2016

MICT1 – Exercise Week 4

Delano Cörvers

Zuyd University

08-Mar-16

MICT1 – Exercise Week 4

Reverse Engineering Data – Exercise week 4

Group	Group 1	
Students	Delano Cörvers	(1306669)
	Davy Heutmekers	(1309730)
	Rik Kierkels	(1354442)
Module	MICT1 – Reverse Engineering Data	
Assignment	Exercise – Week 4	
School year	2015 – 2016	

Table of content

1 What we found

We received a data dump file that was build up out of 162 clusters of 4096 bytes. The total file size is 663,552 bytes.

We were able to recover three image files from the data file, two files are JPEG and one file is a PNG. The two JPEG files are the exact same image with different dimensions. The smaller image is the thumbnail for the larger image and is in fact embedded in the large image. This means that there is actually one file. The JPEG file uses the JFIF encoding and was created, edited or used by Adobe Photoshop (CC) as it contained a lot of metadata. The JPEG file had a total size of 646,929 bytes (including the thumbnail).

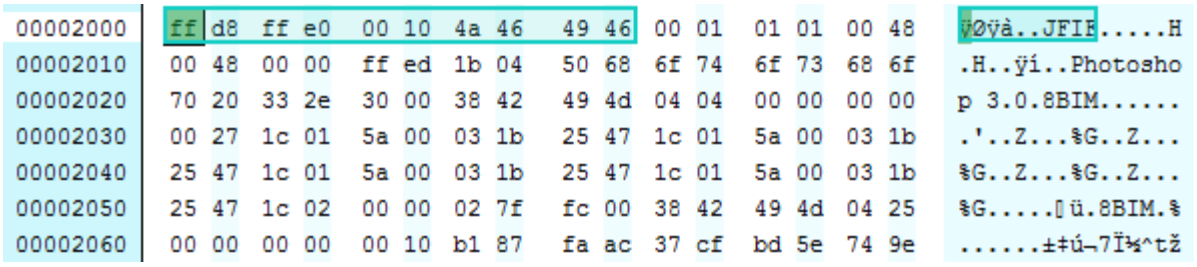


Figure 1: The header of actual JPEG image file.

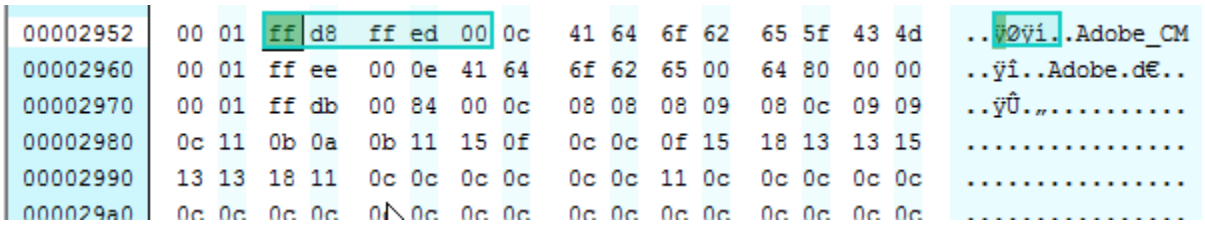
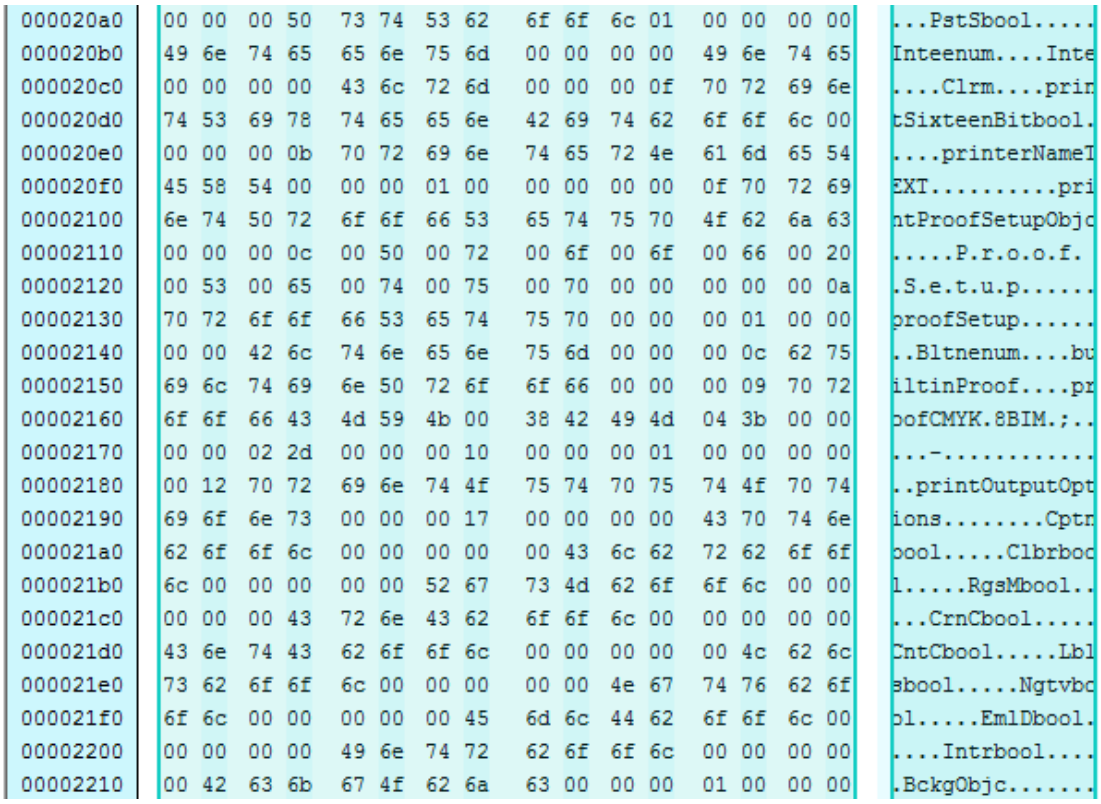


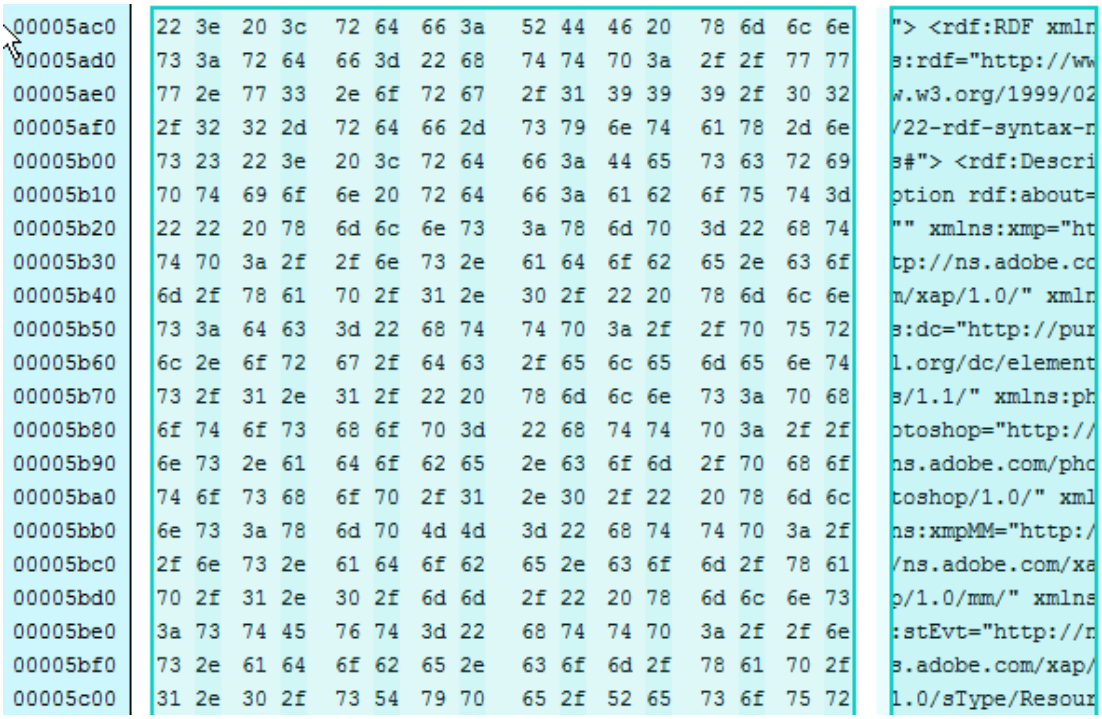
Figure 2: The header of the thumbnail, located inside the actual JPEG.

The next two screenshots will show part of the discovered metadata and print –info and style from the JPEG image file.



```
000020a0  00 00 00 50 73 74 53 62 6f 6f 6c 01 00 00 00 00  ...PstSbool.....
000020b0  49 6e 74 65 65 6e 75 6d 00 00 00 00 49 6e 74 65  Inteenum....Inte
000020c0  00 00 00 00 43 6c 72 6d 00 00 00 0f 70 72 69 6e  ....Clrm....prin
000020d0  74 53 69 78 74 65 65 6e 42 69 74 62 6f 6f 6c 00  tSixteenBitbool.
000020e0  00 00 00 0b 70 72 69 6e 74 65 72 4e 61 6d 65 54  ....printerNameT
000020f0  45 58 54 00 00 00 01 00 00 00 00 00 0f 70 72 69  EXT..........pri
00002100  6e 74 50 72 6f 6f 66 53 65 74 75 70 4f 62 6a 63  ntProofSetupObjc
00002110  00 00 00 0c 00 50 00 72 00 6f 00 6f 00 66 00 20  .....P.r.o.o.f.
00002120  00 53 00 65 00 74 00 75 00 70 00 00 00 00 00 0a  .S.e.t.u.p......
00002130  70 72 6f 6f 66 53 65 74 75 70 00 00 00 01 00 00  proofSetup......
00002140  00 00 42 6c 74 6e 65 6e 75 6d 00 00 00 0c 62 75  ..Bltnenum....bu
00002150  69 6c 74 69 6e 50 72 6f 6f 66 00 00 00 09 70 72  iltinProof....pr
00002160  6f 6f 66 43 4d 59 4b 00 38 42 49 4d 04 3b 00 00  oofCMYK.8BIM.;..
00002170  00 00 02 2d 00 00 00 10 00 00 00 01 00 00 00 00  ...-............
00002180  00 12 70 72 69 6e 74 4f 75 74 70 75 74 4f 70 74  ..printOutputOpt
00002190  69 6f 6e 73 00 00 00 17 00 00 00 00 43 70 74 6e  ions........Cptn
000021a0  62 6f 6f 6c 00 00 00 00 00 43 6c 62 72 62 6f 6f  bool.....Clbrboo
000021b0  6c 00 00 00 00 00 52 67 73 4d 62 6f 6f 6c 00 00  l.....RgsMbool..
000021c0  00 00 00 43 72 6e 43 62 6f 6f 6c 00 00 00 00 00  ...CrnCbool.....
000021d0  43 6e 74 43 62 6f 6f 6c 00 00 00 00 00 4c 62 6c  CntCbool.....Lbl
000021e0  73 62 6f 6f 6c 00 00 00 00 00 4e 67 74 76 62 6f  sbool.....Ngtvbo
000021f0  6f 6c 00 00 00 00 00 45 6d 6c 44 62 6f 6f 6c 00  ol.....EmlDbool.
00002200  00 00 00 00 49 6e 74 72 62 6f 6f 6c 00 00 00 00  ....Intrbool....
00002210  00 42 63 6b 67 4f 62 6a 63 00 00 00 01 00 00 00  .BckgObjc.......
```

Figure 3: JPEG Image print info.



```
00005ac0  22 3e 20 3c 72 64 66 3a 52 44 46 20 78 6d 6c 6e  "> <rdf:RDF xmln
00005ad0  73 3a 72 64 66 3d 22 68 74 74 70 3a 2f 2f 77 77  s:rdf="http://ww
00005ae0  77 2e 77 33 2e 6f 72 67 2f 31 39 39 39 2f 30 32  w.w3.org/1999/02
00005af0  2f 32 32 2d 72 64 66 2d 73 79 6e 74 61 78 2d 6e  /22-rdf-syntax-n
00005b00  73 23 22 3e 20 3c 72 64 66 3a 44 65 73 63 72 69  s#"> <rdf:Descri
00005b10  70 74 69 6f 6e 20 72 64 66 3a 61 62 6f 75 74 3d  ption rdf:about=
00005b20  22 22 20 78 6d 6c 6e 73 3a 78 6d 70 3d 22 68 74  "" xmlns:xmp="ht
00005b30  74 70 3a 2f 2f 6e 73 2e 61 64 6f 62 65 2e 63 6f  tp://ns.adobe.co
00005b40  6d 2f 78 61 70 2f 31 2e 30 2f 22 20 78 6d 6c 6e  m/xap/1.0/" xmln
00005b50  73 3a 64 63 3d 22 68 74 74 70 3a 2f 2f 70 75 72  s:dc="http://pur
00005b60  6c 2e 6f 72 67 2f 64 63 2f 65 6c 65 6d 65 6e 74  l.org/dc/element
00005b70  73 2f 31 2e 31 2f 22 20 78 6d 6c 6e 73 3a 70 68  s/1.1/" xmlns:ph
00005b80  6f 74 6f 73 68 6f 70 3d 22 68 74 74 70 3a 2f 2f  otoshop="http://
00005b90  6e 73 2e 61 64 6f 62 65 2e 63 6f 6d 2f 70 68 6f  ns.adobe.com/pho
00005ba0  74 6f 73 68 6f 70 2f 31 2e 30 2f 22 20 78 6d 6c  toshop/1.0/" xml
00005bb0  6e 73 3a 78 6d 70 4d 4d 3d 22 68 74 74 70 3a 2f  ns:xmpMM="http:/
00005bc0  2f 6e 73 2e 61 64 6f 62 65 2e 63 6f 6d 2f 78 61  /ns.adobe.com/xa
00005bd0  70 2f 31 2e 30 2f 6d 6d 2f 22 20 78 6d 6c 6e 73  p/1.0/mm/" xmlns
00005be0  3a 73 74 45 76 74 3d 22 68 74 74 70 3a 2f 2f 6e  :stEvt="http://n
00005bf0  73 2e 61 64 6f 62 65 2e 63 6f 6d 2f 78 61 70 2f  s.adobe.com/xap/
00005c00  31 2e 30 2f 73 54 79 70 65 2f 52 65 73 6f 75 72  1.0/sType/Resour
```

Figure 4: JPEG Image metadata

The discovered PNG file is one by one pixels large, so the image is not usable (as there is nothing to see).

In the provided file we also found a GIF file and another PNG file. These files were not recoverable. The GIF file had valid magic numbers and trailer, but had no information in between. The PNG also had a header and trailer, but had no data blocks.

If we take a look at the remaining file size of the provided file when we exclude the useable JPEG image, we can conclude that 16,623 bytes of the file is not useable.

Figure 5: The recovered JPEG (JFIF) image file

Figure 6: The recovered thumbnail (located inside the JPEG Image).

2 Where we found it

In this chapter the locations and spans of each file we discovered are noted.

Block	Start	End	Start cluster (4096 KiB)	Ending cluster (4096 KiB)
JFIF	0x00002000 (8,192)	0x0009ff10 (655,120)	3	159
Thumbnail	0x00002952 (10,578)	0x00003a9b (15,003)	3	4
Incomplete GIF	0x0009ff11 (655,121)	0x0009ff35 (655,157)	159	159
PNG (0 frames)	0x000a0000 (655,360)	0x000a0043 (655,427)	160	160
Left-over-data 1	0x00000000 (0)	0x00001fff (8,191)	1	2
Left over-data 2	0x000a0044 (655,428)	0x000a1fff (663,551)	160	162

Figure 7: The locations and spans of each file we discovered.

3 How we found it

We opened the file with Hex Editor Neo to figure out if we could find a structure in the file. While scrolling through, we noticed a few image file headers (JFIF/JPEG, PNG and GIF). At first, we thought it would be easier to work with the file if we split the provided data into smaller files that were the size of each cluster. To split the file in clusters we wrote a small Python script that would do this for us.

```
from os import getcwd


# Split the provided data file into clusters of 4096 bytes.
def cluster_splitter(cluster_size=4096):
    file_path = getcwd() + r"\files\data"

    with open(file_path, "rb") as f:
        cluster_counter = 1

        seeker = 0
        f.seek(seeker)
        index = f.read(cluster_size)

        while index != "":
            create_cluster_file(index, cluster_counter)
            cluster_counter += 1
            seeker += cluster_size

            # Get ready to process the next 4096 byte cluster.
            f.seek(seeker)
            index = f.read(cluster_size)


# Write each discovered cluster to a new file.
def create_cluster_file(array, counter):
    file_path = getcwd() + r"\files\clusters\cluster_"

    with open(file_path + str(counter), "wb") as f:
        f.writelines(array)


# Entry point of the application.
def main():
    print "The tool has started...\n"
    cluster_splitter()
    print "\nFinished running..."


main()
```

Figure 8: Screenshot of the Python script

After having created and used the tool, we started to think about extracting usable files or parts of file from the data. We realized that we didn't actually had to split the files, although it made manually checking the files easier.

With the use of a File Carver, QPhotoRec, we pulled out the image files that we had previously detected manually. QPhotoRec is able to scan drives or image files. Since we had neither, we made a copy of the original data file and added the .img (image) extension to it. This way we were able to open the data using the File Carver.

The File Carver recovered three images, two JPEG files and one PNG file. As said previously in chapter 1 "What we found", the PNG file was not usable and the two JPEG files are in fact one with the smaller file being the thumbnail for the actual image.

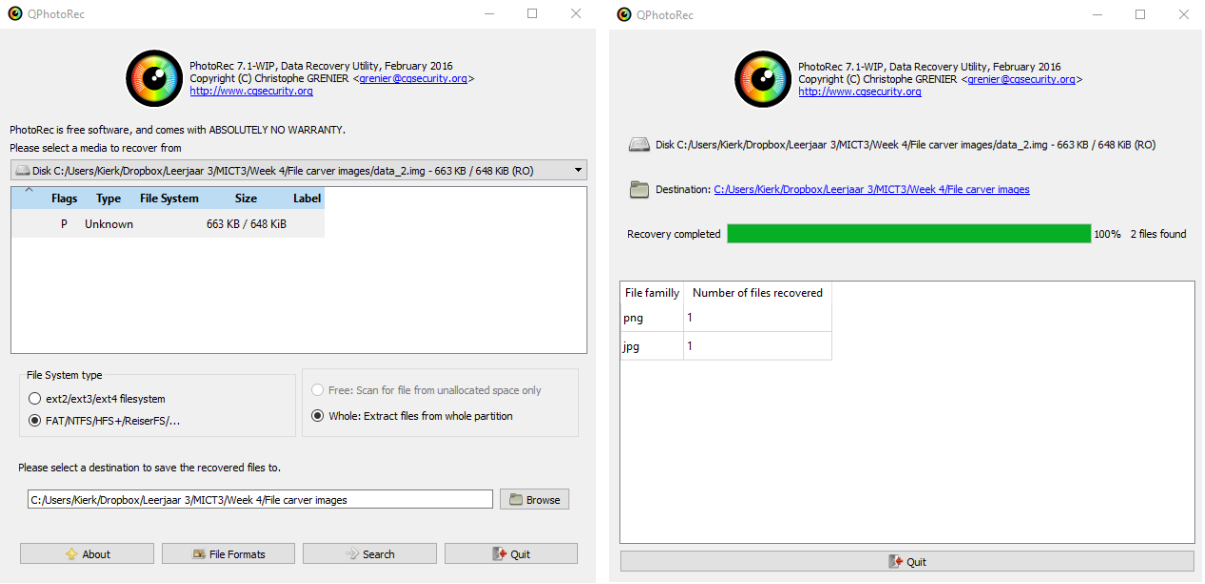


Figure 9: QPhotoRec - Select the data (image file) and the final results.

It is possible to do this task without the use of a File Carver. The File Carver checks for the headers or magic numbers of certain files and then proceeds to find a trailer for that file. It also checks for mandatory blocks to make sure it is a valid file type. Manually you would use a hex value search to find the start of a block using the magic numbers for various image files. As we already found some files ourselves when we opened the file (JFIF, PNG and GIF) it would make sense to look for these magic numbers and then proceed to find the ending signature. When you would find these you can extract the image and test it.

After we recovered the image files, it seemed like there was a lot of random data inside the images. We figured this was left over data from other files. To confirm our thoughts we used Jeffrey's Exif Viewer (http://regex.info/exif.cgi). This is how we actually found out about the metadata in the JPEG image which was most likely added to the file by Adobe Photoshop. This gives us reason to believe the image file was created or, more likely, modified using Adobe Photoshop CC.

Jeffrey's Exif Viewer

Jeffrey Friedl's Image Metadata Viewer
(How to use)

Note: extra functionality is enabled when you use Firefox or another Gecko-based browser..

Some of my other stuff
· My Blog · Lightroom plugins · Pretty Photos
· "Photo Tech"

From Web / From File — Image URL: [ ] View Image At Url [CLEAR IMAGE]

Basic Image Information

Target file: f0000016.jpg

Date:	**July 22, 2015** 3:10:29PM (timezone is 2 hours ahead of GMT) (7 months, 16 days, 21 hours, 2 minutes, 54 seconds ago)
File:	**2,400 × 1,200** JPEG (**2.9** megapixels) 646,929 bytes (632 kilobytes)
Color Encoding:	Embedded color profile: "sRGB"

Extracted **160 × 90** 4.3-kilobyte "Photoshop:PhotoshopThumbnail" JPG
Displayed here at 200% ($^{1}/_{50}$ the area of the original)

Click image to isolate; click this text to show histogram

Main JPG image displayed here at 19% width ($^{1}/_{28}$ the area of the original)

Click image to isolate; click this text to show histogram

Here's the full data:

XMP

XMP Toolkit	Adobe XMP Core 5.5-c021 79.155772, 2014/01/13-19:44:00
Creator Tool	Adobe Photoshop CS6 (Windows)
Create Date	**2015:07:22** 15:10:29+02:00 7 months, 16 days, 21 hours, 2 minutes, 54 seconds ago
Modify Date	**2016:01:13** 12:07:13-08:00 1 month, 25 days, 15 hours, 6 minutes, 10 seconds ago
Metadata Date	**2016:01:13** 12:07:13-08:00 1 month, 25 days, 15 hours, 6 minutes, 10 seconds ago
Format	image/jpeg
Color Mode	RGB
ICC Profile Name	sRGB IEC61966-2.1
Instance ID	xmp.iid:45c7f710-9e41-40bb-8d80-6783e4e4cd4d

Figure 10: Screenshot of Jeffrey's Exif Viewer