CS 365: Digital Forensics
Spring 2020
Assignment #6

Arun Dunna
adunna@cs.umass.edu

Revision 1.1 - April 10, 2020

Due: May 1, 11:55pm

Submission Instructions

1. This assignment is split into two portions that will each count as their own full assignment: programming and written response. We will be providing **minimal** support for the programming portion, as it is intended that you research as needed.
2. Your programming solutions for this assignment are to be written in **Python 3.6**.
3. Your solutions to written response portions of this assignment are to be typed and submitted to the Gradescope written assignment in PDF format with the following specifications: **12pt font, 1-inch margins, clearly labeled and appropriately sectioned solutions, full name included, student ID number included, submission date included.**
4. Your programming solutions are to be submitted to the corresponding Gradescope programming assignment. These should be submitted in a **ZIP file** containing your Python code files, in the following format:

```
submission.zip
├─ main.py
└─ ...
```

   This will match the format of the assignment code distributable. Make sure to include the Python code files you have created. Make sure to include **`main.py`** in your submission.

   Alternatively, you can upload each Python file individually to Gradescope instead of using a ZIP file.

Prerequisites

1. Ensure that you are enrolled in the course on Gradescope. The assignment submission will open a few days before the deadline. If you have not been automatically enrolled, you can use the following enrollment code: **9DZ53K**
2. Setup and test your own environment for executing **Python 3.6** code. For example, this can be in an IDE such as PyCharm, or in a terminal with the **`python3.6`** command.
3. Obtain the assignment distributable from the course website: **`asgn06distrib.zip`**

1 Programming

The assignment distributable contains a “runfile” **`main.py`** which you should use to interact with your program. Create your own Python code file(s) and import them into the **`main.py`** code using import statements. Then, make your program interface-able by adding arguments and control logic to **`main.py`**, such that **`main.py`** behaves identical to the tool you are modeling. If your program takes a file as input, which is passed in through the argument parser in **`main.py`**, you can assume that the file exists.

Therefore, **`main.py`** should **ONLY** be modified in the following ways:

1. Include your written Python file(s) through import(s).
2. Add arguments to the included Argument Parser to interface with your program.
3. Add control logic and function calls using the Argument Parser arguments.

1.1 Tool Implementation [100 points]

Pick a digital forensics tool from the following list, or find one you like and get it approved via a private Piazza post, and re-implement the functions and behavior of the tool as described in the list:

- **`dd`**: You must implement the operands **bs, count, if, of, seek, skip, conv**. You must implement the conversions **lcase, ucase, sparse** where case conversions are done only for ASCII, not Unicode, characters. You must handle multiplicative suffixes (ex. K, MB, M, etc.) as per **dd**’s specifications. Files can only be passed in via **if**, not stdin. All else should NOT be re-implemented. No print output is needed.
- **`fatcat`**: You must implement options **-i, -l, -L, -r, -R, -s, -2, -k**. Assume the filesystem always starts with the first byte of the input file. Only handle FAT16 - not other types. All else should NOT be re-implemented.
- **`exiftool`**: You must implement a subset of functions within exiftool, specifically parsing document properties in **.docx, .pptx, and .xlsx** (i.e. Microsoft Office XML) files. Despite its origins as an EXIF parser, exiftool is a generic file-format parsing utility and this option will not involve EXIF parsing. Mirror the output of **exiftool --ZIP:all --Main:all**, and implement the optional arguments **-s and -sort** which can be combined with display arguments **-json and -t**.

If you choose **dd**, then name your Python code file **dd.py**. If you choose **fatcat**, then name your Python code file **fatcat.py**. If you choose **exiftool**, then name your Python code file **exiftool.py**.

Your submission will be graded based on how well it performs the functions of the program. Note that IF your program does not run, or is not interface-able via **`main.py`**, **you will not receive any credit**. Therefore, you MUST ensure that you can run **`main.py`** in an identical manner to the given tool. For example, if `dd` is reimplemented, the following two commands should behave identically:

```
dd if=asgn06.pdf of=asgn06_snap.pdf bs=512 count=1
python3 main.py if=asgn06.pdf of=asgn06_snap.pdf bs=512 count=1
```

Likewise, for `fatcat`:

```
fatcat -i fat16fs.img
python3 main.py -i fat16fs.img
```

And for `exiftool`:

```
exiftool --ZIP:all --Main:all -s -sort -json Document.docx
python3 main.py -s -sort -json Document.docx
```

2 Written Response

The following questions should be answered in complete, readable sentences, and adhere to the relevant specifications in the submission instructions. They should be labeled according to their numbers.

2.1 NTFS and Data Recovery [25 points]

1. What are three of the primary differences between NTFS and FAT? What are three similarities?
2. In your own words, describe the structure of NTFS with respect to and using the following terms:
   (1) $Boot / boot sector
   (2) Files
   (3) Metafiles
   (4) MFT
   (5) File records and record headers
   (6) Attributes and standard attribute headers
   (7) Resident and non-resident
   (8) Data runs
3. Describe how NTFS keeps track of directories and files by using:
   (1) File record headers (Hint: flags 0x01 and 0x02)
   (2) INDEX_ROOTs
   (3) Index node headers
   (4) Index entries
4. If a directory entry describing a file is overwritten with null bytes in a FAT filesystem, how can we try to recover the file contents?

2.2 Malware [25 points]

1. List three types of malware we covered. For each type:
   (a) Describe how the type of malware works.
   (b) Describe some of the possible outcomes for a system infected with the type of malware (ex. ransomware causing unsuspecting person to lose access to important files).
   (c) Describe an example of a(n) (in)famous historical malware of the type, making sure to include the time period, target platform(s), how it worked, how it spread, why it was dangerous, and why it became well-known.
2. What are two advantages and two disadvantages of a given operating system requiring software to be signed before it can be installed? Assume that keys are only maintained for developers who register and pay an annual fee to the operating system developer.
3. Describe both static and dynamic malware analysis. Pick one of the two and detail three advantages and three disadvantages to that type of malware analysis.
4. List three types of malware defenses we covered. For each type:
   (a) Describe how the defense works.
   (b) Describe the methods (if any) researchers can use to circumvent the defense in analyzing the malware.
   (c) Describe the methods (if any) automated antivirus software can use to detect malware using the given malware defense.

5. Pick two malware identification methods and for each:

   (a) Describe how it works.

   (b) Detail the extent to which it can be automated.

2.3 Desktop and Mobile Forensics [25 points]

1. Describe three reasons why forensics tools can and do vary between desktop operating systems. Find a common desktop forensic tool that is only available on one desktop operating system due to technical limitation(s), and describe the limitation(s) and why they exist. If the tool was built for the operating system, it is not enough to say that the limitation is that the information does not exist on other operating systems; you must explain why, or if it exists on other systems then why it cannot be retrieved with this tool.
2. How does user authentication vary between desktop and mobile systems, and how does this impact forensic investigations?
3. Describe the role that cloud infrastructure plays in mobile forensics. How can cloud infrastructure impact mobile forensic investigations?
4. List and elaborate on three reasons why mobile devices may be more important than any other devices in a forensic investigation.
5. Find a Supreme Court case via `https://caselaw.findlaw.com/court/us-supreme-court` within the last 10 years in which mobile forensics was not applied but could have been, and give a brief synopsis (don't include the opinion —just the case details) of the case. Then, describe how mobile forensics could have been applied to aid either side (pick one) of the case in building their argument.

2.4 Biometrics [25 points]

1. Describe the general processes for:

   (1) Seeding a biometric system.

   (2) Authenticating a user in a biometric system.

   (3) Identifying a user in a biometric system.
2. List three physiological characteristics that are often targets for biometrics applications, and describe an application for each characteristic. Each application must be unique (ex. all three cannot be "user authentication").
3. Forensics often deals with the analysis of evidence after an incident has occurred. However, most biometrics applications are concerned with real-time usage. How can/do we bridge this gap to apply biometrics to forensics?
4. Describe two applications of biometrics to real-time crime detection.