EE6012/ET4027 – Module Assignment

J. Walker, February 2022

Objectives

- Learn how to uses a disk hex editor to explore a disk drive
- Learn how to use an imaging tool to image disk drives and mount images
- Learn one disk partition scheme in detail
- Learn the FAT file system in detail
- Learn the NTFS file system in detail
- Learn how to write a basic file system forensic tool

Assumptions

1) You will use the **Sample_1.dd** image file as your reference disk drive for the work. This file can be downloaded from an S3 bucket located at:
https://imagedataforee6012et4027.s3.eu-west-1.amazonaws.com/Sample_1.dd

   There is also a small text file which is the report made by the FTK Imager tool on the reference disk drive when used to make the image, located at:
https://imagedataforee6012et4027.s3.eu-west-1.amazonaws.com/Sample1.001.txt

2) You will assume that your forensic tool will only be used on a disk drive image that has a standard MBR and the first partition will always be formatted as a FAT-16.

Requirements

Write a program (your forensic tool) to do the following:

a) **Partition information -** Display the number of partitions on the disk and for each partition display the start sector, size of partition and file system type.

b) **FAT Volume information** – For the first partition only, display the number of sectors per cluster, the size of the FAT area, the size of the Root Directory, and the sector address of Cluster #2.

   For the first deleted file on the volume's root directory, display the name and size of that file, and the number of the first cluster. Display the first 16 characters of the content of that file (assume it is a simple text file).

c) **NTFS Volume information** – Display information for an NTFS partition as follows:
[See Addendum-1 for detail]

- How many byes per sector for this NTFS volume
- How many sectors per cluster for this NTFS volume
- What is the sector address for the $MFT file record
- What is the type and length of the first two attributes in the $MFT record

User interface

There are no special user interface requirements. A simple text console will suffice. Your program will be launched as: **Yourprog** *Sample_1.dd* (decide on a suitable name for your program, pass the image file name to your program)

Programming language and platform

Any suitable programming language can be used. An executable file (or runnable script etc.) should be produced to be demonstrated to the lecturer in the lab. Depending upon the platform and language used it may be necessary to submit code with instructions for compiling an executable. Acceptable computer platforms are: Windows PC, Linux PC or Apple OSX computer.

Deliverables

The project assignment has two (2) phases as follows:

PHASE 1) Forensic tool to display the partition information only

- Write a program that will give a tool to display the **partition** information only, i.e. **part a)** of the requirements only.
- Test the tool using the *Sample_1.dd* image file to confirm that you can read the correct information.
- Make a **new image** of some USB disk key. Test your program against your new disk image file and confirm the results. (You will need to learn the FTK Imager - or an equivalent tool.) As part of testing provide evidence that you have **mounted the image** as a way of confirming what information is contained on the image drive. Note that you may do this using FTK Imager or mounting the *image* itself as part of a filesystem.

  The **FTK Imager** is available from Access Data on:
  https://accessdata.com/product-download
  Note that you have to enter an email address on the website to get a link to download the product, however no further commitment is required.

SEE ADDENDUM-3 FOR EXAMPLE PROGRAM SOLUTION

Submission requirements:

Submit via SULIS:

1) Document report (in pdf format) that follows the outline given in the Addendum-2 to this document.
2) The source code.
3) The executable will be demonstrated (including with unseen test disk images) in a lab session following the submission deadline.

(DO NOT SUBMIT ANY IMAGE FILES – THEY ARE TOO BIG!).

PHASE 2) Full tool

Note – the Phase 2 work is **inclusive of** the Phase 1 work.

Write the full tool to display *all* the required features as stated above:

a) Partition information,
b) FAT Volume formation
c) NTFS Volume information (see Addendum-1)

Submission requirements:

Submit via SULIS:

1) Document report (in pdf format) that follows the outline given in the Addendum-2 to this document.
2) The source code.
3) The executable *or* code plus instructions to compile for testing in a lab session.

GROUPS OR INDIVIDUAL PROJECTS

Students can work in groups of up to three persons – meaning that you can work as an individual, in a group of 2 or a group of 3 persons.

You must decide at the start of the project on your individual or group status and this cannot be changed once the project is started. <u>Once you have decided on your group (if any) please notify the lecturer by email by the end of Week 5, Friday 25th February 2022</u>.

The same project standard is expected regardless of individual or group assignments.

Submission

Submit project work via **SULIS** in the normal project submission fashion.

PHASE 1

This phase of the exercise represents **20%** of the total module assessment.

Students/groups will submit via SULIS their submission to the lecturer by Wednesday, 17:00 hours, of week 7 (9th March 2022). Late reports will not be accepted.

PHASE 2

This phase of the exercise represents **20%** of the total module assessment.

Student/group will submit via SULIS their submission to the lecturer by Monday, 17:00 hours, of week 11 (4th April 2022). Late reports will not be accepted.

ALTERNATIVE SUBMISSION SCHEDULE

If a student/group wants to submit the full project (i.e. phase 2) by the PHASE 1 deadline – then just a single submission will suffice. Simply submit a single assignment to SULIS for the **Assignment Phase 1** deadline.

Evaluation and assessment

The lecturer will study your submitted project report and the source code and you will then run the program against some sample image files in the lab demonstration sessions for each phase. For any **group** submission the lecturer will talk to each group member individually before deciding on a grade. If you do a **group** submission please indicate clearly who did what in the report as indicated in the instructions in Addendum 2.

This exercise represents **40%** of the total module assessment. There will be an **exam question on the final exam paper** based on this assignment. The report quality will represent **35%** of the project marking (in each phase or combined).

ADDENDUM 1

NTFS support information

In this section you will get support on how to write a program to explore some features of the NTFS partition.

OBJECTIVE:

Learn how to write a **forensic analysis tool** for the NTFS file with some introductory features only.

Program requirements

You program will examine the NTFS file partition volume and report the following information to the screen:

- How many **bytes per sector** for this NTFS volume
- How many **sectors per cluster** for this NTFS volume
- What is the **sector address** for the $MFT file record
- What is the **type** and **length** of the first attribute in the $MFT record
- What ids the **type** and **length** of the second attribute in the $MFT record

GUIDELINE INSTRUCTIONS

NTFS BOOT SECTOR INFORMATION

Go to the NTFS.COM site and study the following page so that you understand the NTFS Partition Boot Sector, the PBS as follows:

http://ntfs.com/ntfs-partition-boot-sector.htm

Write your program based on the following steps:

Go to the first sector of an NTFS partition on your disk. On the 'Sample_1.dd' image this is sector address 1606500 in decimal.

From the NTFS.COM page above you will know where to find the following information from the NTFS PBS:

Offset 0x0B Bytes Per Sector (200 hex or 512 decimal)
Offset 0x0D Sectors Per Cluster (8 decimal)
Offset 0x30 Logical Cluster Number for file $MFT (4 in our sample image)

On the 'Sample_1.dd' image the Logical Cluster Number for file $MFT will be 0x0004, so this means the sector address will be 4*8 which is 32 sectors.

GO TO $MFT FILE RECORD FOR NEXT INFORMATION

Go to start of $MFT record. For our 'Sample_1.dd' image this is sector address (1606500 + 32) in decimal, i.e. 1606532 sectors into the disk.

In the NTFS Course Notes see table N5 "Data structure for a basic MFT record" and you can see the 'Offset to the first attribute' is in bytes 20 .. 21, or 0x14 .. 0x15 in hex. The value here for the sample image will be 0x0038 bytes; so this offset, 0x38 bytes from the start of the $MFT record, will put you are at the start of the first attribute.

In the NTFS Course Notes see table N3 "Attribute header data structure for a resident attribute record" and you can see the first four bytes gives the Attribute type identifier, which is 0x10, or 16 decimal, in the sample image which is a $STD_INFO type attribute as expected.

Now look in bytes 4..7 which will give the attribute length, which is 0x60 in the sample image. Add this length to the offset to the first attribute (0x60 + 0x38) and this 0x98 bytes is the offset to the second attribute from the start of the $MFT record. At this location in the sample you will find the value 0x30, or 48 decimal, which is the $FILE_NAME type attribute as expected. You can also find the length of this attribute.

FINISH

Once you write the program to the requirements above you will be well on your way to writing a full NTFS file system forensics tool. However, you have done enough now to satisfy yourself that you have the knowledge and ability to do the job, so we can stop here.

ADDENDUM 2

The document file

Each submission for this laboratory project will include a document report. Note, the document does not at all need to be long and wordy – but must be of good quality and presented in the **format of a technical report** and will include all items listed below:

1) The file will be submitted in **PDF format.**

2) The document will have the following information and sections

Front page
Title page with student name or group member names, date, module code and name of assignment.

Requirements
Briefly summarise the assignment requirements from the assignment instructions in the handout.

Description of solution
Briefly describe the features of your solution for each part of the assignment commenting on any special problems or issues.

Testing and results
State how you tested your program or programs and record any results. This applied to both phases. Specifically include the following:

(i) Describe how you tested your program using the given image file,
(ii) Describe how you tested your program using your own image file, including evidence of mounting the image as described in Phase 1 Deliverables.
(iii) Use **screen shots** as evidence of output for (i) and (ii) above.

Instructions
Briefly write instructions on how to run your program, clearly stating any necessary tools, compilers etc.

Statement of completion
Briefly state that you, or the group, have completed all of the requirements as **original** work, or summarise any aspects that you could not complete. For group work briefly state the contribution of each group member.

Source code
Include your source code as part of this document.

……end

ADDENDUM 3

Sample C program that meets Phase I requirements

```
// MiniPart.c    D. Heffernan  15/Nov/2016
//////////////////////////////////////////////////////
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Partition{ char type; int start_sect; int size;} part_entry[4];  // 4 x partition table entry

int main(int argc, char *argv[])
{
 // Define some variables
        int i, offset = 16, not_exist = 0;int *tmp_ptr;
        FILE *fp;
        char buf_part_table[64], vol_type[12];

        fp = fopen(argv[1], "rb"); // Open file for reading - binary mode. Should use error check!
        fseek(fp, 0x1BE, SEEK_SET); // Seek to the start of the part_entry list
        fread(buf_part_table, 1, 64, fp); // Read the 64-byte block to memory buffer

        for (i=0; i <4; i++)            {
                part_entry[i].type = *(char*)(buf_part_table + 0x04 +(i * offset) );
                if (part_entry[i].type == 0) not_exist++;
                part_entry[i].start_sect = *(int*)(buf_part_table + 0x08 +(i * offset));
                part_entry[i].size = *(int*)(buf_part_table + 0x0C + (i * offset)) ;

              switch (part_entry[i].type)  {
                case 00 :           strcpy ( vol_type, "NOT-VALID");  break;
                case 06 :           strcpy ( vol_type, "FAT-16");  break;
                case 07 :           strcpy ( vol_type, "NTFS");  break;
                case 0x0B:          strcpy ( vol_type, "FAT-32");  break;
                default:    strcpy ( vol_type, "NOT-DECODED");  break;
               }

// Print out partition content
printf ("Partition %d: Type: %-12s Start: %-12d Size: %-12d\n", i, vol_type, part_entry[i].start_sect,
part_entry[i].size);
        }
printf("\n\nThe total number of valid partitions is: %d\n\n", (4 - not_exist));
        fclose(fp);
  return(0);
}
```

SAMPLE DISPLAY OUTPUT FOR ABOVE PROGRAM

$ partInfo Sample_1.dd

The expected result would be as follows:

Partition 0: Type: FAT-16 Start: 63 Size: 514017
Partition 1: Type: FAT-32 Start: 578340 Size: 1028160
Partition 2: Type: NTFS Start: 1606500 Size: 369495
Partition 3: Type: NOT-VALID Start: 0 Size: 0

Total number of valid partitions is: 3
