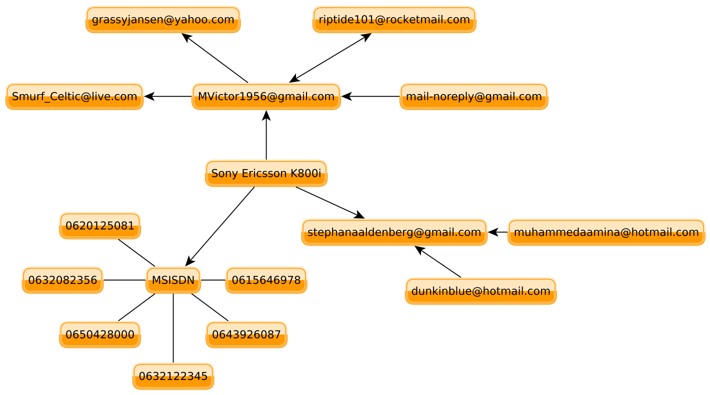


Figure 14: Contacts and their links found on the phone

3.5.3 Timeline

The results of the investigation using the described methodology are presented in a time line added to this document as Appendix A. This graphical overview shows communications between the suspect and different individuals. Each individual has its own colour and the communication between two parties are presented in a horizontal line. The meaning of the colours and horizontal line are indicated by the legend.

4 Conclusion

This report shows which data can still be recovered from a physical memory dump. The developed tools were very useful during the investigation. The ability of FAT Walker to extract Directory Table entries from any memory dump using the FAT file system, can provide useful information for initial research of the dump. Representing data in a table and the possibility to query any of the fields can be of great use when determining the user behaviour on what files where created, deleted and if recovery of the file is still possible. Xarver was very beneficial in recovering XML data. By filtering out any non-printable character a human readable result remained. Forensic soundness is provided by showing the offsets where data was found and a notification was given when non-printable characters were removed. In this way forensic professionals are able to verify the presented evidence. Combining the tools showed that it was possible to link evidence to a specific message, thereby showing their collaborative value.

Evidence connecting Monsieur Victor to the sale of arms through Smurf Celtic.

Four pictures were found which showed rockets, missiles and other weaponry as Appendix E shows, however these pictures cannot be tied to any form of communication. In Table 18 it shows that Monsieur Victor was emailing 'Smurf_Celtic@live.com' numerous times, with subjects like 'Contact' and 'Buy' it clearly shows an intention of making some sort of sale. Combining these two pieces of information shows that it is highly likely that Monsieur Victor is an arms dealer and had contact with Smurf Celtic.

Evidence of the receipt of payment to RipTide Security.

While no information on bank accounts or money transfers was found, find the metadata of an email which indicates that there is connection concerning money between Monsieur Victor and RipTide Security. Table 18 showed that 'riptide101@rocketmail.com' sent an email to Monsieur Victor with the subject 'payment', after emails concerning missiles, rockets and jet engines have been going back and forth. Sadly, it appears as the content of that email is either missing, damaged or deleted, hence we could not retrieve further details of the payment.

Recovery of any leads that might connect Monsieur Victor to other individuals, companies, or bank accounts that are involved in his international arms business.

During this research no leads were found on other companies or bank accounts. The relevant leads connecting Monsieur Victor to other individuals were shown in Figure 14. Individuals are either connected to Monsieur Victor by their MSISDN or E-mail address. A detailed time line is found in Appendix A. The time line shows the connection of other individuals via different communication channels.

5 Future Research

Due to our limited time and damage to the memory dumps we did not pursue repairing the filesystem. However, while it might not be completely successful it can still be fruitful to at least attempt repairing it. The gains of successfully repairing the filesystem can be enormous, making carving possibly unnecessary as it might give the full files like pictures, XML and much more. A restored file system in combination with FAT Walker allows one to see what files were deleted and if restoring a file is possible.

Another part which could be fruitful is examining the installed navigation software. Often navigation software displays the current address, and keeping track of recently entered addresses for the navigation. This might give more insights into where the phone has been, with the best case scenario giving us a full route with time stamps. This could link Monsieur Victor to numerous locations and times, possibly giving the investigators more leads on addresses, people and companies to further investigate.

The last interesting part would be reverse engineering the phone book which is most likely stored in a proprietary format. If successful it could give investigators (new) leads on people or companies stored in the phone book.

Acknowledgement

We would like to thank KPMG Forensic Technology department for offering a workplace and assistance throughout our research and investigation. A special thanks to Hans Wim Tinholt for his guidance and constructive feedback.

References

[1] Digital Forensic Research Workshop *url:* http://www.dfrws.org

[2] DFRWS 2010 Forensics Challenge *url:* http://www.dfrws.org/2010/challenge/

[3] Bez, R.; Camerlenghi, E.; Modelli, A.; Visconti, A. (2003). ”Introduction to flash memory”, *url:* http://ieeexplore.ieee.org/iel5/5/26994/01199079.pdf?tp=&arnumber=1199079&isnumber=26994

[4] Datasheet NAND512R3A, Manufacturer: ST MicroElectronics, accessed: 04-06-2010, *url:* http://www.digchip.com/datasheets/parts/datasheet/456/NAND512R3A-pdf.php

[5] Datasheet Norflash PF38F5060M0Y0BE, Manufacturer: Intel/Numonyx, accessed: 04-06-2010, *url:* http://datasheetz.com/data/Integrated$%$20Circuits$%$20(ICs)/Memory/886439-datasheetz.html

[6] Foremost: linear file carving *url:* http://foremost.sourceforge.net

[7] Hex Workshop v6 - hexadecimal editor *url:* www.bpsoft.com/

[8] Internet Calendaring and Scheduling Core Object Specification *url:* http://tools.ietf.org/html/rfc5545

[9] Overcoming Impediments to Cell Phone Forensics, 2008, ISBN ISSN:1530-1605 , 0-7695-3075-8

[10] A MIME Content-Type for Directory Information *url*: http://www.rfc-editor.org/rfc/rfc2425.txt

[11] IrDA specifications of vBookmark *url:* http://www.pday.com.cn/technology/irda_documents/IrMC_v1p1.pdf

[12] NFI Defraser *url:* http://sourceforge.net/projects/defraser/

[13] Google Goggles *url:* http://www.google.com/mobile/goggles/

[14] Extensible Markup Language (XML) 1.0 (Fifth Edition) *url:* http://www.w3.org/TR/2008/REC-xml-20081126/

[15] Microsoft Extensible Firmware Initiative FAT32 File System Specification, *url*, http://www.microsoft.com/whdc/system/platform/firmware/fatgen.mspx

[16] Sony Ericsson K800i - White paper *url:* http://developer.sonyericsson.com/cws/download/1/708/062/1260889839/83458-wp_k800_r1a.pdf

[17] Forensic Data Recovery from Flash Memory, 2007, Small Scale Digital Forensics Journal*url:* http://www.ssddfj.org/papers/SSDDFJ_V1_1_Breeuwsma_et_al.pdf

[18] An Integrated Approach to Recovering Deleted File from NAND Flash Data, Small Scale Digital Forensics Journal, 2008, *url:* http://www.ssddfj.org/papers/SSDDFJ_V2_1_Luck_Stokes.pdf

A Timeline

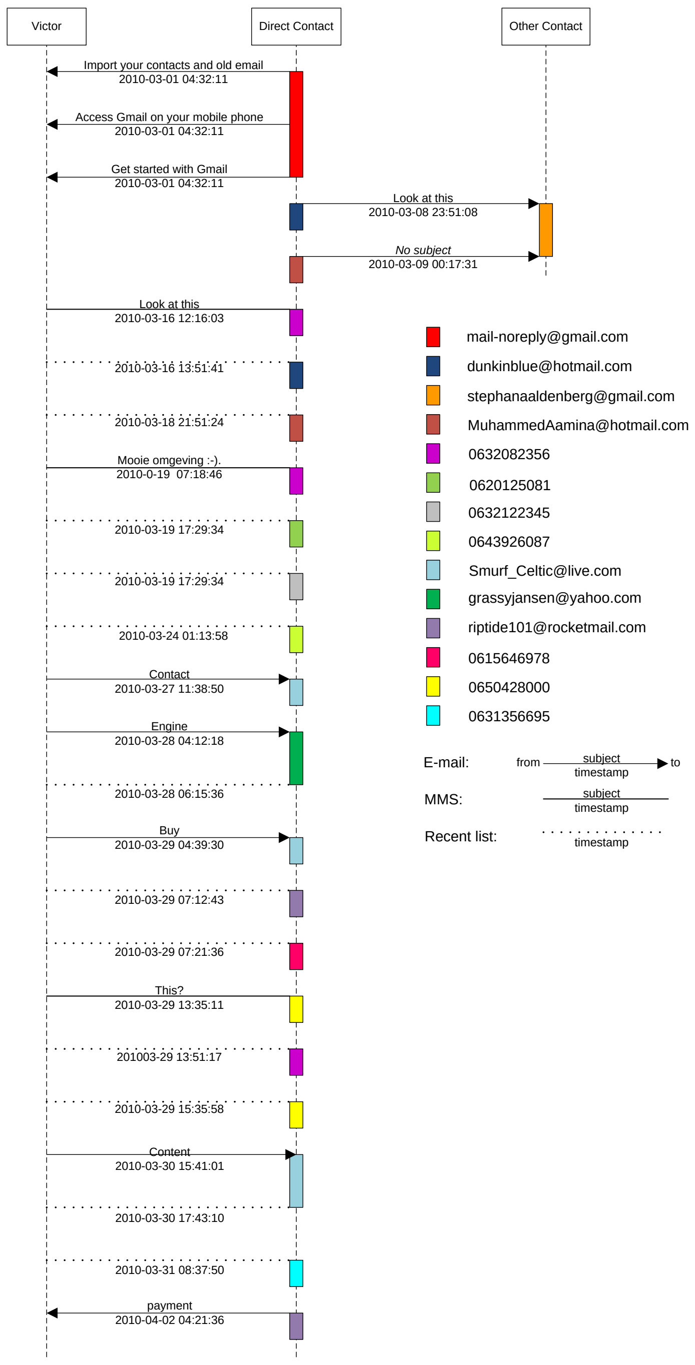


Figure 15: Communication time line

B File System Finder

```
#!/bin/bash
#Usage: ./FileSystemFinder.sh ToScan_file Output_file SectorSize

i=0
size=`du -b $1 | awk '{print $1}'`
size2= `expr $size / $3`
echo $size2
echo $3

while [ "$i" -le `expr $size / $3` ]
do
  `dd bs=$3 count=1 skip=$i if=$1 of=output`
  file_output=`file output`
  if [[ $file_output =~ .*sector.* ]]
  then
    offset=`expr $i \* $3`
    `echo Offset: $offset >> $2`
    `file output >> $2`
    `echo  >> $2`
  fi
  i=`expr $i + 1`
done
rm output
```

C Spare area remover

```
#!/usr/bin/python
#Usage: ./sa-remover.py inputfile sectorsize sparearea
import sys

f = open('se-nand.bin', 'rb')
w = file('new-nand.bin', 'wb')
n = file('spare.bin', 'wb')

inputfile  = sys.argv[1]
sectorsize = int(sys.argv[2])
sparearea = int(sys.argv[3])
#header = int(sys.argv[4])
#footer = int(sys.argv[5])

print sectorsize
print sparearea
print header
print footer

with open(inputfile, 'rb') as a_file:
        nn = a_file.read(sectorsize)
        while nn != "":
                w.write(nn)
                sa = a_file.read(sparearea)
                n.write(sa)
                nn = a_file.read(sectorsize)
w.close()
n.close()
```

D Phone book filter

```
#!/bin/sh

i=0
for file in ./*.adr ; do
```

```
tr -cd '[[:print:]]' < $file | sed 's/\x7c/\xa/g' | sed 's/\x60/\xa/g' > temp
if [ `wc -l temp | awk '{print $1}'` -ge 3 ] ; then
mv temp $i.tmp
i=`expr $i + 1`
fi
done

i=0
for file2 in ./*.tmp ; do

if [ `head $file2 -n1 | wc -w ` -ge 4 ] ; then
head $file2 -n1 | awk '{print $1 " " $2}' | tr -d '[[:digit:]]' > temp
lines=`wc -l $file2 | awk '{print $1}'`
lines=`expr $lines - 1`
head $file2 -n1 | awk '{print $3 " " $4}' >> temp
else
head $file2 -n1 | awk '{print $1}' | tr -d '[[:digit:]]' > temp
lines=`wc -l $file2 | awk '{print $1}'`
lines=`expr $lines - 1`
head $file2 -n1 | awk '{print $2 " " $3}' >> temp
fi
tail $file2 -n $lines >> temp
mv temp $i.txt
i=`expr $i + 1`
done

rm *.tmp

yes 1 | fdupes -d .
```

E Carved pictures

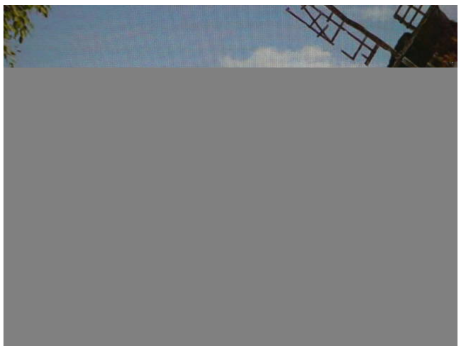


Figure 16: Offset: 0xB2C000, Resolution: 640x480, Google goggles: Shrek Wallpaper

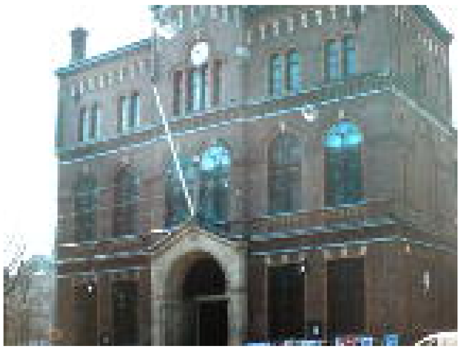


Figure 17: Offset: 0xE1C000, Resolution: 160x120, Thumbnail only

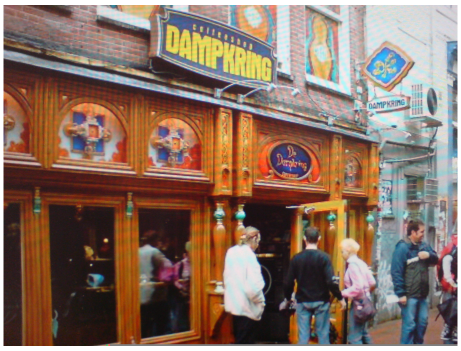


Figure 18: Offset: 0x1C5D000, Resolution: 2048x1536, De Dampkring

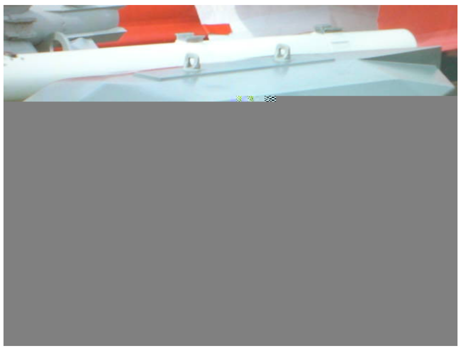


Figure 19: Offset: 0xF16000, Resolution: 640x480

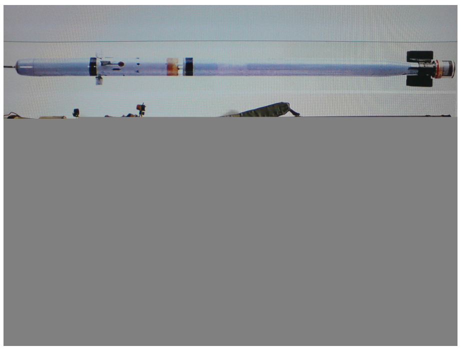


Figure 20: Offset: 0x1657000, Resolution: 2048x1536, Google goggles: 9K38 Igla

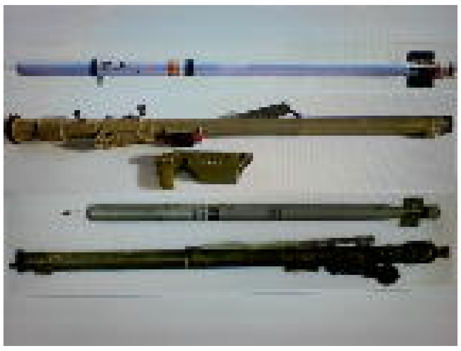


Figure 21: Resolution: 160x120, Note: Thumbnail of 9K38 Igla

Figure 22: Offset: 0x3D0E000, Resolution: 2048x1536, Completely damaged image

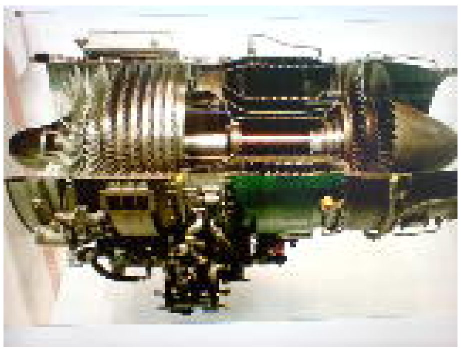


Figure 23: Offset: xxxxx, Resolution: 160x120, Recovered thumbnail of the left image, Google goggles: 17a turboject engine

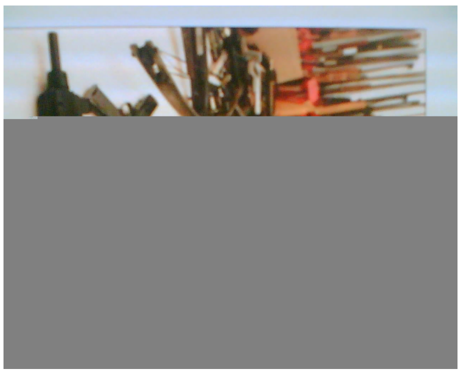


Figure 24: Offset: 0x1EA4000, Resolution: 1280x1024, Google goggles: weapons

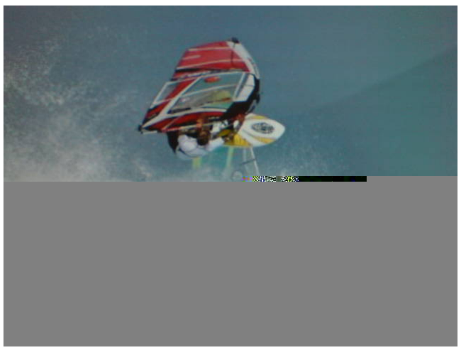


Figure 25: Offset: 0x2065000, Resolution: 640x480

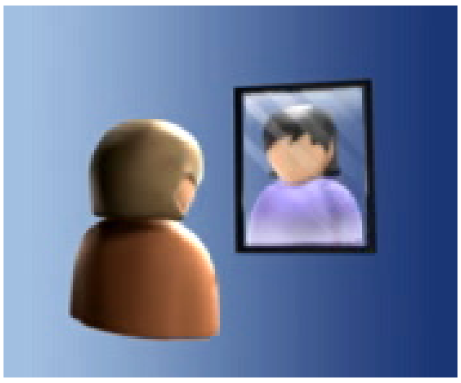


Figure 26: Offset: 0x2C12000, Resolution: 176x144

F Address book entries

The 9 entries:

Arie Handex
Schietweg 3
Alkmaar
Noord-Holland
8564 SR
Netherlands

Smurf Celtic.
Paddolaan 5
Meppel
Drenthe
1245 TG
Netherlands

Dunkin Blue
Groeneweg 4
Zoetermeer
Zuid-Holland
5428 GH
Netherlands

Manon Blaay
Bgastraat 6
Houten
2548RE
Netherlands

Hera Gordijn
Wolkstraat 4
Winsum
Noord-Holland
4785 FR
Netherlands

Hans Brinker
Dijkweg 1
Middelburg
Zeeland
8965 PU
Netherlands

Fadhel Alhassouni
Taylor street 5
Bagdad
FL 32530
Iraq

RipTide Security
Hekweg 5
Scheveningen
Zuid-Holland
2495PG
Netherlands

Stephan Aaldenberg
Vaalserberg 5
Maastricht
Limburg
7854FD
Netherlands