|  |  |
| --- | --- |
| **Digital Forensics**  Diploma in CSF/IT  Year 2/3 (2020/21) Semester 4/6 | Week 4-17 |
|  |
| **Write-up on Open Source Forensic Tools - Individual** | |

|  |  |  |
| --- | --- | --- |
| Student No. | Student Name | Tutorial Group |
| S10198161 | Tan Jia Shun | P02 |

## Name of Open Source Tool(s):

## Tool name: Magic Rescue

Description: Magic Rescue is an application to read a devices and scan for file types in that device and calls an external program to extract them. This application looks for “magic bytes” in the file contents to look for the file, therefore, it can be used both as an undelete utility and for recovering of data from a corrupted drive or partition.

OS requirement: Linux and windows (if Ubuntu LTS is installed)

File system supported: any

Installation:

1. Run “sudo apt-get install magicrescue” in the terminal.

Features:

* Look for a file using file type
* Recover deleted file that is not overwritten
* Recover file from corrupted drive or partition

Usage:

* magicrescue [ options ] [device]
* Option list:
  + “-b” is used to set the blocksize
  + “-d” is used to set output directory
  + “-r” is used to specify recipe(file type)
    - There is a few pre-make recipe
    - We can create own customed recipe too
  + “M” is used to set output mode

Link: <https://github.com/jbj/magicrescue>

## Tool Name: AVML

Description: AVML is a volatile memory acquisition tool that allow user to acquire system memory without knowing the target OS distribution or kernel a priori which is better for new user compare to LIME, another well-known volatile memory acquisition tool.

OS requirement: Linux

OS supported: Linux

Installation:

1. Go to <https://github.com/microsoft/avml/releases> and download AVML  
   
2. Open terminal and change directory to Downloads using “cd Downloads”
3. Change the permission of the file using “chmod 755 avml” so we can run the application

Features:

* Save recorded images to external locations via Azure Blob Store or HTTP PUT
* Automatic Retry with exponential backoff for uploading to Azure Blob Store
* Able to use Snappy page level compression.
* Able to use output using LiME format (when not using compression).

Usage:

* avml [FLAGS] [OPTIONS] <filename>
* Flags:
  + “--compress” is used to enable compression using snappy
  + “--delete” is used to delete dump file after upload
  + “--help” is used to print help information
  + “--version” is used to show software version
* Options:
  + “--sas\_block\_size <sas\_block\_size>” is used to specify maximum block size in MiB
  + “--sas\_url <sas\_url>” is used to declare upload link to Azure Blob Store upon acquisition
  + --url <url>” is used to declare upload link to HTTP PUT acquisition

Link: <https://github.com/microsoft/avml>

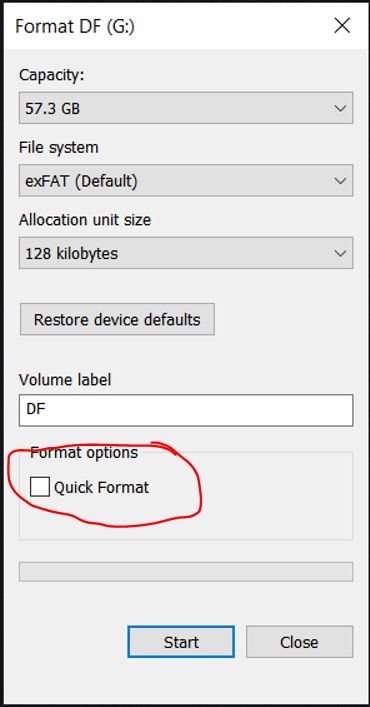
## Description of demonstration:

## Case scenario

For this demonstration case, a spy from another company tried to steal data from the current company. You are tasked to look for evidence file from the spy device and USB drive.

## Evidence files

The evidence file will be created by myself. First, we will need a USB thumb drive which is new or full formatted. To full format a USB thumb drive, we need to uncheck the quick format option.



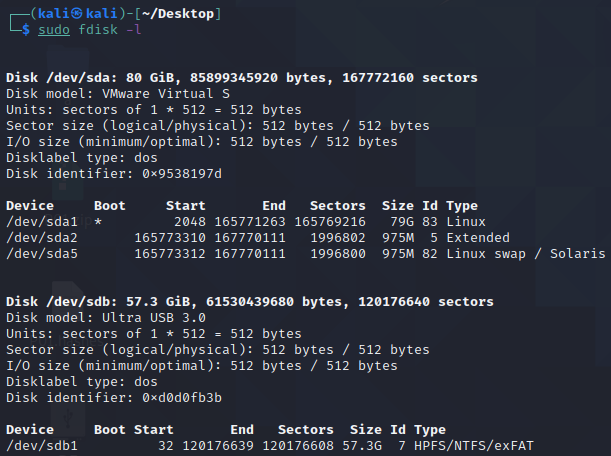
Next, we will use Kali Linux to copy a few files, including the two evidence file (evidence 1.doc and evidence 2.doc) to the formatted USB thumb drive. When done, we will delete the one of the evidence files (evidence 2.doc). With this, the Kali Linux system will have memory of file transferred to the USB thumb drive and creation of the evidence file is completed.

## Tool 1: Magic Rescue

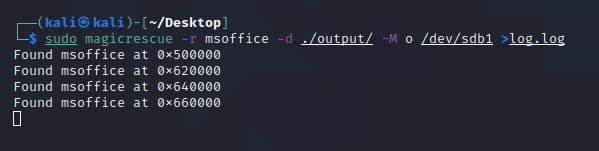
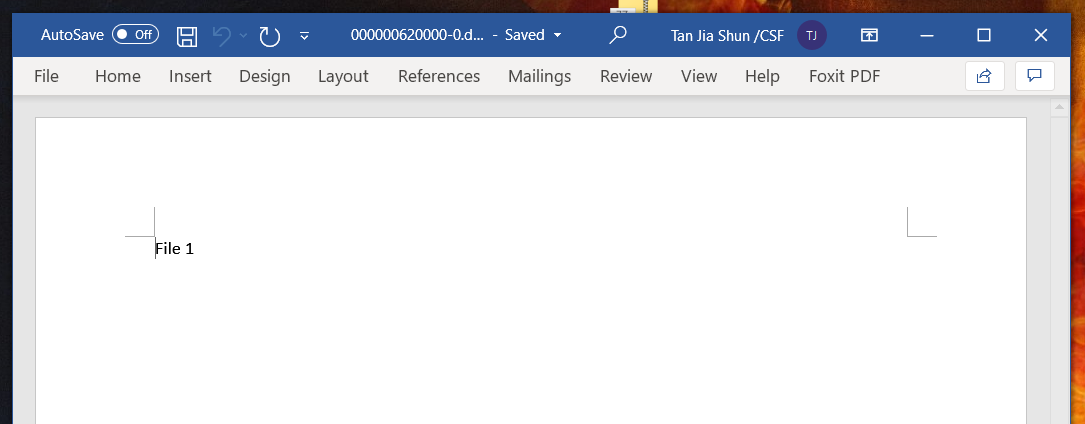
### Requirements:

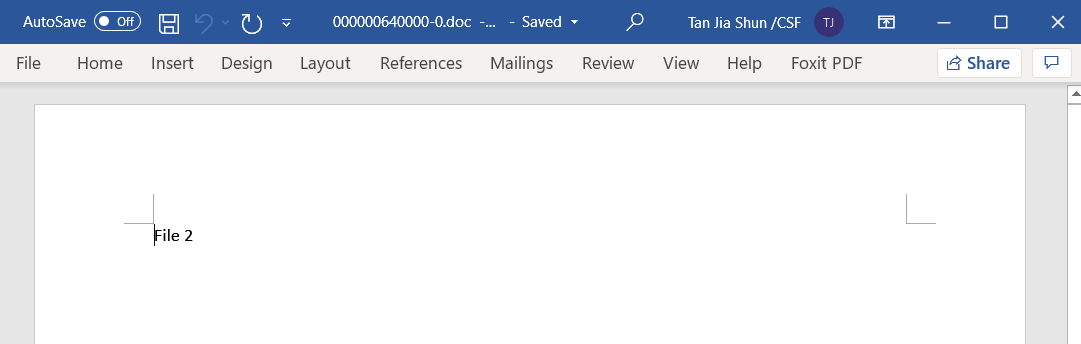
* A Linux system (For this demonstration, I will be using Kali Linux)
* Magic Rescue
* Git
* Evidence device/image (Can be a thumb drive or hard drive or mobile phone)

### Setup:

1. Look for your device name by running “sudo fdisk -l”  
   

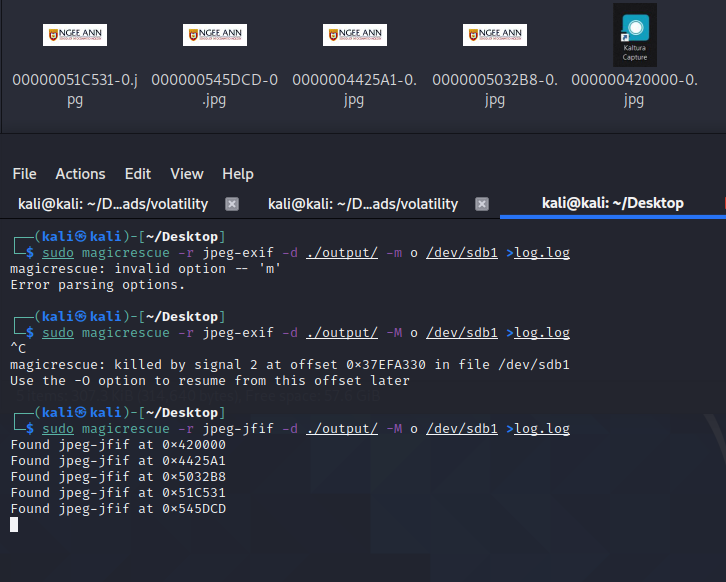
### Obtaining DOC file:

1. Run “sudo magicrescue -r jpeg-jfif -d ./output/ -M o /dev/sdb1 >log.log”
2. When it found a doc file, it will copy the file to output folder and log in log.log  
   
3. As shown, it found 4 doc file. Let open all 4 file and see can we found the 2 evidence file   
   

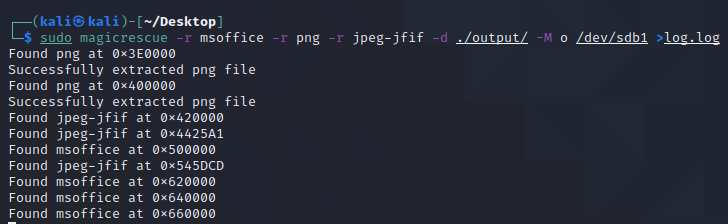


1. It manages to find both the file, even the deleted file.

### Obtaining JPEG file:

1. Run “sudo magicrescue -r jpeg-jfif -d ./output/ -M o /dev/sdb1 >log.log”
2. when it found a jpeg file, it will copy the file to output folder and log in log.log  
   

### Obtaining multiple file type:

1. By adding -r [recipe], it will scan the file type you looking for. So we run “sudo magicrescue -r jpeg-jfif -r png -r msoffice -d ./output/ -M o /dev/sdb1 >log.log”  
   

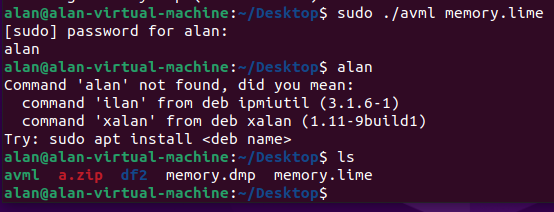
## Tool 2: AWML (recovery memory dump)

### Requirements:

* A Linux system (For this demonstration, I will be using Kali Linux)
* AVML
* Git
* Volatility

### Obtaining memory dump:

1. Open terminal
2. Change directory to where you downloaded SVML (for me it will be Downloads) using “cd Downloads”
3. Run “sudo ./avml memory.dmp”
4. Run “ls”. If you see memory.lime file, that mean we manage to get our memory dump. The dump file size should be the size as the size of the RAM of the system.



### Obtaining compress memory dump:

AVML support compression of memory dump file. To do so, you add “--compress” to the command; 1. “sudo ./avml memory.dmp --compress”

### Upload memory dump:

AVML support Azure VM, AWS S3 and GCP cloud storage upload. This allows digital forensic investigator to immediately upload the evidence file to the cloud, incase there is any physical damage to the device.

For Azure VM:

1. Run “avml --sas\_url [SAS\_URL] --delete memory.lime”

For AWS S3 and GCP cloud storage:

1. Run “avml --put [URL] --delete memory.lime”

### Reading the memory dump (extra):

To read the memory dump, we will need another application, for this case we will be using Volatility, with it, will can see CLS command run, application open, netstat and many more

1. Download volatility using “sudo apt-get install -y volatility”
2. Run “sudo apt install dwarfdump” to install dwarfdump which is a library that volatility needed
3. Run “cd ./volatility/tools/linux”
4. Run “make” to create a profile for Volatility to specific which Linux system you are using
5. Change directory to where you memory.dmp is located (for me it will be Downloads) using “cd Downloads”
6. Now we need to install the Volatility profile. To do so, we run “sudo zip [DISTRO\_KERNEL].zip ./tools/linux/module.dwarf /boot/System.map-[KERNEL VERSION]”. (For my case, it will be “sudo zip kali\_5.9.0-kali5-amd64.zip ./volatility/tools/linux/module.dwarf /boot/System.map-5.9.0-kali5-amd64”)  
   You can get the Kernel information by running “uname -a”
7. Move the profile file to volatility/plugins/overlays/linux using “mv kali\_5.9.0-kali5-amd64.zip ./volatility/plugins/overlays/linux”
8. Now let go to volatility using “cd ./volatility/”
9. And run it using “python vol.py -f ../memory.dmp –profile= kali\_5.9.0-kali5-amd64 [Plugin]”