James Madison University

CS633 Project 1

Author:
Josh Feehs

February 16, 2014

1 Executive Summary

I found that clusters 2373-2552 had junk data. I could not identify those clusters as belonging to their own f

Table 1: Table representing the structure of the disk image

2 Analysis

2.1 Initial Investigation

The first steps in my investigation were to download the floppy image and verify its SHA256 checksum. Once I verified the checksum, I copied the file and verified the copies checksum as well. This way, I was able to preserve the original verified copy of the image and ensure that the rest of my investigation does not change the original data. Once that was complete, I decided to mount the image to see if that would give me any information on what could be on the disk. Given that I did my work on a Mac, I made a copy, changed it to a .dmg file, and mounted the file using `hdiutil mount lab_image.dmg`. Once the disk was mounted, I used the terminal to go into the disk and see what I could find. A normal `ls` command did not show anything, but `ls -al` returned

```
1 drwxrwxrwx@ 1 batman  staff  7168 Feb  4 12:46 .
2 drwxrwxrwt@ 4 root    admin   136 Feb  4 12:46 ..
3 drwxrwxrwx@ 1 batman  staff   512 Feb  4 12:46 .Trashes
4 -rwxrwxrwx  1 batman  staff  4096 Feb  4 12:46 ._.Trashes
5 drwxrwxrwx  1 batman  staff   512 Feb  4 12:47 .fseventsd
```

As expected, this technique was not going to be very helpful for discovering files on the floppy image. In fact, some of the things discovered (like .Trashes) were actually added by OS X when I mounted the drive.

2.2 Inspection of the Boot Sector

The next step in recovering the disk was going to be to use the boot sector to identify where each part of the floppy begins. Inspection of the boot sector showed that each sector was 512 bytes in size, that there is one sector per cluster, that there is only one reserved sector, that there are two FATs which are 9 clusters in length each, and that the root directory has 224 entries of 32 bytes each, meaning that it is comprised of 14 sectors. This means that the disk is roughly laid out as follows:

Image Section	Starting Sector	Length
Boot Sector	0	1
FAT # 1	1	9
FAT # 2	10	9
Root Directory	19	14
Cluster 2	33	1
Cluster 3	34	1
Clusters cont'd	35	1 each, until end of disk

Table 2: Table representing the structure of the disk image

2.3 Inspection of the FATs and Root Directory

The next step was to look at the FATs and Root Directory to see if they had any useful information in them. Unlike the FATs in the scan24 practice image, these FATs were completely empty. Unfortunately, the FATs were not going to be useful in helping me recover files. Likewise, the entirety of the root directory was empty. Whatever had happened to the file before I received the image, all potentially useful information in the FATs or root directory was erased and zeroed out.

2.4 Recovering Files

This section contains the meat of my analysis, as the previous sections did not assist in recovering any files. The remainder of the section is split up into subsections where the acquisition of each file is discussed in detail. — **JPG files: name is ***— My first test was to look for the known header of a JPG file, as we had talked about it in class. I searched for "JFIF" using grep, and found two potential headers at 0x00013600 (sector 155, cluster 124) and 0x00024a00 (sector 293, cluster 262). Given that these are both at the beginning of clusters, they may actually be JPG headers (versus being random noise or data for a text file). I then wanted to find the potential footers for those files, which would be signified as 0xFFd9 near the end of a cluster. I was able to find two such entries, one at 0x000586d0 and the other at 0x000f8e70. These are near the end of clusters beginning at 0x00058600 (sector 707, cluster 676) and at 0x000f8e00(sector 1991, cluster 1959).**

From there, I wanted to find out how many consecutive sectors there were starting from the sector 155. In order to do this, I used `dd` to copy out one sector (starting at the header) and inspected the resulting file to see if it had been distorted at all. I used increasingly large boundaries (powers of 10) for the count parameter until I got distortion, and then used binary search to find the boundary of the file data. In this case, I found that there were 138 consecutive sectors. This means that the first part of the JPG data is contained between sectors 155 and 292.

From there, I wanted to find out how many consecutive sectors there were starting from the sector 293. In order to do this, I used `dd` to copy out one sector (starting at the header) and inspected the resulting file to see if it had been distorted at all. I used increasingly large boundaries (powers of 10) for the count parameter until I got distortion, and then used binary search to find the boundary of the file data. In this case, I found that there were 304 consecutive sectors. This means that the first part of the JPG data is contained between sectors 293 and 596.

2nd image commands:

```
1 dd if=working_lab1_image of=partial2.jpg skip=293 count
    =304
2 dd if=working_lab1_image skip=1361 count=98 >>partial2.jpg
3 dd if=working_lab1_image skip=1792 count=200 >> partial2.
    jpg
```

PNG

Using strings, I found that it was likely that there would be PNG files on the floppy disk. However, when I searched the file for PNG headers, I was not able to find any headers at the beginning of clusters. However, I was pretty sure that there were PNG files in the floppy, which would mean that there were likely ZIP files that contained PNG's.

ZIP Files

I found two ZIP file headers: at 0x0007d600 (sector 1003)and 0x00142e00 (sector 2583) to 00168000- (sector) 2880 First one says it is 393 or 97 clusters long –¿ its central directory entry is in df760 Second one may go to 2880, then continue elsewhere. I think it continues in 4200 (sector 33) I think the central directory is in df770, also look at a9640

First zip:

```
1 dd if=working_lab1_image of=zip1.zip skip=1003 count=158
2 dd if=working_lab1_image skip=1459 count=333 >> zip1.zip
```

second one (142e) has checksum: d0 a0 47 58 I found one ZIP footer: 0xDFF00, (sector 1791) –¿ this is definitely the ending.

to get the second one:

```
1 dd if=working_lab1_image of=zip2.zip skip=2583
2 dd if=working_lab1_image skip=33 count=122 >>zip2.zip
3 dd if=working_lab1_image skip=1355 count=6 >>zip2.zip
```

mp3 708-1002 1991-2403

3 Answers to Questions