Lab 4 - memory dumps and carving

4.0 Objectives

The main objective of this lab is to understand SMS on a low level and learn how carving of artifacts for mobile forensics can be done with own tools. Specific objectives of the lab are:

- Learn how to interpret SMS/PDU (protocol description unit) strings.
- Carve for SMS (or other artifacts) with custom developed tools.
- Use tools for specialized carving of artifacts and more advanced functions.

4.1 Methodology

4.1.1 SMS/PDU format

Scenario

You have obtained a physical dump of a telephone memory, for which you have enumerated data structures of. You have successfully carved out the PDU strings for two SMS out of the dump as fond below (PDU SMS-1 and PDU SMS-2).
However your carver is not especially good, so the carved data may have additional garbage in the header and footer of the hex-strings. Also note that PDU SMS-2 probably is longer than 140 bytes.

PDU SMS-1:

0000000000000000FFFFFFFFFFFFFFFFFF00FFFFFFFFFFFF0107916407970980F1040B9
16407967987F0000080216222049540 07D6F0BAEC06FD00FFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFF

PDU SMS-2:

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0107916407058099F9440B916437305999F100088
02142118471408C050003200201000A000A06340641062A0020062706440628062D063106
43062F064A0634002006430628064A0631064406480643062706460020064306440648002 0
0643062F064A063400200628062F0648002006320639062A0631061F002000200647064406
43062F002006450634062A062706430644002E002E0021002000200020002000200020000A
07916407058099F9440B916437305999F1000880214211840240 7A050003200202000A000
A000A00530065006E0061007300740020006900 6D006F00720067006F006E00200041006E
006E0061007200730020007300 6B00690063006B0061007200200076006900200065006E00
200073006A00E4006C0076006D006F0072006400730062006F006D0062006100720065002
1000AFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

References

- SMS and the PDU format - http://www.gsm-modem.de/sms-pdu-mode.html
- **The site may be down so I have included a zipfile with the site stripped (makes it able to browse locally).**

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

4.1.2 SMS carving

We will now work with flash memory hex dumps from mobile feature (simple/not smart) phones trying to extract and decode SMS/PDU data from them.

Requirements

- Compressed flash memory dumps of a phone with file name: SonyEricsson_W800i_***.7z located at [server]\embedded_forensics\cell_phone_dumps\.
    - The phone dumps are either available as a XRY project (.xry file) which have some limited system/user data in the dump decoded or as raw binary (.bin) files.

It is **NOT** necessary to install any specific mobile forensic tool for this part of the lab other than the ones below:

- The XRY Reader 6.x is included in the large MSAB Forensic Pack, but it is also available in a separate file. You can use the small no license standalone XRY_READER_NOIST_*.zip to view captured logical MSAB XRY data which is found on [server]\embedded_forensics\MSAB.com.
  It is also possible to use the newer viewer tool "XRY viewer" if the XRY file format is in version 6 or newer.

- It is sufficient to use the XACT hex viewer tool at: [server]\embedded_forensics\MSAB.com\MSAB Forensic Pack v6.7**msab XACT 6.7 no installation.7z.** for some of the tasks.

Tasks and questions

a) Create and/or modify a SMS carver program/script which can find and carve out non-file based information such as SMS/PDU strings. You can for example do a script that carve out the interesting memory areas, decode it and write it to a text file for later examination.

I recommend either using Eclipse with the PyDev plugin or Visual Studio with PVTS to solve the programming task.

In the included files smsdecode.py (this is what I use for my decoding of SMS) or pdu.py you have a good start for the decoding part of the task. You can of course use a faster executing language for solving the problem but then you need to find or create the decoding capability yourself.

More specific you have to create a program that steps forward in the dump one or more bytes at time, registering potential SMS headers and footers. When the right information is gathered (offsets where a potential SMS starts and ends) save the string between the offsets (SMS header and footer) and call the `sms = pdu.decode(s, status)` function where s == SMS candidate and then print the result to a file. See appendix 1 for more help.

To know how the SMS header and footer format looks you need to examine the dump and know how the SMS:s are stored. You can use XACT to verify your findings using Menu > Edit > Find and the Menu > View > String and set Encoding to "7 Bit" and Codepage to some suitable as ASCII, Latin 1 or Unicode etc. When you have found a SMS you have to step

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

forward in the hex dump until you reach the begin index for the user data payload letting XACT decode the message for you in the String tab.

By working like this and keeping exact count/control/identification of all the fields in the SMS you can parse and decode any SMS even though they do not follow the standard formatting, or have garbage strings inside them.

b) Theoretically - if you both do a factory reset and install a new firmware on your mobile phone. How is this affecting (or not affecting) the potential evidence found in the mobile phones available flash memories? Explain in detail what may happen or is going to happen?

References

- Python Development with PyDev and Eclipse – Tutorial: http://www.vogella.de/articles/Python/article.html
- Python File Objects: http://docs.python.org/2/library/stdtypes.html#file-objects
- Regular expressions in Python: http://docs.python.org/library/re.html
- A web page that have a good description of the GSM standard for SMS is: http://www.gsm-modem.de/sms-pdu-mode.html
- PTVS (Python Tools for Visual Studio): http://microsoft.github.io/PTVS/

4.1.3 Specialized carving of artifacts and more advanced methods

Before beginning doing this part of the lab examine:

- DFRWS 2010 Forensics Challenge scenario background at: http://www.dfrws.org/2010/challenge/
- Also read thru the SSDDFJ_V1_1_Breeuwsma_et_al.pdf document at: [server]\embedded_forensics\docs\common_documents\ssddfj.org.
- Everything downloadable from DFRWS 2010 (which may be needed for this task) is also found at: [server]\embedded_forensics\DFRWS.org\2010 - cell phone forensics.
- You can view the included .xry report of a logical extraction from the SE K800i with MSAB XRY reader or the new XRY viewer if the XRY file format is in version 6 or newer.
- Example of a physical dump decoding problem in appendix 3.

Tasks and questions

a) Follow Solal Jacob's technical submission (winner of the 2010 DFRWS challenge) where he describes his tools in chapter II (2) in dfrws-2010-challenge-analysis.pdf and chapter VI (6) in dfrws-2010-challenge-technical.pdf.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

I have put an updated description in appendix 2 for how to perform the task with a newer version of DFF.

- The evidence files are the SonyEricsson_K800i_***.bin images at: [server]\embedded_forensics\cell_phone_dumps\.
- The DFF (Digital Forensics Framework) **version 1.2.0** software is found at: [server]\embedded_forensics\common-tools\digital-forensic.org.

Note!
The DFF 1.3.0 version does not have the mobile modules bundled and seems not to import modules correctly. You can try to put the modules directly below: "C:\Program Files (x86)\DFF\dff\modules\phone\k800i***.py" to see if it works.
Note that exporting the part* files to your hard drive you can further examine them with image carvers and tools as Bulk extractor, FTK and FTK Imager etc.

b) In order for carving to work well with larger evidence files found in a memory dump which have been grabbed with a memory toolkit according image 1, you need to fix the flash memory. To clarify - you have successfully been able to extract a full raw dump from the telephone flash chip to your computer hard drive.

Image 1. Memory tool-kit for low level byte-to-byte chip reading.

c) Use (you need to modify it) your program from 4.1.2 to get all SMS:s from the SonyEricsson_K800i_*.bin image.

Notes

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

If you have a working program suitable for SE W800i it is very little you have to change. Only some code to read away garbage data embedded in the SMS/PDU and to recognize new headers/status bytes. It is mainly in the PDU parsing class you need to do the changes if you are using Python.

Example of reading away 4 bytes of garbage in the PDU string.

```
dummy = d.read(4)
```

References

- Digital Forensics Framework - http://www.digital-forensic.org/
- Submissions for the DFRWS 2010 Forensics Challenge - http://www.dfrws.org/2010/challenge/ and [server]\embedded_forensics\DFRWS.org\2010 - cell phone forensics
- Regular Expressions & Search Terms for Phone Examiners -
- http://www.controlf.net/regexps/
- Data Extraction from a Physical Dump - http://www.dfinews.com/articles/2010/09/data-extraction-physical-dump

4.2 Reporting

4.2.1 SMS/PDU format

The SMS/PDU strings does not say much! Your task is to find out what they are containing for both of the SMS:es?

1. Who sent (sender number) these SMS:es?
2. What is the time stamps (TP-SCTS) of this SMS:es etc?
3. What is the message (TP-User-Data)?

In short, we need to know the information which is forensic interesting!

4.2.2 SMS carving

a) Hand in the **SMS program source code** and **output** in a packed archive with a description of added modules and their function. According to the investigators you need to answer the following questions. **Note that you do not need to handle long/multi-part SMS in your carving solution.**

- Find a location where someone (the suspect) is found.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

- Which person is this mobile phone probably belonging to? For example at the site http://www.hitta.se/ you can search for telephone numbers.

b)

- Motivate and explain whats happening and the reasons for this?

4.2.3 Specialized carving of artifacts and more advanced methods

a) Screen shots of the DFF K800i module working on your computer with the evidence found on the relevant **USB**, **system** and **TPA** partitions. See chapter VI (6) in dfrws-2010-challenge-analysis.pdf.

b) Answer the following questions

1. What problems are you facing and how would you try to fix these problems?
2. How does the memory contents and arrangement looks like?
3. Does everything come arranged nicely and neatly in clear text?

c) There is more than double the amount of the 13 SMS found in the DFRWS XRY logical extraction to find in the raw flash dump. The potential found SMS:s should be reported in a readable format by the investigator in those cases it is possible.

4.3 Lab Feedback

a) Was this a relevant and appropriate lab and what about length etc?
b) What corrections and/or improvements do you suggest for this lab?

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Appendix 1 - Hints to help solving the carving task in Python

The best approach in my view is to translate the hex strings (headers/footers) to long int numbers and compare them. To encode/decode and do hex/int conversions of strings you can use libraries as binascii and built in functions. Check out the following demo log from the Python Interactive Shell.

```
>>> "hello".encode("hex")
'68656c6c6f'
>>> "68656c6c6f".decode("hex")
'hello'
>>> int('0x10AFCC', 16)
1093580
>>> hex(1093580)
'0x10afcc'
```

Remember to open files in rb or wb (read/write binary) mode. For various examples check out: http://stackoverflow.com/questions/1035340/reading-binary-file-in-python

Pseudocode solution

Basically you either go one step (byte) forward in the binary dump and test if a certain pattern is found, then try to decode the SMS from that spot in the dump. Otherwise you optimize the search for the pattern a bit according to some algorithm so the program can search faster. File based functions as seek() and read() etc. (File Objects in Python) are going to be needed.

1. read arguments
2. open files for input and output
3. define the header and footer. You can use a long number (8 bytes) in hex as smsFooter = long('0xFFFFFFFFFFFFFFFF', 16)
4. init the pdu decoding routine/class
5. iterate thru the dump file while the file end have NOT been reached
5.1. get current header and foot candidates in the dump file (read a number of bytes)
5.2. convert header and foot to hex numbers
5.3. compare if we got a header to the defined header
5.3.1. if we got a header save the SMS start offset and remember that we visited a header
5.4. compare if we got a footer and that we already visited a header
5.4.1. if we got a footer calculate the string length (start and end offset) in the file
5.4.2. move to the SMS start offset
5.4.3. read the amount bytes we need to carve out of the file to a new SMS candidate buffer
5.4.4. remember to encode the SMS string to hex
5.4.5. detect the type of the SMS and decode it with one of the decoding modules
5.4.6. write strings to the file, undecoded and decoded
5.4.7. do other stuff as print status etc.
5.4.8. reset header visited since we visited the footer
5.5 advance/seek 1 or more steps (bytes) in the file and goto 5 (remember that reading a file automatically moves the file pointer forward)

I recommend to use Eclipse with the PyDev plugin. You can follow Lars Vogels tutorial "Python Development with PyDev and Eclipse - Tutorial" at: http://www.vogella.de/articles/Python/article.html to get you started.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Another solution is to use PTVS (Python Tools for Visual Studio): http://pytools.codeplex.com/. A quick overview: https://www.youtube.com/watch?v=JNNAOypc6Ek.

SMS carving code example

This is not a complete program, it shows partial solutions to the problem. You should be able to figure out the rest by yourself. Note you need Python v2.7.x for the example script to run.

```
try:
    argv1 = sys.argv[1]
    argv2 = sys.argv[2]
except:
    print("""
    Usage: python smscarve.py <path\\in-file-image> <path\\out-file>
    Example args: c:\\tmp\\nand_dump.dd c:\\tmp\\output.txt
    """)
    sys.exit(1)

# open image file in read binary mode and print its size
fd1 = open(argv1, mode='rb')
fSize = os.path.getsize(argv1)
print("length of image file: " + str(fSize/1048576) + " MB")

# SMS header and footer we search for
smsHeader = long('0x079164', 16) # SMS header \x07\x91\x64
smsFooter = long('0xFFFFFFFFFF', 16) # SMS footer \xFF\xFF\xFF\xFF\xFF
hLen = 5
bufStep = 1 # the normal advancement in the file
offset = 0
headerVisited = 0
footerVisited = 0

# iterate thru the whole file advancing one byte at time
while(offset < (fSize - hLen)):

    buff = fd1.read(hLen) # read 5 bytes at a time into a temporary buffer

    # get current header candidates
    qwordhead = str(buff[0] + buff[1] + buff[2]) # unsigned long int
    qwordfoot = str(buff[0] + buff[1] + buff[2] + buff[3] + buff[4]) # unsigned
long int

    # convert word to number so we can compare against head/foot
    header = long(qwordhead.encode("hex"), 16)
    foot = long(qwordfoot.encode("hex"), 16)

    if header == smsHeader:
        headerVisited = 1
        smsbegin = offset

    elif foot == smsFooter and headervisited == 1:
        footerVisited = 1
        fd1.seek(smsbegin - bytesbefore)  # goto where SMS begin minus the amount
of bytes we want to collect before the header
```


Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

```
            numBytes = offset + bufStep - smsbegin # this is the full length of our
potential SMS
            s = fd1.read(numBytes).encode("hex")     # read the whole SMS and make hex
            print s
            status = s[:2]

            # here we do the actual job of check status decoding the SMS,
            # print SMS parameters to a file, etc., etc. see the smsdecode.py example

            headerVisited = 0 # reset visited

        if foot == smsFooter and footerVisited == 1 and offset < (fSize - hLen):
            offset += numBytes  # advance the whole SMS
            footerVisited = 0
        elif foot == smsFooter and offset < (fSize - hLen): # no header is visited yet
            offset += hLen # advance hLen steps if we got many 0xFFFF...
        elif header == smsHeader:
            offset += len(qwordhead) # advance the length of header in bytes
        else:
            offset += bufStep

        fd1.seek(offset) # advance keeping offset and read position in sync

    # end for, close allocated resources and print eventual summary
```

Another better approach may be to just search for a potential SMS header and when found try to parse the TP-UDL (User data length, length of message) byte.

With this method you can calculate the SMS length and do not have to rely on a footer!

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

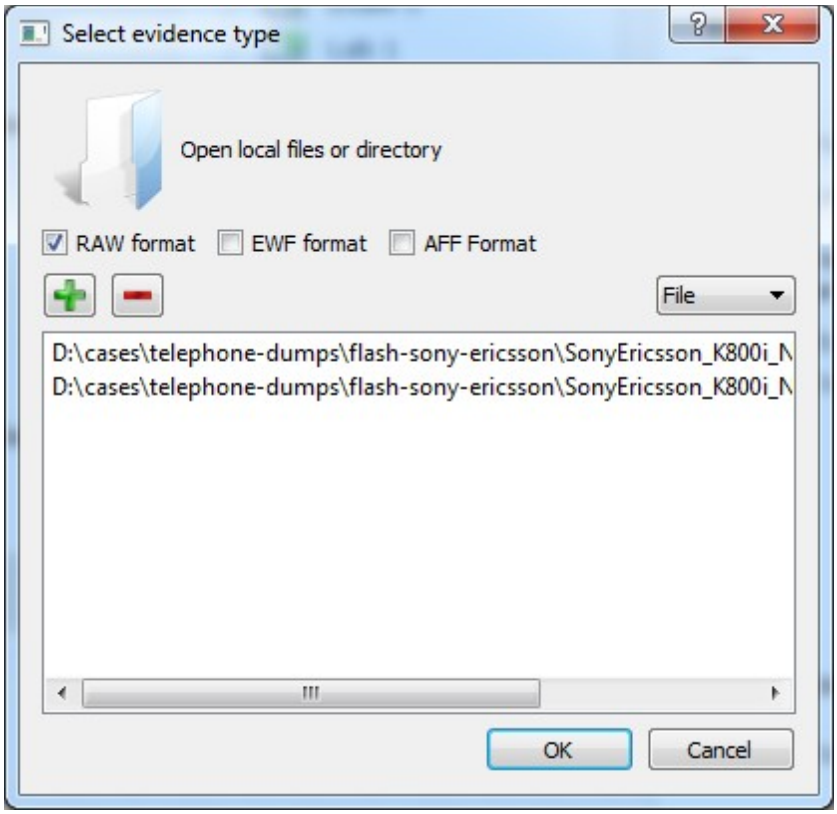
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Appendix 2 - DFF tutorial

This is a howto for DFF version 1.2.0. The full description is found in Solal Jacob's technical submission (winner of the 2010 DFRWS challenge) chapter II (2) in dfrws-2010-challenge-analysis.pdf and chapter VI (6) in dfrws-2010-challenge-technical.pdf.

The best result is achieved by running the K800i-Recover module. The K800i-Recover use the K800i module.

1. File > Open evidence files(s).



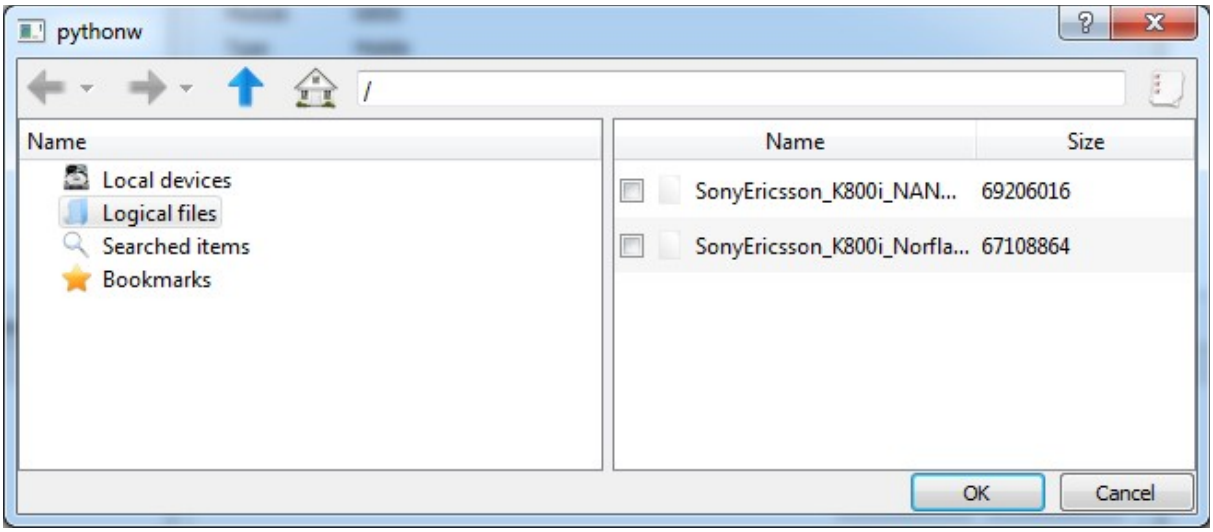
Add with green cross button and RAW selected.
SonyEricsson_K800i_NAND_NAND512R3A.bin and
SonyEricsson_K800i_Norflash_PF38F5060M0Y0BE.dmp

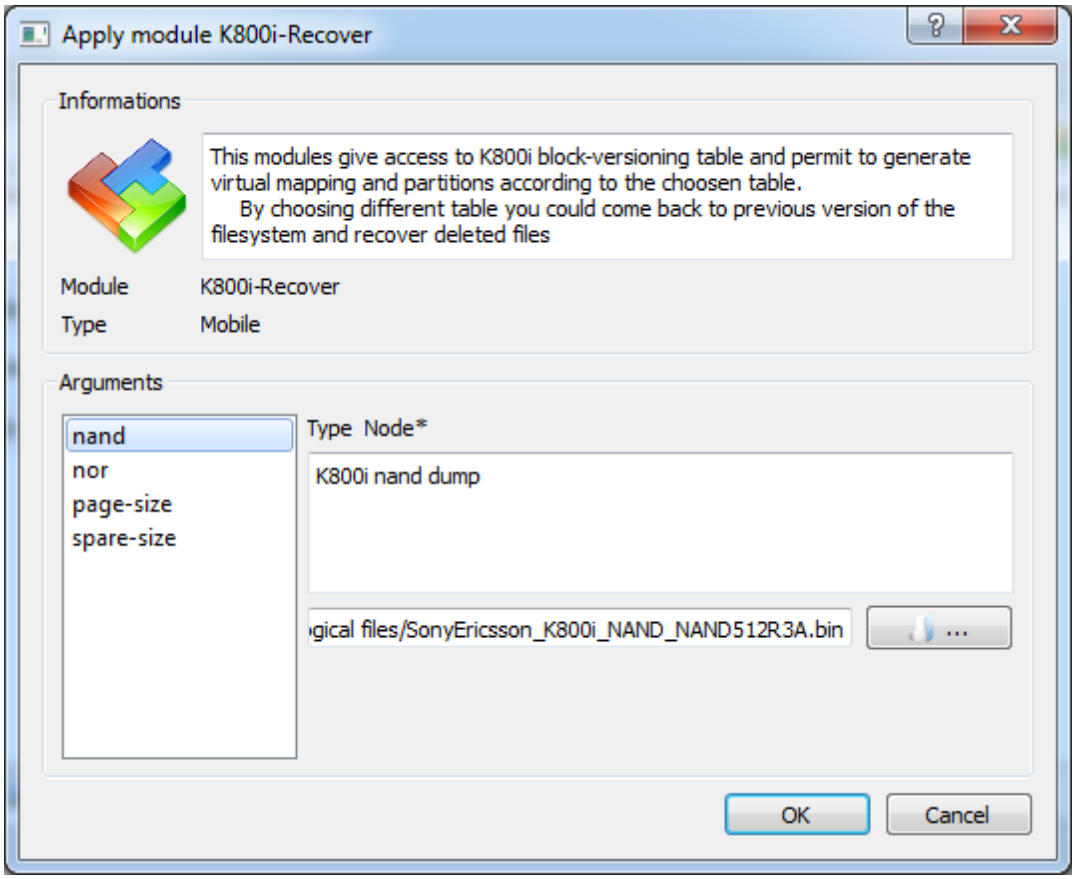
Press OK.

Select Modules > Mobile > k800i-Recover and make sure the Apply Module dialogue have the right information for NOR, NAND and spare/page size 16/512.

Now we need to select or bind the nand and nor file to the Arguments.
Mark the nand argument > press the button with 3 dots on > mark Logical files in the Name part of the pythonw dialog > select the nand file and press OK**.** Then mark the nor argument and repeat the procedure for the nor file.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/





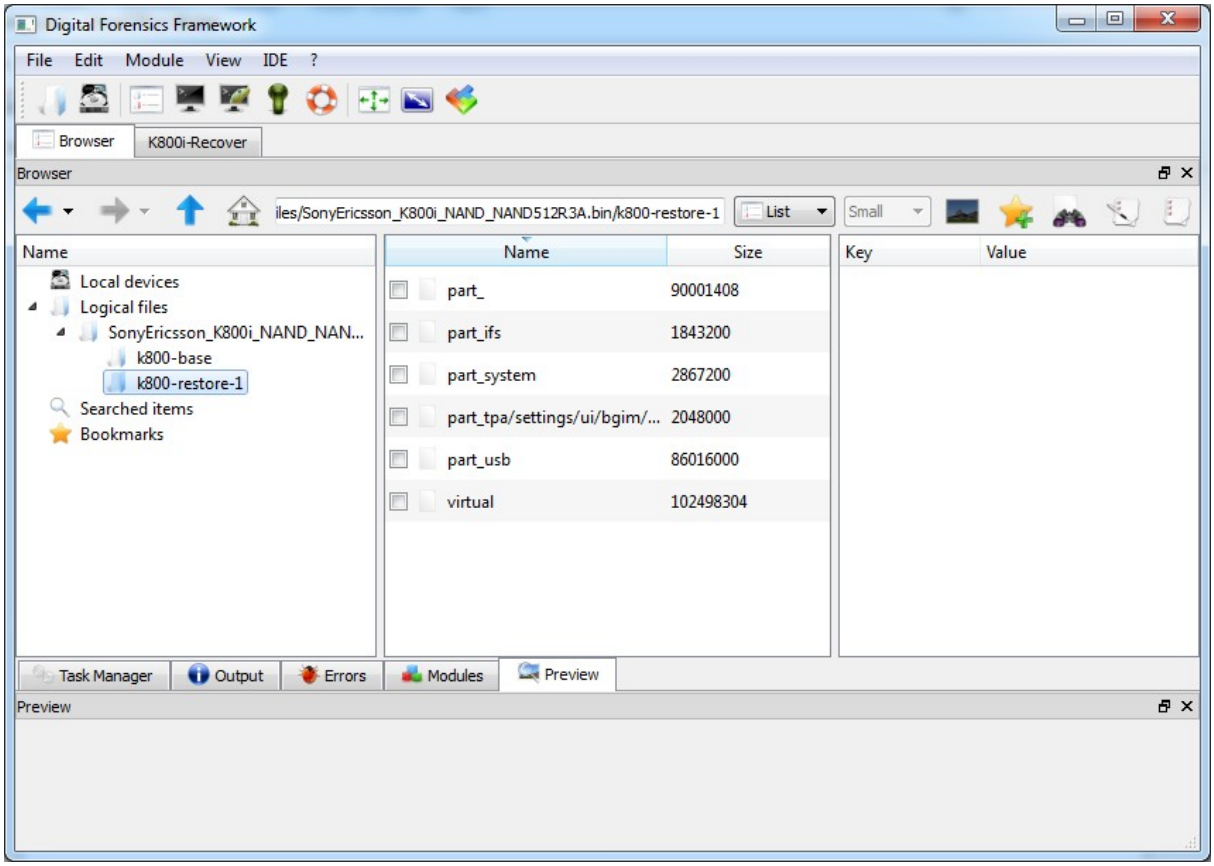
Press OK when finished.

After finished a K800i-Recover tab with a table is visible, see page 7 in dfrws-2010-challenge-analysis.pdf. Choose the following values for the table (leave default for others):
Tables 0x9 set to 0x4cc2
Tables 0x10 set to 0x4c85
Tables 0x4 set to 0x4c9e (may not be possible)
Tables 0x18 set to 0x4c86

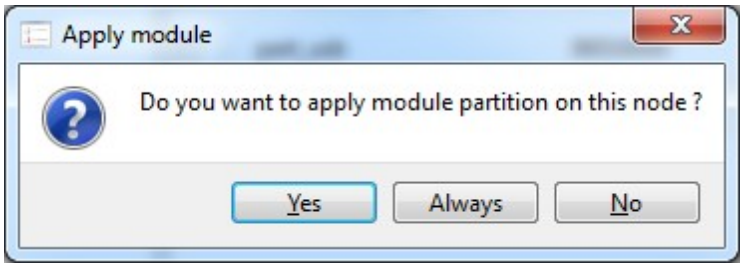
Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

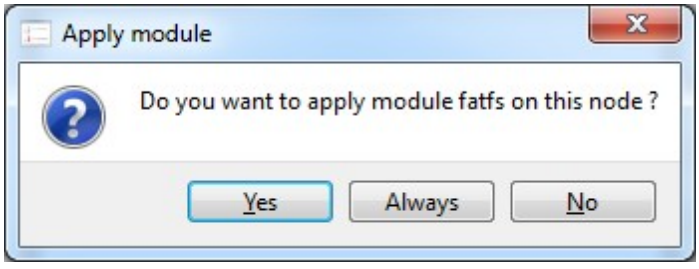
After you have done this press the Create dump button below the tables in the K800i-Recover tab so a k800-restore-1 node is created in the browser tab according to the image below.



Double click on the “part_” partition name and answer yes on the question.



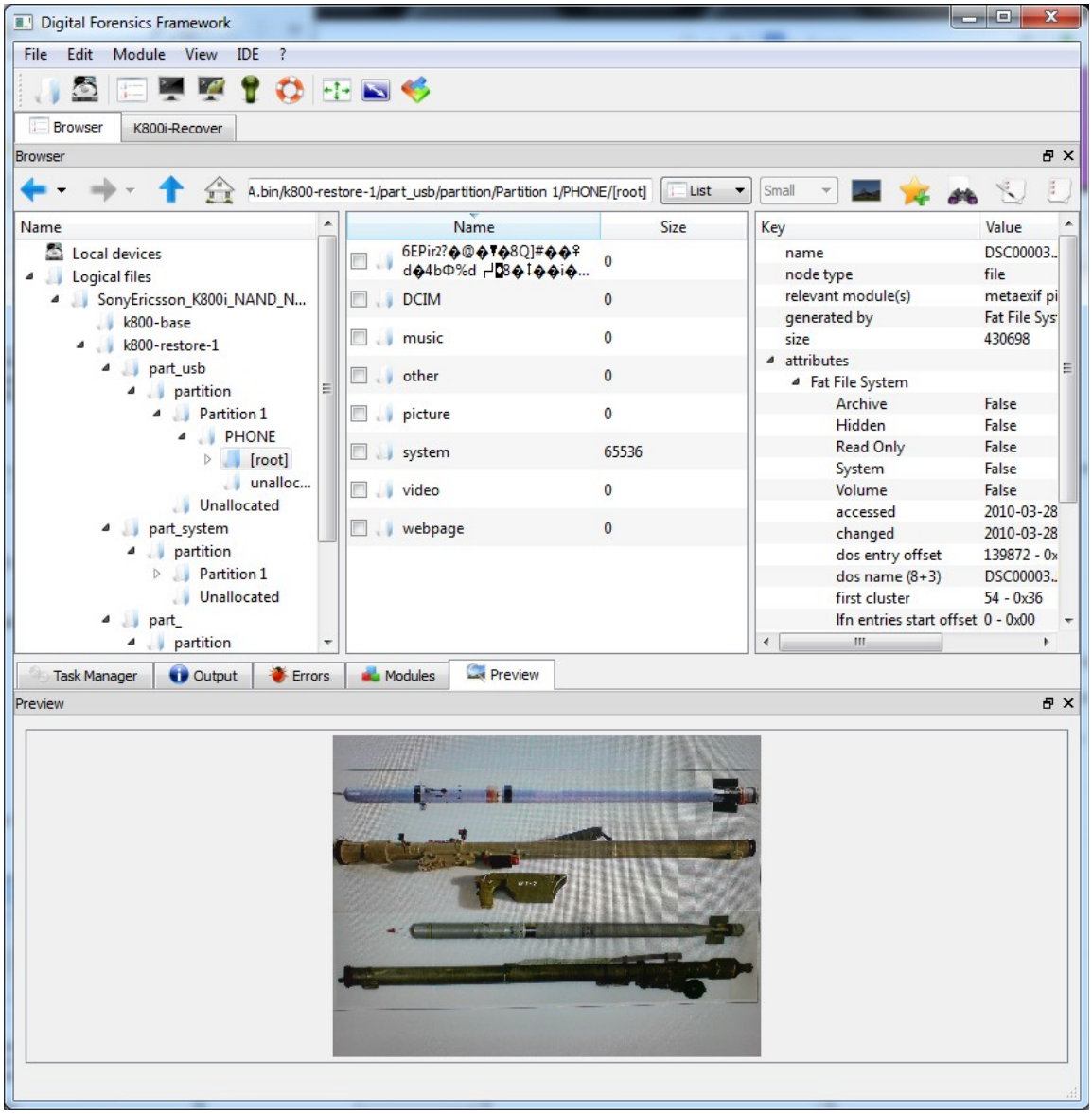
After finished one can expand the k800-restore-1 node (it has a triangle in front of it). Browse to the “part_” node which have been created and double click on the Partition 1 name and answer yes on the question.



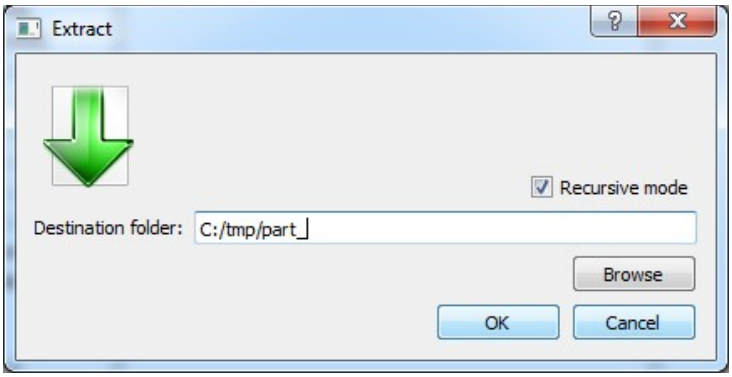
Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Continue and do the same for part_usb and part_system. When finished you can browse the phones file system and view files content.



Is possible to export interesting evidence out of the image recursive if you want. Mark the Partition 1 node in left hand window and right click for example the PHONE sub folder in the middle window and select Extract.



Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Appendix 3 - Example of decoding problem

When performing a logical extraction of the specific phone with MSAB, MMS and E-mails are not supported according to image 1. Doing an extraction with physical decoding means that no features at all is available according to image 2. So now it is up to you to find the evidence!

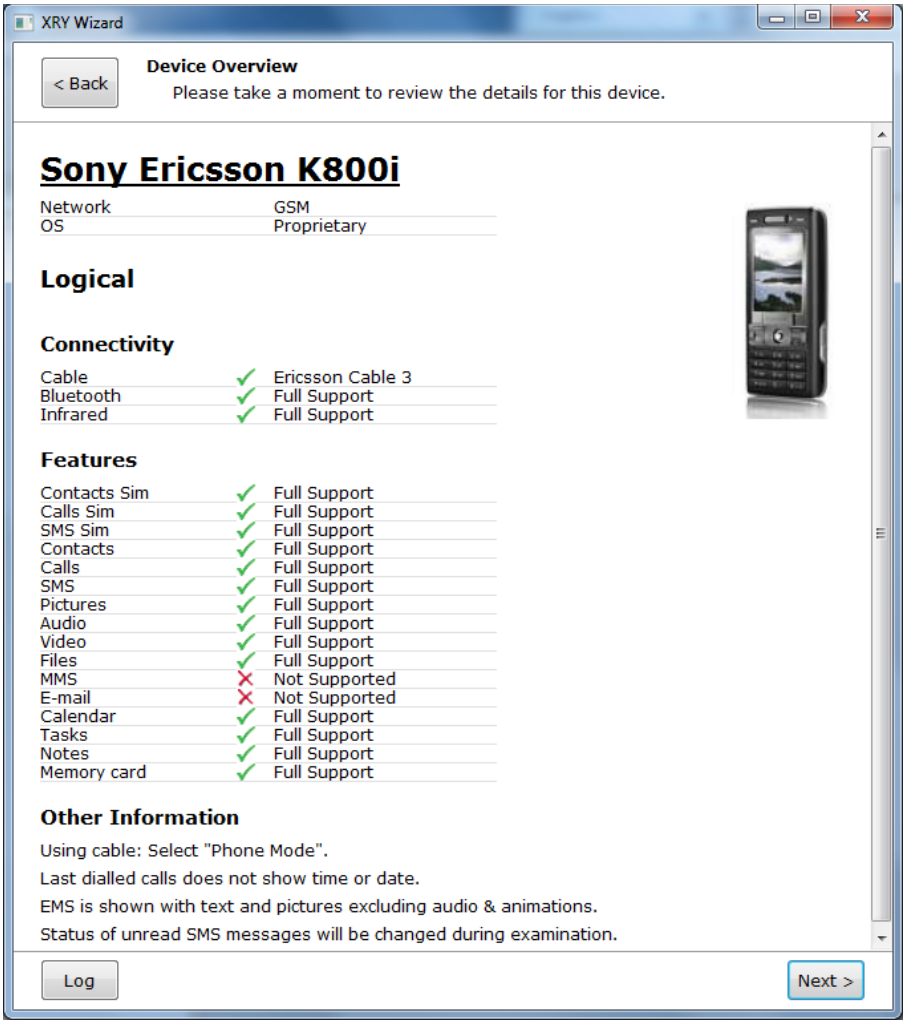


Image 1. Logical decode with XRY.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

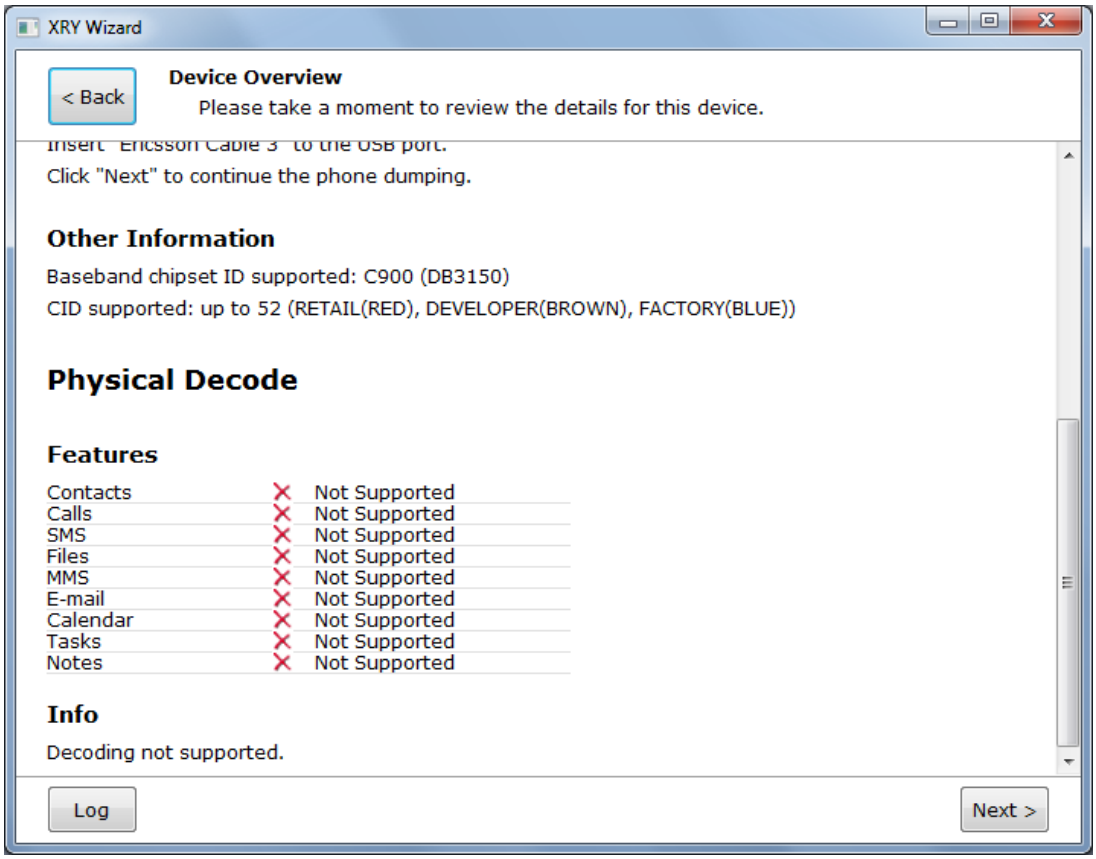


Image 2. Physical decode with XRY.

Appendix 4 – Bulk extractor

If you want you can use the tool bulk_extractor to find other artifacts than SMS in the memory dumps you have examined.

Reference

- http://www.forensicswiki.org/wiki/Bulk_extractor

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/