Andrew Meyer
DFR project report: Weeks 1-2 (5/27/19 - 6/9/19)

What I have done:

- Ordered and received SSD and flash drives.
- Set up VMs for Windows 10 and Ubuntu 18.04 on the external SSD
- Installed Autopsy on both VMs
- Verified I can redirect the flash drives to the VM when needed
- Verified I can pass a filesystem image to the VM as read-only

What I am working on:

- Getting as familiar as I can with the FAT filesystem
- Coming up with ideas for test cases

I had a few thoughts about creating the test images that I think are worth discussing:

- When preparing each image we should start by completely writing over the partition or drive with zeros before writing the filesystem to it, to keep our tests as consistent as possible.
- We should try to keep each image as simple as possible, containing only the files necessary for create the chosen scenario. This way there can be no question about what scenarios cause a tool to fail, and it makes it easier for a person to manually check the filesystem.
- Is it worth making a separate set of cases for different implementations of the same filesystem i.e. Windows FAT vs. Linux FAT? Or is the important factor simply the state of the filesystem when we analyze it? What about FAT12 vs. FAT16 vs. FAT32?

This is my current understanding of how the four core features apply to FAT:
DFR-CR-1: The tool should identify all directory entries that have been deleted (first byte is 0xe5) but not overwritten.
DFR-CR-2: For each deleted directory entry it identifies, the tool should attempt to reconstruct the associated file.
DFR-CR-3: The tool should recover all sectors indicated by the directory entry, besides those which have been reallocated to another file.
DFR-CR-4: The tool should only recover sectors which were a part of the deleted file (i.e. if the file was fragmented, unallocated sectors in between the fragments should not be recovered).

Possible FAT test cases:

1. File A is written to contiguous clusters and deleted.
2. File A is written to non-contiguous clusters (A is fragmented) and deleted.
3. File A is written to non-contiguous clusters and deleted, and clusters between the fragments are de-allocated.
4. File A is written to contiguous clusters and deleted, and file B is written over one of those clusters.
    i. B is written over A's first cluster.
    ii. B is written over the middle of A such that deleted data from A is present before and after B.
    iii. Data from B does not precisely fit the cluster size, so one cluster will contain the end of B followed by deleted data from A.

5. File A is written to contiguous clusters and deleted, and file B is written over one of those clusters and deleted.
    i. B is written over A's first cluster.
    ii. B is written over the middle of A such that deleted data from A is present before and after B.
    iii. Data from B does not precisely fit the cluster size, so one cluster will contain the end of B followed by deleted data from A.
6. File A is deleted and its directory entry is overwritten by file B's.

I expect cases 3 and 5 are the most likely to fail, specifically on DFR-CR-4, because it is easy for unrelated or reused clusters to blend in with the Deleted Block Pool.