Digital Forensics Masterclass Labsheet

File System Forensics

Contents

1 Lab Overview

The learning objective for this lab is for participants to gain practical experience in digital forensics specifically file system analysis. During the lab you will gain experience in the following tools, environments and tasks:

- Kali Linux[1]
- The Sleuth Kit (TSK)[2] file system analysis tool kit.
- Forensic image creation.
- Identifying Attributes about file systems (data uni size file type).
- Deleted file recover procedure.
- Searching for evidence on a file system.
- Event timeline creation?????

As an *optional* bit of fun for the lab, we will be adding a capture the flag (CTF) style element. For each task there is a set goal, that may be finding a word or phrase in unallocated space, the hash of a recovered file etc. After each task is completed please enter the given evidence as an argument to the `flagfound` script as follows:

```
$  ./flagfound "YOUR EVIDENCE HERE"
```

2 Removable Media Resource Credits

The USB resources we will be using in today's lab are from the National Institute of Standards and Technology (NIST) as part of their Computer Forensic Reference Data Sets (CFReDS)[3]. We will be focusing on a subsection of this case using "Removable Media #2", in order to investigate material covered in the lecture. The scenario section below will set the scene.

If you are interested in exploring this scenario more the whole case can be fond here:

`https://www.cfreds.nist.gov/data_leakage_case/data-leakage-case.html`

[1] https://www.kali.org/
[2] https://www.sleuthkit.org/
[3] https://www.cfreds.nist.gov/

3 Environment Setup

Installing Kali Linux in Virtual Box

Before your digital investigation can begin you are required to create a lab environment, for this we will be using a virtual machine (VM) instance of Kali Linux. Your user accounts should have been given access to virtual box in the university lab (EM2.50). We are using a virtual machine to no only give you complete control over the operating system, but so this lab can be carried out safely. Below we will show you how to create a Kali Linux VM.

1. Open a terminal window and enter the following command to open Virtual Box:
   `$ virtualbox &`
2. Return to the terminal window and follow `LoadingKaliVM.pdf` instructions, created by Mike Just.
3. Start your Kali VM

4 Kali Linux Login Information

username: `root` password: `toor`

4.1 Note About Root

In Kali Linux you are logged in as the root user of the system, therefore the operating system assumes you know what you are doing. Double check the commands you are entering especially if they are related to `rm`.

4.2 Create a Case Directory for Investigation

Within your home directory create the a `cases` directory and inside that create a directory called `001` to represent the case number you are working on. Make this your current working directory in a terminal window.

5 Task 1: Create a Forensic Image

There should be USB flash drives available for this lab, with the removable media evidence on them. In this lab we will be working with .dd images also known as raw images. There are many tools available to create the a forensic image with Kali Linux comes installed with both GUI and command line versions. It is up to you which you use however I would recommend `Guymager`.

NOTE: *Sadly we were not able to attain writeblockers for this lab, please remember that writeblockers are important for prevention of unwanted meta data changes to media by the investigators operating system. Writeblockers are used in industry when creating forensic images of a system.*

Insert USB Stick into computer and confirm it has been recognised by the Kali Linux VM, you can check this in either in files that a usb device is visable in the left hand pane or by running `df -h` in the command line and checking for a 4GB device located in `/media/root/`.

Open a terminal window (we will use this later anyway) and enter: `guymager &` the following GUI should appear:

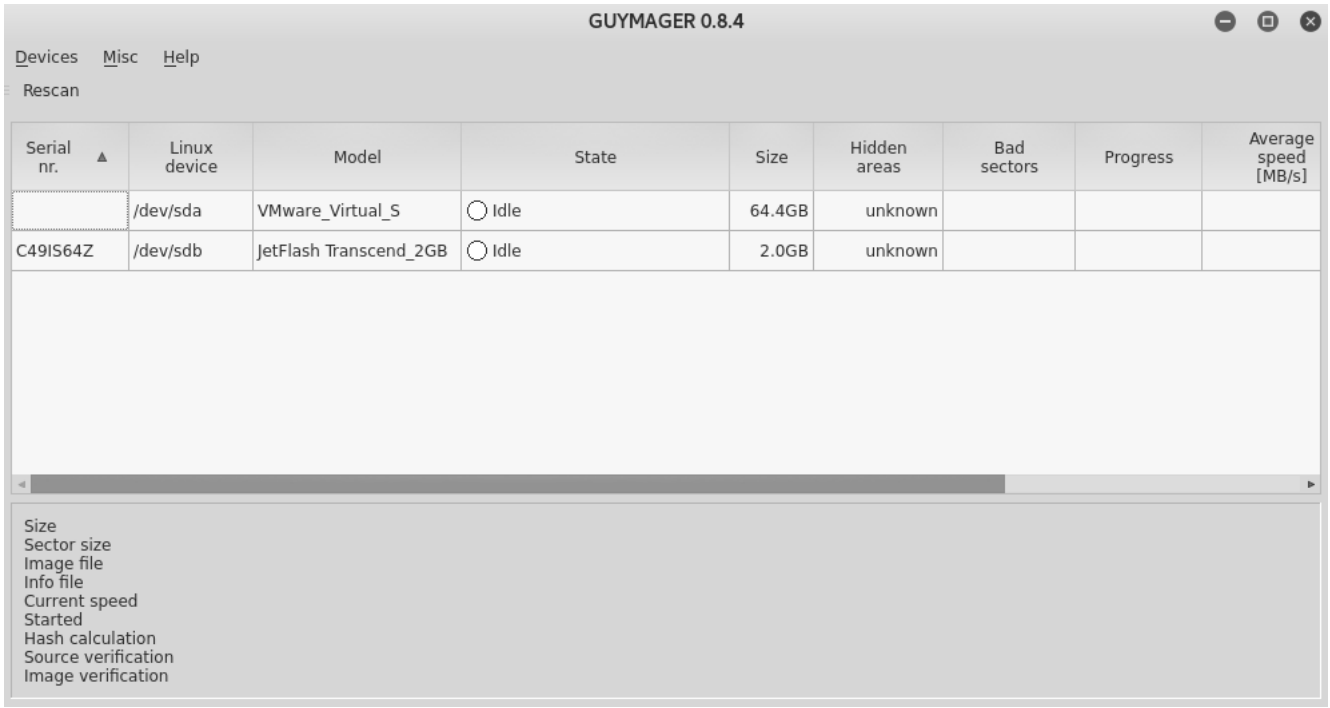

Figure 1: *guymager gui window example*

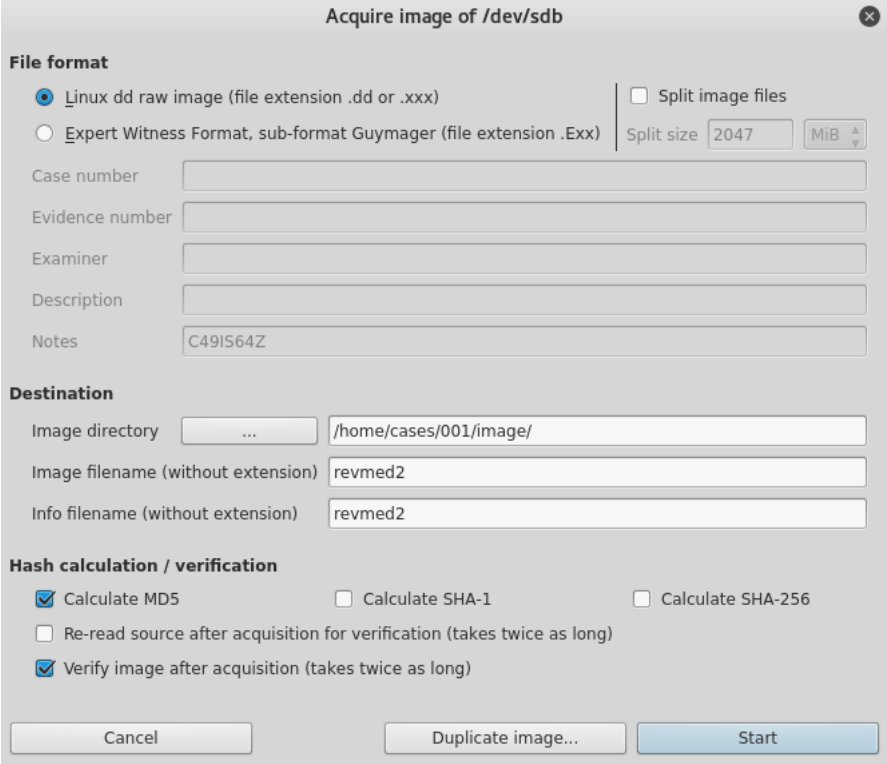

Figure 2: *guymager acquisition window example*

If the 4GB usb drive does not show up press `f5` to rescan devices. From here you can select the drive you wish to create an image of, right click the 4GB drive and select the `Aquire image` option. Fill out the guy form making sure you have the following options then click start:

- Linux dd raw image
- Uncheck split image
- Destination is set to `/home/cases/001/image/`
- Image filename set to `revmed2`
- Info filename set to `revmed2`
- Check calculate md5 hash
- Check verify image after acquisition

Once you have clicked start you shall be returned to the main guymager window and a progress bar should be displayed. once the progress bar states "Finished - Verified & ok" your forensic image has been created. To confirm list the contents of `/home/cases/001/image/` in a terminal window, you should find the following files:

- `revmed2.dd` : the image file we can now perform analysis on.
- `revmed2.info` : acquisition info such as cryptographic hash.

Verify The Media Hash

in order to confirm that our image is the same as the contents of the USB we must generate a hash of the .dd file and compare it to the hash listed in the .info file. in a terminal window run the following: `md5sum image/revmed2.dd` the output should match the hash listed in the .info file you just created.

Remove the USB from your machine and return it to the evidence box.

Optional: *enter the hash into the* `findflag` *program to receive your first flag.*

6 Task 2: File System Information Gathering using TSK

Now that we have a forensic image we can start to analyse the evidence, using The Sleuth Kit (TSK).

Commands we will be using to gather information on the system

- **`mmls`**:

  `mmls` displays the layout of the partitions in a volume system, which include partition tables and disk labels.

- **`fsstat`**:

  `fsstat` displays the details associated with a file system. (requires partition starting point in sectors)

Find the following Information:

Using the tools above and files you currently have (.dd, .info) find the following information: (write it down you will need to reference it later)

1. what is the cluster size of the file system?
2. what is the sector size of the file system?
3. Given that a cluster contains consecutive sectors, how many sectors are in one cluster?
4. What partition has content data in it?
5. What is the staring point of the File System partition (in sectors)?
6. Can you confirm the file system type is fat32 using a TSK tool or other information you have gathered (.info) file?

7 Task 3: Searching Unallocated Space for Evidence

Commands we will be using to gather information on the system

- **blkls**:

  `blkls` opens the named image(s) and copies file system data units (blocks). By default, `blkls` copies the contents of unallocated data blocks.

- **fsstat**:

  `fsstat` displays the details associated with a file system. (requires partition starting point in sectors)

8 Task 4: Deleted File Recovery using Meta Data Pointers

Your another member of your digital investigation team has mentioned there is something suspicious with a file called `progress/my_friends.svg` that was removed from the file image, they have asked to to check if the deleted file is recoverable and verify the data is of format `.svg` then compare the recovered data to his found SHA1 hash.

Co-Workers SHA1 hash:

`95175c35be9d74ad3e271550e358e634154ba105 progress/my_friends.svg`

Commands used in this task

- **fsstat**:

  `fsstat` Displays the details associated with a file system. (requires partition starting point in sectors).

- **fls**:

  `fls` List file and directory names in a disk image, including its associated *inode* number.

- **istat**:

  `istat` displays the uid, gid, mode, size, link number, modified, accessed, changed times, and all the *disk units* a structure has allocated.

- **blkstat**: displays the allocation status of the given data unit.

- **blkls**:

  `blkls` opens the named image(s) and copies file system data units (blocks). By default, blkls copies the contents of unallocated data blocks.

- **sha1sum**: `sha1sum` - compute and check SHA1 message digest

- **`icat`**:

  icat Outputs the content of a file based on its *inode* number according to the details in its meta data pointer.

- **`file`**: file - determine file type

Talk Walkthrough

Task 4.1: List the names and inode info of all deleted files on the image:

We are looking for a deleted file and have been given a name to find within this .dd image. We know that data in file systems have various pointers for metadata, one of which is a file name pointer. We can use TSK tools that proceed with an `f` in this instance we want to see all files that were deleted on the data partition, `fls` will let us do this.

```
# fls -rd -o 128 [path-to-image]
```

4.1 Questions:

1. What is the *iNode Address* for this `my_friends.svg`?

Task 4.2: Based on the inode for my_friends.svg extract stats on it:

Using this inode (meta data) pointer, we can gather stats on the data units in question using `istat`.

```
# istat -o 128 [path-to-image]  1651335
```

4.2 Questions:

1. What is the file size of the Image?
2. What sector does the data unit start at?
3. what sector does the data unit end at?

Task 4.3: Get clusters sectors are in:

```
# blkstat -o 128 [path-to-image]  111408
# blkstat -o 128 [path-to-image]  111522
```

4.3 Questions:

1. What is difference between start and end clusters?

Task 4.4: Recover those sectors from unallocated space:

Given we now know the start and end sectors and have verified that their clusters are not overwritten with other data, we can begin the recovery process. `blkls` will extract all unallocated data units between a given start and end point and write it to a new file (if no points are given it will gather all unallocated space).

```
# blkls -o 128 [path-to-image] > output/recovered-file.blkls
```

Task 4.4: Calculate hash for recoveredfile.blkls:

We have successfully recovered the requested file now verify that its hash matches the one our co-worker found as follows: `# sha1sum output/recovered-file.blkls`

Verify it is a `.svg` file with the file command:

Task 4.5: Check the file type of the recovered data units:

```
# file output/recoved-file.blkls
```

4.5 Questions:

1. What is actual type of the file?
2. Do you know why the file extension contradicts the actual type?