Automated Forensic Analysis using Digital Forensic Framework

BLOSSOM
Manchester Metropolitan University
(Funded by Higher Education Academy)
l.han@mmu.ac.uk

1. Learning Objectives

This lab aims to use Digital Forensic Framework to conduct automated forensic analysis.

2. Preparation

1) Under Linux environment

2) Some files that you will need from /home/user/BlossomFiles/AutomatedAnalysisExtCapabilities:

- 'ntfs1-gen2.aff'
- 'DFF_1.2.0-x86.deb'

3) Some documents that you may need to refer to:

- 'Virtual-MachineGuide.pdf'
- 'Linux-Guide.pdf'
- 'BLOSSOM-UserGuide.pdf'

3. Tasks

Setup & Installation:

- Start a single virtual machine as you have done with previous exercises (see Virtual Machine Guide)

  # kvm -cdrom /var/tmp/BlossomFiles/blossom-0.98.iso -m 512 -net nic,macaddr=52:54:00:12:34:57 -net vde -name node-one

- Use the following command to install Digital Forensics Framework (DFF):

  # dpkg -i DFF_1.2.0-x86.deb

Task 1: DFF Basics & Automation

1.1 In this section, we will look at graphical investigation environments since it is significantly more fitting for what we will be looking at. The advantages of using a graphical environment consist of such factors as integrated case management, simple keyword searching functionality and a more logical structure for viewing files.

The tool we will be using is called Digital Forensics Framework (DFF), which provides multiple different levels of functionality when managing a case during an investigation. Start running DFF's graphical interface:

#dff -g

1.2 After DFF has started, we can see the logical structure of the layout and some of the tools available for helping us analyse evidence. We are going to look at a test file called 'ntfs1-gen2.aff' which is a file of AFF format containing a Windows partition from an NTFS drive.

In order to open up the file within the investigation environment, first select the 'File' menu, and then navigate to 'Open evidence file(s)'. Here we will be presented with a prompt to select a file or directory, select 'AFF Format' and then click on the '+' symbol, and proceed to navigate to the 'ntfs1-gen2.aff' file. After this, select OK and the drive will be loaded in to the 'Logical files' folder within the framework.

1.3 With the drive mounted within DFF, we can now proceed to analyse it, but first we need some sort of logical structure for the file system. DFF can perform this automatically by recognising key files and entries within a drive, such as the MFT within this NTFS drive. By double clicking on the file 'ntfs1-gen2.aff' within the framework, it will prompt us asking whether or not we would like to apply the module 'ntfs' to the file, select OK and the drive will become structured as if it was being viewed with a file browser.

Most of the files we can view initially are system files, such as '$MFT' and '$Bitmap', but there are a few PDF, TXT and JPG files stored within the drive that can be viewed with the viewers contained within DFF. Take a look at the files 'NIST_logo.jpg' and 'logfile1.txt' from the directory 'RAW' using DFF's viewing capabilities, and if the file is not viewable from within DFF, such as the PDF files, we can right click on the file and select 'Extract' from the drop-down menu, and then view it with an external viewer.

1.4 The file system being used here is quite small as it is for demonstration purposes only, but if the file system was larger, it would be more appropriate to search for information using a keyword search tool.

In order to search for a keyword using DFF select the button on the far right that looks like a pair of binoculars, this will open up the search engine which allows us to input a search query in order to easily locate

files. For example, searching for 'jpg' should display the three 'NIST_logo.jpg' files in the system; however, more complex GREP searches can be used in order to be more specific when searching for files.

1.5 Another important feature of DFF is the support for file carving using simple header/footer signature carving, that works by storing details relating to the start and end of specific files, such as JPEG files. The header for a JPEG file is usually 'FF D8' , and the footer is usually 'FF D9', so by performing a file search for these, we can potentially recover files that have been deleted or corrupted.

DFF provides sets of predefined patterns for carving, as well as allowing the user to define their own by entering the header and footer of a file type. We will look at using the predefined JPG carver to recover some image files which are not visible initially.

First of all, right click on the NTFS drive and select 'Open with' from the dropdown menu, which will show the list of all the modules that can be used within DFF. Scroll down to 'Search' and select 'carvergui'. This will bring up an interface , listing the various predefined patterns that are available. Select images and choose 'jpg', then begin the search.

Following this, a significant number of 'headers' will be found, relating to jpg files that were stored within the other areas of the drive. In order to view them, go back to the 'Browser' and the carved files will be stored within the directory 'carved' within the framework. View the images within the picture viewer.

Question: Try using some of the other carving presets, how many other types of files are discovered through this method?

1.6 There are many other tools & modules contained within DFF that provide an edge when performing a digital investigation, such as 'metaexif' under the 'Metadata' modules, which provides metadata for certain files, and 'hash' under the 'Hash' modules, which provides a hash for a certain file. Experiment with these tools and understand their purpose within the forensic environment.

Task 2: Timelines

2.1 Timelines are a extremely valuable forensic analysis tool which allows for an investigator to develop context with regards to the events being observed. The most common way of establishing a timeline of events is with the file metadata, providing information regarding creation, modification and access times of a specific files; however, with more recent timeline advancements, we can pull in additional information from multiple data sources, allowing us to view other events such as a user navigating to the Start menu to launch some sort of program.

Another reason why timelines are important, is that they help reinforce or verify the authenticity of creation, access and modification times, as the metadata information of a file can be quite easily modified, but if there was a registry key that contained information relating to the modification time of the a file, that could be used to verify the modification time.

DFF has a built in timeline function, which we can use to develop a more in depth understanding of when various events occurred on the file system that we are analysing. Right click on the NTFS drive and select 'Statistics' and then 'timeline'.

2.2 Now that we have created a timeline, we can see that all the events are broken down in to three separate time frames. We can then further analyse the time frames of the timeline by selecting the individual sections by clicking on the graph and dragging a selection box over each separate part, selecting the 'Navigation' menu on the right hand side, and then clicking on 'Export'. All of the files that relate to that specific time frame will now be listed in the browser under the directory 'timeline'.

There is a third option called 'Display' on the right hand side of the timeline, which allows for various colours to be set for a variety of different events such as file accessed time and file altered time. Viewing each result independently allows us to figure out what occurred at what time.

Question: With each section now split in to three different time frames, and knowing how to view which events occurred when, discover which file is registered as the most recent in the timeline, and also figure out what event occurred to that file.