# Activity – Data Recovery by File Carving

## Introduction

This activity introduces various tools to carve data from forensic images. The tools are close in capabilities, sharing many common features, however it is important to understand that results will vary. Each tool has its own unique detection processes, some more capable than others, and it is not uncommon for one tool to find a file that another tool will not.

## Requirements

* A Kali Virtual Machine (VM)
* Four (4) images provided by your facilitator:
  + Mod6-1.dd
  + Mod6-2.dd
  + Mod6-3.dd
  + usbdrive20211021.dd

## Task 1: Copy images to your Kali VM

Copy the four (4) image files provided by your facilitator to your Kali VM. Filename and associated hashes are provided below in Table 1.

|  |  |
| --- | --- |
| Filename | Hash (MD5) |
| Mod6-1.dd | 197b1d7d19fcf214db276dcfbbc233e3 |
| Mod6-2.dd | b47ee7fcd8338060b22ca1b682d37c8f |
| Mod6-3.dd | 12842b4388337f07a7776204ca61620e |
| usbdrive20211021.dd | 76763e3ce29bf364e77ef1f5aa39b20c |

Table - Artefact Filenames and Associated Hashes

Open a terminal window and navigate to the directory where you copied the image files. You can verify the file hashes by typing the following command:

md5sum \*.dd

The output from md5sum should match the hashes listed in Table 1. If there is a hash mismatch, apply your problem-solving skills and determine why this occurred and how you can correct the issue.

## Task 2: File Carving with Foremost

Foremost is a command line program used to recover files based on their headers, footers, and internal data structures. Foremost can work on image files, such as those generated by dd. The headers and footers can be specified in a configuration file, or as command line switches to specify built-in file types.

You can extend the file types that Foremost will carve by modifying the configuration file at /etc/foremost.conf. This configuration file is used to add file types in addition to the built-in file types that Foremost can process.

The structure of the configuration file makes adding additional file types straightforward. Opening the configuration file in your favourite editor, you would add line entries that include details of the file type you want to carve.

Below we see a line entry that configures the GIF file type for carving. This file type is included by default and does not really need to be added. The structure of the line is extension, case sensitivity, max file size, header (bytes) and footer (bytes).

gif y 155000000 \x47\x49\x46\x38\x37\x61 \x00\x3b

Foremost will perform a search for a byte sequence in your source data that matches the file type header defined in the configuration file. If a match is found, it will then look for the footer. If both header and footer are found, the data between these two ranges, including the header and footer, are saved to disk with the associated file extension. If the footer is not found, the max file size is used to calculate how much data to carve. Case sensitivity applies to the header and footer.

You can view the Foremost command line arguments by running the foremost -h command:

$ **foremost -h**

foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus.

$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w-d] [-t <type>] [-s <blocks>] [-k <size>]

[-b <size>] [-c <file>] [-o <dir>] [-i <file]

-V - display copyright information and exit

-t - specify file type. (-t jpeg,pdf ...)

-d - turn on indirect block detection (for UNIX file-systems)

-i - specify input file (default is stdin)

-a - Write all headers, perform no error detection (corrupted files)

-w - Only write the audit file, do not write any detected files to the disk

-o - set output directory (defaults to output)

-c - set configuration file to use (defaults to foremost.conf)

-q - enables quick mode. Search are performed on 512 byte boundaries.

-Q - enables quiet mode. Suppress output messages.

-v - verbose mode. Logs all messages to screen

To run Foremost against the first image file, execute the following command:

foremost -v -t all -i Mod6-1.dd -o foremost1

Foremost will start data carving the Mod6-1.dd image file. A new directory will be created in your current working directory called foremost1, unless you specified a different directory name.

The output from the data carving is shown below. With a default Foremost configuration, having no additional file types, the result is two (2) files, a .wmv and .wav file, carved from this image. Foremost provides some insight in to where these files were found inside our image, and the file size of each.

$ **foremost -v -t all -i Mod6-1.dd -o foremost1**

Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus

Audit File

Foremost started at Thu Aug 29 20:48:28 2024

Invocation: foremost -v -t all -i Mod6-1.dd -o foremost1

Output directory: /home/kali/Desktop/datarec/foremost1

Configuration file: /etc/foremost.conf

Processing: Mod6-1.dd

|------------------------------------------------------------------

File: Mod6-1.dd

Start: Thu Aug 29 20:48:28 2024

Length: 34 MB (36465664 bytes)

Num Name (bs=512) Size File Offset Comment

0: 00069122.wmv 1 MB 35390464

1: 00022057.wav 4 MB 11293184

\*|

Finish: Thu Aug 29 20:48:29 2024

2 FILES EXTRACTED

wmv:= 1

rif:= 1

------------------------------------------------------------------

Foremost finished at Thu Aug 29 20:48:29 2024

Viewing the created output folder using tree shows the files that have been carved and an audit.txt file that contains a capture of the execution output.

$ **tree foremost1**

foremost1

├── audit.txt

├── wav

│   └── 00022057.wav

└── wmv

└── 00069122.wmv

3 directories, 3 files

Use Foremost to perform data carving on the Mod6-2.dd and Mod6-3.dd image files. You will need to specify a new folder for each execution as Foremost will not allow an existing folder to be used. The reason for this is to maintain the integrity of carved data and avoid contamination of your results by overwriting and mixing artefacts from different image files.

The data carved from Mod6-2.dd is shown below.

$ **tree foremost2**

foremost2

├── audit.txt

├── docx

│   ├── 00061210.docx

│   └── 00071219.docx

├── jpg

│   └── 00081300.jpg

├── pdf

│   ├── 00010000.pdf

│   └── 00026204.pdf

├── png

│   └── 00081364.png

├── pptx

│   └── 00081227.pptx

└── xlsx

├── 00041137.xlsx

└── 00051183.xlsx

7 directories, 10 files

And finally, the data from the Mod6-3.dd image is below.

$ **tree foremost3**

foremost3

├── audit.txt

├── gif

│   └── 00126243.gif

├── jpg

│   └── 00056902.jpg

└── png

└── 00010000.png

4 directories, 4 files

Carving data is just the first step in your analysis, and you will need to perform further analysis on the files recovered. The analysis would depend on the scope of your investigation, but it is suggested to always check that the extracted files are in fact valid, and the file matches its detected file extension.

The data carved from Mod6-2.dd contains some image files and Word documents. Verifying the images are valid can be performed with any image viewer. The Word documents should be verified with caution as they may contain malicious code.

## Task 3: File Carving with Scalpel

Scalpel is a tool developed to improve on the performance of Foremost. It claims to be faster and less memory exhaustive. Scalpel and Foremost are similar, with one major difference being Scalpel does not have pre-defined default file types. Any file types you want to carve with Scalpel must be defined in the /etc/scalpel/scalpel.conf configuration file.

Editing the Scalpel configuration file is the same process as Foremost. Both applications share a similar format for defining new file types. In /etc/scalpel/scalpel.conf there are commented out pre-defined file types that can be enabled by removing the # at the start of the type definitions. Note that Scalpel has a hard-coded file type limit of 100 so you can not add every file type known to mankind.

For this task you need to replace the existing /etc/scalpel/scalpel.conf file with an updated version available at the Scalpel GitHub repository. Failing to perform this step will result in no detections.

From the command line, run the following multi-line command to update your Scalpel configuration:

sudo curl \

--output /etc/scalpel/scalpel.conf \

https://raw.githubusercontent.com/sleuthkit/scalpel/master/scalpel.conf

This command will execute curl as root and download the latest scalpel.conf from Github. The downloaded configuration file will be saved in the /etc/scalpel folder, hence the requirement for root.

You can view the Scalpel command line arguments by running the scalpel -h command:

$ **scalpel -h**

Scalpel version 1.60

Written by Golden G. Richard III, based on Foremost 0.69.

Carves files from a disk image based on file headers and footers.

Usage: scalpel [-b] [-c <config file>] [-d] [-h|V] [-i <file>]

[-m blocksize] [-n] [-o <outputdir>] [-O num] [-q clustersize]

[-r] [-s num] [-t <blockmap file>] [-u] [-v]

<imgfile> [<imgfile>] ...

-b Carve files even if defined footers aren't discovered within

maximum carve size for file type [foremost 0.69 compat mode].

-c Choose configuration file.

-d Generate header/footer database; will bypass certain optimizations

and discover all footers, so performance suffers. Doesn't affect

the set of files carved. \*\*EXPERIMENTAL\*\*

-h Print this help message and exit.

-i Read names of disk images from specified file.

-m Generate/update carve coverage blockmap file. The first 32bit

unsigned int in the file identifies the block size. Thereafter

each 32bit unsigned int entry in the blockmap file corresponds

to one block in the image file. Each entry counts how many

carved files contain this block. Requires more memory and

disk. \*\*EXPERIMENTAL\*\*

-n Don't add extensions to extracted files.

-o Set output directory for carved files.

-O Don't organize carved files by type. Default is to organize carved files

into subdirectories.

-p Perform image file preview; audit log indicates which files

would have been carved, but no files are actually carved.

-q Carve only when header is cluster-aligned.

-r Find only first of overlapping headers/footers [foremost 0.69 compat mode].

-s Skip n bytes in each disk image before carving.

-t Set directory for coverage blockmap. \*\*EXPERIMENTAL\*\*

-u Use carve coverage blockmap when carving. Carve only sections

of the image whose entries in the blockmap are 0. These areas

are treated as contiguous regions. \*\*EXPERIMENTAL\*\*

-V Print copyright information and exit.

-v Verbose mode.

To run Scalpel against our first image file, execute the following command:

scalpel -o scalpel1 Mod6-1.dd

Scalpel is verbose by default, outputting details of the file types it is searching for and any positive results. Once completed, you will have a new directory created titled scalpel1 that contains files successfully carved.

Compare the results returned from Foremost and Scalpel for the Mod6-1.dd image file. Note that Foremost found wav and wmv files whereas Scalpel found wav and mpg files. Viewing the wav files, both have different names, and different file sizes. This is why further analysis is important.

Run Scalpel against the two (2) other Mod6-\*.dd image files and compare the results with those found from Foremost. If file types are not being detected by Scalpel, research how you would add these file types. Knowing what file types are needed to be carved is one challenge with data carving and is why using multiple tools are recommended.

## Task 4: File Carving with PhotoRec

PhotoRec is an open-source data recovery utility that is actively maintained and updated. It is another tool that should be part of your data carving arsenal, often finding file types that the previous two tools were not able to find.

To install PhotoRec in your Kali environment, run the following command:

sudo apt install testdisk

You can view the command line options for PhotoRec by running the photorec -h command.

$ **photorec --help**

PhotoRec 7.1, Data Recovery Utility, July 2019

Christophe GRENIER <grenier@cgsecurity.org>

https://www.cgsecurity.org

Usage: photorec [/log] [/debug] [/d recup\_dir] [file.dd|file.e01|device]

photorec /version

/log : create a photorec.log file

/debug : add debug information

PhotoRec searches for various file formats (JPEG, Office...). It stores files

in the recup\_dir directory.

Much like the previous tools, we can execute PhotoRec against our image file, and define an output directory, by running the following command:

photorec /d photorec1 Mod6-1.dd

PhotoRec requires some additional actions before it will start data carving. The navigation takes place within the command terminal and can be navigated using your cursor keys. The first action is to confirm the file that you want to perform data carving on. You should have a similar interface as shown in Figure 1.

A screenshot of a computer

Description automatically generated

Figure - PhotoRec Media Select Screen

We provided our input image file at the command line, and this is reflected in this screen. The highlighted disk is named Mod6-1.dd which is the filename we provided. We continue to the next screen by selecting **Proceed**, which is already highlighted, and pressing **Enter**.

The next screen, shown in Figure 2, allows us to select which partition we want to data carve. Not every dump of a device will be a valid disk image with partitions, as we can see in this image. No partition has been detected.

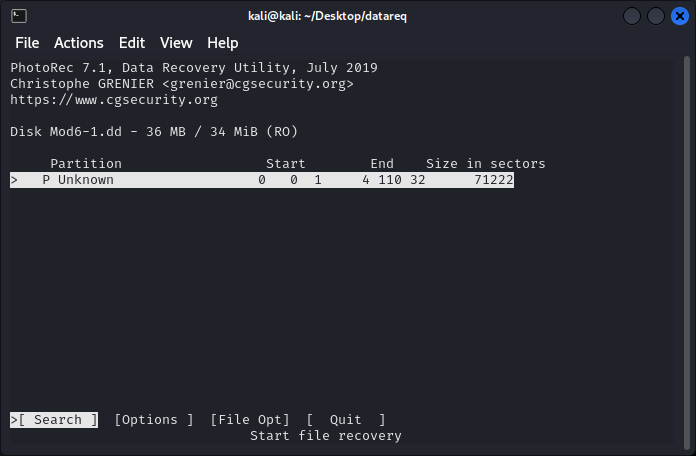


Figure - PhotoRec Partition Select Screen

Since there is no partition detected, we will proceed with our data carving over the entire input file data. Before we start data carving, we can look at the file types that are going to be searched for. Navigate to the **File Opt** selection using the right arrow. The highlighted box at the bottom of the terminal window will move as you navigate. Once **File Opt** is selected, press **Enter**. The file types enabled for detection will be listed, shown in Figure 3.

A screenshot of a computer

Description automatically generated

Figure - PhotoRec File Signatures Screen

You can navigate the available file types using the Up and Down arrows. Have a look through the available files and if there are any additional file types you want to include, enable them by pressing the **Spacebar**. You can disable file types in the same manner.

Once you are finished in this screen, the only option available is **Quit**, which is highlighted by default, so press **Enter**. You will be returned to the partition select screen. At the partition screen, select **Search** and press **Enter**.

PhotoRec supports a range of different file systems and will attempt to auto detect the type that is associated with the input data. For our input data PhotoRec has detected the file type as **Other**, shown in Figure 4, which we will accept by pressing **Enter**.

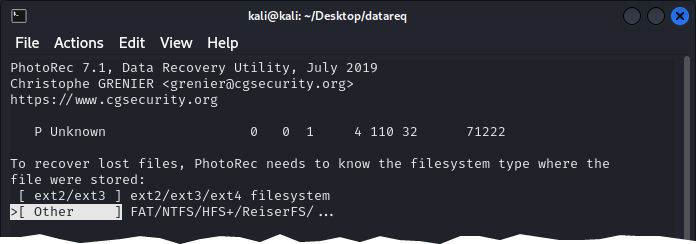


Figure - PhotoRec Filesystem Screen

Data carving will now start and on completion a count of files carved, and the output directory will be shown, see Figure 5. For our image file PhotoRec has carved 4 files and saved them to the photorec1 directory. Press **Enter** to go back to the partition screen, then navigate to **Quit** and press **Enter**, as many times as required, to exit PhotoRec.

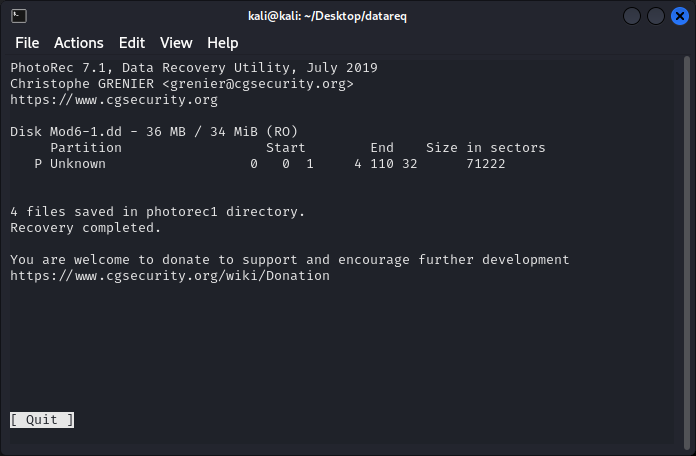


Figure - PhotoRec Completed Screen

Once back at the terminal, navigate to the output folder where PhotoRec saved the carved files. Note that the folder will be named photorec1.X. The X will increment if you provide the same folder name for output to avoid contamination of your output data.

Changing to the output directory and listing out the files with ls -la shows us the files that have been carved.

$ **cd photorec1.1**

$ **ls -la**

total 15632

drwxrwxr-x 2 kali kali 4096 Aug 31 21:15 .

drwxrwxr-x 10 kali kali 4096 Aug 31 21:19 ..

-rw-rw-r-- 1 kali kali 1053174 Aug 31 21:15 f0010000.mp3

-rw-rw-r-- 1 kali kali 4612660 Aug 31 21:15 f0022057.wav

-rw-rw-r-- 1 kali kali 9243672 Aug 31 21:15 f0041067.au

-rw-rw-r-- 1 kali kali 1074873 Aug 2 2012 f0069122.wma

-rw-rw-r-- 1 kali kali 2199 Aug 31 21:17 report.xml

Once again we have different results to the other tools we have previously used. This time we have an mp3 file that Foremost and Scalpel did not detect. The report.xml is generated by PhotoRec and contains details about the carving that has occurred.

## Task 5: File Carving with Autopsy

The last tool we will discuss is Autopsy. Autopsy performs data carving by sending unallocated space to PhotoRec. Note that Autopsy only data carves unallocated space. If files are in allocated space in an image, there is no need to carve them, so they are ignored.

Using Autopsy, create a new case and select Mod6-1.dd as the data source. We have used Autopsy before, so these steps are not duplicated here. Once your data source has been processed by Autopsy, you can navigate to the carved files using the navigation tree on the left-hand side, shown in Figure 6.

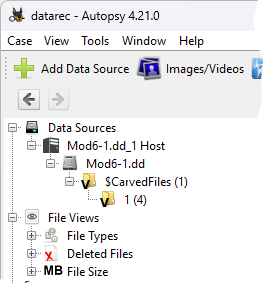


Figure - Autopsy Navigation Tree

Selecting the folder under $CarvedFiles shows four (4) files have been carved, consistent with what we carved used PhotoRec. The listing of files Autopsy has found is shown below in Figure 7.

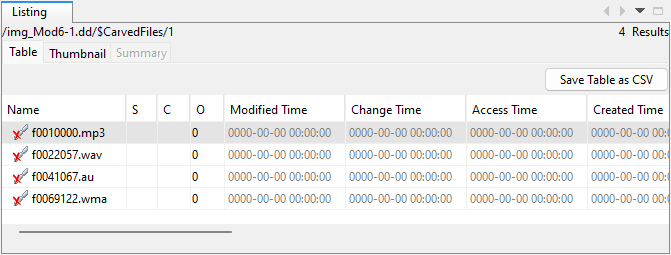


Figure - Autopsy Carved Files Listing

You can now use Autopsy to extract and save these files by right-clicking each file or selected multiple files and right-clicking. A context menu will be shown that allows you to **Extract Files(s)**. When extracting multiple files, it is worth noting that an Autopsy generated internal ID for the file is prefixed to the filename. Multiple file exactions at once will result in different filenames to those listed. If you wish to remain filename integrity, extract files one at a time, or rename post extraction to remove the ID.

## Task 6: Data carving from a USB Thumb Drive

In this task you will simulate the actions of a normal user storing files on a thumb drive, accidentally erasing the files, and then looking to recover any files that were on the drive. Imagine there have been write cycles that prevent a simply restore of data on the thumb drive.

If you need to erase your thumb drive, do so. The best way to ensure there will be no residual data is to write zeros where the image will go. For this activity we will use the first 200Mb of our USB thumb drive.

First, we need to ensure we know which USB device we want to use for this activity. Using the wrong device will lead to undesirable results. We can use the tool lsblk to see which file in the /dev directory represents our USB thumb drive.

To list attached USB devices, the associated /dev file and vendor, run the following command:

lsblk -o name,vendor,size,type

The output from executing lsblk will be a list of all USB devices that your system has detected. You can then verify you are working with the correct device.

$ **lsblk -o name,vendor,size,type**

NAME VENDOR SIZE TYPE

sda VMware, 80.1G disk

└─sda1 80.1G part

sdb TOSHIBA 7.2G disk

└─sdb1 7.2G part

Before executing lsblk we inserted a Toshiba USB with a capacity of 8GB. Looking at the output we would use /dev/sdb for the rest of this activity.

**Verify your USB thumb drive device before continuing**

To write 200MB of zeros to the start of the USB thumb drive, represented by /dev/sdb, which may differ on your system, run the following command:

$ sudo dd if=/dev/zero of=/dev/**sdb** bs=1M count=200

The above command will use dd with the input file of /dev/zero. This is a special file in Unix-like operating systems that provides as many null characters (ASCII NUL, 0x00) as are read from it. Basically, any time you read the /dev/zero file you will get null characters in return.

**Verify your USB thumb drive device before continuing**

Using dd, dump the image file named usbdrive20211021.dd to the USB thumb drive. This can be performed by running the following command:

$ sudo dd if=usbdrive20211021.dd of=/dev/**sdb** bs=4096

You may need to eject the thumb drive and put it back in to initialise the device and its file system.

Scenario

You are handed a USB thumb drive that has been confiscated from a user who is alleged to have committed IP theft. Staff from the IT department have already examined the contents of the device and found nothing suspicious. The head of the IT department has read that files can exist on USB devices even after being deleted and would like you to analyse the device. The IT staff let you use a local Linux workstation to do your analysis that has a collection of open-source command-line forensic tools. GUI tools such as Autopsy are not available to you in this scenario.

You have been tasked to:

1. Forensically examine the disk using command-line accessible tooling. The aim of this task is to perform enough analysis using command line tools to write observations and recover any data.
2. Document your actions, and any observations whilst examining this thumb drive device.
3. Are there any concerns you may have about analysing this USB thumb drive and any evidence collected from it?
4. Is there a file (document, image, pdf, etc) that proves conclusively that the employee did or did not steal IP?